
Goulib Documentation

Release 1.9.21

Ph. Guglielmetti, <https://github.com/goulu/Goulib>

Aug 10, 2017

Contents

1	Requirements	3
2	Modules	5
2.1	Goulib.colors module	6
2.2	Goulib.container module	11
2.3	Goulib.datetime2 module	17
2.4	Goulib.decorators module	27
2.5	Goulib.drawing module	27
2.6	Goulib.expr module	62
2.7	Goulib.geom module	67
2.8	Goulib.geom3d module	98
2.9	Goulib.graph module	107
2.10	Goulib.image module	148
2.11	Goulib.interval module	158
2.12	Goulib.itertools2 module	166
2.13	Goulib.markup module	175
2.14	Goulib.math2 module	191
2.15	Goulib.motion module	203
2.16	Goulib.optim module	214
2.17	Goulib.piecewise module	222
2.18	Goulib.plot module	224
2.19	Goulib.polynomial module	226
2.20	Goulib.statemachine module	229
2.21	Goulib.stats module	231
2.22	Goulib.table module	240
2.23	Goulib.tests module	246
2.24	Goulib.workdays module	253
3	Classes	257
4	Indices and tables	259
	Python Module Index	261

library of useful Python code for scientific + technical applications

see the [IPython notebook](#) for an overview of features

author Philippe Guglielmetti goulib@goulu.net

installation “pip install Goulib”

distribution <https://pypi.python.org/pypi/Goulib>

documentation <https://readthedocs.org/>

notebook <http://nbviewer.ipython.org/github/Goulu/Goulib/blob/master/notebook.ipynb>

source <https://github.com/goulu/Goulib>

CHAPTER 1

Requirements

Goulib uses “lazy” requirements. Many modules and functions do not require any other packages, packages listed in requirements.txt are needed only by some classes or functions

[Sphinx](#) is needed to generate this documentation, [Pythoscope](#) is used to generate nose unit tests

CHAPTER 2

Modules

<i>colors</i>	color conversion in various colorspaces and palettes
<i>container</i>	advanced containers : Record (struct), SortedCollection, and INFINITE Sequence
<i>datetime2</i>	additions to <i>datetime</i> standard library
<i>decorators</i>	useful decorators
<i>drawing</i>	Read/Write and handle vector graphics in .dxf, .svg and .pdf formats
<i>expr</i>	simple symbolic math expressions
<i>geom</i>	2D geometry
<i>geom3d</i>	3D geometry
<i>graph</i>	efficient Euclidian Graphs for <i>networkx</i> and related algorithms
<i>image</i>	
<i>interval</i>	operations on [a..b] intervals
<i>itertools2</i>	additions to <i>itertools</i> standard library
<i>markup</i>	simple HTML/XML generation (forked from <i>markup</i>)
<i>math2</i>	more math than <i>math</i> standard library, without numpy
<i>motion</i>	motion simulation (kinematics)
<i>optim</i>	various optimization algorithms : knapsack, traveling salesman, simulated annealing, differential evolution
<i>piecewise</i>	piecewise-defined functions
<i>plot</i>	plotable rich object display on IPython/Jupyter notebooks
<i>polynomial</i>	simple manipulation of polynomials (without SimPy)
<i>statemachine</i>	state machines with graph representation
<i>stats</i>	very basic statistics functions
<i>table</i>	“mini pandas.DataFrame”
<i>tests</i>	utilities for unit tests (using nose)
<i>workdays</i>	WorkCalendar class with datetime operations on working hours, handling holidays

Goulib.colors module

color conversion in various colorspaces and palettes

`Goulib.colors.rgb2hex` (*c*, *illuminant='ignore'*)

`Goulib.colors.hex2rgb` (*c*, *illuminant='ignore'*)

`Goulib.colors.rgb2cmyk` (*rgb*, ***kwargs*)

Parameters `rgb` – 3-tuple of floats of red,green,blue in [0..1] range

Returns 4-tuple of floats (cyan, magenta, yellow, black) in [0..1] range

`Goulib.colors.cmyk2rgb` (*cmyk*, ***kwargs*)

Parameters `cmyk` – 4-tuple of floats (cyan, magenta, yellow, black) in [0..1] range

Result 3-tuple of floats (red,green,blue)

warning : rgb is out the [0..1] range for some cmyk

`Goulib.colors.xyz2xyy` (*xyz*, ***kwargs*)

Convert from XYZ to xyY

Based on formula from http://brucelindbloom.com/Equn_XYZ_to_xyY.html

Implementation Notes: 1. Watch out for black, where $X = Y = Z = 0$. In that case, x and y are set to the chromaticity coordinates of the reference whitepoint.

2.The output Y value is in the nominal range [0.0, Y[XYZ]].

`Goulib.colors.yyy2xyz` (*xyY*, ***kwargs*)

Convert from xyY to XYZ to

Based on formula from http://brucelindbloom.com/Equn_xyY_to_XYZ.html

Implementation Notes:

1.Watch out for the case where $y = 0$. In that case, you may want to set $X = Y = Z = 0$.

2.The output XYZ values are in the nominal range [0.0, 1.0].

`Goulib.colors.converter` (*c*, *illuminant='ignore'*)

`Goulib.colors.convert` (*color*, *source*, *target*)

convert a color between colorspaces, eventually using intermediary steps

class `Goulib.colors.Color` (*value*, *space='RGB'*, *name=None*, *illuminant='D65'*)

Bases: `object`

A color with math operations and conversions `Color` is immutable (`._values` caches representations)

constructor :param *value*: string color name, hex string, or values tuple :param *space*: string defining the color space of value :param *name*: string for color name :param *illuminant*: string in {"A", "D50", "D55", "D65", "D75", "E"}

•D65 is used by default in skimage, see <http://scikit-image.org/docs/dev/api/skimage.color.html>

•D50 is used in Pantone and other graphic arts

___**init**___ (*value*, *space='RGB'*, *name=None*, *illuminant='D65'*)

constructor :param *value*: string color name, hex string, or values tuple :param *space*: string defining the color space of value :param *name*: string for color name :param *illuminant*: string in {"A", "D50", "D55", "D65", "D75", "E"}

- D65 is used by default in skimage, see <http://scikit-image.org/docs/dev/api/skimage.color.html>

- D50 is used in Pantone and other graphic arts

name

convert (*target*, ***kwargs*)

Parameters *target* – str of desired colorspace, or none for default

Returns color in target colorspace

str (*mode=None*)

native

rgb

hex

lab

luv

cmyk

hsv

xyz

xyY

__hash__ ()

__repr__ ()

compose (*other*, *f*, *mode='rgb'*)
compose colors in given mode

__add__ (*other*)

__radd__ (*other*)
only to allow sum(colors) easily

__sub__ (*other*)

__mul__ (*factor*)

__neg__ ()
complementary color

deltaE (*other*)
color difference according to CIEDE2000 https://en.wikipedia.org/wiki/Color_difference

isclose (*other*, *abs_tol=1*)
<http://zschuessler.github.io/DeltaE/learn/> <= 1.0 Not perceptible by human eyes. 1 - 2 Perceptible through close observation. 2 - 10 Perceptible at a glance. 11 - 49 Colors are more similar than opposite 100 Colors are exact opposite

__eq__ (*other*)

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

`__dir__` () → list
default dir() implementation

`__format__` ()
default object formatter

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__gt__`
Return self>value.

`__le__`
Return self<=value.

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__` () → int
size of object in memory, in bytes

`__str__`
Return str(self).

class Goulib.colors.Palette (data=[], keys=256)

Bases: `collections.OrderedDict`

dict of Colors indexed by anything

`__init__` (data=[], keys=256)

update (data, keys=256)

updates the dictionary with new colors :param data: colors to add :param keys: keys to use in dict, or int to discretize the Colormap

index (c, dE=5)

Returns key of c or nearest color, None if distance is larger than deltaE

`__repr__` ()

patches (wide=64, size=(16, 16))

Image made of each palette color

pil

Returns a sequence of integers, or a string containing a binary

representation of the palette. In both cases, the palette contents should be ordered (r, g, b, r, g, b, ...). The palette can contain up to 768 entries (3*256). If a shorter palette is given, it is padded with zeros. #http://effbot.org/zone/creating-palette-images.htm

sorted (*key*=<function Palette.<lambda>>)

__class__

alias of `type`

__contains__ ()

True if D has a key k, else False.

__delattr__

Implement `delattr(self, name)`.

__delitem__

Delete `self[key]`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__getitem__ ()

`x.__getitem__(y) <==> x[y]`

__gt__

Return `self>value`.

__hash__ = None

__iter__

Implement `iter(self)`.

__le__

Return `self<=value`.

__len__

Return `len(self)`.

__lt__

Return `self<value`.

__ne__

Return `self!=value`.

__new__ ()

Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()

Return state information for pickling

__reduce_ex__ ()

helper for pickle

`__reversed__()` \iff `reversed(od)`

`__setattr__`

Implement `setattr(self, name, value)`.

`__setitem__`

Set `self[key]` to value.

`__sizeof__()`

`__str__`

Return `str(self)`.

`clear()` \rightarrow None. Remove all items from od.

`copy()` \rightarrow a shallow copy of od

`fromkeys(S[, v])` \rightarrow New ordered dictionary with keys from S.

If not specified, the value defaults to None.

`get(k[, d])` \rightarrow `D[k]` if `k` in `D`, else `d`. `d` defaults to None.

`items()`

`keys()`

`move_to_end()`

Move an existing element to the end (or beginning if `last==False`).

Raises `KeyError` if the element does not exist. When `last=True`, acts like a fast version of `self[key]=self.pop(key)`.

`pop(k[, d])` \rightarrow `v`, remove specified key and return the corresponding value. If key is not found, `d` is returned if given, otherwise `KeyError` is raised.

`popitem()` \rightarrow (`k`, `v`), return and remove a (key, value) pair.

Pairs are returned in LIFO order if `last` is true or FIFO order if false.

`setdefault(k[, d])` \rightarrow `od.get(k,d)`, also set `od[k]=d` if `k` not in `od`

`values()`

`Goulib.colors.ColorTable(colors, key=None, width=10)`

`Goulib.colors.color_to_aci(x, nearest=True)`

Returns int Autocad Color Index of color `x`

`Goulib.colors.aci_to_color(x, block_color=None, layer_color=None)`

`Goulib.colors.deltaE(c1, c2)`

`Goulib.colors.nearest_color(c, l=None, opt=<built-in function min>, comp=<function deltaE>)`

Parameters

- `x` – Color
- `l` – list or dict of Color, color by default
- `opt` – with `opt=max` you can find the most different color ...

Returns nearest Color of `x` in `l`

`Goulib.colors.color_range(n, start, end, space='hsv')`

Parameters

- `n` – int number of colors to generate

- **start** – string hex color or color name
- **end** – string hex color or color name

Result list of n Color interpolated between start and end, included

Goulib.container module

advanced containers : Record (struct), SortedCollection, and INFINITE Sequence

class Goulib.container.**Record** (*args, **kwargs)

Bases: collections.OrderedDict

__init__ (*args, **kwargs)

__getattr__ (name)

__setattr__ (name, value)

__str__ ()

__class__

alias of type

__contains__ ()

True if D has a key k, else False.

__delattr__

Implement delattr(self, name).

__delitem__

Delete self[key].

__dir__ () → list

default dir() implementation

__eq__

Return self==value.

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__getitem__ ()

x.__getitem__(y) <=> x[y]

__gt__

Return self>value.

__hash__ = None

__iter__

Implement iter(self).

__le__

Return self<=value.

`__len__`

Return len(self).

`__lt__`

Return self<value.

`__ne__`

Return self!=value.

`__new__` ()

Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()

Return state information for pickling

`__reduce_ex__` ()

helper for pickle

`__repr__`

Return repr(self).

`__reversed__` () <==> *reversed(od)*

`__setitem__`

Set self[key] to value.

`__sizeof__` ()

`clear` () → None. Remove all items from od.

`copy` () → a shallow copy of od

`fromkeys` (S[, v]) → New ordered dictionary with keys from S.
If not specified, the value defaults to None.

`get` (k[, d]) → D[k] if k in D, else d. d defaults to None.

`items` ()

`keys` ()

`move_to_end` ()

Move an existing element to the end (or beginning if last==False).

Raises `KeyError` if the element does not exist. When last=True, acts like a fast version of self[key]=self.pop(key).

`pop` (k[, d]) → v, remove specified key and return the corresponding value. If key is not found, d is returned if given, otherwise `KeyError` is raised.

`popitem` () → (k, v), return and remove a (key, value) pair.
Pairs are returned in LIFO order if last is true or FIFO order if false.

`setdefault` (k[, d]) → od.get(k,d), also set od[k]=d if k not in od

`update` ()

`values` ()

class Goulib.container.**SortedCollection** (iterable=(), key=None)

Bases: `object`

Sequence sorted by a key function.

SortedCollection() is much easier to work with than using bisect() directly. It supports key functions like those use in sorted(), min(), and max(). The result of the key function call is saved so that keys can be searched efficiently.

Instead of returning an insertion-point which can be hard to interpret, the five find-methods return a specific item in the sequence. They can scan for exact matches, the last item less-than-or-equal to a key, or the first item greater-than-or-equal to a key.

Once found, an item's ordinal position can be located with the index() method. New items can be added with the insert() and insert_right() methods. Old items can be deleted with the remove() method.

The usual sequence methods are provided to support indexing, slicing, length lookup, clearing, copying, forward and reverse iteration, contains checking, item counts, item removal, and a nice looking repr.

Finding and indexing are $O(\log n)$ operations while iteration and insertion are $O(n)$. The initial sort is $O(n \log n)$.

The key function is stored in the 'key' attribute for easy introspection or so that you can assign a new key function (triggering an automatic re-sort).

In short, the class was designed to handle all of the common use cases for bisect but with a simpler API and support for key functions.

```
>>> from pprint import pprint
>>> from operator import itemgetter
```

```
>>> s = SortedCollection(key=itemgetter(2))
>>> for record in [
...     ('roger', 'young', 30),
...     ('angela', 'jones', 28),
...     ('bill', 'smith', 22),
...     ('david', 'thomas', 32)]:
...     s.insert(record)
```

```
>>> pprint(list(s))           # show records sorted by age
[('bill', 'smith', 22),
 ('angela', 'jones', 28),
 ('roger', 'young', 30),
 ('david', 'thomas', 32)]
```

```
>>> s.find_le(29)           # find oldest person aged 29 or younger
('angela', 'jones', 28)
>>> s.find_lt(28)          # find oldest person under 28
('bill', 'smith', 22)
>>> s.find_gt(28)          # find youngest person over 28
('roger', 'young', 30)
```

```
>>> r = s.find_ge(32)       # find youngest person aged 32 or older
>>> s.index(r)             # get the index of their record
3
>>> s[3]                   # fetch the record at that index
('david', 'thomas', 32)
```

```
>>> s.key = itemgetter(0)   # now sort by first name
>>> pprint(list(s))
[('angela', 'jones', 28),
 ('bill', 'smith', 22),
```

```
('david', 'thomas', 32),  
( 'roger', 'young', 30)]
```

__init__ (*iterable=()*, *key=None*)

key
key function

clear ()

copy ()

__len__ ()

__getitem__ (*i*)

__iter__ ()

__reversed__ ()

__repr__ ()

__reduce__ ()

__contains__ (*item*)

index (*item*)
Find the position of an item. Raise ValueError if not found.

count (*item*)
Return number of occurrences of item

insert (*item*)
Insert a new item. If equal keys are found, add to the left

insert_right (*item*)
Insert a new item. If equal keys are found, add to the right

pop (*i=-1*)

remove (*item*)
Remove first occurrence of item. Raise ValueError if not found

find (*k*)
Return first item with a key == k. Raise ValueError if not found.

find_le (*k*)
Return last item with a key <= k. Raise ValueError if not found.

find_lt (*k*)
Return last item with a key < k. Raise ValueError if not found.

find_ge (*k*)
Return first item with a key >= equal to k. Raise ValueError if not found

find_gt (*k*)
Return first item with a key > k. Raise ValueError if not found

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

`__dir__()` → list
default dir() implementation

`__eq__`
Return self==value.

`__format__()`
default object formatter

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__gt__`
Return self>value.

`__hash__`
Return hash(self).

`__le__`
Return self<=value.

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__()`
Create and return a new object. See help(type) for accurate signature.

`__reduce_ex__()`
helper for pickle

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return str(self).

class `Goulib.container.Sequence` (*iterf=None, itemf=None, containf=None, desc=''*)
Bases: `object`

combines a generator and a read-only list used for INFINITE numeric (integer) sequences

Parameters

- **iterf** – optional iterator, or a function returning an iterator
- **itemf** – optional function(i) returning the i-th element
- **containf** – optional function(n) return bool True if n belongs to Sequence
- **desc** – string description

`__init__` (*iterf=None, itemf=None, containf=None, desc=''*)

Parameters

- **iterf** – optional iterator, or a function returning an iterator

- **itemf** – optional function(i) returning the i-th element
- **containf** – optional function(n) return bool True if n belongs to Sequence
- **desc** – string description

__repr__ ()

save (*filename*, *comment=None*, *n=1000*, *maxtime=10*)

__iter__ ()

reset the generator

Returns a tee-ed copy of iterf

__getitem__ (*i*)

index (*v*)

__contains__ (*n*)

__add__ (*other*)

__sub__ (*other*)

__or__ (*other*)

Returns Sequence with items from both operands

__and__ (*other*)

Returns Sequence with items in both operands

__mod__ (*other*)

Returns Sequence with items from left operand not in right

apply (*f*, *containf=None*, *desc=''*)

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__le__

Return `self<=value`.

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__()`
Create and return a new object. See help(type) for accurate signature.

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return str(self).

filter (*f, desc=''*)

accumulate (*op=<built-in function add>, skip_first=False*)

pairwise (*op, skip_first=False*)

sort (*key=None, buffer=100*)

unique (*buffer=100*)

Parameters **buffer** – int number of last elements found.

if two identical elements are separated by more than this number of elements in self, they might be generated twice in the resulting Sequence :return: Sequence made of unique elements of this one

Goulib.datetime2 module

additions to `datetime` standard library

class Goulib.datetime2.**datetime2** (**args, **kwargs*)

Bases: `datetime.datetime`

`__init__` (**args, **kwargs*)

`__sub__` (*other*)

`__add__`
Return self+value.

`__class__`
alias of `type`

`__delattr__`
Implement delattr(self, name).

`__dir__` () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
Formats self with strftime.

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__radd__
Return value+self.

__reduce__ () -> (cls, state)

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__rsub__
Return value-self.

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

astimezone ()
tz -> convert to local time in new timezone tz

combine ()
date, time -> datetime with same date and time fields

ctime ()
Return ctime() style string.

date ()
Return date object with same year, month and day.

day

dst ()

Return self.tzinfo.dst(self).

fromordinal ()

int -> date corresponding to a proleptic Gregorian ordinal.

fromtimestamp ()

timestamp[, tz] -> tz's local time from POSIX timestamp.

hour

isocalendar ()

Return a 3-tuple containing ISO year, week number, and weekday.

isoformat ()

[sep] -> string in ISO 8601 format, YYYY-MM-DDTHH:MM:SS[.mmmmmm][+HH:MM].

sep is used to separate the year from the time, and defaults to 'T'.

isoweekday ()

Return the day of the week represented by the date. Monday == 1 ... Sunday == 7

max = datetime.datetime(9999, 12, 31, 23, 59, 59, 999999)

microsecond

min = datetime.datetime(1, 1, 1, 0, 0)

minute

month

now ()

Returns new datetime object representing current time local to tz.

tz Timezone object.

If no tz is specified, uses local timezone.

replace ()

Return datetime with new specified fields.

resolution = datetime.timedelta(0, 0, 1)

second

strftime ()

format -> strftime() style string.

strptime ()

string, format -> new datetime parsed from a string (like time.strptime()).

time ()

Return time object with same time but with tzinfo=None.

timestamp ()

Return POSIX timestamp as float.

timetuple ()

Return time tuple, compatible with time.localtime().

timetz ()

Return time object with same time and tzinfo.

today ()

Current date or datetime: same as `self.__class__.fromtimestamp(time.time())`.

toordinal ()

Return proleptic Gregorian ordinal. January 1 of year 1 is day 1.

tzinfo

tzname ()

Return `self.tzinfo.tzname(self)`.

utcfromtimestamp ()

Construct a naive UTC datetime from a POSIX timestamp.

utcnow ()

Return a new datetime representing UTC day and time.

utcoffset ()

Return `self.tzinfo.utcoffset(self)`.

utctimetuple ()

Return UTC time tuple, compatible with `time.localtime()`.

weekday ()

Return the day of the week represented by the date. Monday == 0 ... Sunday == 6

year

class `Goulib.datetime2.date2`

Bases: `datetime.date`

__init__ (*args, **kwargs)

__add__

Return `self+value`.

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

Formats `self` with `strftime`.

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__init__

Initialize `self`. See `help(type(self))` for accurate signature.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__()
Create and return a new object. See help(type) for accurate signature.

__radd__
Return value+self.

__reduce__() -> (cls, state)

__reduce_ex__()
helper for pickle

__repr__
Return repr(self).

__rsub__
Return value-self.

__setattr__
Implement setattr(self, name, value).

__sizeof__() → int
size of object in memory, in bytes

__str__
Return str(self).

__sub__
Return self-value.

ctime()
Return ctime() style string.

day

fromordinal()
int -> date corresponding to a proleptic Gregorian ordinal.

fromtimestamp()
timestamp -> local date from a POSIX timestamp (like time.time()).

isocalendar()
Return a 3-tuple containing ISO year, week number, and weekday.

isoformat()
Return string in ISO 8601 format, YYYY-MM-DD.

isoweekday()
Return the day of the week represented by the date. Monday == 1 ... Sunday == 7

max = datetime.date(9999, 12, 31)

min = datetime.date(1, 1, 1)

month

replace()

Return date with new specified fields.

resolution = datetime.timedelta(1)

strftime()

format -> strftime() style string.

timetuple()

Return time tuple, compatible with time.localtime().

today()

Current date or datetime: same as self.__class__.fromtimestamp(time.time()).

toordinal()

Return proleptic Gregorian ordinal. January 1 of year 1 is day 1.

weekday()

Return the day of the week represented by the date. Monday == 0 ... Sunday == 6

year

class Goulib.datetime2.**time2**(*args, **kwargs)

Bases: `datetime.time`

__init__(*args, **kwargs)

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__() → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__()

Formats self with `strftime`.

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__le__

Return `self<=value`.

__lt__

Return `self<value`.

__ne__

Return `self!=value`.

__new__()

Create and return a new object. See `help(type)` for accurate signature.

```

__reduce__ () -> (cls, state)

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () -> int
    size of object in memory, in bytes

__str__
    Return str(self).

dst ()
    Return self.tzinfo.dst(self).

hour

isoformat ()
    Return string in ISO 8601 format, HH:MM:SS[.mmmmmm][+HH:MM].

max = datetime.time(23, 59, 59, 999999)

microsecond

min = datetime.time(0, 0)

minute

replace ()
    Return time with new specified fields.

resolution = datetime.timedelta(0, 0, 1)

second

strftime ()
    format -> strftime() style string.

tzinfo

tzname ()
    Return self.tzinfo.tzname(self).

utcoffset ()
    Return self.tzinfo.utcoffset(self).

class Goulib.datetime2.timedelta2 (*args, **kwargs)
    Bases: datetime.timedelta

    __init__ (*args, **kwargs)

    isoformat ()

    __abs__
        abs(self)

    __add__
        Return self+value.

    __bool__
        self != 0

```

`__class__`
alias of `type`

`__delattr__`
Implement `delattr(self, name)`.

`__dir__` () → list
default `dir()` implementation

`__divmod__`
Return `divmod(self, value)`.

`__eq__`
Return `self==value`.

`__floordiv__`
Return `self//value`.

`__format__` ()
default object formatter

`__ge__`
Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self>value`.

`__hash__`
Return `hash(self)`.

`__le__`
Return `self<=value`.

`__lt__`
Return `self<value`.

`__mod__`
Return `self%value`.

`__mul__`
Return `self*value`.

`__ne__`
Return `self!=value`.

`__neg__`
`-self`

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__pos__`
`+self`

`__radd__`
Return `value+self`.

`__rdivmod__`
Return `divmod(value, self)`.

`__reduce__` () → (*cls, state*)

__reduce_ex__ ()
 helper for pickle

__repr__
 Return repr(self).

__rfloordiv__
 Return value//self.

__rmod__
 Return value%self.

__rmul__
 Return value*self.

__rsub__
 Return value-self.

__rtruediv__
 Return value/self.

__setattr__
 Implement setattr(self, name, value).

__sizeof__ () → int
 size of object in memory, in bytes

__str__
 Return str(self).

__sub__
 Return self-value.

__truediv__
 Return self/value.

days
 Number of days.

max = datetime.timedelta(999999999, 86399, 999999)

microseconds
 Number of microseconds (>= 0 and less than 1 second).

min = datetime.timedelta(-999999999)

resolution = datetime.timedelta(0, 0, 1)

seconds
 Number of seconds (>= 0 and less than 1 day).

total_seconds ()
 Total seconds in the duration.

Goulib.datetime2.**datetime** (d, t=None, fmt='%Y-%m-%d')

“converts something to a datetime :param d: can be:

- datetime : result is a copy of d with time optionally replaced
- date : result is date at time t, (00:00AM by default)
- int or float : if fmt is None, d is considered as Excel date numeric format (see <http://answers.oreilly.com/topic/1694-how-excel-stores-date-and-time-values/>)
- string or specified format: result is datetime parsed using specified format string

Parameters

- **fmt** – format string. See <http://docs.python.org/2/library/datetime.html#strftime-strptime-behavior>
- **t** – optional time. replaces the time of the datetime obtained from d. Allows `datetimeef(date,time)`

Returns datetime

`Goulib.datetime2.datetimeef(d,fmt='%Y-%m-%d')`
converts something to a date. See `datetimeef`

`Goulib.datetime2.timeef(t,fmt='%H:%M:%S')`
converts something to a time. See `datetimeef`

`Goulib.datetime2.timedeltaf(t,fmt=None)`
converts something to a timedelta. :param t: can be: * already a timedelta, or a time, or a int/float giving a number of days (Excel) * or a string in HH:MM:SS format by default * or a string in timedelta str() output format

`Goulib.datetime2.strftimedelta(t,fmt='%H:%M:%S')`

Parameters **t** – float seconds or timedelta

`Goulib.datetime2.tdround(td,s=1)`
return timedelta rounded to s seconds

`Goulib.datetime2.minutes(td)`

Parameters **td** – timedelta**Returns** float timedelta in minutes

`Goulib.datetime2.hours(td)`

Parameters **td** – timedelta**Returns** float timedelta in hours

`Goulib.datetime2.daysgen(start,length,step=datetime.timedelta(1))`
returns a range of dates or datetimes

`Goulib.datetime2.days(start,length,step=datetime.timedelta(1))`

`Goulib.datetime2.timedelta_sum(timedeltas)`

`Goulib.datetime2.timedelta_div(t1,t2)`
divides a timedelta by a timedelta or a number. should be a method of timedelta...

`Goulib.datetime2.timedelta_mul(t1,t2)`
multiplies a timedelta. should be a method of timedelta...

`Goulib.datetime2.time_sub(t1,t2)`
subtracts 2 time. should be a method of time...

`Goulib.datetime2.time_add(t,d)`
adds delta to time. should be a method of time...

`Goulib.datetime2.add_months(date,months)`

`Goulib.datetime2.date_add(date,years=0,months=0,weeks=0,days=0)`

`Goulib.datetime2.equal(a,b,epsilon=datetime.timedelta(0,0,500000))`
approximately equal. Use this instead of `a==b` :return: True if a and b are less than seconds apart

`Goulib.datetime2.datetime_intersect (t1, t2)`
 returns timedelta overlap between 2 intervals (tuples) of datetime

`Goulib.datetime2.time_intersect (t1, t2)`
 returns timedelta overlap between 2 intervals (tuples) of time

Goulib.decorators module

useful decorators

`Goulib.decorators.memoize (obj)`

`Goulib.decorators.debug (func)`

`Goulib.decorators.noddebug (func)`

`Goulib.decorators.get_thread_pool ()`

`Goulib.decorators.timeout (timeout)`

`Goulib.decorators.itimeout (iterable, timeout)`
 timeout for loops :param iterable: any iterable :param timeout: float max running time in seconds :yield: items
 in iterator until timeout occurs :raise: multiprocessing.TimeoutError if timeout occurred

Goulib.drawing module

Read/Write and handle vector graphics in .dxf, .svg and .pdf formats

requires

- [svg.path](#) for svg input
- [matplotlib](#) for bitmap + svg and pdf output
- [dxfwrite](#) for dxf output

optional

- [dxfgrabber](#) for dxf input
- [pdfminer.six](#) for pdf input

`Goulib.drawing.Trans (scale=1, offset=None, rotation=None)`

Parameters

- **scale** – float or (scalex,scaley) tuple of scale factor
- **offset** – `Vector3`
- **rotation** – float angle in degrees

Returns `Matrix3` of generalized scale+offset+rotation

class `Goulib.drawing.BBox (p1=None, p2=None)`

Bases: `Goulib.interval.Box`

bounding box

Parameters

- **pt1** – `Point2` first corner (any)

- **pt2** – Point2 opposite corner (any)

`__init__` (*p1=None, p2=None*)

Parameters

- **pt1** – Point2 first corner (any)
- **pt2** – Point2 opposite corner (any)

xmin

ymin

xmax

ymax

xmed

ymed

width

height

area

`__contains__` (*other*)

Returns True if other lies in bounding box.

`__iadd__` (*pt*)

enlarge box if required to contain specified point :param pt1: `geom.Point2` point to add

`__call__` ()

Returns list of flatten corners

size ()

Returns `geom.Vector2` with xy sizes

center ()

Returns Pt center

trans (*trans*)

Parameters **trans** – Xform

Returns `BBox` = self transformed by trans

class `Goulib.drawing.Entity`

Bases: `Goulib.plot.Plot`

Base class for all drawing entities

color = 'black'

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

end

`__repr__` ()

center

bbox ()

Returns *BBox* bounding box of Entity

isclosed ()

isline ()

isvertical (*tol=0.01*)

ishorizontal (*tol=0.01*)

to_dxf (***attr*)

Parameters *attr* – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

static from_svg (*path, color*)

Parameters *path* – svg path

Returns Entity of correct subtype

static from_pdf (*path, trans, color*)

Parameters *path* – pdf path

Returns Entity of correct subtype

static from_dxf (*e, mat3*)

Parameters

- *e* – dxf.entity
- *mat3* – Matrix3 transform

Returns Entity of correct subtype

patches (***kwargs*)

Returns list of (a single) *Patch* corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

static figure (*box, **kwargs*)

Parameters

- *box* – `drawing.BBox` bounds and clipping box
- *kwargs* – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

draw (*fig=None, **kwargs*)

draw entities :param *fig*: matplotlib figure where to draw. `figure(g)` is called if missing :return: `fig.patch`

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

class `Goulib.drawing.Spline` (*points*)

Bases: `Goulib.drawing.Entity`, `Goulib.geom.Geometry`

cubic spline segment

Parameters *points* – list of (x,y) tuples

```
__init__(points)
    Parameters points – list of (x,y) tuples
start
end
xy
length
    Returns float (very) approximate length
bbox()
swap()
    swap start and end
__abstractmethods__ = frozenset()
```

class Goulib.drawing.**Group**

Bases: list, Goulib.drawing._Group

group of Entities but it is a Geometry since we can intersect, connect and compute distances between Groups

color

layer

append(entity, **kwargs)

append entity to group :param entity: Entity :param kwargs: dict of attributes copied to entity :return: Group (or Chain) to which the entity was added, or None if entity was None

extend(entities, **kwargs)

__copy__()

swap()

swap start and end

chainify(mergeable)

merge all possible entities into chains

from_dxf(dxf, layers=None, only=[], ignore=['POINT'], trans=Matrix3(1.0, 0, 0, 0, 1.0, 0, 0, 0, 1.0), flatten=False)

Parameters

- **dxf** – dxf.entity
- **layers** – list of layer names to consider. entities not on these layers are ignored. default=None: all layers are read
- **only** – list of dxf entity types names that are read. default=[]: all are read
- **ignore** – list of dxf entity types names that are ignored. default=['POINT']: points and null length segments are ignored
- **trans** – *Trans* optional transform matrix

Parm flatten bool flatten block structure

Returns *Entity* of correct subtype

__abstractmethods__ = frozenset()

class Goulib.drawing.**Instance**(group, trans)

Bases: Goulib.drawing._Group

Parameters

- **group** – Group
- **trans** – optional mat3 of transformation

`__init__` (*group, trans*)

Parameters

- **group** – Group
- **trans** – optional mat3 of transformation

static `from_dxf` (*e, blocks, mat3*)

Parameters

- **e** – dxf.entity
- **blocks** – dict of Groups indexed by name
- **mat3** – Matrix3 transform

`__repr__` ()

`__iter__` ()

`__abstractmethods__` = frozenset()

class `Goulib.drawing.Chain` (*data=[]*)

Bases: `Goulib.drawing.Group`

group of contiguous Entities (Polyline or similar)

`__init__` (*data=[]*)

start

end

`__repr__` ()

contiguous (*edge, tol=1e-06, allow_swap=True*)

check if edge can be appended to the chain :param edge: `Entity` to append :param tol: float tolerance on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :return: int,bool index where to append in chain, swap of edge required

append (*entity, tol=1e-06, allow_swap=True, mergeable=None, **attrs*)

append entity to chain, ensuring contiguity :param entity: `Entity` to append :param tol: float tolerance on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :param mergeable: function of the form f(e1,e2) returning True if entities e1,e2 can be merged :param attrs: attributes passed to Group.append :return: self, or None if edge is not contiguous

static `from_pdf` (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

See http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf p. 132

static `from_svg` (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

static from_dxf (*e, mat3*)

Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

to_dxf (*split=False, **attr*)

Parameters

- **split** – bool if True, each segment in Chain is saved separately
- **attr** – dict of graphic attributes

Returns polyline or list of entities along the chain

__abstractmethods__ = frozenset()

Goulib.drawing.**chains** (*group, tol=1e-06, mergeable=None*)

build chains from all possible segments in group :param mergeable: function(e1,e2) returning True if entities e1,e2 can be merged

class Goulib.drawing.**Rect** (**args*)

Bases: *Goulib.drawing.Chain*

a rectangle starting at low/left and going trigowise through top/right

__init__ (**args*)

p1

p2

__repr__ ()

__abstractmethods__ = frozenset()

class Goulib.drawing.**Text** (*text, point, size=12, rotation=0*)

Bases: *Goulib.drawing.Entity*

Parameters

- **text** – string
- **point** – Point2
- **size** – size in points
- **rotation** – float angle in degrees trigowise

__init__ (*text, point, size=12, rotation=0*)

Parameters

- **text** – string
- **point** – Point2
- **size** – size in points
- **rotation** – float angle in degrees trigowise

bbox ()

length

Returns float length of the text contour in mm

intersect (*other*)

to_dxf (***attr*)

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

class `Goulib.drawing.Drawing` (*data=[]*, ***kwargs*)

Bases: `Goulib.drawing.Group`

list of Entities representing a vector graphics drawing

__abstractmethods__ = frozenset()

__init__ (*data=[]*, ***kwargs*)

load (*filename*, ***kwargs*)

read_pdf (*filename*, ***kwargs*)

reads a vector graphics on a .pdf file only the first page is parsed

read_svg (*content*, ***kwargs*)

appends svg content to drawing :param content: string, either filename or svg content

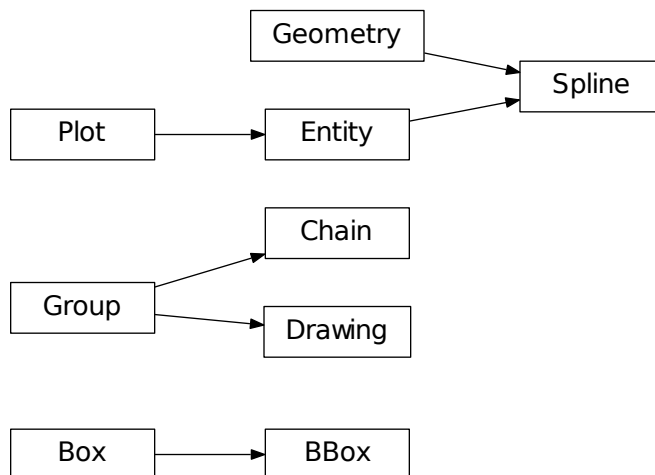
read_dxf (*filename*, *options=None*, ***kwargs*)

reads a .dxf file :param filename: string path to .dxf file to read :param options: passed to `from_dxf`

save (*filename*, ***kwargs*)

save graph in various formats

Classes



graphics in .dxf, .svg and .pdf formats

requires

Read/Write and handle vector

- `svg.path` for svg input
- `matplotlib` for bitmap + svg and pdf output
- `dxfwrite` for dxf output

optional

- `dxgrabber` for dxf input
- `pdfminer.six` for pdf input

`Goulib.drawing.Trans` (*scale=1, offset=None, rotation=None*)

Parameters

- **scale** – float or (scalex,scaley) tuple of scale factor
- **offset** – `Vector3`
- **rotation** – float angle in degrees

Returns `Matrix3` of generalized scale+offset+rotation

class `Goulib.drawing.BBox` (*p1=None, p2=None*)

Bases: `Goulib.interval.Box`

bounding box

Parameters

- **pt1** – `Point2` first corner (any)
- **pt2** – `Point2` opposite corner (any)

`__init__` (*p1=None, p2=None*)

Parameters

- **pt1** – `Point2` first corner (any)
- **pt2** – `Point2` opposite corner (any)

xmin

ymin

xmax

ymax

xmed

ymed

width

height

area

`__contains__` (*other*)

Returns True if other lies in bounding box.

`__iadd__` (*pt*)

enlarge box if required to contain specified point :param pt1: `geom.Point2` point to add

`__call__` ()

Returns list of flatten corners

size()

Returns `geom.Vector2` with xy sizes

center()

Returns Pt center

trans (*trans*)

Parameters **trans** – Xform

Returns *BBBox* = self transformed by trans

__add__ (*other*)

enlarge box if required to contain specified point :param other: Box or (list of) N-tuple point(s) :return: new Box containing both

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__delitem__

Delete `self[key]`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__getitem__ ()

`x.__getitem__(y) <==> x[y]`

__gt__

Return `self>value`.

__hash__ = None

__imul__

Implement `self*=value`.

__iter__

Implement `iter(self)`.

__le__

Return `self<=value`.

__len__

Return `len(self)`.

__lt__

Return `self<value`.

__mul__
Return self*value.n

__ne__
Return self!=value.

__new__()
Create and return a new object. See help(type) for accurate signature.

__nonzero__()

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__
Return repr(self).

__reversed__()
L.__reversed__() – return a reverse iterator over the list

__rmul__
Return self*value.

__setattr__
Implement setattr(self, name, value).

__setitem__
Set self[key] to value.

__sizeof__()
L.__sizeof__() – size of L in memory, in bytes

__str__
Return str(self).

append (*object*) → None – append object to end

clear () → None – remove all items from L

copy () → list – a shallow copy of L

corner (*n*)
return n-th corner of box 0-th corner is “start” made of all minimal values of intervals -1.th corner is “end”, made of all maximal values of intervals

count (*value*) → integer – return number of occurrences of value

empty ()
Returns True iff Box is empty.

end

extend (*iterable*) → None – extend list by appending elements from the iterable

index (*value* [, *start* [, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

insert ()
L.insert(index, object) – insert object before index

max

min

pop (*[index]*) → item – remove and return item at index (default last).
Raises `IndexError` if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.
Raises `ValueError` if the value is not present.

reverse ()
L.reverse() – reverse *IN PLACE*

sort (*key=None, reverse=False*) → None – stable sort **IN PLACE**

start

class Goulib.drawing.**Entity**

Bases: *Goulib.plot.Plot*

Base class for all drawing entities

color = 'black'

setattr (***kwargs*)
set (graphic) attributes to entity :param *kwargs*: dict of attributes copied to entity

start

end

__repr__ ()

center

bbox ()

Returns *BBBox* bounding box of Entity

isclosed ()

isline ()

isvertical (*tol=0.01*)

ishorizontal (*tol=0.01*)

to_dxf (***attr*)

Parameters *attr* – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

static from_svg (*path, color*)

Parameters *path* – svg path

Returns Entity of correct subtype

static from_pdf (*path, trans, color*)

Parameters *path* – pdf path

Returns Entity of correct subtype

static from_dxf (*e, mat3*)

Parameters

- *e* – dxf.entity
- *mat3* – Matrix3 transform

Returns Entity of correct subtype

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

Note this is the only method that needs to be overridden in descendants for `draw`, `render` and `IPython_repr_XXX_` to work

static figure (*box, **kwargs*)

Parameters

- **box** – `drawing.BBox` bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

draw (*fig=None, **kwargs*)

draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig.patch

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__init__

Initialize self. See `help(type(self))` for accurate signature.

__le__

Return `self<=value`.

__lt__

Return `self<value`.

__ne__

Return `self!=value`.

__new__ ()

Create and return a new object. See `help(type)` for accurate signature.

```

__reduce__()
    helper for pickle

__reduce_ex__()
    helper for pickle

__setattr__
    Implement setattr(self, name, value).

__sizeof__() → int
    size of object in memory, in bytes

__str__
    Return str(self).

html (**kwargs)

plot (**kwargs)
    renders on IPython Notebook (alias to make usage more straightforward)

png (**kwargs)

save (filename, **kwargs)

svg (**kwargs)

```

```

class Goulib.drawing.Spline (points)
    Bases: Goulib.drawing.Entity, Goulib.geom.Geometry

```

cubic spline segment

Parameters *points* – list of (x,y) tuples

```

__init__ (points)
    Parameters points – list of (x,y) tuples

```

start

end

xy

length

Returns float (very) approximate length

bbox ()

```

swap ()
    swap start and end

```

```

__abstractmethods__ = frozenset()

```

```

__class__
    alias of ABCMeta

```

```

__contains__ (pt)

```

```

__delattr__
    Implement delattr(self, name).

```

```

__dir__ () → list
    default dir() implementation

```

```

__eq__
    Return self==value.

```

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__ ()

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

center

color = 'black'

connect (*other*)
Returns Geometry shortest (Segment2 or Segment3) that connects self to other

distance (*other*)

draw (*fig=None, **kwargs*)
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

figure (*box, **kwargs*)
Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (*e, mat3*)

Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

intersect (*other*)

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter u

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

svg (***kwargs*)

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter u

to_dxf (***attr*)

Parameters **attr** – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

class Goulib.drawing.**Group**

Bases: list, Goulib.drawing._Group

group of Entities but it is a Geometry since we can intersect, connect and compute distances between Groups

color

layer

append (*entity*, ***kwargs*)

append entity to group :param entity: Entity :param kwargs: dict of attributes copied to entity :return: Group (or Chain) to which the entity was added, or None if entity was None

extend (*entities*, ***kwargs*)

__copy__ ()

swap ()

swap start and end

chainify (*mergeable*)

merge all possible entities into chains

from_dxf (*dxf*, *layers=None*, *only=[]*, *ignore=['POINT']*, *trans=Matrix3(1.0, 0, 0, 0, 1.0, 0, 0, 0, 1.0)*, *flatten=False*)

Parameters

- **dxf** – dxf.entity
- **layers** – list of layer names to consider. entities not on these layers are ignored. default=None: all layers are read
- **only** – list of dxf entity types names that are read. default=[]: all are read
- **ignore** – list of dxf entity types names that are ignored. default=['POINT']: points and null length segments are ignored
- **trans** – *Trans* optional transform matrix

Parm flatten bool flatten block structure

Returns *Entity* of correct subtype

__abstractmethods__ = frozenset()

__add__

Return self+value.

__class__

alias of ABCMeta

__contains__

Return key in self.

__delattr__

Implement delattr(self, name).

__delitem__

Delete self[key].

__dir__ () → list

default dir() implementation

__eq__

Return self==value.

`__format__` ()
 default object formatter

`__ge__`
 Return self>=value.

`__getattr__`
 Return getattr(self, name).

`__getitem__` ()
 x.__getitem__(y) <==> x[y]

`__gt__`
 Return self>value.

`__hash__` = None

`__iadd__`
 Implement self+=value.

`__imul__`
 Implement self*=value.

`__init__`
 Initialize self. See help(type(self)) for accurate signature.

`__iter__`
 Implement iter(self).

`__le__`
 Return self<=value.

`__len__`
 Return len(self).

`__lt__`
 Return self<value.

`__mul__`
 Return self*value.n

`__ne__`
 Return self!=value.

`__new__` ()
 Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
 helper for pickle

`__reduce_ex__` ()
 helper for pickle

`__repr__`
 Return repr(self).

`__reversed__` ()
 L.__reversed__() – return a reverse iterator over the list

`__rmul__`
 Return self*value.

`__setattr__`
 Implement setattr(self, name, value).

`__getitem__`

Set self[key] to value.

`__sizeof__` ()

L.`__sizeof__`() – size of L in memory, in bytes

`__str__`

Return str(self).

bbox (*filter=None*)

Parameters **filter** – optional function(entity):bool returning True if entity should be considered in box

Returns *BBox* bounding box of Entity

center

clear () → None – remove all items from L

connect (*other*)

copy () → list – a shallow copy of L

count (*value*) → integer – return number of occurrences of value

distance (*other*)

draw (*fig=None, **kwargs*)

draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (*box, **kwargs*)

Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.

Raises ValueError if the value is not present.

insert ()

L.insert(index, object) – insert object before index

intersect (*other*)

Parameters **other** – *geom.Entity*

Result generate tuples (Point2,Entity_self) of intersections between other and each Entity

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

length

patches (***kwargs*)

Returns list of `Patch` corresponding to group

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

pop (*[index]*) → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.

Raises `ValueError` if the value is not present.

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

reverse ()

`L.reverse()` – reverse *IN PLACE*

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param *kwargs*: dict of attributes copied to entity

sort (*key=None, reverse=False*) → None – stable sort **IN PLACE**

start

svg (***kwargs*)

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter *u*

to_dxf (***kwargs*)

Returns flatten list of dxf entities

class `Goulib.drawing.Instance` (*group, trans*)

Bases: `Goulib.drawing._Group`

Parameters

- **group** – Group
- **trans** – optional mat3 of transformation

__init__ (*group, trans*)

Parameters

- **group** – Group

- **trans** – optional mat3 of transformation

static from_dxf (*e*, *blocks*, *mat3*)

Parameters

- **e** – dxf.entity
- **blocks** – dict of Groups indexed by name
- **mat3** – Matrix3 transform

__repr__ ()

__iter__ ()

__abstractmethods__ = frozenset()

__class__

alias of ABCMeta

__contains__ (*pt*)

__delattr__

Implement delattr(self, name).

__dir__ () → list

default dir() implementation

__eq__

Return self==value.

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__gt__

Return self>value.

__hash__

Return hash(self).

__le__

Return self<=value.

__lt__

Return self<value.

__ne__

Return self!=value.

__new__ ()

Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

helper for pickle

__reduce_ex__ ()

helper for pickle

__setattr__
 Implement setattr(self, name, value).

__sizeof__ () → int
 size of object in memory, in bytes

__str__
 Return str(self).

bbox (*filter=None*)
Parameters **filter** – optional function(entity):bool returning True if entity should be considered in box
Returns *BBox* bounding box of Entity

center

color = 'black'

connect (*other*)

distance (*other*)

draw (*fig=None, **kwargs*)
 draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (*box, **kwargs*)
Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to *~matplotlib.pyplot.figure*

Returns matplotlib axis suitable for drawing

from_pdf (*path, trans, color*)
Parameters **path** – pdf path
Returns Entity of correct subtype

from_svg (*path, color*)
Parameters **path** – svg path
Returns Entity of correct subtype

html (***kwargs*)

intersect (*other*)
Parameters **other** – *geom.Entity*
Result generate tuples (Point2,Entity_self) of intersections between other and each Entity

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

length

patches (***kwargs*)

Returns list of `Patch` corresponding to group

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter *u*

to_dxf (***kwargs*)

Returns flatten list of dxf entities

class `Goulib.drawing.Chain` (*data=[]*)

Bases: `Goulib.drawing.Group`

group of contiguous Entities (Polyline or similar)

__init__ (*data=[]*)

start

end

__repr__ ()

contiguous (*edge, tol=1e-06, allow_swap=True*)

check if edge can be appended to the chain :param edge: `Entity` to append :param tol: float tolerance on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :return: int,bool index where to append in chain, swap of edge required

append (*entity, tol=1e-06, allow_swap=True, mergeable=None, **attrs*)

append entity to chain, ensuring contiguity :param entity: `Entity` to append :param tol: float tolerance on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :param mergeable: function of the form `f(e1,e2)` returning True if entities `e1,e2` can be merged :param attrs: attributes passed to `Group.append` :return: self, or None if edge is not contiguous

static from_pdf (*path, trans, color*)

Parameters `path` – pdf path

Returns Entity of correct subtype

See http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf p. 132

static from_svg (*path, color*)

Parameters `path` – svg path

Returns Entity of correct subtype

static from_dxf (*e*, *mat3*)

Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

to_dxf (*split=False*, ***attr*)

Parameters

- **split** – bool if True, each segment in Chain is saved separately
- **attr** – dict of graphic attributes

Returns polyline or list of entities along the chain

__abstractmethods__ = frozenset()

__add__

Return self+value.

__class__

alias of ABCMeta

__contains__

Return key in self.

__copy__ ()

__delattr__

Implement delattr(self, name).

__delitem__

Delete self[key].

__dir__ () → list

default dir() implementation

__eq__

Return self==value.

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__getitem__ ()

x.__getitem__(y) <==> x[y]

__gt__

Return self>value.

__hash__ = None

__iadd__

Implement self+=value.

`__imul__`
Implement `self*=value`.

`__iter__`
Implement `iter(self)`.

`__le__`
Return `self<=value`.

`__len__`
Return `len(self)`.

`__lt__`
Return `self<value`.

`__mul__`
Return `self*value.n`

`__ne__`
Return `self!=value`.

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__reversed__` ()
`L.__reversed__()` – return a reverse iterator over the list

`__rmul__`
Return `self*value`.

`__setattr__`
Implement `setattr(self, name, value)`.

`__setitem__`
Set `self[key]` to value.

`__sizeof__` ()
`L.__sizeof__()` – size of L in memory, in bytes

`__str__`
Return `str(self)`.

`bbox` (*filter=None*)

Parameters **filter** – optional function(entity):bool returning True if entity should be considered in box

Returns `BBBox` bounding box of Entity

`center`

`chainify` (*mergeable*)

merge all possible entities into chains

`clear` () → None – remove all items from L

`color`

`connect` (*other*)

copy () → list – a shallow copy of L

count (value) → integer – return number of occurrences of value

distance (other)

draw (fig=None, **kwargs)

draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

extend (entities, **kwargs)

figure (box, **kwargs)

Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

html (**kwargs)

index (value[, start[, stop]]) → integer – return first index of value.

Raises ValueError if the value is not present.

insert ()

L.insert(index, object) – insert object before index

intersect (other)

Parameters other – *geom.Entity*

Result generate tuples (Point2,Entity_self) of intersections between other and each Entity

isclosed ()

ishorizontal (tol=0.01)

isline ()

isvertical (tol=0.01)

layer

length

patches (**kwargs)

Returns list of `Patch` corresponding to group

plot (**kwargs)

renders on IPython Notebook (alias to make usage more straightforward)

png (**kwargs)

point (u)

Returns Point2 or Point3 at parameter u

pop ([index]) → item – remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range.

remove (value) → None – remove first occurrence of value.

Raises ValueError if the value is not present.

render (fmt, **kwargs)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

reverse ()
L.reverse() – reverse *IN PLACE*

save (filename, **kwargs)

setattr (**kwargs)
set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

sort (key=None, reverse=False) → None – stable sort **IN PLACE**

svg (**kwargs)

swap ()
swap start and end

tangent (u)
Returns Vector2 or Vector3 tangent at parameter u

Goulib.drawing.**chains** (group, tol=1e-06, mergeable=None)
build chains from all possible segments in group :param mergeable: function(e1,e2) returning True if entities e1,e2 can be merged

class Goulib.drawing.**Rect** (*args)
Bases: *Goulib.drawing.Chain*
a rectangle starting at low/left and going trigowise through top/right

__init__ (*args)

p1

p2

__repr__ ()

__abstractmethods__ = frozenset()

__add__
Return self+value.

__class__
alias of ABCMeta

__contains__
Return key in self.

__copy__ ()

__delattr__
Implement delattr(self, name).

__delitem__
Delete self[key].

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

`__getattr__`
Return `getattr(self, name)`.

`__getitem__` (`()`)
`x.__getitem__(y) <==> x[y]`

`__gt__`
Return `self > value`.

`__hash__` = `None`

`__iadd__`
Implement `self += value`.

`__imul__`
Implement `self *= value`.

`__iter__`
Implement `iter(self)`.

`__le__`
Return `self <= value`.

`__len__`
Return `len(self)`.

`__lt__`
Return `self < value`.

`__mul__`
Return `self * value.n`

`__ne__`
Return `self != value`.

`__new__` (`()`)
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` (`()`)
helper for pickle

`__reduce_ex__` (`()`)
helper for pickle

`__reversed__` (`()`)
`L.__reversed__()` – return a reverse iterator over the list

`__rmul__`
Return `self * value`.

`__setattr__`
Implement `setattr(self, name, value)`.

`__setitem__`
Set `self[key]` to value.

`__sizeof__` (`()`)
`L.__sizeof__()` – size of L in memory, in bytes

`__str__`
Return `str(self)`.

`append` (`entity, tol=1e-06, allow_swap=True, mergeable=None, **attrs`)
append `entity` to chain, ensuring contiguity :param `entity`: *Entity* to append :param `tol`: float tolerance

on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :param mergeable: function of the form f(e1,e2) returning True if entities e1,e2 can be merged :param attrs: attributes passed to Group.append :return: self, or None if edge is not contiguous

bbox (*filter=None*)

Parameters **filter** – optional function(entity):bool returning True if entity should be considered in box

Returns *BBox* bounding box of Entity

center

chainify (*mergeable*)

merge all possible entities into chains

clear () → None – remove all items from L

color

connect (*other*)

contiguous (*edge, tol=1e-06, allow_swap=True*)

check if edge can be appended to the chain :param edge: *Entity* to append :param tol: float tolerance on contiguity :param allow_swap: if True (default), tries to swap edge or self to find contiguity :return: int,bool index where to append in chain, swap of edge required

copy () → list – a shallow copy of L

count (*value*) → integer – return number of occurrences of value

distance (*other*)

draw (*fig=None, **kwargs*)

draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

extend (*entities, **kwargs*)

figure (*box, **kwargs*)

Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (*e, mat3*)

Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

See http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf p. 132

from_svg (*path*, *color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

insert ()

L.insert(index, object) – insert object before index

intersect (*other*)

Parameters **other** – *geom.Entity*

Result generate tuples (Point2,Entity_self) of intersections between other and each Entity

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

layer

length

patches (***kwargs*)

Returns list of `Patch` corresponding to group

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

pop ([*index*]) → item – remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.
Raises ValueError if the value is not present.

render (*fmt*, ***kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

reverse ()

L.reverse() – reverse *IN PLACE*

save (*filename*, ***kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

sort (*key=None*, *reverse=False*) → None – stable sort **IN PLACE**

start

svg (***kwargs*)

swap ()
swap start and end

tangent (*u*)
Returns Vector2 or Vector3 tangent at parameter *u*

to_dxf (*split=False, **attr*)

Parameters

- **split** – bool if True, each segment in Chain is saved separately
- **attr** – dict of graphic attributes

Returns polyline or list of entities along the chain

class Goulib.drawing.**Text** (*text, point, size=12, rotation=0*)
Bases: *Goulib.drawing.Entity*

Parameters

- **text** – string
- **point** – Point2
- **size** – size in points
- **rotation** – float angle in degrees trigowise

__init__ (*text, point, size=12, rotation=0*)

Parameters

- **text** – string
- **point** – Point2
- **size** – size in points
- **rotation** – float angle in degrees trigowise

bbox ()

length

Returns float length of the text contour in mm

intersect (*other*)

to_dxf (***attr*)

patches (***kwargs*)

Returns list of (a single) *Patch* corresponding to entity

__class__
alias of *type*

__delattr__
Implement *delattr*(self, name).

__dir__ () → list
default *dir*() implementation

__eq__
Return *self==value*.

```

__format__ ()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__ ()

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

center
color = 'black'
draw (fig=None, **kwargs)
    draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch
end
figure (box, **kwargs)
    Parameters
        • box – drawing.BBox bounds and clipping box
        • kwargs – parameters passed to ~matplotlib.pyplot.figure
    Returns matplotlib axis suitable for drawing
from_dxf (e, mat3)
    Parameters

```

- `e` – `dx.f.entity`
- `mat3` – Matrix3 transform

Returns Entity of correct subtype

`from_pdf` (*path, trans, color*)

Parameters `path` – pdf path

Returns Entity of correct subtype

`from_svg` (*path, color*)

Parameters `path` – svg path

Returns Entity of correct subtype

`html` (***kwargs*)

`isclosed` ()

`ishorizontal` (*tol=0.01*)

`isline` ()

`isvertical` (*tol=0.01*)

`plot` (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

`png` (***kwargs*)

`render` (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

`save` (*filename, **kwargs*)

`setattr` (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

`start`

`svg` (***kwargs*)

class `Goulib.drawing.Drawing` (*data=[], **kwargs*)

Bases: `Goulib.drawing.Group`

list of Entities representing a vector graphics drawing

`__abstractmethods__` = frozenset()

`__add__`

Return self+value.

`__class__`

alias of ABCMeta

`__contains__`

Return key in self.

`__copy__` ()

`__delattr__`

Implement delattr(self, name).

`__delitem__`

Delete self[key].

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__getitem__ ()
x.__getitem__(y) <==> x[y]

__gt__
Return self>value.

__hash__ = None

__iadd__
Implement self+=value.

__imul__
Implement self*=value.

__init__ (data=[], **kwargs)

__iter__
Implement iter(self).

__le__
Return self<=value.

__len__
Return len(self).

__lt__
Return self<value.

__mul__
Return self*value.n

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__reversed__ ()
L.__reversed__() – return a reverse iterator over the list

`__rmul__`

Return self*value.

`__setattr__`

Implement setattr(self, name, value).

`__setitem__`

Set self[key] to value.

`__sizeof__` ()

L.__sizeof__() – size of L in memory, in bytes

`__str__`

Return str(self).

append (*entity*, ***kwargs*)

append entity to group :param entity: Entity :param kwargs: dict of attributes copied to entity :return: Group (or Chain) to which the entity was added, or None if entity was None

bbox (*filter=None*)

Parameters **filter** – optional function(entity):bool returning True if entity should be considered in box

Returns *BBBox* bounding box of Entity

center

chainify (*mergeable*)

merge all possible entities into chains

clear () → None – remove all items from L

color

connect (*other*)

copy () → list – a shallow copy of L

count (*value*) → integer – return number of occurrences of value

distance (*other*)

draw (*fig=None*, ***kwargs*)

draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

extend (*entities*, ***kwargs*)

figure (*box*, ***kwargs*)

Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (*dxf*, *layers=None*, *only=[]*, *ignore=['POINT']*, *trans=Matrix3(1.0, 0, 0, 0, 1.0, 0, 0, 0, 1.0)*, *flatten=False*)

Parameters

- **dxf** – dxf.entity

- **layers** – list of layer names to consider. entities not on these layers are ignored. default=None: all layers are read
- **only** – list of dxf entity types names that are read. default=[]: all are read
- **ignore** – list of dxf entity types names that are ignored. default=['POINT']: points and null length segments are ignored
- **trans** – *Trans* optional transform matrix

Parm flatten bool flatten block structure

Returns *Entity* of correct subtype

from_pdf (*path*, *trans*, *color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path*, *color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

insert ()

L.insert(index, object) – insert object before index

intersect (*other*)

Parameters **other** – *geom.Entity*

Result generate tuples (Point2,Entity_self) of intersections between other and each Entity

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

layer

length

patches (***kwargs*)

Returns list of *Patch* corresponding to group

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter u

pop ([*index*]) → item – remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.
Raises ValueError if the value is not present.

render (*fmt*, ***kwargs*)
render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

reverse ()
L.reverse() – reverse *IN PLACE*

setattr (***kwargs*)
set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

sort (*key=None*, *reverse=False*) → None – stable sort **IN PLACE**

start

svg (***kwargs*)

swap ()
swap start and end

tangent (*u*)
Returns Vector2 or Vector3 tangent at parameter u

to_dxf (***kwargs*)
Returns flatten list of dxf entities

load (*filename*, ***kwargs*)

read_pdf (*filename*, ***kwargs*)
reads a vector graphics on a .pdf file only the first page is parsed

read_svg (*content*, ***kwargs*)
appends svg content to drawing :param content: string, either filename or svg content

read_dxf (*filename*, *options=None*, ***kwargs*)
reads a .dxf file :param filename: string path to .dxf file to read :param options: passed to `from_dxf`

save (*filename*, ***kwargs*)
save graph in various formats

Goulib.expr module

simple symbolic math expressions

`Goulib.expr.eval` (*node*, *ctx={}*)
safe eval of ast node : only functions and operators listed above can be used

Parameters

- **node** – ast.AST to evaluate
- **ctx** – dict of varname : value to substitute in node

Returns number or expression string

`Goulib.expr.get_function_source` (*f*)
returns cleaned code of a function or lambda currently only supports: - lambda x:formula_of_(x) - def anything(x): return formula_of_(x)

class `Goulib.expr.Expr` (*f*)

Bases: `Goulib.plot.Plot`

Math expressions that can be evaluated like standard functions combined using standard operators and plotted in IPython/Jupyter notebooks

Parameters *f* – function or operator, Expr to copy construct, or formula string

`__init__` (*f*)

Parameters *f* – function or operator, Expr to copy construct, or formula string

`__call__` (*x=None, **kwargs*)

evaluate the Expr at *x* OR compose `self(x)`

`__repr__` ()

`__str__` ()

`latex` ()

Returns string LaTeX formula

`apply` (*f, right=None*)

function composition `self o f = f(self(x))`

`applx` (*f, var='x'*)

function composition `f o self = self(f(x))`

`isconstant`

Returns True if Expr evaluates to a constant number of bool

`__eq__` (*other*)

`__lt__` (*other*)

`__add__` (*right*)

`__sub__` (*right*)

`__neg__` ()

`__mul__` (*right*)

`__rmul__` (*right*)

`__truediv__` (*right*)

`__pow__` (*right*)

`__div__` (*right*)

`__invert__` ()

`__and__` (*right*)

`__or__` (*right*)

`__xor__` (*right*)

`__lshift__` (*dx*)

`__rshift__` (*dx*)

`__class__`

alias of `type`

`__delattr__`
Implement `delattr(self, name)`.

`__dir__` () → list
default `dir()` implementation

`__format__` ()
default object formatter

`__ge__`
Return `self >= value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self > value`.

`__hash__` = None

`__le__`
Return `self <= value`.

`__ne__`
Return `self != value`.

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__setattr__`
Implement `setattr(self, name, value)`.

`__sizeof__` () → int
size of object in memory, in bytes

`html` (**kwargs)

`plot` (**kwargs)
renders on IPython Notebook (alias to make usage more straightforward)

`png` (**kwargs)

`render` (fmt='svg', **kwargs)

`save` (filename, **kwargs)

`svg` (**kwargs)

class `Goulib.expr.TextVisitor` (*dialect*)
Bases: `ast.NodeVisitor`

Parameters `dialect` – int index in operators of symbols to use

`__init__` (*dialect*)

Parameters `dialect` – int index in operators of symbols to use

`prec` (*n*)

`prec_UnaryOp` (*n*)

```

prec_BinOp (n)
visit_Call (n)
visit_Name (n)
visit_NameConstant (node)
visit_UnaryOp (n)
visit_BinOp (n)
__class__
    alias of type
__delattr__
    Implement delattr(self, name).
__dir__ () → list
    default dir() implementation
__eq__
    Return self==value.
__format__ ()
    default object formatter
__ge__
    Return self>=value.
__getattr__
    Return getattr(self, name).
__gt__
    Return self>value.
__hash__
    Return hash(self).
__le__
    Return self<=value.
__lt__
    Return self<value.
__ne__
    Return self!=value.
__new__ ()
    Create and return a new object. See help(type) for accurate signature.
__reduce__ ()
    helper for pickle
__reduce_ex__ ()
    helper for pickle
__repr__
    Return repr(self).
__setattr__
    Implement setattr(self, name, value).
__sizeof__ () → int
    size of object in memory, in bytes

```

__str__
Return str(self).

visit (*node*)
Visit a node.

visit_Compare (*n*)

visit_Num (*n*)

generic_visit (*n*)

class Goulib.expr.LatexVisitor
Bases: *Goulib.expr.TextVisitor*

__class__
alias of *type*

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

```

__sizeof__() → int
    size of object in memory, in bytes

__str__
    Return str(self).

generic_visit (n)

prec (n)

prec_BinOp (n)

prec_UnaryOp (n)

visit (node)
    Visit a node.

visit_BinOp (n)

visit_Compare (n)

visit_Name (n)

visit_NameConstant (node)

visit_Num (n)

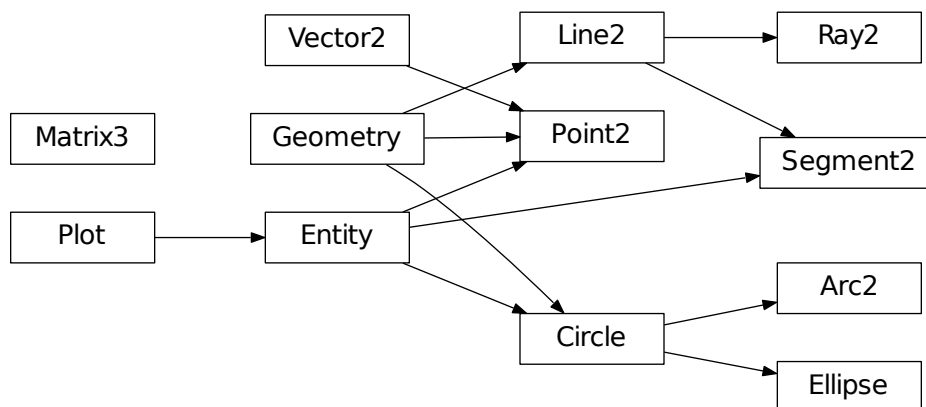
__init__()

visit_Call (n)

visit_UnaryOp (n)

```

Goulib.geom module



ometry

```
class Goulib.geom.Geometry(*args)
```

2D ge-

Bases: `object`

The following classes are available for dealing with simple 2D geometry. The interface to each shape is similar; in particular, the `connect` and `distance` methods are defined identically for each.

For example, to find the closest point on a line to a circle:

```
>>> circ = Circle(Point2(3., 2.), 2.)
>>> line = Line2(Point2(0., 0.), Point2(-1., 1.))
>>> line.connect(circ).p1
Point2(0.50, -0.50)
```

To find the corresponding closest point on the circle to the line:

```
>>> line.connect(circ).p2
Point2(1.59, 0.59)
```

this constructor is called by descendant classes at copy it is replaced to copy some graphics attributes in module drawings

`__init__` (*args)

this constructor is called by descendant classes at copy it is replaced to copy some graphics attributes in module drawings

`point` (u)

Returns Point2 or Point3 at parameter u

`tangent` (u)

Returns Vector2 or Vector3 tangents at parameter u

`intersect` (other)

`connect` (other)

Returns Geometry shortest (Segment2 or Segment3) that connects self to other

`distance` (other)

`__contains__` (pt)

`__abstractmethods__` = frozenset()

`__class__`

alias of ABCMeta

`__delattr__`

Implement `delattr(self, name)`.

`__dir__` () → list

default `dir()` implementation

`__eq__`

Return `self==value`.

`__format__` ()

default object formatter

`__ge__`

Return `self>=value`.

`__getattr__`

Return `getattr(self, name)`.

`__gt__`
Return self>value.

`__hash__`
Return hash(self).

`__le__`
Return self<=value.

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return repr(self).

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__` () → int
size of object in memory, in bytes

`__str__`
Return str(self).

Goulib.geom.**argPair** (*x*, *y=None*)
Process a pair of values passed in various ways.

class Goulib.geom.**Vector2** (*args)
Bases: `object`

Mutable 2D vector:

Construct a vector in the obvious way:

```
>>> Vector2(1.5, 2.0)
Vector2(1.50, 2.00)

>>> Vector3(1.0, 2.0, 3.0)
Vector3(1.00, 2.00, 3.00)
```

Element access

Components may be accessed as attributes (examples that follow use *Vector3*, but all results are similar for *Vector2*, using only the *x* and *y* components):

```
>>> v = Vector3(1, 2, 3)
>>> v.x
1
>>> v.y
2
```

```
>>> v.z
3
```

Vectors support the list interface via slicing:

```
>>> v = Vector3(1, 2, 3)
>>> len(v)
3
>>> v[0]
1
>>> v[:]
(1, 2, 3)
```

You can also “swizzle” the components (*a la* GLSL or Cg):

```
>>> v = Vector3(1, 2, 3)
>>> v.xyz
(1, 2, 3)
>>> v.zx
(3, 1)
>>> v.zzzz
(3, 3, 3, 3)
```

Operators

Addition and subtraction are supported via operator overloading (note that in-place operators perform faster than those that create a new object):

```
>>> v1 = Vector3(1, 2, 3)
>>> v2 = Vector3(4, 5, 6)
>>> v1 + v2
Vector3(5.00, 7.00, 9.00)
>>> v1 -= v2
>>> v1
Vector3(-3.00, -3.00, -3.00)
```

Multiplication and division can be performed with a scalar only:

```
>>> Vector3(1, 2, 3) * 2
Vector3(2.00, 4.00, 6.00)
>>> v1 = Vector3(1., 2., 3.)
>>> v1 /= 2
>>> v1
Vector3(0.50, 1.00, 1.50)
```

The magnitude of a vector can be found with `abs`:

```
>>> v = Vector3(1., 2., 3.)
>>> abs(v)
3.7416573867739413
```

A vector can be normalized in-place (note that the in-place method also returns `self`, so you can chain it with further operators):

```
>>> v = Vector3(1., 2., 3.)
>>> v.normalize()
Vector3(0.27, 0.53, 0.80)
```

```
>>> v
Vector3(0.27, 0.53, 0.80)
```

The following methods do *not* alter the original vector or their arguments:

magnitude() Returns the magnitude of the vector; equivalent to `abs(v)`. Example:

```
>>> v = Vector3(1., 2., 3.)
>>> v.magnitude()
3.7416573867739413
```

magnitude_squared() Returns the sum of the squares of each component. Useful for comparing the length of two vectors without the expensive square root operation. Example:

```
>>> v = Vector3(1., 2., 3.)
>>> v.magnitude_squared()
14.0
```

normalized() Return a unit length vector in the same direction. Note that this method differs from `normalize` in that it does not modify the vector in-place. Example:

```
>>> v = Vector3(1., 2., 3.)
>>> v.normalized()
Vector3(0.27, 0.53, 0.80)
>>> v
Vector3(1.00, 2.00, 3.00)
```

dot(other) Return the scalar “dot” product of two vectors. Example:

```
>>> v1 = Vector3(1., 2., 3.)
>>> v2 = Vector3(4., 5., 6.)
>>> v1.dot(v2)
32.0
```

cross() and **cross(other)** Return the cross product of a vector (for **Vector2**), or the cross product of two vectors (for **Vector3**). The return type is a vector. Example:

```
>>> v1 = Vector3(1., 2., 3.)
>>> v2 = Vector3(4., 5., 6.)
>>> v1.cross(v2)
Vector3(-3.00, 6.00, -3.00)
```

In two dimensions there can be no argument to `cross`:

```
>>> v1 = Vector2(1., 2.)
>>> v1.cross()
Vector2(2.00, -1.00)
```

reflect(normal) Return the vector reflected about the given normal. In two dimensions, *normal* is the normal to a line, in three dimensions it is the normal to a plane. The normal must have unit length. Example:

```
>>> v = Vector3(1., 2., 3.)
>>> v.reflect(Vector3(0, 1, 0))
Vector3(1.00, -2.00, 3.00)
>>> v = Vector2(1., 2.)
>>> v.reflect(Vector2(1, 0))
Vector2(-1.00, 2.00)
```

rotate_around(axes, theta) For 3D vectors, return the vector rotated around axis by the angle theta.

```
>>> v = Vector3(1., 2., 3.)
>>> axes = Vector3(1.,1.,0)
>>> v.rotate_around(axes,math.pi/4)
Vector3(2.65, 0.35, 2.62)
```

Constructor. :param *args: x,y values

`__init__` (*args)

Constructor. :param *args: x,y values

xy

Returns tuple (x,y)

`__repr__` ()

`__hash__` ()

`__eq__` (other)

Tests for equality include comparing against other sequences:

```
>>> v2 = Vector2(1, 2)
>>> v2 == Vector2(3, 4)
```

False >>> v2 != Vector2(1, 2) False >>> v2 == (1, 2) True

```
>>> v3 = Vector3(1, 2, 3)
>>> v3 == Vector3(3, 4, 5)
False
>>> v3 != Vector3(1, 2, 3)
False
>>> v3 == (1, 2, 3)
True
```

`__len__` ()

`__iter__` ()

`__add__` (other)

`__radd__` (other)

`__iadd__` (other)

`__sub__` (other)

`__rsub__` (other)

Point2 - Vector 2 subtraction :param other: Point2 or (x,y) tuple :return: Vector2

`__mul__` (other)

`__rmul__` (other)

`__imul__` (other)

`__div__` (other)

`__rdiv__` (other)

`__floordiv__` (other)

`__rfloordiv__` (other)

`__truediv__` (other)

```

__rtruediv__ (other)
__neg__ ()
__pos__ ()
__abs__ ()
mag ()
length
mag2 ()
normalize ()
normalized ()
dot (other)
cross ()
reflect (normal)
angle (other=None, unit=False)
    angle between two vectors. :param unit: bool True if vectors are unit vectors. False increases computations
    :return: float angle in radians to the other vector, or self direction if other=None
project (other)
    Return the projection (the component) of the vector on other.
__class__
    alias of type
__delattr__
    Implement delattr(self, name).
__dir__ () → list
    default dir() implementation
__format__ ()
    default object formatter
__ge__
    Return self >= value.
__getattr__
    Return getattr(self, name).
__gt__
    Return self > value.
__le__
    Return self <= value.
__lt__
    Return self < value.
__ne__
    Return self != value.
__new__ ()
    Create and return a new object. See help(type) for accurate signature.
__reduce__ ()
    helper for pickle

```

`__reduce_ex__()`
helper for pickle

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return str(self).

class Goulib.geom.**Point2**(*args)

Bases: Goulib.geom.Vector2, Goulib.geom.Geometry, Goulib.drawing.Entity

A point on a 2D plane. Construct in the obvious way:

```
>>> p = Point2(1.0, 2.0)
>>> p
```

Point2(1.00, 2.00)

Point2 subclasses **Vector2**, so all of **Vector2** operators and methods apply. In particular, subtracting two points gives a vector:

```
>>> Point2(2.0, 3.0) - Point2(1.0, 0.0)
```

Vector2(1.00, 3.00)

connect(other) Returns a **Segment2** which is the minimum length line segment that can connect the two shapes. other may be a **Point2**, **Line2**, **Ray2**, **Segment2** or **Circle**.

Constructor. :param *args: x,y values

distance(other)
absolute minimum distance to other object :param other: Point2, Line2 or Circle :return: float positive distance between self and other

`__contains__`(pt)
Returns True if self and pt are the same point, False otherwise
needed for coherency

intersect(other)
Point2/object intersection :return: Point2 copy of self if on other object, None if not

connect(other)

`__abs__()`

`__abstractmethods__` = frozenset()

`__add__`(other)

`__class__`
alias of ABCMeta

`__delattr__`
Implement delattr(self, name).

`__dir__`() → list
default dir() implementation

`__div__`(other)

__eq__ (*other*)

Tests for equality include comparing against other sequences:

```

>>> v2 = Vector2(1, 2)
>>> v2 == Vector2(3, 4)

```

False >>> v2 != Vector2(1, 2) False >>> v2 == (1, 2) True

```

>>> v3 = Vector3(1, 2, 3)
>>> v3 == Vector3(3, 4, 5)
False
>>> v3 != Vector3(1, 2, 3)
False
>>> v3 == (1, 2, 3)
True

```

__floordiv__ (*other*)

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__gt__

Return self>value.

__hash__ ()

__iadd__ (*other*)

__imul__ (*other*)

__init__ (**args*)

Constructor. :param **args*: x,y values

__iter__ ()

__le__

Return self<=value.

__len__ ()

__lt__

Return self<value.

__mul__ (*other*)

__ne__

Return self!=value.

__neg__ ()

__new__ ()

Create and return a new object. See help(type) for accurate signature.

__pos__ ()

__radd__ (*other*)

__rdiv__ (*other*)

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__repr__()`

`__rfloordiv__(other)`

`__rmul__(other)`

`__rsub__(other)`
Point2 - Vector 2 subtraction :param other: Point2 or (x,y) tuple :return: Vector2

`__rtruediv__(other)`

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return str(self).

`__sub__(other)`

`__truediv__(other)`

`angle (other=None, unit=False)`
angle between two vectors. :param unit: bool True if vectors are unit vectors. False increases computations
:return: float angle in radians to the other vector, or self direction if other=None

`bbox()`
Returns `BBBox` bounding box of Entity

`center`

`color = 'black'`

`cross()`

`dot (other)`

`draw (fig=None, **kwargs)`
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

`end`

`figure (box, **kwargs)`
Parameters

- `box` – `drawing.BBox` bounds and clipping box
- `kwargs` – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

`from_dxf (e, mat3)`
Parameters

- `e` – `dxf.entity`
- `mat3` – `Matrix3` transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

length

mag ()

mag2 ()

normalize ()

normalized ()

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 or Point3 at parameter *u*

project (*other*)

Return the projection (the component) of the vector on *other*.

reflect (*normal*)

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

tangent (*u*)

Returns Vector2 or Vector3 tangent at parameter *u*

`to_dxf (**attr)`

Parameters `attr` – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

xy

Returns tuple (x,y)

`Goulib.geom.Polar (mag, angle)`

class `Goulib.geom.Line2 (*args)`

Bases: `Goulib.geom.Geometry`

A **Line2** is a line on a 2D plane extending to infinity in both directions; a **Ray2** has a finite end-point and extends to infinity in a single direction; a **Segment2** joins two points.

All three classes support the same constructors, operators and methods, but may behave differently when calculating intersections etc.

You may construct a line, ray or line segment using any of:

- another line, ray or line segment
- two points
- a point and a vector
- a point, a vector and a length

For example:

```
>>> Line2(Point2(1.0, 1.0), Point2(2.0, 3.0))
Line2(<1.00, 1.00> + u<1.00, 2.00>)
>>> Line2(Point2(1.0, 1.0), Vector2(1.0, 2.0))
Line2(<1.00, 1.00> + u<1.00, 2.00>)
>>> Ray2(Point2(1.0, 1.0), Vector2(1.0, 2.0), 1.0)
Ray2(<1.00, 1.00> + u<0.45, 0.89>)
```

Internally, lines, rays and line segments store a `Point2 p` and a `Vector2 v`. You can also access (but not set) the two endpoints `p1` and `p2`. These may or may not be meaningful for all types of lines.

The following methods are supported by all three classes:

intersect (other) If `other` is a **Line2**, **Ray2** or **Segment2**, returns a **Point2** of intersection, or `None` if the lines are parallel.

If `other` is a **Circle**, returns a **Segment2** or **Point2** giving the part of the line that intersects the circle, or `None` if there is no intersection.

connect (other) Returns a **Segment2** which is the minimum length line segment that can connect the two shapes. For two parallel lines, this line segment may be in an arbitrary position. `other` may be a **Point2**, **Line2**, **Ray2**, **Segment2** or **Circle**.

distance (other) Returns the absolute minimum distance to `other`. Internally this simply returns the length of the result of `connect`.

Segment2 also has a `length` property which is read-only.

`__init__ (*args)`

`__eq__ (other)`

lines are “equal” only if base points and vector are strictly equal. to compare if lines are “same”, use `line1.distance(line2)==0`

`__repr__()`

`point(u)`
Returns Point2 at parameter u

`tangent(u)`
Returns Vector2 tangent at parameter u. Warning : tangent is generally not a unit vector

`intersect(other)`

`connect(other)`

`__abstractmethods__ = frozenset()`

`__class__`
 alias of ABCMeta

`__contains__(pt)`

`__delattr__`
 Implement delattr(self, name).

`__dir__()` → list
 default dir() implementation

`__format__()`
 default object formatter

`__ge__`
 Return self>=value.

`__getattr__`
 Return getattr(self, name).

`__gt__`
 Return self>value.

`__hash__ = None`

`__le__`
 Return self<=value.

`__lt__`
 Return self<value.

`__ne__`
 Return self!=value.

`__new__()`
 Create and return a new object. See help(type) for accurate signature.

`__reduce__()`
 helper for pickle

`__reduce_ex__()`
 helper for pickle

`__setattr__`
 Implement setattr(self, name, value).

`__sizeof__()` → int
 size of object in memory, in bytes

__str__

Return str(self).

distance (*other*)

class Goulib.geom.**Ray2** (*args)

Bases: *Goulib.geom.Line2*

__abstractmethods__ = frozenset()

__class__

alias of ABCMeta

__contains__ (*pt*)

__delattr__

Implement delattr(self, name).

__dir__ () → list

default dir() implementation

__eq__ (*other*)

lines are “equal” only if base points and vector are strictly equal. to compare if lines are “same”, use line1.distance(line2)==0

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

__gt__

Return self>value.

__hash__ = None

__init__ (*args)

__le__

Return self<=value.

__lt__

Return self<value.

__ne__

Return self!=value.

__new__ ()

Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

helper for pickle

__reduce_ex__ ()

helper for pickle

__repr__ ()

__setattr__

Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return `str(self)`.

`connect` (*other*)

`distance` (*other*)

`intersect` (*other*)

`point` (*u*)

Returns Point2 at parameter *u*

`tangent` (*u*)

Returns Vector2 tangent at parameter *u*. Warning : tangent is generally not a unit vector

class `Goulib.geom.Segment2` (**args*)

Bases: `Goulib.geom.Line2`, `Goulib.drawing.Entity`

`p1`

`p2`

`__repr__()`

`__abs__()`

`mag2()`

`length`

`swap()`

`midpoint()`

`bisect()`

`__abstractmethods__` = frozenset()

`__class__`
alias of `ABCMeta`

`__contains__` (*pt*)

`__delattr__`
Implement `delattr(self, name)`.

`__dir__()` → list
default `dir()` implementation

`__eq__` (*other*)
lines are “equal” only if base points and vector are strictly equal. to compare if lines are “same”, use `line1.distance(line2)==0`

`__format__()`
default object formatter

`__ge__`
Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

__gt__
Return self>value.

__hash__ = None

__init__ (*args)

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

bbox ()
Returns BBox bounding box of Entity

center

color = 'black'

connect (*other*)

distance (*other*)

draw (*fig=None, **kwargs*)
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig.patch

end

figure (*box, **kwargs*)
Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (*e, mat3*)
Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

intersect (*other*)

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and IPython `_repr_XXX_` to work

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns `Point2` at parameter `u`

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

tangent (*u*)

Returns `Vector2` tangent at parameter `u`. Warning : tangent is generally not a unit vector

to_dxf (***attr*)

Parameters **attr** – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

class `Goulib.geom.Circle` (**args*)

Bases: `Goulib.geom.Geometry`, `Goulib.drawing.Entity`

Circles are constructed with a center **Point2** and a radius:

```
>>> c = Circle(Point2(1.0, 1.0), 0.5)
>>> c
```

Circle(<1.00, 1.00>, radius=0.50)

Internally there are two attributes: *c*, giving the center point and *r*, giving the radius.

The following methods are supported:

connect (other) Returns a **Segment2** which is the minimum length line segment that can connect the two shapes. *other* may be a **Point2**, **Line2**, **Ray2**, **Segment2** or **Circle**.

distance (other) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of `connect`.

Parameters args – can be

- Circle
- center, point on circle
- center, radius

`__init__` (*args)

Parameters args – can be

- Circle
- center, point on circle
- center, radius

`__eq__` (other)

`__repr__` ()

`__abs__` ()

Returns float perimeter

length

point (u)

Returns Point2 at angle u radians

tangent (u)

Returns Vector2 tangent at angle u. Warning : tangent has magnitude $r \neq 1$

`__contains__` (pt)

Returns True if pt is ON or IN the circle

intersect (other)

Parameters other – Line2, Ray2 or Segment2**, **Ray2** or **Segment2**, returns

a **Segment2** giving the part of the line that intersects the circle, or None if there is no intersection.

connect (other)

swap ()

`__abstractmethods__` = frozenset()

```

__class__
    alias of ABCMeta

__delattr__
    Implement delattr(self, name).

__dir__ () → list
    default dir() implementation

__format__ ()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__ = None

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

bbox ()
    Returns BBOX bounding box of Entity

center

color = 'black'

distance (other)

draw (fig=None, **kwargs)
    draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig.patch

end

figure (box, **kwargs)

```

Parameters

- **box** – `drawing.BBox` bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (*e, mat3*)

Parameters

- **e** – `dxf.entity`
- **mat3** – `Matrix3` transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and `IPython_repr_XXX_` to work

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

to_dxf (***attr*)

Parameters **attr** – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

`Goulib.geom.circle_from_3_points(a, b, c)`
 constructs Circle passing through 3 distinct points :param a,b,c: Point2 :return: the unique Circle through the three points a, b, c

`Goulib.geom.arc_from_3_points(a, b, c)`
 constructs Arc2 starting in a, going through b and ending in c :param a,b,c: Point2 :return: the unique Arc2 starting in a, going through b and ending in c

class `Goulib.geom.Arc2` (*center, p1=0, p2=6.283185307179586, r=None, dir=1*)
 Bases: `Goulib.geom.Circle`

Parameters

- **center** – Point2 or (x,y) tuple
- **p1** – starting Point2 or angle in radians
- **p2** – ending Point2 or angle in radians
- **r** – float radius, needed only if p1 or p2 is an angle
- **dir** – arc direction. +1 is trig positive (CCW) and -1 is Clockwise

`__init__` (*center, p1=0, p2=6.283185307179586, r=None, dir=1*)

Parameters

- **center** – Point2 or (x,y) tuple
- **p1** – starting Point2 or angle in radians
- **p2** – ending Point2 or angle in radians
- **r** – float radius, needed only if p1 or p2 is an angle
- **dir** – arc direction. +1 is trig positive (CCW) and -1 is Clockwise

angle (*b=None*)

Returns float signed arc angle

`__abs__` ()

Returns float arc length

point (*u*)

Returns Point2 at parameter u

tangent (*u*)

Returns Vector2 tangent at parameter u

`__eq__` (*other*)

`__repr__` ()

swap ()

`__contains__` (*pt*)

Returns True if pt is ON the Arc

intersect (*other*)

`__abstractmethods__` = frozenset()

`__class__`

alias of ABCMeta

`__delattr__`
Implement `delattr(self, name)`.

`__dir__` () → list
default `dir()` implementation

`__format__` ()
default object formatter

`__ge__`
Return `self >= value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self > value`.

`__hash__` = None

`__le__`
Return `self <= value`.

`__lt__`
Return `self < value`.

`__ne__`
Return `self != value`.

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__setattr__`
Implement `setattr(self, name, value)`.

`__sizeof__` () → int
size of object in memory, in bytes

`__str__`
Return `str(self)`.

`bbox` ()
Returns `BBBox` bounding box of Entity

`center`

`color` = 'black'

`connect` (*other*)

`distance` (*other*)

`draw` (*fig=None, **kwargs*)
draw entities :param *fig*: matplotlib figure where to draw. `figure(g)` is called if missing :return: `fig.patch`

`end`

`figure` (*box, **kwargs*)

Parameters

- **box** – `drawing.BBox` bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (*e, mat3*)

Parameters

- **e** – `dxf.entity`
- **mat3** – `Matrix3` transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)

Parameters **path** – pdf path

Returns Entity of correct subtype

from_svg (*path, color*)

Parameters **path** – svg path

Returns Entity of correct subtype

html (***kwargs*)

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

length

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

Note this is the only method that needs to be overridden in descendants for `draw`, `render` and `IPython_repr_XXX_` to work

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param kwargs: dict of attributes copied to entity

start

svg (***kwargs*)

to_dxf (***attr*)

Parameters **attr** – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

class `Goulib.geom.Ellipse` (*args)

Bases: `Goulib.geom.Circle`

Parameters `args` – can be

- Ellipse
- center, corner point
- center, r1,r2,angle

`__init__` (*args)

Parameters `args` – can be

- Ellipse
- center, corner point
- center, r1,r2,angle

`__repr__` ()

`__eq__` (other)

`__abs__` ()

Returns float perimeter

`__abstractmethods__` = frozenset()

`__class__`

alias of ABCMeta

`__contains__` (pt)

Returns True if pt is ON or IN the circle

`__delattr__`

Implement `delattr`(self, name).

`__dir__` () → list

default `dir`() implementation

`__format__` ()

default object formatter

`__ge__`

Return `self >= value`.

`__getattr__`

Return `getattr`(self, name).

`__gt__`

Return `self > value`.

`__hash__` = None

`__le__`

Return `self <= value`.

`__lt__`

Return `self < value`.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

bbox ()
Returns BBox bounding box of Entity

center

color = 'black'

connect (*other*)

distance (*other*)

draw (*fig=None, **kwargs*)
draw entities :param fig: matplotlib figure where to draw. figure(g) is called if missing :return: fig,patch

end

figure (*box, **kwargs*)
Parameters

- **box** – drawing.BBox bounds and clipping box
- **kwargs** – parameters passed to `~matplotlib.pyplot.figure`

Returns matplotlib axis suitable for drawing

from_dxf (*e, mat3*)
Parameters

- **e** – dxf.entity
- **mat3** – Matrix3 transform

Returns Entity of correct subtype

from_pdf (*path, trans, color*)
Parameters **path** – pdf path
Returns Entity of correct subtype

from_svg (*path, color*)
Parameters **path** – svg path
Returns Entity of correct subtype

html (***kwargs*)

intersect (*other*)

Parameters *other* – Line2, Ray2 or Segment2**, **Ray2** or **Segment2**, returns a **Segment2** giving the part of the line that intersects the circle, or None if there is no intersection.

isclosed ()

ishorizontal (*tol=0.01*)

isline ()

isvertical (*tol=0.01*)

length

patches (***kwargs*)

Returns list of (a single) `Patch` corresponding to entity

Note this is the only method that needs to be overridden in descendants for draw, render and `IPython_repr_XXX_` to work

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

point (*u*)

Returns Point2 at angle *u* radians

render (*fmt, **kwargs*)

render graph to bitmap stream :return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

setattr (***kwargs*)

set (graphic) attributes to entity :param *kwargs*: dict of attributes copied to entity

start

svg (***kwargs*)

swap ()

tangent (*u*)

Returns Vector2 tangent at angle *u*. Warning : tangent has magnitude $r \neq 1$

to_dxf (***attr*)

Parameters *attr* – dict of attributes passed to the dxf entity, overriding those defined in self

Returns dxf entity

class Goulib.geom.**Matrix3** (**args*)

Bases: `object`

Two matrix classes are supplied, *Matrix3*, a 3x3 matrix for working with 2D affine transformations, and *Matrix4*, a 4x4 matrix for working with 3D affine transformations.

The default constructor initializes the matrix to the identity:


```

>>> Matrix3()
Matrix3([ 1.00  0.00  0.00
         0.00  1.00  0.00
         0.00  0.00  1.00])
>>> Matrix4()
Matrix4([ 1.00  0.00  0.00  0.00
         0.00  1.00  0.00  0.00
         0.00  0.00  1.00  0.00
         0.00  0.00  0.00  1.00])

```

Element access

Internally each matrix is stored as a set of attributes named a to p. The layout for Matrix3 is:

```

# a b c
# e f g
# i j k

```

and for Matrix4:

```

# a b c d
# e f g h
# i j k l
# m n o p

```

If you wish to set or retrieve a number of elements at once, you can do so with a slice:

```

>>> m = Matrix4()
>>> m[:]
[1.0, 0, 0, 0, 0, 1.0, 0, 0, 0, 0, 0, 1.0, 0, 0, 0, 0, 1.0]
>>> m[12:15] = (5, 5, 5)
>>> m
Matrix4([ 1.00  0.00  0.00  5.00
         0.00  1.00  0.00  5.00
         0.00  0.00  1.00  5.00
         0.00  0.00  0.00  1.00])

```

Note that slices operate in column-major order, which makes them suitable for working directly with OpenGL's `glLoadMatrix` and `glGetFloatv` functions.

Class constructors

There are class constructors for the most common types of transform.

new_identity Equivalent to the default constructor. Example:

```

>>> m = Matrix4.new_identity()
>>> m
Matrix4([ 1.00  0.00  0.00  0.00
         0.00  1.00  0.00  0.00
         0.00  0.00  1.00  0.00
         0.00  0.00  0.00  1.00])

```

new_scale(x, y) and **new_scale(x, y, z)** The former is defined on **Matrix3**, the latter on **Matrix4**. Equivalent to the OpenGL call `glScalef`. Example:

```

>>> m = Matrix4.new_scale(2.0, 3.0, 4.0)
>>> m
Matrix4([ 2.00  0.00  0.00  0.00

```

```

0.00    3.00    0.00    0.00
0.00    0.00    4.00    0.00
0.00    0.00    0.00    1.00])

```

new_translate(x, y) and **new_translate(x, y, z)** The former is defined on **Matrix3**, the latter on **Matrix4**. Equivalent to the OpenGL call `glTranslatef`. Example:

```

>>> m = Matrix4.new_translate(3.0, 4.0, 5.0)
>>> m
Matrix4([
    1.00    0.00    0.00    3.00
    0.00    1.00    0.00    4.00
    0.00    0.00    1.00    5.00
    0.00    0.00    0.00    1.00])

```

new_rotate(angle) Create a **Matrix3** for a rotation around the origin. *angle* is specified in radians, anti-clockwise. This is not implemented in **Matrix4** (see below for equivalent methods). Example:

```

>>> import math
>>> m = Matrix3.new_rotate(math.pi / 2)
>>> m
Matrix3([
    0.00   -1.00    0.00
    1.00    0.00    0.00
    0.00    0.00    1.00])

```

The following constructors are defined for **Matrix4** only.

new_rotatex(angle), **new_rotatey(angle)**, **new_rotatez(angle)** Create a **Matrix4** for a rotation around the X, Y or Z axis, respectively. *angle* is specified in radians. Example:

```

>>> m = Matrix4.new_rotatex(math.pi / 2)
>>> m
Matrix4([
    1.00    0.00    0.00    0.00
    0.00    0.00   -1.00    0.00
    0.00    1.00    0.00    0.00
    0.00    0.00    0.00    1.00])

```

new_rotate_axis(angle, axis) Create a **Matrix4** for a rotation around the given axis. *angle* is specified in radians, and *axis* must be an instance of **Vector3**. It is not necessary to normalize the axis. Example:

```

>>> m = Matrix4.new_rotate_axis(math.pi / 2, Vector3(1.0, 0.0, 0.0))
>>> m
Matrix4([
    1.00    0.00    0.00    0.00
    0.00    0.00   -1.00    0.00
    0.00    1.00    0.00    0.00
    0.00    0.00    0.00    1.00])

```

new_rotate_euler(heading, attitude, bank) Create a **Matrix4** for the given Euler rotation. *heading* is a rotation around the Y axis, *attitude* around the X axis and *bank* around the Z axis. All rotations are performed simultaneously, so this method avoids “gimbal lock” and is the usual method for implemented 3D rotations in a game. Example:

```

>>> m = Matrix4.new_rotate_euler(math.pi / 2, math.pi / 2, 0.0)
>>> m
Matrix4([
    0.00   -0.00    1.00    0.00
    1.00    0.00   -0.00    0.00
   -0.00    1.00    0.00    0.00
    0.00    0.00    0.00    1.00])

```

new_perspective(fov_y, aspect, near, far) Create a **Matrix4** for projection onto the 2D viewing plane. This method is equivalent to the OpenGL call `gluPerspective`. *fov_y* is the view angle in the Y direction, in radians. *aspect* is the aspect ration *width / height* of the viewing plane. *near* and *far* are the distance to the near and far clipping planes. They must be positive and non-zero. Example:

```
>>> m = Matrix4.new_perspective(math.pi / 2, 1024.0 / 768, 1.0, 100.0)
>>> m
Matrix4([
    0.75    0.00    0.00    0.00
    0.00    1.00    0.00    0.00
    0.00    0.00   -1.02   -2.02
    0.00    0.00   -1.00    0.00])
```

Operators

Matrices of the same dimension may be multiplied to give a new matrix. For example, to create a transform which translates and scales:

```
>>> m1 = Matrix3.new_translate(5.0, 6.0)
>>> m2 = Matrix3.new_scale(1.0, 2.0)
>>> m1 * m2
Matrix3([
    1.00    0.00    5.00
    0.00    2.00    6.00
    0.00    0.00    1.00])
```

Note that multiplication is not commutative (the order that you apply transforms matters):

```
>>> m2 * m1
Matrix3([
    1.00    0.00    5.00
    0.00    2.00   12.00
    0.00    0.00    1.00])
```

In-place multiplication is also permitted (and optimised):

```
>>> m1 *= m2
>>> m1
Matrix3([
    1.00    0.00    5.00
    0.00    2.00    6.00
    0.00    0.00    1.00])
```

Multiplying a matrix by a vector returns a vector, and is used to transform a vector:

```
>>> m1 = Matrix3.new_rotate(math.pi / 2)
>>> m1 * Vector2(1.0, 1.0)
Vector2(-1.00, 1.00)
```

Note that translations have no effect on vectors. They do affect points, however:

```
>>> m1 = Matrix3.new_translate(5.0, 6.0)
>>> m1 * Vector2(1.0, 2.0)
Vector2(1.00, 2.00)
>>> m1 * Point2(1.0, 2.0)
Point2(6.00, 8.00)
```

Multiplication is currently incorrect between matrices and vectors – the projection component is ignored. Use the **Matrix4.transform** method instead.

Matrix4 also defines **transpose** (in-place), **transposed** (functional), **determinant** and **inverse** (functional) methods.

A **Matrix3** can be multiplied with a **Vector2** or any of the 2D geometry objects (**Point2**, **Line2**, **Circle**, etc).

A **Matrix4** can be multiplied with a **Vector3** or any of the 3D geometry objects (**Point3**, **Line3**, **Sphere**, etc).

For convenience, each of the matrix constructors are also available as in-place operators. For example, instead of writing:

```
>>> m1 = Matrix3.new_translate(5.0, 6.0)
>>> m2 = Matrix3.new_scale(1.0, 2.0)
>>> m1 *= m2
```

you can apply the scale directly to *m1*:

```
>>> m1 = Matrix3.new_translate(5.0, 6.0)
>>> m1.scale(1.0, 2.0)
Matrix3([ 1.00  0.00  5.00
          0.00  2.00  6.00
          0.00  0.00  1.00])
>>> m1
Matrix3([ 1.00  0.00  5.00
          0.00  2.00  6.00
          0.00  0.00  1.00])
```

Note that these methods operate in-place (they modify the original matrix), and they also return themselves as a result. This allows you to chain transforms together directly:

```
>>> Matrix3().translate(1.0, 2.0).rotate(math.pi / 2).scale(4.0, 4.0)
Matrix3([ 0.00 -4.00  1.00
          4.00  0.00  2.00
          0.00  0.00  1.00])
```

All constructors have an equivalent in-place method. For **Matrix3**, they are `identity`, `translate`, `scale` and `rotate`. For **Matrix4**, they are `identity`, `translate`, `scale`, `rotatex`, `rotatey`, `rotatez`, `rotate_axis` and `rotate_euler`. Both **Matrix3** and **Matrix4** also have an in-place transpose method.

The `copy` method is also implemented in both matrix classes and behaves in the obvious way.

```
__init__(*args)
__repr__()
__iter__()
__getitem__(key)
__setitem__(key, value)
__eq__(other)
__sub__(other)
__imul__(other)
__mul__(other)
__call__(other)
identity()
scale(x, y=None)
__class__
    alias of type
```

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__ = None

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__slotnames__ = []

__str__
Return str(self).

offset ()

angle (*angle=0*)

Parameters **angle** – angle in radians of a unit vector starting at origin

Returns float bearing in radians of the transformed vector

mag (*v=None*)
Return the net (uniform) scaling of this transform.

translate (**args*)

Parameters ***args** – x,y values

rotate (*angle*)

```
classmethod new_identity ()
classmethod new_scale (x, y)
classmethod new_translate (x, y)
classmethod new_rotate (angle)
mag2 ()
__abs__ ()
transpose ()
transposed ()
determinant ()
inverse ()
orientation ()
```

Returns 1 if matrix is right handed, -1 if left handed

Goulib.geom3d module

3D geometry

class Goulib.geom3d.**Vector3** (*args)

Bases: `object`

Mutable 3D Vector. See ‘Vector2’ documentation

Constructor. :param *args: x,y,z values

```
__init__ (*args)
```

Constructor. :param *args: x,y,z values

xyz

Returns tuple (x,y,z)

```
__repr__ ()
```

```
__eq__ (other)
```

```
__ne__ (other)
```

```
__bool__ ()
```

```
__nonzero__ ()
```

```
__len__ ()
```

```
__iter__ ()
```

```
__add__ (other)
```

```
__radd__ (other)
```

```
__iadd__ (other)
```

```
__sub__ (other)
```

```
__rsub__ (other)
```

```
__mul__ (other)
```

```

__rmul__(other)
__imul__(other)
__div__(other)
__rdiv__(other)
__floordiv__(other)
__rfloordiv__(other)
__truediv__(other)
__rtruediv__(other)
__neg__()
__pos__()
__abs__()
mag()
mag2()
normalize()
normalized()
dot(other)
cross(other)
reflect(normal)
rotate_around(axis, theta)
    Return the vector rotated around axis through angle theta. Right hand rule applies
angle(other)
    angle between two vectors. :param other: Vector3 :return: float angle in radians to the other vector, or self
    direction if other=None
project(other)
    Return one vector projected on the vector other
__hash__ = None

```

class Goulib.geom3d.**Point3**(*args)
 Bases: *Goulib.geom3d.Vector3, Goulib.geom.Geometry*

A point on a 3D plane. Construct in the obvious way:

```

>>> p = Point3(1.0, 2.0, 3.0)
>>> p
Point3(1.00, 2.00, 3.00)

```

Point3 subclasses **Vector3**, so all of **Vector3** operators and methods apply. In particular, subtracting two points gives a vector:

```

>>> Point3(1.0, 2.0, 3.0) - Point3(1.0, 0.0, -2.0)
Vector3(0.00, 2.00, 5.00)

```

The following methods are also defined:

intersect(other) If *other* is a **Sphere**, returns True iff the point lies within the sphere.

connect (other) Returns a **LineSegment3** which is the minimum length line segment that can connect the two shapes. *other* may be a **Point3**, **Line3**, **Ray3**, **LineSegment3**, **Sphere** or **Plane**.

distance (other) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of `connect`.

Constructor: :param **args*: x,y,z values

intersect (other)

Point3/object intersection :return: self Point3 if on other object, None if not

connect (other)

`__abstractmethods__ = frozenset()`

class Goulib.geom3d.**Line3** (**args*)

Bases: *Goulib.geom.Geometry*

A **Line3** is a line on a 3D plane extending to infinity in both directions; a **Ray3** has a finite end-point and extends to infinity in a single direction; a **LineSegment3** joins two points.

All three classes support the same constructors, operators and methods, but may behave differently when calculating intersections etc.

You may construct a line, ray or line segment using any of:

- another line, ray or line segment
- two points
- a point and a vector
- a point, a vector and a length

For example:

```
>>> Line3(Point3(1.0, 1.0, 1.0), Point3(1.0, 2.0, 3.0))
Line3(<1.00, 1.00, 1.00> + u<0.00, 1.00, 2.00>)
>>> Line3(Point3(0.0, 1.0, 1.0), Vector3(1.0, 1.0, 2.0))
Line3(<0.00, 1.00, 1.00> + u<1.00, 1.00, 2.00>)
>>> Ray3(Point3(1.0, 1.0, 1.0), Vector3(1.0, 1.0, 2.0), 1.0)
Ray3(<1.00, 1.00, 1.00> + u<0.41, 0.41, 0.82>)
```

Internally, lines, rays and line segments store a **Point3** *p* and a **Vector3** *v*. You can also access (but not set) the two endpoints *p1* and *p2*. These may or may not be meaningful for all types of lines.

The following methods are supported by all three classes:

intersect (other) If *other* is a **Sphere**, returns a **LineSegment3** which is the intersection of the sphere and line, or None if there is no intersection.

If *other* is a **Plane**, returns a **Point3** of intersection, or None.

connect (other) Returns a **LineSegment3** which is the minimum length line segment that can connect the two shapes. For two parallel lines, this line segment may be in an arbitrary position. *other* may be a **Point3**, **Line3**, **Ray3**, **LineSegment3**, **Sphere** or **Plane**.

distance (other) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of `connect`.

LineSegment3 also has a *length* property which is read-only.

`__init__` (**args*)

`__repr__` ()

p1

p2

point (*u*)

Returns Point3 at parameter *u*

intersect (*other*)

connect (*other*)

__abstractmethods__ = frozenset()

class Goulib.geom3d.**Ray3** (*args)

Bases: *Goulib.geom3d.Line3*

__abstractmethods__ = frozenset()

class Goulib.geom3d.**Segment3** (*args)

Bases: *Goulib.geom3d.Line3*

__repr__ ()

__abs__ ()

mag2 ()

swap ()

length

__abstractmethods__ = frozenset()

Goulib.geom3d.**Spherical** (*r, theta, phi*)

class Goulib.geom3d.**Sphere** (*args)

Bases: *Goulib.geom.Geometry*

Spheres are constructed with a center **Point3** and a radius:

```
>>> s = Sphere(Point3(1.0, 1.0, 1.0), 0.5)
>>> s
```

Sphere(<1.00, 1.00, 1.00>, radius=0.50)

Internally there are two attributes: *c*, giving the center point and *r*, giving the radius.

The following methods are supported:

intersect (**other**): If *other* is a **Point3**, returns True iff the point lies within the sphere.

If *other* is a **Line3**, **Ray3** or **LineSegment3**, returns a **LineSegment3** giving the intersection, or None if the line does not intersect the sphere.

distance (**other**) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of **connect**.

Parameters **args** – can be

- Sphere
- center, point on sphere
- center, radius

__init__ (*args)

Parameters *args* – can be

- Sphere
- center, point on sphere
- center, radius

`__repr__()`

`__contains__(pt)`

Returns True if pt is ON or IN the sphere

point (*u, v*)

Parameters

- **u** – float angle from “north pole” (=radians(90-lat) in radians
- **v** – float angle from 0 meridian

Returns Point3 on sphere at specified coordinates

intersect (*other*)

connect (*other*)

minimal joining segment between Sphere and other 3D Object :param other: Point3, Line3, Sphere, Plane
:return: LineSegment3 of minimal length

distance_on_sphere (*phi1, theta1, phi2, theta2*)

Parameters

- **phi1** – float angle from “north pole” (=radians(90-lat) in radians
- **theta1** – float angle from 0 meridian
- **phi2** – float angle from “north pole” (=radians(90-lat) in radians
- **theta2** – float angle from 0 meridian

`__abstractmethods__ = frozenset()`

class Goulib.geom3d.**Plane** (**args*)

Bases: *Goulib.geom.Geometry*

Planes can be constructed with any of:

- three **Point3**'s lying on the plane
- a **Point3** on the plane and the **Vector3** normal
- a **Vector3** normal and *k*, described below.

Internally, planes are stored with the normal *n* and constant *k* such that $n.p = k$ for any point on the plane *p*.

The following methods are supported:

intersect (**other**) If *other* is a **Line3**, **Ray3** or **LineSegment3**, returns a **Point3** of intersection, or None if there is no intersection.

If *other* is a **Plane**, returns the **Line3** of intersection.

connect (**other**) Returns a **LineSegment3** which is the minimum length line segment that can connect the two shapes. *other* may be a **Point3**, **Line3**, **Ray3**, **LineSegment3**, **Sphere** or **Plane**.

distance (*other*) Returns the absolute minimum distance to *other*. Internally this simply returns the length of the result of `connect`.

`__init__` (*args)

`__repr__` ()

intersect (*other*)

connect (*other*)

`__abstractmethods__` = frozenset()

class Goulib.geom3d.**Matrix4** (*args)

Bases: `object`

`__init__` (*args)

`__repr__` ()

`__iter__` ()

`__getitem__` (*key*)

`__setitem__` (*key*, *value*)

`__mul__` (*other*)

`__call__` (*other*)

`__imul__` (*other*)

ttransform (*other*)

identity ()

scale (*x*, *y*, *z*)

translate (*x*, *y*, *z*)

rotatex (*angle*)

rotatey (*angle*)

rotatez (*angle*)

rotate_axis (*angle*, *axis*)

rotate_euler (*heading*, *attitude*, *bank*)

rotate_triple_axis (*x*, *y*, *z*)

transpose ()

transposed ()

classmethod new (**values*)

classmethod new_identity ()

classmethod new_scale (*x*, *y*, *z*)

classmethod new_translate (*x*, *y*, *z*)

classmethod new_rotatex (*angle*)

classmethod new_rotatey (*angle*)

classmethod new_rotatez (*angle*)

classmethod new_rotate_axis (*angle*, *axis*)

classmethod `new_rotate_euler` (*heading, attitude, bank*)

classmethod `new_rotate_triple_axis` (*x, y, z*)

classmethod `new_look_at` (*eye, at, up*)

classmethod `new_perspective` (*fov_y, aspect, near, far*)

determinant ()

inverse ()

class `Goulib.geom3d.Quaternion` (*w=1, x=0, y=0, z=0*)

Bases: `object`

A quaternion represents a three-dimensional rotation or reflection transformation. They are the preferred way to store and manipulate rotations in 3D applications, as they do not suffer the same numerical degradation that matrices do.

The quaternion constructor initializes to the identity transform:

```
>>> q = Quaternion()
>>> q
Quaternion(real=1.00, imag=<0.00, 0.00, 0.00>)
```

Element access

Internally, the quaternion is stored as four attributes: `x`, `y` and `z` forming the imaginary vector, and `w` the real component.

Constructors

Rotations can be formed using the constructors:

new_identity() Equivalent to the default constructor.

new_rotate_axis(*angle, axis*) Equivalent to the `Matrix4` constructor of the same name. *angle* is specified in radians, *axis* is an instance of `Vector3`. It is not necessary to normalize the axis. Example:

```
>>> q = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q
Quaternion(real=0.71, imag=<0.71, 0.00, 0.00>)
```

new_rotate_euler(*heading, attitude, bank*) Equivalent to the `Matrix4` constructor of the same name. *heading* is a rotation around the Y axis, *attitude* around the X axis and *bank* around the Z axis. All angles are given in radians. Example:

```
>>> q = Quaternion.new_rotate_euler(math.pi / 2, math.pi / 2, 0)
>>> q
Quaternion(real=0.50, imag=<0.50, 0.50, 0.50>)
```

new_interpolate(*q1, q2, t*) Create a quaternion which gives a (SLERP) interpolated rotation between *q1* and *q2*. *q1* and *q2* are instances of `Quaternion`, and *t* is a value between 0.0 and 1.0. For example:

```
>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q2 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(0, 1, 0))
>>> for i in range(11):
...     print Quaternion.new_interpolate(q1, q2, i / 10.0)
...
Quaternion(real=0.71, imag=<0.71, 0.00, 0.00>)
Quaternion(real=0.75, imag=<0.66, 0.09, 0.00>)
```

```

Quaternion(real=0.78, imag=<0.61, 0.17, 0.00>)
Quaternion(real=0.80, imag=<0.55, 0.25, 0.00>)
Quaternion(real=0.81, imag=<0.48, 0.33, 0.00>)
Quaternion(real=0.82, imag=<0.41, 0.41, 0.00>)
Quaternion(real=0.81, imag=<0.33, 0.48, 0.00>)
Quaternion(real=0.80, imag=<0.25, 0.55, 0.00>)
Quaternion(real=0.78, imag=<0.17, 0.61, 0.00>)
Quaternion(real=0.75, imag=<0.09, 0.66, 0.00>)
Quaternion(real=0.71, imag=<0.00, 0.71, 0.00>)

```

Operators

Quaternions may be multiplied to compound rotations. For example, to rotate 90 degrees around the X axis and then 90 degrees around the Y axis:

```

>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q2 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(0, 1, 0))
>>> q1 * q2
Quaternion(real=0.50, imag=<0.50, 0.50, 0.50>)

```

Multiplying a quaternion by a vector gives a vector, transformed appropriately:

```

>>> q = Quaternion.new_rotate_axis(math.pi / 2, Vector3(0, 1, 0))
>>> q * Vector3(1.0, 0, 0)
Vector3(0.00, 0.00, -1.00)

```

Similarly, any 3D object can be multiplied (e.g., **Point3**, **Line3**, **Sphere**, etc):

```

>>> q * Ray3(Point3(1., 1., 1.), Vector3(1., 1., 1.))
Ray3(<1.00, 1.00, -1.00> + u<1.00, 1.00, -1.00>)

```

As with the matrix classes, the constructors are also available as in-place operators. These are named `identity`, `rotate_euler` and `rotate_axis`. For example:

```

>>> q1 = Quaternion()
>>> q1.rotate_euler(math.pi / 2, math.pi / 2, 0)
Quaternion(real=0.50, imag=<0.50, 0.50, 0.50>)
>>> q1
Quaternion(real=0.50, imag=<0.50, 0.50, 0.50>)

```

Quaternions are usually unit length, but you may wish to use sized quaternions. In this case, you can find the magnitude using `abs`, `magnitude` and `magnitude_squared`, as with the vector classes. Example:

```

>>> q1 = Quaternion()
>>> abs(q1)
1.0
>>> q1.magnitude()
1.0

```

Similarly, the class implements `normalize` and `normalized` in the same way as the vectors.

The following methods do not alter the quaternion:

conjugated() Returns a quaternion that is the conjugate of the instance. For example:

```

>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q1.conjugated()
Quaternion(real=0.71, imag=<-0.71, -0.00, -0.00>)

```

```
>>> q1
Quaternion(real=0.71, imag=<0.71, 0.00, 0.00>)
```

get_angle_axis() Returns a tuple (angle, axis), giving the angle to rotate around an axis equivalent to the quaternion. For example:

```
>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q1.get_angle_axis()
(1.5707963267948966, Vector3(1.00, 0.00, 0.00))
```

get_matrix() Returns a **Matrix4** implementing the transformation of the quaternion. For example:

```
>>> q1 = Quaternion.new_rotate_axis(math.pi / 2, Vector3(1, 0, 0))
>>> q1.get_matrix()
Matrix4([
    1.00    0.00    0.00    0.00
    0.00    0.00    -1.00   0.00
    0.00    1.00    0.00    0.00
    0.00    0.00    0.00    1.00])
```

__init__ (*w=1, x=0, y=0, z=0*)

__repr__ ()

__mul__ (*other*)

__imul__ (*other*)

mag2 ()

__abs__ ()

mag ()

identity ()

rotate_axis (*angle, axis*)

rotate_euler (*heading, attitude, bank*)

rotate_matrix (*m*)

conjugated ()

normalize ()

normalized ()

get_angle_axis ()

get_euler ()

get_matrix ()

classmethod new_identity ()

classmethod new_rotate_axis (*angle, axis*)

classmethod new_rotate_euler (*heading, attitude, bank*)

classmethod new_rotate_matrix (*m*)

classmethod new_interpolate (*q1, q2, t*)

Goulib.graph module

efficient Euclidian Graphs for `networkx` and related algorithms

requires

- `networkx`
- `matplotlib`

optional

- `scipy` for delauney triangulation
- `rtree` for faster GeoGraph algorithms

class `Goulib.graph.index`

Bases: `object`

class `Property`

Bases: `object`

set_dimension (*n*)

class `index.Index` (*properties*)

Bases: `dict`

fallback for `rtree.index`

__init__ (*properties*)

count (*ignored*)

insert (*k, p, _*)

delete (*k, _*)

nearest (*p, num_results, objects='raw'*)

very inefficient, but remember it's a fallback...

class `Goulib.graph.AGraph`

Bases: `object`

`Goulib.graph.to_networkx_graph` (*data, create_using=None, multigraph_input=False*)

Make a NetworkX graph from a known data structure. enhances `networkx.convert.to_networkx_graph` :param data: any type handled by `convert.to_networkx_graph`, plus: * `scipy.spatial.qhull.Delaunay` to enable building a graph from a delauney triangulation

If `create_using` is a :class:'GeoGraph' and data is a Graph where nodes have a 'pos' attribute, then this attribute will be used to rename nodes as (x,y,...) tuples suitable for GeoGraph.

class `Goulib.graph.GeoGraph` (*data=None, nodes=None, **kwargs*)

Bases: `Goulib.graph._Geo`, `networkx.classes.multigraph.MultiGraph`

Undirected graph with nodes positions can be set to non multiedges anytime with attribute `multi=False`

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

__init__ (*data=None, nodes=None, **kwargs*)

Parameters

- **data** – see `to_networkx_graph()` for valid types

- **kwargs** – other parameters will be copied as attributes, especially:

class Goulib.graph.**DiGraph** (*data=None, nodes=None, **kwargs*)

Bases: Goulib.graph._Geo, networkx.classes.multidigraph.MultiDiGraph

directed graph with nodes positions can be set to non multiedges anytime with attribute multi=False

Parameters

- **data** – see *to_networkx_graph()* for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

__init__ (*data=None, nodes=None, **kwargs*)

Parameters

- **data** – see *to_networkx_graph()* for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

Goulib.graph.**figure** (*g, box=None, **kwargs*)

Parameters

- **g** – _Geo derived Graph
- **box** – optional interval.Box if g has no box

Returns matplotlib axis suitable for drawing graph g

Goulib.graph.**draw_networkx** (*g, pos=None, **kwargs*)

improves nx.draw_networkx :param g: NetworkX Graph :param pos: can be either :

- optional dictionary of (x,y) node positions
- function of the form lambda node:(x,y) that maps node positions.
- None. in this case, nodes are directly used as positions if graph is a GeoGraph, otherwise nx.draw_shell is used

Parameters ****kwargs** – passed to nx.draw method as described in http://networkx.lanl.gov/reference/generated/networkx.drawing.nx_pylab.draw_networkx.html with one tweak:

- if edge_color is a function of the form lambda data:color string, it is mapped over all edges

Goulib.graph.**to_drawing** (*g, d=None, edges=[]*)

draws Graph to a *Drawing* :param g: Graph :param d: existing Drawing to draw onto, or None to create a new Drawing :param edges: iterable of edges (with data) that will be added, in the same order. By default all edges are drawn :return: Drawing

Graph edges with an ‘entity’ property

Goulib.graph.**write_dxf** (*g, filename*)

writes networkx.Graph in .dxf format

Goulib.graph.**write_dot** (*g, filename*)

Goulib.graph.**to_json** (*g, **kwargs*)

Returns string JSON representation of a graph

Goulib.graph.**write_json** (*g, filename, **kwargs*)

write a JSON file, suitable for D*.js representation

`Goulib.graph.delauney_triangulation` (*nodes*, *qhull_options*='', *incremental*=False, ***kwargs*)
https://en.wikipedia.org/wiki/Delaunay_triangulation :param *nodes*: _Geo graph or list of (x,y) or (x,y,z) node positions :param *qhull_options*: string passed to `scipy.spatial.Delaunay()`, which passes it to Qhull (<http://www.qhull.org/>) *'Qt' ensures all points are connected *'Qz' required when nodes lie on a sphere *'QJ' solves some singularity situations

Parameters *kwargs* – passed to the *GeoGraph* constructor

Returns *GeoGraph* with delauney triangulation between nodes

`Goulib.graph.euclidean_minimum_spanning_tree` (*nodes*, ***kwargs*)

Parameters *nodes* – list of (x,y) nodes positions

Returns *GeoGraph* with minimum spanning tree between nodes

see https://en.wikipedia.org/wiki/Euclidean_minimum_spanning_tree

`Goulib.graph.points_on_sphere` (*N*)

Classes

efficient Euclidian Graphs for `networkx` and related algorithms

requires

- `networkx`
- `matplotlib`

optional

- `scipy` for delauney triangulation
- `rtree` for faster *GeoGraph* algorithms

class `Goulib.graph.index`

Bases: `object`

class `Property`

Bases: `object`

set_dimension (*n*)

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__
Return self>value.

__hash__
Return hash(self).

__init__
Initialize self. See help(type(self)) for accurate signature.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

class `index.Index` (*properties*)
Bases: `dict`

fallback for `rtree.index`

__init__ (*properties*)

count (*ignored*)

insert (*k, p, _*)

delete (*k, _*)

nearest (*p, num_results, objects='raw'*)
very inefficient, but remember it's a fallback...

__class__
alias of `type`

__contains__ ()
True if D has a key k, else False.

__delattr__
Implement delattr(self, name).

`__delitem__`
Delete self[key].

`__dir__` () → list
default dir() implementation

`__eq__`
Return self==value.

`__format__` ()
default object formatter

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__getitem__` ()
x.__getitem__(y) <==> x[y]

`__gt__`
Return self>value.

`__hash__` = None

`__iter__`
Implement iter(self).

`__le__`
Return self<=value.

`__len__`
Return len(self).

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return repr(self).

`__setattr__`
Implement setattr(self, name, value).

`__setitem__`
Set self[key] to value.

`__sizeof__` () → size of D in memory, in bytes

`__str__`
Return str(self).

`clear` () → None. Remove all items from D.

copy () → a shallow copy of D

fromkeys ()
Returns a new dict with keys from iterable and values equal to value.

get (*k*, *d*) → D[*k*] if *k* in D, else *d*. *d* defaults to None.

items () → a set-like object providing a view on D's items

keys () → a set-like object providing a view on D's keys

pop (*k*, *d*) → *v*, remove specified key and return the corresponding value.
If key is not found, *d* is returned if given, otherwise `KeyError` is raised

popitem () → (*k*, *v*), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

setdefault (*k*, *d*) → D.get(*k*,*d*), also set D[*k*]=*d* if *k* not in D

update (*E*, ***F*) → None. Update D from dict/iterable *E* and *F*.
If *E* is present and has a `.keys()` method, then does: for *k* in *E*: D[*k*] = *E*[*k*] If *E* is present and lacks a `.keys()` method, then does: for *k*, *v* in *E*: D[*k*] = *v* In either case, this is followed by: for *k* in *F*: D[*k*] = *F*[*k*]

values () → an object providing a view on D's values

index. **__class__**
alias of `type`

index. **__delattr__**
Implement `delattr(self, name)`.

index. **__dir__** () → list
default `dir()` implementation

index. **__eq__**
Return `self==value`.

index. **__format__** ()
default object formatter

index. **__ge__**
Return `self>=value`.

index. **__getattr__**
Return `getattr(self, name)`.

index. **__gt__**
Return `self>value`.

index. **__hash__**
Return `hash(self)`.

index. **__init__**
Initialize self. See `help(type(self))` for accurate signature.

index. **__le__**
Return `self<=value`.

index. **__lt__**
Return `self<value`.

index. **__ne__**
Return `self!=value`.

`index.__new__()`
 Create and return a new object. See `help(type)` for accurate signature.

`index.__reduce__()`
 helper for pickle

`index.__reduce_ex__()`
 helper for pickle

`index.__repr__`
 Return `repr(self)`.

`index.__setattr__`
 Implement `setattr(self, name, value)`.

`index.__sizeof__()` → int
 size of object in memory, in bytes

`index.__str__`
 Return `str(self)`.

class `Goulib.graph.AGraph`

Bases: `object`

`__class__`
 alias of `type`

`__delattr__`
 Implement `delattr(self, name)`.

`__dir__()` → list
 default `dir()` implementation

`__eq__`
 Return `self==value`.

`__format__()`
 default object formatter

`__ge__`
 Return `self>=value`.

`__getattr__`
 Return `getattr(self, name)`.

`__gt__`
 Return `self>value`.

`__hash__`
 Return `hash(self)`.

`__init__`
 Initialize self. See `help(type(self))` for accurate signature.

`__le__`
 Return `self<=value`.

`__lt__`
 Return `self<value`.

`__ne__`
 Return `self!=value`.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

Goulib.graph.**to_networkx_graph** (data, create_using=None, multigraph_input=False)

Make a NetworkX graph from a known data structure. enhances `networkx.convert.to_networkx_graph` :param data: any type handled by `convert.to_networkx_graph`, plus: * `scipy.spatial.qhull.Delaunay` to enable building a graph from a delauney triangulation

If create_using is a :class:‘GeoGraph’ and data is a Graph where nodes have a ‘pos’ attribute, then this attribute will be used to rename nodes as (x,y,...) tuples suitable for GeoGraph.

class Goulib.graph.**GeoGraph** (data=None, nodes=None, **kwargs)

Bases: Goulib.graph._Geo, networkx.classes.multigraph.MultiGraph

Undirected graph with nodes positions can be set to non multiedges anytime with attribute multi=False

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

__init__ (data=None, nodes=None, **kwargs)

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

__bool__ ()

Returns True if graph has at least one node

__class__
alias of `type`

__contains__ (n)
Return True if n is a node, False otherwise. Use the expression ‘n in G’.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> 1 in G
True
```

__delattr__
Implement delattr(self, name).

`__dir__()` → list
default dir() implementation

`__eq__(other)`

Returns True if self and other are equal

`__format__()`
default object formatter

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__getitem__(n)`
Return a dict of neighbors of node n. Use the expression 'G[n]'.
n [node] A node in the graph.

adj_dict [dictionary] The adjacency dictionary for nodes connected to n.

adj_dict [dictionary] The adjacency dictionary for nodes connected to n.

G[n] is similar to G.neighbors(n) but the internal data dictionary is returned instead of a list.

Assigning G[n] will corrupt the internal graph data structure. Use G[n] for reading data only.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G[0]
{1: {}}
```

`__gt__`
Return self>value.

`__hash__ = None`

`__iter__()`
Iterate over the nodes. Use the expression 'for n in G'.

niter [iterator] An iterator over all nodes in the graph.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
```

`__le__`
Return self<=value.

`__len__()`
Return the number of nodes. Use the expression 'len(G)'.

nnodes [int] The number of nodes in the graph.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> len(G)
4
```

`__lt__`
Return self<value.

`__ne__`

Return self!=value.

`__new__` ()

Create and return a new object. See help(type) for accurate signature.

`__nonzero__` ()

Returns True if graph has at least one node

`__reduce__` ()

helper for pickle

`__reduce_ex__` ()

helper for pickle

`__repr__`

Return repr(self).

`__setattr__`

Implement setattr(self, name, value).

`__sizeof__` () → int

size of object in memory, in bytes

`__str__` ()

Returns string representation, used mainly for logging and debugging

`add_cycle` (nodes, **attr)

Add a cycle.

nodes: iterable container A container of nodes. A cycle will be constructed from the nodes (in order) and added to the graph.

attr [keyword arguments, optional (default= no attributes)] Attributes to add to every edge in cycle.

add_path, add_star

```
>>> G=nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_cycle([0,1,2,3])
>>> G.add_cycle([10,11,12],weight=7)
```

`add_edge` (u, v, k=None, attr_dict=None, **attrs)

add an edge to graph :return: edge data from created or existing edge

`add_edges_from` (ebunch, attr_dict=None, **attr)

Add all the edges in ebunch.

ebunch [container of edges] Each edge given in the container will be added to the graph. The edges can be:

- 2-tuples (u,v) or
- 3-tuples (u,v,d) for an edge attribute dict d, or
- 4-tuples (u,v,k,d) for an edge identified by key k

attr_dict [dictionary, optional (default= no attributes)] Dictionary of edge attributes. Key/value pairs will update existing data associated with each edge.

attr [keyword arguments, optional] Edge data (or labels or objects) can be assigned using keyword arguments.

`add_edge` : add a single edge `add_weighted_edges_from` : convenient way to add weighted edges

Adding the same edge twice has no effect but any edge data will be updated when each duplicate edge is added.

Edge attributes specified in edges take precedence over attributes specified generally.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edges_from([(0,1), (1,2)]) # using a list of edge tuples
>>> e = zip(range(0,3), range(1,4))
>>> G.add_edges_from(e) # Add the path graph 0-1-2-3
```

Associate data to edges

```
>>> G.add_edges_from([(1,2), (2,3)], weight=3)
>>> G.add_edges_from([(3,4), (1,4)], label='WN2898')
```

add_node (*p*, *attr_dict=None*, ***attr*)

add a node or return one already very close :return (x,y,...) node id

add_nodes_from (*nodes*, ***attr*)

must be here because Graph.add_nodes_from doesn't call add_node cleanly as it should...

add_path (*nodes*, ***attr*)

Add a path.

nodes [iterable container] A container of nodes. A path will be constructed from the nodes (in order) and added to the graph.

attr [keyword arguments, optional (default= no attributes)] Attributes to add to every edge in path.

`add_star`, `add_cycle`

```
>>> G=nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.add_path([10,11,12],weight=7)
```

add_star (*nodes*, ***attr*)

Add a star.

The first node in nodes is the middle of the star. It is connected to all other nodes.

nodes [iterable container] A container of nodes.

attr [keyword arguments, optional (default= no attributes)] Attributes to add to every edge in star.

`add_path`, `add_cycle`

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_star([0,1,2,3])
>>> G.add_star([10,11,12],weight=2)
```

add_weighted_edges_from (*ebunch*, *weight='weight'*, ***attr*)

Add all the edges in ebunch as weighted edges with specified weights.

ebunch [container of edges] Each edge given in the list or container will be added to the graph. The edges must be given as 3-tuples (u,v,w) where w is a number.

weight [string, optional (default= 'weight')] The attribute name for the edge weights to be added.

attr [keyword arguments, optional (default= no attributes)] Edge attributes to add/update for all edges.

`add_edge` : add a single edge `add_edges_from` : add multiple edges

Adding the same edge twice for `Graph/DiGraph` simply updates the edge data. For `MultiGraph/MultiDiGraph`, duplicate edges are stored.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_weighted_edges_from([(0,1,3.0), (1,2,7.5)])
```

`adjacency_iter()`

Return an iterator of (node, adjacency dict) tuples for all nodes.

This is the fastest way to look at every edge. For directed graphs, only outgoing adjacencies are included.

adj_iter [iterator] An iterator of (node, adjacency dictionary) for all nodes in the graph.

`adjacency_list`

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> [(n,nbrdict) for n,nbrdict in G.adjacency_iter()]
[(0, {1: {}}), (1, {0: {}, 2: {}}), (2, {1: {}, 3: {}}), (3, {2: {}})]
```

`adjacency_list()`

Return an adjacency list representation of the graph.

The output adjacency list is in the order of `G.nodes()`. For directed graphs, only outgoing adjacencies are included.

adj_list [lists of lists] The adjacency structure of the graph as a list of lists.

`adjacency_iter`

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.adjacency_list() # in order given by G.nodes()
[[1], [0, 2], [1, 3], [2]]
```

`adjlist_dict_factory`

alias of `dict`

`box()`

Returns nodes bounding box as (xmin,ymin,...),(xmax,ymax,...)

`box_size()`

Returns (x,y) size

`clear()`

`closest_edges(p, data=False)`

Returns container of edges close to p and distance

`closest_nodes(p, n=1, skip=False)`

nodes closest to a given position :param p: (x,y) position tuple :param skip: optional bool to skip n itself :return: list of nodes, minimal distance

`contiguity(pts)`

Returns int number of points from pts already in graph

`copy()`

Returns copy of self graph

degree (*nbunch=None, weight=None*)

Return the degree of a node or nodes.

The node degree is the number of edges adjacent to that node.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights adjacent to the node.

nd [dictionary, or number] A dictionary with nodes as keys and degree as values or a number if a single node is specified.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.degree(0)
1
>>> G.degree([0,1])
{0: 1, 1: 2}
>>> list(G.degree([0,1]).values())
[1, 2]
```

degree_iter (*nbunch=None, weight=None*)

Return an iterator for (node, degree).

The node degree is the number of edges adjacent to the node.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights adjacent to the node.

nd_iter [an iterator] The iterator returns two-tuples of (node, degree).

degree

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> list(G.degree_iter(0)) # node 0 with degree 1
[(0, 1)]
>>> list(G.degree_iter([0,1]))
[(0, 1), (1, 2)]
```

dist (*u, v*)

Returns float distance between nodes *u* and *v*

draw (***kwargs*)

draw graph with default params

edge_attr_dict_factory

alias of `dict`

edge_key_dict_factory

alias of `dict`

edges (*nbunch=None, data=False, keys=False, default=None*)

Return a list of edges.

Edges are returned as tuples with optional data and keys in the order (node, neighbor, key, data).

nbunch [iterable container, optional (default= all nodes)] A container of nodes. The container will be iterated through once.

data [bool, optional (default=False)] Return two tuples (u,v) (False) or three-tuples (u,v,data) (True).

keys [bool, optional (default=False)] Return two tuples (u,v) (False) or three-tuples (u,v,key) (True).

edge_list: list of edge tuples Edges that are adjacent to any node in nbunch, or a list of all edges if nbunch is not specified.

edges_iter : return an iterator over the edges

Nodes in nbunch that are not in the graph will be (quietly) ignored. For directed graphs this returns the out-edges.

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2])
>>> G.add_edge(2,3,weight=5)
>>> G.edges()
[(0, 1), (1, 2), (2, 3)]
>>> G.edges(data=True) # default edge data is {} (empty dictionary)
[(0, 1, {}), (1, 2, {}), (2, 3, {'weight': 5})]
>>> list(G.edges_iter(data='weight', default=1))
[(0, 1, 1), (1, 2, 1), (2, 3, 5)]
>>> G.edges(keys=True) # default keys are integers
[(0, 1, 0), (1, 2, 0), (2, 3, 0)]
>>> G.edges(data=True, keys=True) # default keys are integers
[(0, 1, 0, {}), (1, 2, 0, {}), (2, 3, 0, {'weight': 5})]
>>> list(G.edges(data='weight', default=1, keys=True))
[(0, 1, 0, 1), (1, 2, 0, 1), (2, 3, 0, 5)]
>>> G.edges([0,3])
[(0, 1), (3, 2)]
>>> G.edges(0)
[(0, 1)]
```

edges_iter (*nbunch=None, data=False, keys=False, default=None*)

Return an iterator over the edges.

Edges are returned as tuples with optional data and keys in the order (node, neighbor, key, data).

nbunch [iterable container, optional (default= all nodes)] A container of nodes. The container will be iterated through once.

data [string or bool, optional (default=False)] The edge attribute returned in 3-tuple (u,v,ddict[data]). If True, return edge attribute dict in 3-tuple (u,v,ddict). If False, return 2-tuple (u,v).

default [value, optional (default=None)] Value used for edges that dont have the requested attribute. Only relevant if data is not True or False.

keys [bool, optional (default=False)] If True, return edge keys with each edge.

edge_iter [iterator] An iterator of (u,v), (u,v,d) or (u,v,key,d) tuples of edges.

edges : return a list of edges

Nodes in nbunch that are not in the graph will be (quietly) ignored. For directed graphs this returns the out-edges.

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2])
>>> G.add_edge(2,3,weight=5)
>>> [e for e in G.edges_iter()]
[(0, 1), (1, 2), (2, 3)]
>>> list(G.edges_iter(data=True)) # default data is {} (empty dict)
[(0, 1, {}), (1, 2, {}), (2, 3, {'weight': 5})]
>>> list(G.edges_iter(data='weight', default=1))
[(0, 1, 1), (1, 2, 1), (2, 3, 5)]
>>> list(G.edges(keys=True)) # default keys are integers
[(0, 1, 0), (1, 2, 0), (2, 3, 0)]
>>> list(G.edges(data=True,keys=True)) # default keys are integers
[(0, 1, 0, {}), (1, 2, 0, {}), (2, 3, 0, {'weight': 5})]
>>> list(G.edges(data='weight',default=1,keys=True))
[(0, 1, 0, 1), (1, 2, 0, 1), (2, 3, 0, 5)]
>>> list(G.edges_iter([0,3]))
[(0, 1), (3, 2)]
>>> list(G.edges_iter(0))
[(0, 1)]
```

get_edge_data (*u, v, key=None, default=None*)

Return the attribute dictionary associated with edge (u,v).

u, v : nodes

default [any Python object (default=None)] Value to return if the edge (u,v) is not found.

key [hashable identifier, optional (default=None)] Return data only for the edge with specified key.

edge_dict [dictionary] The edge attribute dictionary.

It is faster to use `G[u][v][key]`.

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_edge(0,1,key='a',weight=7)
>>> G[0][1]['a'] # key='a'
{'weight': 7}
```

Warning: Assigning `G[u][v][key]` corrupts the graph data structure. But it is safe to assign attributes to that dictionary,

```
>>> G[0][1]['a']['weight'] = 10
>>> G[0][1]['a']['weight']
10
>>> G[1][0]['a']['weight']
10
```

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2,3])
>>> G.get_edge_data(0,1)
{0: {}}
>>> e = (0,1)
>>> G.get_edge_data(*e) # tuple form
{0: {}}
>>> G.get_edge_data('a','b',default=0) # edge not in graph, return 0
0
```

has_edge (*u, v, key=None*)

Return True if the graph has an edge between nodes *u* and *v*.

u, v [nodes] Nodes can be, for example, strings or numbers.

key [hashable identifier, optional (default=None)] If specified return True only if the edge with key is found.

edge_ind [bool] True if edge is in the graph, False otherwise.

Can be called either using two nodes *u,v*, an edge tuple (*u,v*), or an edge tuple (*u,v,key*).

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2,3])
>>> G.has_edge(0,1) # using two nodes
True
>>> e = (0,1)
>>> G.has_edge(*e) # e is a 2-tuple (u,v)
True
>>> G.add_edge(0,1,key='a')
>>> G.has_edge(0,1,key='a') # specify key
True
>>> e=(0,1,'a')
>>> G.has_edge(*e) # e is a 3-tuple (u,v,'a')
True
```

The following syntax are equivalent:

```
>>> G.has_edge(0,1)
True
>>> 1 in G[0] # though this gives KeyError if 0 not in G
True
```

has_node (*n*)

Return True if the graph contains the node *n*.

n : node

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
>>> G.has_node(0)
True
```

It is more readable and simpler to use

```
>>> 0 in G
True
```

html (***kwargs*)

is_directed ()

Return True if graph is directed, False otherwise.

is_multigraph ()

used internally in constructor

length (*edges=None*)

Parameters **edges** – iterator over edges either as (u,v,data) or (u,v,key,data). If None, all edges are taken

Returns sum of ‘length’ attributes of edges

multi

name

nbunch_iter (*nbunch=None*)

Return an iterator of nodes contained in nbunch that are also in the graph.

The nodes in nbunch are checked for membership in the graph and if not are silently ignored.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

niter [iterator] An iterator over nodes in nbunch that are also in the graph. If nbunch is None, iterate over all nodes in the graph.

NetworkXError If nbunch is not a node or or sequence of nodes. If a node in nbunch is not hashable.

Graph.__iter__

When nbunch is an iterator, the returned iterator yields values directly from nbunch, becoming exhausted when nbunch is exhausted.

To test whether nbunch is a single node, one can use “if nbunch in self:”, even after processing with this routine.

If nbunch is not a node or a (possibly empty) sequence/iterator or None, a NetworkXError is raised. Also, if any object in nbunch is not hashable, a NetworkXError is raised.

neighbors (*n*)

Return a list of the nodes connected to the node n.

n [node] A node in the graph

nlist [list] A list of nodes that are adjacent to n.

NetworkXError If the node n is not in the graph.

It is usually more convenient (and faster) to access the adjacency dictionary as G[n]:

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge('a', 'b', weight=7)
>>> G['a']
{'b': {'weight': 7}}
```

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.neighbors(0)
[1]
```

neighbors_iter (*n*)

Return an iterator over all neighbors of node n.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
```

```
>>> [n for n in G.neighbors_iter(0)]
[1]
```

It is faster to use the idiom “in G[0]”, e.g.

```
>>> G = nx.path_graph(4)
>>> [n for n in G[0]]
[1]
```

node_dict_factory

alias of `dict`

nodes (*data=False*)

Return a list of the nodes in the graph.

data [boolean, optional (default=False)] If False return a list of nodes. If True return a two-tuple of node and node data dictionary

nlist [list] A list of nodes. If `data=True` a list of two-tuples containing (node, node data dictionary).

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
>>> G.nodes()
[0, 1, 2]
>>> G.add_node(1, time='5pm')
>>> G.nodes(data=True)
[(0, {}), (1, {'time': '5pm'}), (2, {})]
```

nodes_iter (*data=False*)

Return an iterator over the nodes.

data [boolean, optional (default=False)] If False the iterator returns nodes. If True return a two-tuple of node and node data dictionary

niter [iterator] An iterator over nodes. If `data=True` the iterator gives two-tuples containing (node, node data, dictionary)

If the node data is not required it is simpler and equivalent to use the expression ‘for n in G’.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
```

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
```

```
>>> [d for n,d in G.nodes_iter(data=True)]
[{}, {}, {}]
```

nodes_with_selfloops ()

Return a list of nodes with self loops.

A node with a self loop has an edge with both ends adjacent to that node.

nodelist [list] A list of nodes with self loops.

`selfloop_edges`, `number_of_selfloops`


```

>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge(1,1)
>>> G.add_edge(1,2)
>>> G.nodes_with_selfloops()
[1]

```

number_of_edges (*u=None, v=None*)

Return the number of edges between two nodes.

u, v [nodes, optional (default=all edges)] If u and v are specified, return the number of edges between u and v. Otherwise return the total number of all edges.

nedges [int] The number of edges in the graph. If nodes u and v are specified return the number of edges between those nodes.

size

```

>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.number_of_edges()
3
>>> G.number_of_edges(0,1)
1
>>> e = (0,1)
>>> G.number_of_edges(*e)
1

```

number_of_nodes (*doublecheck=False*)

number_of_selfloops ()

Return the number of selfloop edges.

A selfloop edge has the same node at both ends.

nloops [int] The number of selfloops.

nodes_with_selfloops, selfloop_edges

```

>>> G=nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge(1,1)
>>> G.add_edge(1,2)
>>> G.number_of_selfloops()
1

```

order ()

Return the number of nodes in the graph.

nnodes [int] The number of nodes in the graph.

number_of_nodes, __len__ which are identical

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

pos (*nodes=None*)

Parameters **nodes** – a single node, an iterator of all nodes if None

Returns the position of node(s)

remove_edge (*u, v=None, key=None, clean=False*)

Parameters

- **u** – Node or Edge (Nodes tuple)
- **v** – Node if u is a single Node
- **clean** – bool removes disconnected nodes. must be False for certain nx algos to work

Result return attributes of removed edge

remove edge from graph. NetworkX graphs do not remove unused nodes

remove_edges_from (*ebunch*)

Remove all edges specified in ebunch.

ebunch: list or container of edge tuples Each edge given in the list or container will be removed from the graph. The edges can be:

- 2-tuples (u,v) All edges between u and v are removed.
- 3-tuples (u,v,key) The edge identified by key is removed.
- 4-tuples (u,v,key,data) where data is ignored.

remove_edge : remove a single edge

Will fail silently if an edge in ebunch is not in the graph.

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2,3])
>>> ebunch=[(1,2), (2,3)]
>>> G.remove_edges_from(ebunch)
```

Removing multiple copies of edges

```
>>> G = nx.MultiGraph()
>>> G.add_edges_from([(1,2), (1,2), (1,2)])
>>> G.remove_edges_from([(1,2), (1,2)])
>>> G.edges()
[(1, 2)]
>>> G.remove_edges_from([(1,2), (1,2)]) # silently ignore extra copy
>>> G.edges() # now empty graph
[]
```

remove_node (*n*)

Parameters **n** – node tuple

remove node from graph and rtree

remove_nodes_from (*nodes*)

Remove multiple nodes.

nodes [iterable container] A container of nodes (list, dict, set, etc.). If a node in the container is not in the graph it is silently ignored.

remove_node

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
>>> e = G.nodes()
>>> e
[0, 1, 2]
```

```
>>> G.remove_nodes_from(e)
>>> G.nodes()
[]
```

render (*fmt='svg', **kwargs*)

render graph to bitmap stream :param *fmt*: string defining the format. 'svg' by default for INotepads
:return: matplotlib figure as a byte stream in specified format

save (*filename, **kwargs*)

save graph in various formats

selfloop_edges (*data=False, keys=False, default=None*)

Return a list of selfloop edges.

A selfloop edge has the same node at both ends.

data [bool, optional (default=False)] Return selfloop edges as two tuples (u,v) (data=False) or three-tuples (u,v,datadict) (data=True) or three-tuples (u,v,datavalue) (data='attrname')

default [value, optional (default=None)] Value used for edges that dont have the requested attribute. Only relevant if data is not True or False.

keys [bool, optional (default=False)] If True, return edge keys with each edge.

edgelist [list of edge tuples] A list of all selfloop edges.

nodes_with_selfloops, number_of_selfloops

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_edge(1,1)
>>> G.add_edge(1,2)
>>> G.selfloop_edges()
[(1, 1)]
>>> G.selfloop_edges(data=True)
[(1, 1, {})]
>>> G.selfloop_edges(keys=True)
[(1, 1, 0)]
>>> G.selfloop_edges(keys=True, data=True)
[(1, 1, 0, {})]
```

shortest_path (*source=None, target=None*)

size (*weight=None*)

Return the number of edges.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1.

nedges [int] The number of edges or sum of edge weights in the graph.

number_of_edges

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.size()
3
```

```

>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge('a','b',weight=2)
>>> G.add_edge('b','c',weight=4)
>>> G.size()
2
>>> G.size(weight='weight')
6.0

```

stats()

Returns dict of graph data to use in `__repr__` or usable otherwise

subgraph (*nbunch*)

Return the subgraph induced on nodes in *nbunch*.

The induced subgraph of the graph contains the nodes in *nbunch* and the edges between those nodes.

nbunch [list, iterable] A container of nodes which will be iterated through once.

G [Graph] A subgraph of the graph with the same edge attributes.

The graph, edge or node attributes just point to the original graph. So changes to the node or edge structure will not be reflected in the original graph while changes to the attributes will.

To create a subgraph with its own copy of the edge/node attributes use: `nx.Graph(G.subgraph(nbunch))`

If edge attributes are containers, a deep copy can be obtained using: `G.subgraph(nbunch).copy()`

For an inplace reduction of a graph to a subgraph you can remove nodes: `G.remove_nodes_from([n in G if n not in set(nbunch)])`

```

>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> H = G.subgraph([0,1,2])
>>> H.edges()
[(0, 1), (1, 2)]

```

svg (***kwargs*)

to_directed ()

Return a directed representation of the graph.

G [MultiDiGraph] A directed graph with the same name, same nodes, and with each edge (u,v,data) replaced by two directed edges (u,v,data) and (v,u,data).

This returns a “deepcopy” of the edge, node, and graph attributes which attempts to completely copy all of the data and references.

This is in contrast to the similar `D=DiGraph(G)` which returns a shallow copy of the data.

See the Python copy module for more information on shallow and deep copies, <http://docs.python.org/library/copy.html>.

Warning: If you have subclassed `MultiGraph` to use dict-like objects in the data structure, those changes do not transfer to the `MultiDiGraph` created by this method.

```

>>> G = nx.Graph() # or MultiGraph, etc
>>> G.add_path([0,1])
>>> H = G.to_directed()
>>> H.edges()
[(0, 1), (1, 0)]

```

If already directed, return a (deep) copy

```
>>> G = nx.DiGraph() # or MultiDiGraph, etc
>>> G.add_path([0,1])
>>> H = G.to_directed()
>>> H.edges()
[(0, 1)]
```

`to_undirected()`

Return an undirected copy of the graph.

G [Graph/MultiGraph] A deepcopy of the graph.

copy, add_edge, add_edges_from

This returns a “deepcopy” of the edge, node, and graph attributes which attempts to completely copy all of the data and references.

This is in contrast to the similar `G=DiGraph(D)` which returns a shallow copy of the data.

See the Python copy module for more information on shallow and deep copies, <http://docs.python.org/library/copy.html>.

```
>>> G = nx.Graph() # or MultiGraph, etc
>>> G.add_path([0,1])
>>> H = G.to_directed()
>>> H.edges()
[(0, 1), (1, 0)]
>>> G2 = H.to_undirected()
>>> G2.edges()
[(0, 1)]
```

`tol`

class Goulib.graph.**DiGraph** (*data=None, nodes=None, **kwargs*)

Bases: Goulib.graph._Geo, networkx.classes.multidigraph.MultiDiGraph

directed graph with nodes positions can be set to non multiedges anytime with attribute `multi=False`

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

`__init__` (*data=None, nodes=None, **kwargs*)

Parameters

- **data** – see `to_networkx_graph()` for valid types
- **kwargs** – other parameters will be copied as attributes, especially:

`__bool__` ()

Returns True if graph has at least one node

`__class__`

alias of `type`

`__contains__` (*n*)

Return True if *n* is a node, False otherwise. Use the expression ‘*n* in *G*’.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> 1 in G
True
```

`__delattr__`

Implement `delattr(self, name)`.

`__dir__` () → list

default `dir()` implementation

`__eq__` (*other*)

Returns True if `self` and `other` are equal

`__format__` ()

default object formatter

`__ge__`

Return `self >= value`.

`__getattr__`

Return `getattr(self, name)`.

`__getitem__` (*n*)

Return a dict of neighbors of node `n`. Use the expression '`G[n]`'.

n [node] A node in the graph.

adj_dict [dictionary] The adjacency dictionary for nodes connected to `n`.

`G[n]` is similar to `G.neighbors(n)` but the internal data dictionary is returned instead of a list.

Assigning `G[n]` will corrupt the internal graph data structure. Use `G[n]` for reading data only.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G[0]
{1: {}}
```

`__gt__`

Return `self > value`.

`__hash__` = None

`__iter__` ()

Iterate over the nodes. Use the expression '`for n in G`'.

niter [iterator] An iterator over all nodes in the graph.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
```

`__le__`

Return `self <= value`.

`__len__` ()

Return the number of nodes. Use the expression '`len(G)`'.

nnodes [int] The number of nodes in the graph.

```

>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> len(G)
4

```

__lt__

Return self<value.

__ne__

Return self!=value.

__new__()

Create and return a new object. See help(type) for accurate signature.

__nonzero__()

Returns True if graph has at least one node

__reduce__()

helper for pickle

__reduce_ex__()

helper for pickle

__repr__

Return repr(self).

__setattr__

Implement setattr(self, name, value).

__sizeof__() → int

size of object in memory, in bytes

__str__()

Returns string representation, used mainly for logging and debugging

add_cycle (nodes, ***attr*)

Add a cycle.

nodes: iterable container A container of nodes. A cycle will be constructed from the nodes (in order) and added to the graph.

attr [keyword arguments, optional (default= no attributes)] Attributes to add to every edge in cycle.

add_path, add_star

```

>>> G=nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_cycle([0,1,2,3])
>>> G.add_cycle([10,11,12],weight=7)

```

add_edge (u, v, k=None, attr_dict=None, ***attrs*)

add an edge to graph :return: edge data from created or existing edge

add_edges_from (ebunch, attr_dict=None, ***attr*)

Add all the edges in ebunch.

ebunch [container of edges] Each edge given in the container will be added to the graph. The edges can be:

- 2-tuples (u,v) or
- 3-tuples (u,v,d) for an edge attribute dict d, or

- 4-tuples (u,v,k,d) for an edge identified by key k

attr_dict [dictionary, optional (default= no attributes)] Dictionary of edge attributes. Key/value pairs will update existing data associated with each edge.

attr [keyword arguments, optional] Edge data (or labels or objects) can be assigned using keyword arguments.

`add_edge` : add a single edge `add_weighted_edges_from` : convenient way to add weighted edges

Adding the same edge twice has no effect but any edge data will be updated when each duplicate edge is added.

Edge attributes specified in edges take precedence over attributes specified generally.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edges_from([(0,1), (1,2)]) # using a list of edge tuples
>>> e = zip(range(0,3), range(1,4))
>>> G.add_edges_from(e) # Add the path graph 0-1-2-3
```

Associate data to edges

```
>>> G.add_edges_from([(1,2), (2,3)], weight=3)
>>> G.add_edges_from([(3,4), (1,4)], label='WN2898')
```

add_node (*p*, *attr_dict=None*, ***attr*)

add a node or return one already very close :return (x,y,...) node id

add_nodes_from (*nodes*, ***attr*)

must be here because Graph.add_nodes_from doesn't call add_node cleanly as it should...

add_path (*nodes*, ***attr*)

Add a path.

nodes [iterable container] A container of nodes. A path will be constructed from the nodes (in order) and added to the graph.

attr [keyword arguments, optional (default= no attributes)] Attributes to add to every edge in path.

`add_star`, `add_cycle`

```
>>> G=nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.add_path([10,11,12],weight=7)
```

add_star (*nodes*, ***attr*)

Add a star.

The first node in nodes is the middle of the star. It is connected to all other nodes.

nodes [iterable container] A container of nodes.

attr [keyword arguments, optional (default= no attributes)] Attributes to add to every edge in star.

`add_path`, `add_cycle`

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_star([0,1,2,3])
>>> G.add_star([10,11,12],weight=2)
```

add_weighted_edges_from (*ebunch*, *weight='weight'*, ***attr*)

Add all the edges in ebunch as weighted edges with specified weights.

ebunch [container of edges] Each edge given in the list or container will be added to the graph. The edges must be given as 3-tuples (u,v,w) where w is a number.

weight [string, optional (default= 'weight')] The attribute name for the edge weights to be added.

attr [keyword arguments, optional (default= no attributes)] Edge attributes to add/update for all edges.

add_edge : add a single edge add_edges_from : add multiple edges

Adding the same edge twice for Graph/DiGraph simply updates the edge data. For Multi-Graph/MultiDiGraph, duplicate edges are stored.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_weighted_edges_from([(0,1,3.0), (1,2,7.5)])
```

adjacency_iter()

Return an iterator of (node, adjacency dict) tuples for all nodes.

This is the fastest way to look at every edge. For directed graphs, only outgoing adjacencies are included.

adj_iter [iterator] An iterator of (node, adjacency dictionary) for all nodes in the graph.

adjacency_list

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> [(n,nbrdict) for n,nbrdict in G.adjacency_iter()]
[(0, {1: {}}), (1, {0: {}, 2: {}}), (2, {1: {}, 3: {}}), (3, {2: {}})]
```

adjacency_list()

Return an adjacency list representation of the graph.

The output adjacency list is in the order of G.nodes(). For directed graphs, only outgoing adjacencies are included.

adj_list [lists of lists] The adjacency structure of the graph as a list of lists.

adjacency_iter

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.adjacency_list() # in order given by G.nodes()
[[1], [0, 2], [1, 3], [2]]
```

adjlist_dict_factory

alias of dict

box()

Returns nodes bounding box as (xmin,ymin,...),(xmax,ymax,...)

box_size()

Returns (x,y) size

clear()

closest_edges (p, data=False)

Returns container of edges close to p and distance

closest_nodes (p, n=1, skip=False)

nodes closest to a given position :param p: (x,y) position tuple :param skip: optional bool to skip n itself :return: list of nodes, minimal distance

contiguity (*pts*)

Returns int number of points from pts already in graph

copy ()

Returns copy of self graph

degree (*nbunch=None, weight=None*)

Return the degree of a node or nodes.

The node degree is the number of edges adjacent to that node.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights adjacent to the node.

nd [dictionary, or number] A dictionary with nodes as keys and degree as values or a number if a single node is specified.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.degree(0)
1
>>> G.degree([0,1])
{0: 1, 1: 2}
>>> list(G.degree([0,1]).values())
[1, 2]
```

degree_iter (*nbunch=None, weight=None*)

Return an iterator for (node, degree).

The node degree is the number of edges adjacent to the node.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights.

nd_iter [an iterator] The iterator returns two-tuples of (node, degree).

degree

```
>>> G = nx.MultiDiGraph()
>>> G.add_path([0,1,2,3])
>>> list(G.degree_iter(0)) # node 0 with degree 1
[(0, 1)]
>>> list(G.degree_iter([0,1]))
[(0, 1), (1, 2)]
```

dist (*u, v*)

Returns float distance between nodes u and v

draw (***kwargs*)

draw graph with default params

edge_attr_dict_factoryalias of `dict`**edge_key_dict_factory**alias of `dict`**edges** (*nbunch=None, data=False, keys=False, default=None*)

Return a list of edges.

Edges are returned as tuples with optional data and keys in the order (node, neighbor, key, data).

nbunch [iterable container, optional (default= all nodes)] A container of nodes. The container will be iterated through once.**data** [bool, optional (default=False)] Return two tuples (u,v) (False) or three-tuples (u,v,data) (True).**keys** [bool, optional (default=False)] Return two tuples (u,v) (False) or three-tuples (u,v,key) (True).**edge_list: list of edge tuples** Edges that are adjacent to any node in nbunch, or a list of all edges if nbunch is not specified.`edges_iter` : return an iterator over the edges

Nodes in nbunch that are not in the graph will be (quietly) ignored. For directed graphs this returns the out-edges.

```

>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2])
>>> G.add_edge(2,3,weight=5)
>>> G.edges()
[(0, 1), (1, 2), (2, 3)]
>>> G.edges(data=True) # default edge data is {} (empty dictionary)
[(0, 1, {}), (1, 2, {}), (2, 3, {'weight': 5})]
>>> list(G.edges_iter(data='weight', default=1))
[(0, 1, 1), (1, 2, 1), (2, 3, 5)]
>>> G.edges(keys=True) # default keys are integers
[(0, 1, 0), (1, 2, 0), (2, 3, 0)]
>>> G.edges(data=True,keys=True) # default keys are integers
[(0, 1, 0, {}), (1, 2, 0, {}), (2, 3, 0, {'weight': 5})]
>>> list(G.edges(data='weight',default=1,keys=True))
[(0, 1, 0, 1), (1, 2, 0, 1), (2, 3, 0, 5)]
>>> G.edges([0,3])
[(0, 1), (3, 2)]
>>> G.edges(0)
[(0, 1)]

```

edges_iter (*nbunch=None, data=False, keys=False, default=None*)

Return an iterator over the edges.

Edges are returned as tuples with optional data and keys in the order (node, neighbor, key, data).

nbunch [iterable container, optional (default= all nodes)] A container of nodes. The container will be iterated through once.**data** [string or bool, optional (default=False)] The edge attribute returned in 3-tuple (u,v,ddict[data]). If True, return edge attribute dict in 3-tuple (u,v,ddict). If False, return 2-tuple (u,v).**keys** [bool, optional (default=False)] If True, return edge keys with each edge.**default** [value, optional (default=None)] Value used for edges that dont have the requested attribute. Only relevant if data is not True or False.

edge_iter [iterator] An iterator of (u,v), (u,v,d) or (u,v,key,d) tuples of edges.

edges : return a list of edges

Nodes in nbunch that are not in the graph will be (quietly) ignored. For directed graphs this returns the out-edges.

```
>>> G = nx.MultiDiGraph()
>>> G.add_path([0,1,2])
>>> G.add_edge(2,3,weight=5)
>>> [e for e in G.edges_iter()]
[(0, 1), (1, 2), (2, 3)]
>>> list(G.edges_iter(data=True)) # default data is {} (empty dict)
[(0, 1, {}), (1, 2, {}), (2, 3, {'weight': 5})]
>>> list(G.edges_iter(data='weight', default=1))
[(0, 1, 1), (1, 2, 1), (2, 3, 5)]
>>> list(G.edges(keys=True)) # default keys are integers
[(0, 1, 0), (1, 2, 0), (2, 3, 0)]
>>> list(G.edges(data=True,keys=True)) # default keys are integers
[(0, 1, 0, {}), (1, 2, 0, {}), (2, 3, 0, {'weight': 5})]
>>> list(G.edges(data='weight',default=1,keys=True))
[(0, 1, 0, 1), (1, 2, 0, 1), (2, 3, 0, 5)]
>>> list(G.edges_iter([0,2]))
[(0, 1), (2, 3)]
>>> list(G.edges_iter(0))
[(0, 1)]
```

get_edge_data (u, v, key=None, default=None)

Return the attribute dictionary associated with edge (u,v).

u, v : nodes

default [any Python object (default=None)] Value to return if the edge (u,v) is not found.

key [hashable identifier, optional (default=None)] Return data only for the edge with specified key.

edge_dict [dictionary] The edge attribute dictionary.

It is faster to use G[u][v][key].

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_edge(0,1,key='a',weight=7)
>>> G[0][1]['a'] # key='a'
{'weight': 7}
```

Warning: Assigning G[u][v][key] corrupts the graph data structure. But it is safe to assign attributes to that dictionary,

```
>>> G[0][1]['a']['weight'] = 10
>>> G[0][1]['a']['weight']
10
>>> G[1][0]['a']['weight']
10
```

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2,3])
>>> G.get_edge_data(0,1)
{0: {}}
>>> e = (0,1)
```

```
>>> G.get_edge_data(*e) # tuple form
{0: {}}
>>> G.get_edge_data('a','b',default=0) # edge not in graph, return 0
0
```

has_edge (*u*, *v*, *key=None*)

Return True if the graph has an edge between nodes *u* and *v*.

u, v [nodes] Nodes can be, for example, strings or numbers.

key [hashable identifier, optional (default=None)] If specified return True only if the edge with key is found.

edge_ind [bool] True if edge is in the graph, False otherwise.

Can be called either using two nodes *u,v*, an edge tuple (*u,v*), or an edge tuple (*u,v,key*).

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2,3])
>>> G.has_edge(0,1) # using two nodes
True
>>> e = (0,1)
>>> G.has_edge(*e) # e is a 2-tuple (u,v)
True
>>> G.add_edge(0,1,key='a')
>>> G.has_edge(0,1,key='a') # specify key
True
>>> e=(0,1,'a')
>>> G.has_edge(*e) # e is a 3-tuple (u,v,'a')
True
```

The following syntax are equivalent:

```
>>> G.has_edge(0,1)
True
>>> 1 in G[0] # though this gives KeyError if 0 not in G
True
```

has_node (*n*)

Return True if the graph contains the node *n*.

n : node

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
>>> G.has_node(0)
True
```

It is more readable and simpler to use

```
>>> 0 in G
True
```

has_predecessor (*u*, *v*)

Return True if node *u* has predecessor *v*.

This is true if graph has the edge *u*<-*v*.

has_successor (*u*, *v*)

Return True if node *u* has successor *v*.

This is true if graph has the edge *u*->*v*.

html (***kwargs*)

in_degree (*nbunch=None*, *weight=None*)

Return the in-degree of a node or nodes.

The node in-degree is the number of edges pointing in to the node.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights adjacent to the node.

nd [dictionary, or number] A dictionary with nodes as keys and in-degree as values or a number if a single node is specified.

degree, out_degree, in_degree_iter

```
>>> G = nx.DiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2,3])
>>> G.in_degree(0)
0
>>> G.in_degree([0,1])
{0: 0, 1: 1}
>>> list(G.in_degree([0,1]).values())
[0, 1]
```

in_degree_iter (*nbunch=None*, *weight=None*)

Return an iterator for (node, in-degree).

The node in-degree is the number of edges pointing in to the node.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights adjacent to the node.

nd_iter [an iterator] The iterator returns two-tuples of (node, in-degree).

degree, in_degree, out_degree, out_degree_iter

```
>>> G = nx.MultiDiGraph()
>>> G.add_path([0,1,2,3])
>>> list(G.in_degree_iter(0)) # node 0 with degree 0
[(0, 0)]
>>> list(G.in_degree_iter([0,1]))
[(0, 0), (1, 1)]
```

in_edges (*nbunch=None*, *keys=False*, *data=False*)

Return a list of the incoming edges.

nbunch [iterable container, optional (default= all nodes)] A container of nodes. The container will be iterated through once.

data [bool, optional (default=False)] If True, return edge attribute dict with each edge.

keys [bool, optional (default=False)] If True, return edge keys with each edge.

in_edges [list] A list of (u,v), (u,v,d) or (u,v,key,d) tuples of edges.

out_edges: return a list of outgoing edges

in_edges_iter (*nbunch=None, data=False, keys=False*)

Return an iterator over the incoming edges.

nbunch [iterable container, optional (default= all nodes)] A container of nodes. The container will be iterated through once.

data [bool, optional (default=False)] If True, return edge attribute dict with each edge.

keys [bool, optional (default=False)] If True, return edge keys with each edge.

in_edge_iter [iterator] An iterator of (u,v), (u,v,d) or (u,v,key,d) tuples of edges.

edges_iter : return an iterator of edges

is_directed ()

Return True if graph is directed, False otherwise.

is_multigraph ()

used internally in constructor

length (*edges=None*)

Parameters **edges** – iterator over edges either as (u,v,data) or (u,v,key,data). If None, all edges are taken

Returns sum of ‘length’ attributes of edges

multi

name

nbunch_iter (*nbunch=None*)

Return an iterator of nodes contained in nbunch that are also in the graph.

The nodes in nbunch are checked for membership in the graph and if not are silently ignored.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

niter [iterator] An iterator over nodes in nbunch that are also in the graph. If nbunch is None, iterate over all nodes in the graph.

NetworkXError If nbunch is not a node or or sequence of nodes. If a node in nbunch is not hashable.

Graph.__iter__

When nbunch is an iterator, the returned iterator yields values directly from nbunch, becoming exhausted when nbunch is exhausted.

To test whether nbunch is a single node, one can use “if nbunch in self:”, even after processing with this routine.

If nbunch is not a node or a (possibly empty) sequence/iterator or None, a NetworkXError is raised. Also, if any object in nbunch is not hashable, a NetworkXError is raised.

neighbors (*n*)

Return a list of successor nodes of *n*.

`neighbors()` and `successors()` are the same function.

neighbors_iter (*n*)

Return an iterator over successor nodes of *n*.

`neighbors_iter()` and `successors_iter()` are the same.

node_dict_factory

alias of `dict`

nodes (*data=False*)

Return a list of the nodes in the graph.

data [boolean, optional (default=False)] If False return a list of nodes. If True return a two-tuple of node and node data dictionary

nlist [list] A list of nodes. If `data=True` a list of two-tuples containing (node, node data dictionary).

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
>>> G.nodes()
[0, 1, 2]
>>> G.add_node(1, time='5pm')
>>> G.nodes(data=True)
[(0, {}), (1, {'time': '5pm'}), (2, {})]
```

nodes_iter (*data=False*)

Return an iterator over the nodes.

data [boolean, optional (default=False)] If False the iterator returns nodes. If True return a two-tuple of node and node data dictionary

niter [iterator] An iterator over nodes. If `data=True` the iterator gives two-tuples containing (node, node data, dictionary)

If the node data is not required it is simpler and equivalent to use the expression ‘for *n* in *G*’.

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
```

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
```

```
>>> [d for n,d in G.nodes_iter(data=True)]
[{}, {}, {}]
```

nodes_with_selfloops ()

Return a list of nodes with self loops.

A node with a self loop has an edge with both ends adjacent to that node.

nodelist [list] A list of nodes with self loops.

`selfloop_edges`, `number_of_selfloops`


```

>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge(1,1)
>>> G.add_edge(1,2)
>>> G.nodes_with_selfloops()
[1]

```

number_of_edges (*u=None, v=None*)

Return the number of edges between two nodes.

u, v [nodes, optional (default=all edges)] If u and v are specified, return the number of edges between u and v. Otherwise return the total number of all edges.

nedges [int] The number of edges in the graph. If nodes u and v are specified return the number of edges between those nodes.

size

```

>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.number_of_edges()
3
>>> G.number_of_edges(0,1)
1
>>> e = (0,1)
>>> G.number_of_edges(*e)
1

```

number_of_nodes (*doublecheck=False*)

number_of_selfloops ()

Return the number of selfloop edges.

A selfloop edge has the same node at both ends.

nloops [int] The number of selfloops.

nodes_with_selfloops, selfloop_edges

```

>>> G=nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge(1,1)
>>> G.add_edge(1,2)
>>> G.number_of_selfloops()
1

```

order ()

Return the number of nodes in the graph.

nnodes [int] The number of nodes in the graph.

number_of_nodes, __len__ which are identical

out_degree (*nbunch=None, weight=None*)

Return the out-degree of a node or nodes.

The node out-degree is the number of edges pointing out of the node.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights adjacent to the node.

nd [dictionary, or number] A dictionary with nodes as keys and out-degree as values or a number if a single node is specified.

```
>>> G = nx.DiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2,3])
>>> G.out_degree(0)
1
>>> G.out_degree([0,1])
{0: 1, 1: 1}
>>> list(G.out_degree([0,1]).values())
[1, 1]
```

out_degree_iter (*nbunch=None, weight=None*)

Return an iterator for (node, out-degree).

The node out-degree is the number of edges pointing out of the node.

nbunch [iterable container, optional (default=all nodes)] A container of nodes. The container will be iterated through once.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1. The degree is the sum of the edge weights.

nd_iter [an iterator] The iterator returns two-tuples of (node, out-degree).

degree, in_degree, out_degree, in_degree_iter

```
>>> G = nx.MultiDiGraph()
>>> G.add_path([0,1,2,3])
>>> list(G.out_degree_iter(0)) # node 0 with degree 1
[(0, 1)]
>>> list(G.out_degree_iter([0,1]))
[(0, 1), (1, 1)]
```

out_edges (*nbunch=None, keys=False, data=False*)

Return a list of the outgoing edges.

Edges are returned as tuples with optional data and keys in the order (node, neighbor, key, data).

nbunch [iterable container, optional (default= all nodes)] A container of nodes. The container will be iterated through once.

data [bool, optional (default=False)] If True, return edge attribute dict with each edge.

keys [bool, optional (default=False)] If True, return edge keys with each edge.

out_edges [list] An list of (u,v), (u,v,d) or (u,v,key,d) tuples of edges.

Nodes in nbunch that are not in the graph will be (quietly) ignored. For directed graphs edges() is the same as out_edges().

in_edges: return a list of incoming edges

out_edges_iter (*nbunch=None, data=False, keys=False, default=None*)

Return an iterator over the edges.

Edges are returned as tuples with optional data and keys in the order (node, neighbor, key, data).

nbunch [iterable container, optional (default= all nodes)] A container of nodes. The container will be iterated through once.

data [string or bool, optional (default=False)] The edge attribute returned in 3-tuple (u,v,ddict[data]). If True, return edge attribute dict in 3-tuple (u,v,ddict). If False, return 2-tuple (u,v).

keys [bool, optional (default=False)] If True, return edge keys with each edge.

default [value, optional (default=None)] Value used for edges that dont have the requested attribute. Only relevant if data is not True or False.

edge_iter [iterator] An iterator of (u,v), (u,v,d) or (u,v,key,d) tuples of edges.

edges : return a list of edges

Nodes in nbunch that are not in the graph will be (quietly) ignored. For directed graphs this returns the out-edges.

```
>>> G = nx.MultiDiGraph()
>>> G.add_path([0,1,2])
>>> G.add_edge(2,3,weight=5)
>>> [e for e in G.edges_iter()]
[(0, 1), (1, 2), (2, 3)]
>>> list(G.edges_iter(data=True)) # default data is {} (empty dict)
[(0, 1, {}), (1, 2, {}), (2, 3, {'weight': 5})]
>>> list(G.edges_iter(data='weight', default=1))
[(0, 1, 1), (1, 2, 1), (2, 3, 5)]
>>> list(G.edges(keys=True)) # default keys are integers
[(0, 1, 0), (1, 2, 0), (2, 3, 0)]
>>> list(G.edges(data=True,keys=True)) # default keys are integers
[(0, 1, 0, {}), (1, 2, 0, {}), (2, 3, 0, {'weight': 5})]
>>> list(G.edges(data='weight',default=1,keys=True))
[(0, 1, 0, 1), (1, 2, 0, 1), (2, 3, 0, 5)]
>>> list(G.edges_iter([0,2]))
[(0, 1), (2, 3)]
>>> list(G.edges_iter(0))
[(0, 1)]
```

plot (**kwargs)

renders on IPython Notebook (alias to make usage more straightforward)

png (**kwargs)

pos (nodes=None)

Parameters nodes – a single node, an iterator of all nodes if None

Returns the position of node(s)

predecessors (n)

Return a list of predecessor nodes of n.

predecessors_iter (n)

Return an iterator over predecessor nodes of n.

remove_edge (u, v=None, key=None, clean=False)

Parameters

- u – Node or Edge (Nodes tuple)

- **v** – Node if u is a single Node
- **clean** – bool removes disconnected nodes. must be False for certain nx algos to work

Result return attributes of removed edge

remove edge from graph. NetworkX graphs do not remove unused nodes

remove_edges_from (*ebunch*)

Remove all edges specified in ebunch.

ebunch: list or container of edge tuples Each edge given in the list or container will be removed from the graph. The edges can be:

- 2-tuples (u,v) All edges between u and v are removed.
- 3-tuples (u,v,key) The edge identified by key is removed.
- 4-tuples (u,v,key,data) where data is ignored.

remove_edge : remove a single edge

Will fail silently if an edge in ebunch is not in the graph.

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_path([0,1,2,3])
>>> ebunch=[(1,2), (2,3)]
>>> G.remove_edges_from(ebunch)
```

Removing multiple copies of edges

```
>>> G = nx.MultiGraph()
>>> G.add_edges_from([(1,2), (1,2), (1,2)])
>>> G.remove_edges_from([(1,2), (1,2)])
>>> G.edges()
[(1, 2)]
>>> G.remove_edges_from([(1,2), (1,2)]) # silently ignore extra copy
>>> G.edges() # now empty graph
[]
```

remove_node (*n*)

Parameters **n** – node tuple

remove node from graph and rtree

remove_nodes_from (*nbunch*)

Remove multiple nodes.

nodes [iterable container] A container of nodes (list, dict, set, etc.). If a node in the container is not in the graph it is silently ignored.

remove_node

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2])
>>> e = G.nodes()
>>> e
[0, 1, 2]
>>> G.remove_nodes_from(e)
>>> G.nodes()
[]
```

render (*fmt='svg', **kwargs*)

render graph to bitmap stream :param *fmt*: string defining the format. 'svg' by default for INotepads
:return: matplotlib figure as a byte stream in specified format

reverse (*copy=True*)

Return the reverse of the graph.

The reverse is a graph with the same nodes and edges but with the directions of the edges reversed.

copy [bool optional (default=True)] If True, return a new DiGraph holding the reversed edges. If False, reverse the reverse graph is created using the original graph (this changes the original graph).

save (*filename, **kwargs*)

save graph in various formats

selfloop_edges (*data=False, keys=False, default=None*)

Return a list of selfloop edges.

A selfloop edge has the same node at both ends.

data [bool, optional (default=False)] Return selfloop edges as two tuples (u,v) (*data=False*) or three-tuples (u,v,datadict) (*data=True*) or three-tuples (u,v,datavalue) (*data='attrname'*)

default [value, optional (default=None)] Value used for edges that dont have the requested attribute. Only relevant if *data* is not True or False.

keys [bool, optional (default=False)] If True, return edge keys with each edge.

edgelist [list of edge tuples] A list of all selfloop edges.

nodes_with_selfloops, number_of_selfloops

```
>>> G = nx.MultiGraph() # or MultiDiGraph
>>> G.add_edge(1,1)
>>> G.add_edge(1,2)
>>> G.selfloop_edges()
[(1, 1)]
>>> G.selfloop_edges(data=True)
[(1, 1, {})]
>>> G.selfloop_edges(keys=True)
[(1, 1, 0)]
>>> G.selfloop_edges(keys=True, data=True)
[(1, 1, 0, {})]
```

shortest_path (*source=None, target=None*)

size (*weight=None*)

Return the number of edges.

weight [string or None, optional (default=None)] The edge attribute that holds the numerical value used as a weight. If None, then each edge has weight 1.

nedges [int] The number of edges or sum of edge weights in the graph.

number_of_edges

```
>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> G.size()
3
```

```

>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_edge('a', 'b', weight=2)
>>> G.add_edge('b', 'c', weight=4)
>>> G.size()
2
>>> G.size(weight='weight')
6.0

```

stats()

Returns dict of graph data to use in `__repr__` or usable otherwise

subgraph (*nbunch*)

Return the subgraph induced on nodes in *nbunch*.

The induced subgraph of the graph contains the nodes in *nbunch* and the edges between those nodes.

nbunch [list, iterable] A container of nodes which will be iterated through once.

G [Graph] A subgraph of the graph with the same edge attributes.

The graph, edge or node attributes just point to the original graph. So changes to the node or edge structure will not be reflected in the original graph while changes to the attributes will.

To create a subgraph with its own copy of the edge/node attributes use: `nx.Graph(G.subgraph(nbunch))`

If edge attributes are containers, a deep copy can be obtained using: `G.subgraph(nbunch).copy()`

For an inplace reduction of a graph to a subgraph you can remove nodes: `G.remove_nodes_from([n in G if n not in set(nbunch)])`

```

>>> G = nx.Graph() # or DiGraph, MultiGraph, MultiDiGraph, etc
>>> G.add_path([0,1,2,3])
>>> H = G.subgraph([0,1,2])
>>> H.edges()
[(0, 1), (1, 2)]

```

successors (*n*)

Return a list of successor nodes of *n*.

`neighbors()` and `successors()` are the same function.

successors_iter (*n*)

Return an iterator over successor nodes of *n*.

`neighbors_iter()` and `successors_iter()` are the same.

svg (***kwargs*)

to_directed ()

Return a directed copy of the graph.

G [MultiDiGraph] A deepcopy of the graph.

If edges in both directions (*u,v*) and (*v,u*) exist in the graph, attributes for the new undirected edge will be a combination of the attributes of the directed edges. The edge data is updated in the (arbitrary) order that the edges are encountered. For more customized control of the edge attributes use `add_edge()`.

This returns a “deepcopy” of the edge, node, and graph attributes which attempts to completely copy all of the data and references.

This is in contrast to the similar `G=DiGraph(D)` which returns a shallow copy of the data.

See the Python copy module for more information on shallow and deep copies, <http://docs.python.org/library/copy.html>.

```
>>> G = nx.Graph() # or MultiGraph, etc
>>> G.add_path([0,1])
>>> H = G.to_directed()
>>> H.edges()
[(0, 1), (1, 0)]
```

If already directed, return a (deep) copy

```
>>> G = nx.MultiDiGraph()
>>> G.add_path([0,1])
>>> H = G.to_directed()
>>> H.edges()
[(0, 1)]
```

to_undirected (*reciprocal=False*)

Return an undirected representation of the digraph.

reciprocal [bool (optional)] If True only keep edges that appear in both directions in the original digraph.

G [MultiGraph] An undirected graph with the same name and nodes and with edge (u,v,data) if either (u,v,data) or (v,u,data) is in the digraph. If both edges exist in digraph and their edge data is different, only one edge is created with an arbitrary choice of which edge data to use. You must check and correct for this manually if desired.

This returns a “deepcopy” of the edge, node, and graph attributes which attempts to completely copy all of the data and references.

This is in contrast to the similar `D=DiGraph(G)` which returns a shallow copy of the data.

See the Python copy module for more information on shallow and deep copies, <http://docs.python.org/library/copy.html>.

Warning: If you have subclassed MultiGraph to use dict-like objects in the data structure, those changes do not transfer to the MultiDiGraph created by this method.

tol

`Goulib.graph.figure` (*g, box=None, **kwargs*)

Parameters

- **g** – _Geo derived Graph
- **box** – optional interval.Box if g has no box

Returns matplotlib axis suitable for drawing graph g

`Goulib.graph.draw_networkx` (*g, pos=None, **kwargs*)

improves `nx.draw_networkx` :param g: NetworkX Graph :param pos: can be either :

- optional dictionary of (x,y) node positions
- function of the form `lambda node:(x,y)` that maps node positions.
- None. in this case, nodes are directly used as positions if graph is a GeoGraph, otherwise `nx.draw_shell` is used

Parameters ****kwargs** – passed to `nx.draw` method as described in http://networkx.lanl.gov/reference/generated/networkx.drawing.nx_pylab.draw_networkx.html with one tweak:

- if `edge_color` is a function of the form `lambda data:color string`, it is mapped over all edges

`Goulib.graph.to_drawing(g, d=None, edges=[])`

draws `Graph` to a `Drawing` :param `g`: `Graph` :param `d`: existing `Drawing` to draw onto, or `None` to create a new `Drawing` :param `edges`: iterable of edges (with data) that will be added, in the same order. By default all edges are drawn :return: `Drawing`

Graph edges with an 'entity' property

`Goulib.graph.write_dxf(g, filename)`

writes `networkx.Graph` in `.dxf` format

`Goulib.graph.write_dot(g, filename)`

`Goulib.graph.to_json(g, **kwargs)`

Returns string JSON representation of a graph

`Goulib.graph.write_json(g, filename, **kwargs)`

write a JSON file, suitable for `D*.js` representation

`Goulib.graph.delauney_triangulation(nodes, qhull_options='', incremental=False, **kwargs)`

https://en.wikipedia.org/wiki/Delaunay_triangulation :param `nodes`: `_Geo` graph or list of (x,y) or (x,y,z) node positions :param `qhull_options`: string passed to `scipy.spatial.Delaunay()`, which passes it to `Qhull` (<http://www.qhull.org/>) *'Qt' ensures all points are connected *'Qz' required when nodes lie on a sphere *'QJ' solves some singularity situations

Parameters `kwargs` – passed to the `GeoGraph` constructor

Returns `GeoGraph` with delauney triangulation between nodes

`Goulib.graph.euclidean_minimum_spanning_tree(nodes, **kwargs)`

Parameters `nodes` – list of (x,y) nodes positions

Returns `GeoGraph` with minimum spanning tree between nodes

see https://en.wikipedia.org/wiki/Euclidean_minimum_spanning_tree

`Goulib.graph.points_on_sphere(N)`

Goulib.image module

class `Goulib.image.Mode(name, nchannels, type, min, max)`

Bases: `object`

`__init__`(name, nchannels, type, min, max)

`__repr__`()

`__class__`

alias of `type`

`__delattr__`

Implement `delattr(self, name)`.

`__dir__`() → list

default `dir()` implementation

`__eq__`

Return `self==value`.


```

__format__ ()
    default object formatter

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

__str__
    Return str(self).

```

Goulib.image.**nchannels** (*arr*)

Goulib.image.**guessmode** (*arr*)

Goulib.image.**adapt_rgb** (*func*)

Decorator that adapts to RGB(A) images to a gray-scale filter. :param apply_to_rgb: function

Function that returns a filtered image from an image-filter and RGB image. This will only be called if the image is RGB-like.

class Goulib.image.**Image** (*data=None, mode=None, **kwargs*)

Bases: *Goulib.plot.Plot*

Parameters **data** – can be either:

- *PIL.Image* : makes a copy
- string : path of image to load
- None : creates an empty image with kwargs parameters:

** size : (y,x) pixel size tuple ** mode : 'F' (gray) by default ** color: to fill None=black by default ** colormap: Palette or matplotlib colormap

`__init__` (*data=None, mode=None, **kwargs*)

Parameters **data** – can be either:

- *PIL.Image* : makes a copy
- string : path of image to load
- None : creates an empty image with kwargs parameters:

** size : (y,x) pixel size tuple ** mode : 'F' (gray) by default ** color: to fill None=black by default ** colormap: Palette or matplotlib colormap

shape

size

nchannels

npixels

`__nonzero__` ()

`__lt__` (*other*)
is smaller

load (*path*)

save (*path, autoconvert=True, format_str=None, **kwargs*)

saves an image :param path: string with path/filename.ext :param autoconvert: bool, if True converts color planes formats to RGB :param format_str: str of file format. set to 'PNG' by `skimage.io.imsave` :param kwargs: optional params passed to `skimage.io.imsave` :return: self for chaining

render (*fmt='PNG', **kwargs*)

static open (*path*)

PIL(low) compatibility

static new (*mode, size, color='black'*)

PIL(low) compatibility

pil

convert to PIL(low) Image :see: <http://effbot.org/imagingbook/concepts.htm>

getdata (*dtype=<class 'numpy.uint8'>, copy=True*)

split (*mode=None*)

getpixel (*yx*)

putpixel (*yx, value*)

getpalette (*maxcolors=256*)

setpalette (*p*)

getcolors (*maxcolors=256*)

Returns an unsorted list of (count, color) tuples,

where count is the number of times the corresponding color occurs in the image. If the maxcolors value is exceeded, the method stops counting and returns None. The default maxcolors value is 256. To make sure

you get all colors in an image, you can pass in `size[0]*size[1]` (but make sure you have lots of memory before you do that on huge images).

replace (*pairs*)

replace a color by another currently works only for indexed color images :param pairs: iterable of (from,to) ints

optimize (*maxcolors=256*)

remove unused colors from the palette

crop (*lurb*)

Parameters `lurb` – 4-tuple with left,up,right,bottom int coordinates

Returns Image

__getitem__ (*slice*)

resize (*size, filter=None, **kwargs*)

Returns a resized copy of an image.

Parameters

- **size** – int tuple (width, height) requested size in pixels
- **filter** –
 - NEAREST (use nearest neighbour),
 - BILINEAR (linear interpolation in a 2x2 environment),
 - BICUBIC (cubic spline interpolation in a 4x4 environment)
 - ANTIALIAS (a high-quality downsampling filter)
- **kwargs** – astra parameters passed to `skimage.transform.resize`

paste (*image, box, mask=None*)

Pastes another image into this image.

Parameters

- **image** – image to paste, or color given as a single numerical value for single-band images, and a tuple for multi-band images.
- **box** – 2-tuple giving the upper left corner or 4-tuple defining the left, upper, right, and lower pixel coordinate, or None (same as (0, 0)). If a 4-tuple is given, the size of the pasted image must match the size of the region.

:param mask:optional image to update only the regions indicated by the mask. You can use either “1”, “L” or “RGBA” images (in the latter case, the alpha band is used as mask). Where the mask is 255, the given image is copied as is. Where the mask is 0, the current value is preserved. Intermediate values can be used for transparency effects. Note that if you paste an “RGBA” image, the alpha band is ignored. You can work around this by using the same image as both source image and mask.

threshold (*level=None*)

quantize (*colors=256, method=None, kmeans=0, palette=None*)

(PIL.Image compatible) Convert the image to ‘P’ mode with the specified number of colors. :param colors: The desired number of colors, <= 256 :param method: 0 = median cut

1 = maximum coverage 2 = fast octree 3 = libimagequant

Parameters

- **kmeans** – Integer
- **palette** – Quantize to the `PIL.ImagingPalette` palette.

Returns A new image

convert (*mode*, ***kwargs*)

convert image mode :param mode: string destination mode :param kwargs: optional params passed to converter(s). can contain: * **palette** : to force using a palette instead of the image's one for indexed images
:return: image in desired mode

__repr__ ()

average_hash (*hash_size=8*)

Average Hash computation Implementation follows <http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html>

Parameters **hash_size** – int sqrt of the hash size. 8 (64 bits) is perfect for usual photos

Returns list of hash_size*hash_size bool (=bits)

dist (*other*, *hash_size=8*)

distance between images

Parameters **hash_size** – int sqrt of the hash size. 8 (64 bits) is perfect for usual photos

Returns float =0 if images are equal or very similar (same **average_hash**) =1 if images are completely decorrelated (half of the hash bits are the same by luck) =2 if images are inverted

__hash__ ()

__abs__ ()

Returns float Frobenius norm of image

invert ()

__neg__ ()

__inv__ ()

grayscale (*mode=None*)

convert (color) to grayscale :param mode: string target mode (should be in 'FUIL') or automatic if none

colorize (*color0*, *color1=None*)

colorize a grayscale image

Parameters **color0**, **color1** – 2 colors. - If only one is specified, image is colorized from white (for 0) to the specified color (for 1) - if 2 colors are specified, image is colorized from color0 (for 0) to color1 (for 1)

Returns RGB(A) color

dither (*method=None*, *n=2*)

normalize (*newmax=None*, *newmin=None*)

filter (*f*)

correlation (*other*)

Compute the correlation between two, single-channel, grayscale input images. The second image must be smaller than the first. :param other: the Image we're looking for

scale (*s*)

resize image by factor s

Parameters **s** – (sx, sy) tuple of float scaling factor, or scalar s=sx=sy

Returns Image scaled

shift (*dx, dy, **kwargs*)

expand (*size, ox=None, oy=None*)

Returns image in larger canvas size, pasted at ox,oy

compose (*other, a=0.5, b=0.5, mode=None*)
compose new image from a*self + b*other

add (*other, pos=(0, 0), alpha=1, mode=None*)
simply adds other image at px,py (subbixel) coordinates

__add__ (*other*)

__radd__ (*other*)
only to allow sum(images) easily

sub (*other, pos=(0, 0), alpha=1, mode=None*)

__sub__ (*other*)

deltaE (*other*)

__mul__ (*other*)

__div__ (*f*)

__truediv__ (*f*)

__class__
alias of `type`

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__le__
Return self<=value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return str(self).

`html (**kwargs)`

`plot (**kwargs)`
renders on IPython Notebook (alias to make usage more straightforward)

`png (**kwargs)`

`svg (**kwargs)`

`Goulib.image.alpha_composite (front, back)`
Alpha composite two RGBA images.

Source: <http://stackoverflow.com/a/9166671/284318>

Keyword Arguments: front – PIL RGBA Image object back – PIL RGBA Image object

The algorithm comes from http://en.wikipedia.org/wiki/Alpha_compositing

`Goulib.image.alpha_composite_with_color (image, color=(255, 255, 255))`
Alpha composite an RGBA image with a single color image of the specified color and the same size as the original image.

Keyword Arguments: image – PIL RGBA Image object color – Tuple r, g, b (default 255, 255, 255)

`Goulib.image.pure_pil_alpha_to_color_v1 (image, color=(255, 255, 255))`
Alpha composite an RGBA Image with a specified color.

NOTE: This version is much slower than the alpha_composite_with_color solution. Use it only if numpy is not available.

Source: <http://stackoverflow.com/a/9168169/284318>

Keyword Arguments: image – PIL RGBA Image object color – Tuple r, g, b (default 255, 255, 255)

`Goulib.image.pure_pil_alpha_to_color_v2 (image, color=(255, 255, 255))`
Alpha composite an RGBA Image with a specified color.

Simpler, faster version than the solutions above.

Source: <http://stackoverflow.com/a/9459208/284318>

Keyword Arguments: image – PIL RGBA Image object color – Tuple r, g, b (default 255, 255, 255)

`Goulib.image.disk (radius, antialias=1)`

`Goulib.image.fspecial (name, **kwargs)`
mimics the Matlab image toolbox fspecial function <http://www.mathworks.com/help/images/ref/fspecial.html?refresh=true>

`Goulib.image.normalize (a, newmax=255, newmin=0)`

`Goulib.image.read_pdf(filename, **kwargs)`

reads a bitmap graphics on a .pdf file only the first page is parsed

`Goulib.image.fig2img(fig)`

Convert a Matplotlib figure to a PIL Image in RGBA format and return it

Parameters `fig` – matplotlib figure

Returns PIL image

`Goulib.image.quantize(image, N=2, L=None)`

Quantize a gray image. :param image: ndarray input image. :param N: int number of quantization levels. :param L: float max value.

`Goulib.image.randomize(image, N=2, L=None)`

class `Goulib.image.Ditherer(name, method)`

Bases: `object`

`__init__(name, method)`

`__call__(image, N=2)`

`__repr__()`

`__class__`
alias of `type`

`__delattr__`
Implement `delattr(self, name)`.

`__dir__()` → list
default `dir()` implementation

`__eq__`
Return `self==value`.

`__format__()`
default object formatter

`__ge__`
Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self>value`.

`__hash__`
Return `hash(self)`.

`__le__`
Return `self<=value`.

`__lt__`
Return `self<value`.

`__ne__`
Return `self!=value`.

`__new__()`
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__setattr__`
Implement `setattr(self, name, value)`.

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return `str(self)`.

class `Goulib.image.ErrorDiffusion` (*name, positions, weights, wsum=None*)

Bases: `Goulib.image.Ditherer`

`__init__` (*name, positions, weights, wsum=None*)

`__call__` (*image, N=2*)

`__class__`
alias of `type`

`__delattr__`
Implement `delattr(self, name)`.

`__dir__()` → list
default `dir()` implementation

`__eq__`
Return `self==value`.

`__format__()`
default object formatter

`__ge__`
Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self>value`.

`__hash__`
Return `hash(self)`.

`__le__`
Return `self<=value`.

`__lt__`
Return `self<value`.

`__ne__`
Return `self!=value`.

`__new__()`
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__()`
helper for pickle

`__reduce_ex__()`
 helper for pickle

`__repr__()`

`__setattr__`
 Implement setattr(self, name, value).

`__sizeof__()` → int
 size of object in memory, in bytes

`__str__`
 Return str(self).

class `Goulib.image.FloydSteinberg`
 Bases: `Goulib.image.ErrorDiffusion`

`__init__()`

`__call__(image, N=2)`

`__class__`
 alias of `type`

`__delattr__`
 Implement delattr(self, name).

`__dir__()` → list
 default dir() implementation

`__eq__`
 Return self==value.

`__format__()`
 default object formatter

`__ge__`
 Return self>=value.

`__getattr__`
 Return getattr(self, name).

`__gt__`
 Return self>value.

`__hash__`
 Return hash(self).

`__le__`
 Return self<=value.

`__lt__`
 Return self<value.

`__ne__`
 Return self!=value.

`__new__()`
 Create and return a new object. See help(type) for accurate signature.

`__reduce__()`
 helper for pickle

`__reduce_ex__()`
 helper for pickle

`__repr__()`
`__setattr__`
Implement setattr(self, name, value).
`__sizeof__()` → int
size of object in memory, in bytes
`__str__`
Return str(self).

Goulib.image.**dither** (*image, method=3, N=2*)

Quantize a gray image, using dithering. :param image: ndarray input image. :param method: key in dithering dict :param N: int number of quantization levels. References ——— <http://www.efg2.com/Lab/Library/ImageProcessing/DHALF.TXT>

Goulib.image.**gray2bool** (*image, method=3, N=2*)

Quantize a gray image, using dithering. :param image: ndarray input image. :param method: key in dithering dict :param N: int number of quantization levels. References ——— <http://www.efg2.com/Lab/Library/ImageProcessing/DHALF.TXT>

Goulib.image.**rgb2cmyk** (*rgb*)

Goulib.image.**cmyk2rgb** (*cmyk*)

Goulib.image.**gray2rgb** (*im, color0=(0, 0, 0), color1=(1, 1, 1)*)

Goulib.image.**bool2rgb** (*im, color0=(0, 0, 0), color1=(1, 1, 1)*)

Goulib.image.**bool2gray** (*im*)

Goulib.image.**rgb2rgba** (*array*)

Goulib.image.**palette** (*im, ncolors, tol=0.01*)

extract the color palette of image array (in its own colorspace. use Lab for best results) :param im: nparray (x,y,n) containing image :param ncolors: int number of colors :param tol: tolerance for precision/speed compromise. 1/100 means about 100 points per color are taken for kmeans segmentation :return: array of ncolors most used in image (center of kmeans centroids)

Goulib.image.**lab2ind** (*im, colors=256*)

convert a Lab image to indexed colors :param a: nparray (x,y,n) containing image :param colors: int number of colors or predefined Palette :ref: http://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html

Goulib.image.**ind2any** (*im, palette, dest*)

Goulib.image.**ind2rgb** (*im, palette*)

Goulib.image.**convert** (*a, source, target, **kwargs*)

convert an image between modes, eventually using intermediary steps :param a: nparray (x,y,n) containing image :param source: string : key of source image mode in modes :param target: string : key of target image mode in modes

Goulib.interval module

operations on [a..b[intervals

Goulib.interval.**in_interval** (*interval, x, closed=True*)

Returns bool True if x is in interval [a,b] or [b,a] (tuple)

Goulib.interval.**intersect** (*t1, t2*)

Returns bool True if intervals [t1[[t2[intersect

`Goulib.interval.intersection` (*t1*, *t2*)

Returns tuple intersection between 2 intervals (tuples),

or None if intervals don't intersect

`Goulib.interval.intersectlen` (*t1*, *t2*, *none=0*)

Parameters

- **t1** – interval 1 (tuple)
- **t2** – interval 2 (tuple)
- **none** – value to return when t1 does not intersect t2

Returns len of intersection between 2 intervals (tuples),

or none if intervals don't intersect

class `Goulib.interval.Interval` (*start*, *end*)

Bases: list

Represents an interval. Defined as half-open interval [start,end), which includes the start position but not the end. Start and end do not have to be numeric types. They might especially be time, date or timedelta as used in datetime2

inspired from <http://code.activestate.com/recipes/576816-interval/> alternatives could be <https://pypi.python.org/pypi/interval/>

(outdated, no more doc) or <https://pypi.python.org/pypi/pyinterval/>

Construct, start must be <= end.

`__init__` (*start*, *end*)

Construct, start must be <= end.

start

The interval's start

end

The interval's end

`__str__` ()

`__repr__` ()

`__hash__` ()

`__lt__` (*other*)

`__eq__` (*other*)

size

center

separation (*other*)

Returns distance between self and other, negative if overlap

overlap (*other*, *allow_contiguous=False*)

Returns True iff self intersects other.

intersection (*other*)

Returns Intersection with other, or None if no intersection.

`__iadd__` (*other*)

expands self to contain other.

`hull` (*other*)

Returns new Interval containing both self and other.

`__add__` (*other*)

`__contains__` (*x*)

Returns True if *x* in self.

`subset` (*other*)

Returns True iff self is subset of other.

`proper_subset` (*other*)

Returns True iff self is proper subset of other.

`empty` ()

Returns True iff self is empty.

`__nonzero__` ()

`singleton` ()

Returns True iff `self.end - self.start == 1`.

`__class__`

alias of `type`

`__delattr__`

Implement `delattr(self, name)`.

`__delitem__`

Delete `self[key]`.

`__dir__` () → list

default `dir()` implementation

`__format__` ()

default object formatter

`__ge__`

Return `self >= value`.

`__getattr__`

Return `getattr(self, name)`.

`__getitem__` ()

`x.__getitem__(y) <==> x[y]`

`__gt__`

Return `self > value`.

`__imul__`

Implement `self *= value`.

`__iter__`

Implement `iter(self)`.

`__le__`
Return self<=value.

`__len__`
Return len(self).

`__mul__`
Return self*value.n

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__reversed__` ()
L.`__reversed__`() – return a reverse iterator over the list

`__rmul__`
Return self*value.

`__setattr__`
Implement setattr(self, name, value).

`__setitem__`
Set self[key] to value.

`__sizeof__` ()
L.`__sizeof__`() – size of L in memory, in bytes

`append` (*object*) → None – append object to end

`clear` () → None – remove all items from L

`copy` () → list – a shallow copy of L

`count` (*value*) → integer – return number of occurrences of value

`extend` (*iterable*) → None – extend list by appending elements from the iterable

`index` (*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

`insert` ()
L.insert(index, object) – insert object before index

`pop` ([*index*]) → item – remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

`remove` (*value*) → None – remove first occurrence of value.
Raises ValueError if the value is not present.

`reverse` ()
L.reverse() – reverse *IN PLACE*

`sort` (*key=None*, *reverse=False*) → None – stable sort **IN PLACE**

class Goulib.interval.**Intervals** (*iterable=()*, *key=None*)
Bases: *Goulib.container.SortedCollection*

a list of intervals kept in ascending order

__repr__ ()

insert (*item*)

__iadd__ (*item*)

__add__ (*item*)

__call__ (*x*)

returns intervals containing *x*

__class__

alias of `type`

__contains__ (*item*)

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__getitem__ (*i*)

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__init__ (*iterable=()*, *key=None*)

__iter__ ()

__le__

Return `self<=value`.

__len__ ()

__lt__

Return `self<value`.

__ne__

Return `self!=value`.

__new__ ()

Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()

__reduce_ex__ ()

helper for pickle

```

__reversed__ ()
__setattr__
    Implement setattr(self, name, value).
__sizeof__ () → int
    size of object in memory, in bytes
__str__
    Return str(self).
clear ()
copy ()
count (item)
    Return number of occurrences of item
find (k)
    Return first item with a key == k. Raise ValueError if not found.
find_ge (k)
    Return first item with a key >= equal to k. Raise ValueError if not found
find_gt (k)
    Return first item with a key > k. Raise ValueError if not found
find_le (k)
    Return last item with a key <= k. Raise ValueError if not found.
find_lt (k)
    Return last item with a key < k. Raise ValueError if not found.
index (item)
    Find the position of an item. Raise ValueError if not found.
insert_right (item)
    Insert a new item. If equal keys are found, add to the right
key
    key function
pop (i=-1)
remove (item)
    Remove first occurrence of item. Raise ValueError if not found
class Goulib.interval.Box (*args)
    Bases: list
    a N dimensional rectangular box defined by a list of N Intervals
    __init__ (*args)
    corner (n)
        return n-th corner of box 0-th corner is “start” made of all minimal values of intervals -1.th corner is “end”,
        made of all maximal values of intervals
    start
    end
    size
    center

```

`__call__()`

Returns tuple of all intervals as tuples

`__iadd__(other)`

enlarge box if required to contain specified point :param other: *Box* or (list of) N-tuple point(s)

`__add__(other)`

enlarge box if required to contain specified point :param other: *Box* or (list of) N-tuple point(s) :return: new *Box* containing both

`__contains__(other)`

Returns True if x in self.

`__nonzero__()`

`empty()`

Returns True iff *Box* is empty.

`__class__`

alias of `type`

`__delattr__`

Implement `delattr(self, name)`.

`__delitem__`

Delete `self[key]`.

`__dir__()` → list

default `dir()` implementation

`__eq__`

Return `self==value`.

`__format__()`

default object formatter

`__ge__`

Return `self>=value`.

`__getattr__`

Return `getattr(self, name)`.

`__getitem__()`

`x.__getitem__(y) <==> x[y]`

`__gt__`

Return `self>value`.

`__hash__ = None`

`__imul__`

Implement `self*=value`.

`__iter__`

Implement `iter(self)`.

`__le__`

Return `self<=value`.

`__len__`

Return `len(self)`.

__lt__
Return self<value.

__mul__
Return self*value.n

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__reversed__ ()
L.__reversed__() – return a reverse iterator over the list

__rmul__
Return self*value.

__setattr__
Implement setattr(self, name, value).

__setitem__
Set self[key] to value.

__sizeof__ ()
L.__sizeof__() – size of L in memory, in bytes

__str__
Return str(self).

append (*object*) → None – append object to end

clear () → None – remove all items from L

copy () → list – a shallow copy of L

count (*value*) → integer – return number of occurrences of value

extend (*iterable*) → None – extend list by appending elements from the iterable

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

insert ()
L.insert(index, object) – insert object before index

max

min

pop ([*index*]) → item – remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.
Raises ValueError if the value is not present.

reverse ()

L.reverse() – reverse *IN PLACE*

sort (*key=None, reverse=False*) → None – stable sort **IN PLACE**

Goulib.itertools2 module

additions to `itertools` standard library

Goulib.itertools2.**take** (*n, iterable*)

Result first *n* items from iterable

Goulib.itertools2.**index** (*n, iterable*)

Result *nth* item

Goulib.itertools2.**first** (*iterable*)

Result first element in the iterable

Goulib.itertools2.**last** (*iterable*)

Result last element in the iterable

Goulib.itertools2.**takeevery** (*n, iterable, start=0*)

Take an element from iterator every *n* elements

Goulib.itertools2.**every** (*n, iterable, start=0*)

Take an element from iterator every *n* elements

Goulib.itertools2.**drop** (*n, iterable*)

Drop *n* elements from iterable and return the rest

Goulib.itertools2.**ilen** (*it*)

Result int length exhausting an iterator

Goulib.itertools2.**irange** (*start_or_end, optional_end=None*)

Result iterable that counts from start to end (both included).

Goulib.itertools2.**isiterable** (*obj*)

Result bool True if *obj* is iterable (but not a string)

Goulib.itertools2.**iscallable** (*f*)

Goulib.itertools2.**enumerates** (*iterable*)

generalizes enumerate to dicts :result: key,value pair for whatever iterable type

Goulib.itertools2.**arange** (*start, stop=None, step=1*)

range for floats or other types (*numpy.arange* without numpy)

Parameters

- **start** – optional number. Start of interval. The interval includes this value. The default start value is 0.
- **stop** – number. End of interval. The interval does not include this value, except in some cases where step is not an integer and floating point round-off affects the length of out.
- **step** – optional number. Spacing between values. For any output out, this is the distance between two adjacent values, out[i+1] - out[i]. The default step size is 1.

Result iterator

`Goulib.itertools2.linspace` (*start, end, n=100*)

iterator over *n* values linearly interpolated between (and including) *start* and *end* *numpy.linspace* without *numpy*

Parameters

- **start** – number, or iterable vector
- **end** – number, or iterable vector
- **n** – int number of interpolated values

Result iterator

`Goulib.itertools2.flatten` (*l, donotrecursein=(<class 'str'>,)*)

iterator to flatten (depth-first) structure

Parameters

- **l** – iterable structure
- **donotrecursein** – iterable types in which algo doesn't recurse string type by default

`Goulib.itertools2.itemgetter` (*iterable, i*)

`Goulib.itertools2.compact` (*iterable, f=<class 'bool'>*)

Returns iterator skipping *None* values from iterable

`Goulib.itertools2.compress` (*iterable*)

generates (*item, count*) pairs by counting the number of consecutive items in iterable)

`Goulib.itertools2.recurrence` (*f, x*)

`Goulib.itertools2.swap` (*iterable*)

`Goulib.itertools2.tee` (*iterable, n=2, copy=None*)

tee or copy depending on type and goal

Parameters

- **iterable** – any iterable
- **n** – int number of tees/copies to return
- **copy** – optional copy function, for exemple `copy.copy` or `copy.deepcopy`

Result tee of iterable if it's an iterator or generator, or (deep)copies for other types

this function is useful to avoid side effects at a lower memory cost depending on the case

`Goulib.itertools2.groups` (*iterable, n, step=None*)

Make groups of '*n*' elements from the iterable advancing '*step*' elements on each iteration

`Goulib.itertools2.pairwise` (*iterable, op=None, loop=False*)

iterates through consecutive pairs

Parameters

- **iterable** – input iterable *s1, s2, s3, ..., sn*
- **op** – optional operator to apply to each pair
- **loop** – boolean True if last pair should be (*sn, s1*) to close the loop

Result pairs iterator (*s1, s2*), (*s2, s3*) ... (*si, si+1*), ... (*sn-1, sn*) + optional pair to close the loop

Goulib.itertools2.**shape** (*iterable*)

shape of a mutidimensional array, without numpy

Parameters *iterable* – iterable of iterable ... of iterable or numpy arrays...

Result list of n ints corresponding to iterable's len of each dimension

Warning if iterable is not a (hyper) rect matrix, shape is evaluated from

the [0,0,...0] element ... :see: <http://docs.scipy.org/doc/numpy-1.10.1/reference/generated/numpy.ndarray.shape.html>

Goulib.itertools2.**ndim** (*iterable*)

number of dimensions of a mutidimensional array, without numpy

Parameters *iterable* – iterable of iterable ... of iterable or numpy arrays...

Result int number of dimensions

Goulib.itertools2.**reshape** (*data, dims*)

Result data as a n-dim matrix

Goulib.itertools2.**compose** (*f, g*)

Compose two functions -> compose(f, g)(x) -> f(g(x))

Goulib.itertools2.**iterate** (*func, arg*)

After Haskell's iterate: apply function repeatedly.

Goulib.itertools2.**accumulate** (*iterable, func=<built-in function add>, skip_first=False*)

Return running totals. extends *python.accumulate*

accumulate([1,2,3,4,5]) -> 1 3 6 10 15 # accumulate([1,2,3,4,5], operator.mul) -> 1 2 6 24 120

Goulib.itertools2.**tails** (*seq*)

Get tails of a sequence

tails([1,2,3]) -> [1,2,3], [2,3], [3], [].

Goulib.itertools2.**ireduce** (*func, iterable, init=None*)

Like *python.reduce* but using iterators (a.k.a scanl)

Goulib.itertools2.**unique** (*iterable, key=None, buffer=None*)

generate unique elements, preserving order. :param iterable: iterable :param key: optional function defining which elements are considered equal :param buffer: optional integer defining how many of the last unique elements to keep in memory

unique('AAAABBBCCDAABBB') -> A B C D # unique('ABBCcAD', str.lower) -> A B C D

Goulib.itertools2.**count_unique** (*iterable, key=None*)

Count unique elements

count_unique('AAAABBBCCDAABBB') -> 4 # count_unique('ABBCcAD', str.lower) -> 4

Goulib.itertools2.**identity** (*x*)

Do nothing and return the variable untouched

Goulib.itertools2.**occurrences** (*it, exchange=False*)

Return dictionary with occurrences from iterable

Goulib.itertools2.**cartesian_product** (**iterables, **kwargs*)

<http://stackoverflow.com/questions/12093364/cartesian-product-of-large-iterators-itertools>

Goulib.itertools2.**combinations_with_replacement** (*iterable, r*)

combinations_with_replacement('ABC', 2) -> AA AB AC BB BC CC same as combinations_with_replacement except it doesn't generate duplicates

`Goulib.itertools2.any(seq, pred=<class 'bool'>)`

Result bool True if pred(x) is True for at least one element in the iterable

`Goulib.itertools2.all(seq, pred=<class 'bool'>)`

Result bool True if pred(x) is True for all elements in the iterable

`Goulib.itertools2.no(seq, pred=<class 'bool'>)`

Result bool True if pred(x) is False for every element in the iterable

`Goulib.itertools2.takenth(n, iterable, default=None)`

Result nth item of iterable

`Goulib.itertools2.nth(n, iterable, default=None)`

Result nth item of iterable

`Goulib.itertools2.icross(*sequences)`

Cartesian product of sequences (recursive version)

`Goulib.itertools2.quantify(iterable, pred=<class 'bool'>)`

Result int count how many times the predicate is true

`Goulib.itertools2.interleave(l1, l2)`

Parameters

- **l1** – iterable
- **l2** – iterable of same length, or 1 less than l1

Result iterable interleaving elements from l1 and l2, starting by l1[0]

`Goulib.itertools2.shuffle(ary)`

Param array to shuffle by Fisher-Yates algorithm

Result shuffled array (IN PLACE!)

See <http://www.drgoulu.com/2013/01/19/comment-bien-brasser-les-cartes/>

`Goulib.itertools2.rand_seq(size)`

Result range(size) shuffled

`Goulib.itertools2.all_pairs(size)`

generates all i,j pairs for i,j from 0-size

`Goulib.itertools2.index_min(values, key=<function identity>)`

Result min_index, min_value

`Goulib.itertools2.index_max(values, key=<function identity>)`

Result max_index, max_value

`Goulib.itertools2.best(iterable, key=None, n=1, reverse=False)`

generate items corresponding to the n best values of key sort order

`Goulib.itertools2.sort_indexes(iterable, key=<function identity>, reverse=False)`

Returns iterator over indexes of iterable that correspond to the sorted iterable

`Goulib.itertools2.filter2(iterable, condition)`

like `python.filter` but returns 2 lists : - list of elements in iterable that satisfy condition - list of those that don't

Goulib.itertools2.**ifind** (*iterable, f, reverse=False*)
iterates through items in iterable where f(item) == True.

Goulib.itertools2.**iremove** (*iterable, f*)
removes items from an iterable based on condition :param iterable: iterable . will be modified in place :param f:
function of the form lambda line:bool returning True if item should be removed :yield: removed items backwards

Goulib.itertools2.**removef** (*iterable, f*)
removes items from an iterable based on condition :param iterable: iterable . will be modified in place :param f:
function of the form lambda line:bool returning True if item should be removed :result: list of removed items.

Goulib.itertools2.**find** (*iterable, f*)
Return first item in iterable where f(item) == True.

Goulib.itertools2.**isplit** (*iterable, sep, include_sep=False*)
split iterable by separators or condition :param sep: value or function(item) returning True for items that separate
:param include_sep: bool. If True the separators items are included in output, at beginning of each sub-iterator
:result: iterates through slices before, between, and after separators

Goulib.itertools2.**split** (*iterable, sep, include_sep=False*)
like <https://docs.python.org/2/library/stdtypes.html#str.split>, but for iterable :param sep: value or function(item)
returning True for items that separate :param include_sep: bool. If True the separators items are included in
output, at beginning of each sub-iterator :result: list of iterable slices before, between, and after separators

Goulib.itertools2.**dictsplit** (*dic, keys*)
extract keys from dic :param dic: dict source :param keys: iterable of dict keys :result: dict,dict : the first
contains entries present in source, the second the remaining entries

Goulib.itertools2.**next_permutation** (*seq, pred=<function <lambda>>*)
Like C++ std::next_permutation() but implemented as generator. see <http://blog.bjrn.se/2008/04/lexicographic-permutations-using.html> :param seq: iterable :param pred: a function (a,b) that returns a neg-
ative number if a<b, like cmp(a,b) in Python 2.7

class Goulib.itertools2.**iter2** (*iterable*)

Bases: `object`

Takes in an object that is iterable. <http://code.activestate.com/recipes/578092-flattening-an-arbitrarily-deep-list-or-any-iterato/> Allows for the following method calls (that should be built into iterators anyway...) calls: - append - appends another iterable onto the iterator. - insert - only accepts inserting at the 0 place, inserts an iterable before other iterables. - adding. an iter2 object can be added to another object that is iterable. i.e. iter2 + iter (not iter + iter2). It's best to make all objects iter2 objects to avoid syntax errors. :D

`__init__` (*iterable*)

`append` (*iterable*)

`insert` (*place, iterable*)

`__add__` (*iterable*)

`__next__` ()

`next` ()

`__iter__` ()

`__class__`
alias of `type`

`__delattr__`
Implement delattr(self, name).

`__dir__()` → list
default `dir()` implementation

`__eq__`
Return `self==value`.

`__format__()`
default object formatter

`__ge__`
Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self>value`.

`__hash__`
Return `hash(self)`.

`__le__`
Return `self<=value`.

`__lt__`
Return `self<value`.

`__ne__`
Return `self!=value`.

`__new__()`
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__repr__`
Return `repr(self)`.

`__setattr__`
Implement `setattr(self, name, value)`.

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return `str(self)`.

`Goulib.itertools2.subdict(d, keys)`

extract “sub-dictionary” :param d: dict :param keys: container of keys
to extract: :result: dict: :see: <http://stackoverflow.com/questions/5352546/best-way-to-extract-subset-of-key-value-pairs-from-python-dictionary-object/5352649#5352649>

exception `Goulib.itertools2.SortingError`

Bases: `Exception`

`__cause__`
exception cause

`__class__`
alias of `type`

__context__
exception context

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__init__
Initialize self. See help(type(self)) for accurate signature.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__setstate__ ()

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

__suppress_context__

__traceback__

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

Goulib.itertools2.**ensure_sorted**(*iterable*, *key=None*)
makes sure iterable is sorted according to key

Yields items of iterable

Raise SortingError if not

Goulib.itertools2.**sorted_iterable**(*iterable*, *key=None*, *buffer=100*)
sorts an “almost sorted” (infinite) iterable

Parameters

- **iterable** – iterable
- **key** – function used as sort key
- **buffer** – int size of buffer. elements to swap should not be further than that

Goulib.itertools2.**diff**(*iterable1*, *iterable2*)
generate items in sorted iterable1 that are not in sorted iterable2

Goulib.itertools2.**intersect**(**its*)
generates intersection of N iterables

Parameters *its* – any number of SORTED iterables

Yields elements that belong to all iterables

See <http://stackoverflow.com/questions/969709/joining-a-set-of-ordered-integer-yielding-python-iterators>

class Goulib.itertools2.**keep**(*iterable*)

Bases: collections.abc.Iterator

iterator that keeps the last value

__init__(*iterable*)

__next__()

next()

__abstractmethods__ = frozenset()

__class__

alias of ABCMeta

__delattr__

Implement delattr(self, name).

__dir__() → list

default dir() implementation

__eq__

Return self==value.

__format__()

default object formatter

__ge__

Return self>=value.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self > value`.

`__hash__`
Return `hash(self)`.

`__iter__` ()
Return `iter(self)`.

`__le__`
Return `self <= value`.

`__lt__`
Return `self < value`.

`__ne__`
Return `self != value`.

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return `repr(self)`.

`__setattr__`
Implement `setattr(self, name, value)`.

`__sizeof__` () → int
size of object in memory, in bytes

`__slots__` = ()

`__str__`
Return `str(self)`.

`Goulib.itertools2.first_match(iter1, iter2, limit=None)`
” :param limit: int max number of loops :return: integer i first index where `iter1[i]==iter2[i]`

`Goulib.itertools2.floyd(iterable, limit=1000000.0)`
Detect a cycle in iterable using Floyd “tortue hand hare” algorithm

See https://en.wikipedia.org/wiki/Cycle_detection

Parameters

- **iterable** – iterable
- **limit** – int limit to prevent infinite loop. no limit if None

Result (i,l) tuple of integers where i=index of cycle start, l=length if no cycle is found, return (None,None)

`Goulib.itertools2.detect_cycle(iterable, limit=1000000.0)`

Goulib.markup module

simple HTML/XML generation (forked from [markup](#))

`Goulib.markup.cgiprint` (*line=''*, *unbuff=True*, *line_end='\r\n'*)

Print to the stdout. :param line: string to print, followed by line_end :param unbuff: boolean, True to flush the buffer after every write. :param line_end: string to print after each line. By default this is

, which is the standard specified by the RFC for http headers.

`Goulib.markup.style_dict2str` (*style*)

`Goulib.markup.style_str2dict` (*style*)

`Goulib.markup.tag` (*tag*, *between*, ***kwargs*)
generate full tag.

class `Goulib.markup.element` (*tag*, *case='lower'*, *parent=None*)

Bases: `object`

This class handles the addition of a new element.

`__init__` (*tag*, *case='lower'*, *parent=None*)

`__call__` (**args*, ***kwargs*)

render (*t*, *single*, *args*, ***kwargs*)

Append the actual tags to content.

close ()

Append a closing tag unless element has only opening tag.

open (***kwargs*)

Append an opening tag.

`__class__`

alias of `type`

`__delattr__`

Implement `delattr(self, name)`.

`__dir__` () → list

default `dir()` implementation

`__eq__`

Return `self==value`.

`__format__` ()

default object formatter

`__ge__`

Return `self>=value`.

`__getattr__`

Return `getattr(self, name)`.

`__gt__`

Return `self>value`.

`__hash__`

Return `hash(self)`.

`__le__`

Return `self<=value`.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

class Goulib.markup.**page** (*mode='strict_html', case='lower', onetags=None, twotags=None, separator='n', class_=None*)

Bases: `object`

This is our main class representing a document. Elements are added as attributes of an instance of this class.

Stuff that effects the whole document.

Parameters mode – string. can be either:

- ‘strict_html’ for HTML 4.01 (default)
- ‘html’ alias for ‘strict_html’
- ‘loose_html’ to allow some deprecated elements
- ‘xml’ to allow arbitrary elements

Parameters case – string. can be either:

- ‘lower’ element names will be printed in lower case (default)
- ‘upper’ they will be printed in upper case
- ‘given’ element names will be printed as they are given

Parameters

- **onetags** – list or tuple of valid elements with opening tags only
- **twotags** – list or tuple of valid elements with both opening and closing tags

these two keyword arguments may be used to select the set of valid elements in ‘xml’ mode invalid elements will raise appropriate exceptions

Parameters

- **separator** – string to place between added elements, defaults to newline

- **class** – a class that will be added to every element if defined

__init__ (*mode='strict_html', case='lower', onetags=None, twotags=None, separator='\n', class_=None*)

Stuff that effects the whole document.

Parameters mode – string. can be either:

- 'strict_html' for HTML 4.01 (default)
- 'html' alias for 'strict_html'
- 'loose_html' to allow some deprecated elements
- 'xml' to allow arbitrary elements

Parameters case – string. can be either:

- 'lower' element names will be printed in lower case (default)
- 'upper' they will be printed in upper case
- 'given' element names will be printed as they are given

Parameters

- **onetags** – list or tuple of valid elements with opening tags only
- **twotags** – list or tuple of valid elements with both opening and closing tags

these two keyword arguments may be used to select the set of valid elements in 'xml' mode invalid elements will raise appropriate exceptions

Parameters

- **separator** – string to place between added elements, defaults to newline
- **class** – a class that will be added to every element if defined

__getattr__ (*attr*)

__str__ ()

__call__ (*escape=False*)

Return the document as a string.

escape – **False print normally**

True replace < and > by < and > the default escape sequences in most browsers

add (*text*)

This is an alias to addcontent.

addfooter (*text*)

Add some text to the bottom of the document

addheader (*text*)

Add some text to the top of the document

addcontent (*text*)

Add some text to the main part of the document

init (*lang='en', css=None, metainfo=None, title=None, header=None, footer=None, charset=None, encoding=None, doctype=None, bodyattrs=None, script=None, base=None*)

This method is used for complete documents with appropriate doctype, encoding, title, etc information. For an /XML snippet omit this method.

lang – language, usually a two character string, will appear as `<html lang='en'>` in html mode (ignored in xml mode)

css – Cascading Style Sheet filename as a string or a list of strings for multiple css files (ignored in xml mode)

metainfo – a dictionary in the form { 'name': 'content' } to be inserted into meta element(s) as `<meta name='name' content='content'>` (ignored in xml mode)

base – set the `<base href="...">` tag in `<head>`

bodyattrs – a dictionary in the form { 'key': 'value', ... } which will be added as attributes of the `<body>` element as `<body key='value' ... >` (ignored in xml mode)

script – dictionary containing src:type pairs, <script type='text/type' src=src></script> or a list of ['src1', 'src2', ...] in which case 'javascript' is assumed for all

title – the title of the document as a string to be inserted into a title element as `<title>my title</title>` (ignored in xml mode)

header – some text to be inserted right after the <body> element (ignored in xml mode)

footer – some text to be inserted right before the </body> element (ignored in xml mode)

charset – a string defining the character set, will be inserted into a `<meta http-equiv='Content-Type' content='text/html; charset=myset'>` element (ignored in xml mode)

encoding – a string defining the encoding, will be put into to first line of the document as `<?xml version='1.0' encoding='myencoding' ?>` in xml mode (ignored in html mode)

doctype – the document type string, defaults to `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN'>` in html mode (ignored in xml mode)

css (*filelist*)

This convenience function is only useful for html. It adds css stylesheet(s) to the document via the `<link>` element.

metainfo (*mydict*)

This convenience function is only useful for html. It adds meta information via the `<meta>` element, the argument is a dictionary of the form { 'name': 'content' }.

scripts (*mydict*)

Only useful in html, mydict is dictionary of src:type pairs or a list of script sources ['src1', 'src2', ...] in which case 'javascript' is assumed for type. Will be rendered as `<script type='text/type' src=src></script>`

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

```

__ge__
    Return self>=value.

__getattr__
    Return getattr(self, name).

__gt__
    Return self>value.

__hash__
    Return hash(self).

__le__
    Return self<=value.

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

```

Goulib.markup.**escape** (*text*, *newline=False*)
Escape special html characters.

Goulib.markup.**unescape** (*text*)
Inverse of escape.

class Goulib.markup.**dummy**
Bases: `object`

A dummy class for attaching attributes.

```

__class__
    alias of type

__delattr__
    Implement delattr(self, name).

__dir__ () → list
    default dir() implementation

__eq__
    Return self==value.

__format__ ()
    default object formatter

```

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__gt__`
Return self>value.

`__hash__`
Return hash(self).

`__init__`
Initialize self. See help(type(self)) for accurate signature.

`__le__`
Return self<=value.

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return repr(self).

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__` () → int
size of object in memory, in bytes

`__str__`
Return str(self).

class Goulib.markup.**russell**

Bases: `object`

A dummy class that contains anything.

`__contains__` (*item*)

`__class__`
alias of `type`

`__delattr__`
Implement delattr(self, name).

`__dir__` () → list
default dir() implementation

`__eq__`
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__init__
Initialize self. See help(type(self)) for accurate signature.

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

exception Goulib.markup.**MarkupError**

Bases: Exception

All our exceptions subclass this.

__str__ ()

__cause__
exception cause

__class__
alias of `type`

__context__
exception context

`__delattr__`
Implement `delattr(self, name)`.

`__dir__` () → list
default `dir()` implementation

`__eq__`
Return `self==value`.

`__format__` ()
default object formatter

`__ge__`
Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self>value`.

`__hash__`
Return `hash(self)`.

`__init__`
Initialize self. See `help(type(self))` for accurate signature.

`__le__`
Return `self<=value`.

`__lt__`
Return `self<value`.

`__ne__`
Return `self!=value`.

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return `repr(self)`.

`__setattr__`
Implement `setattr(self, name, value)`.

`__setstate__` ()

`__sizeof__` () → int
size of object in memory, in bytes

`__suppress_context__`

`__traceback__`

args

`with_traceback` ()
Exception.`with_traceback(tb)` – set `self.__traceback__` to `tb` and return self.

exception `Goulib.markup.ClosingError` (*tag*)
 Bases: `Goulib.markup.MarkupError`

__init__ (*tag*)

__cause__
 exception cause

__class__
 alias of `type`

__context__
 exception context

__delattr__
 Implement `delattr(self, name)`.

__dir__ () → list
 default `dir()` implementation

__eq__
 Return `self==value`.

__format__ ()
 default object formatter

__ge__
 Return `self>=value`.

__getattr__
 Return `getattr(self, name)`.

__gt__
 Return `self>value`.

__hash__
 Return `hash(self)`.

__le__
 Return `self<=value`.

__lt__
 Return `self<value`.

__ne__
 Return `self!=value`.

__new__ ()
 Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()

__reduce_ex__ ()
 helper for pickle

__repr__
 Return `repr(self)`.

__setattr__
 Implement `setattr(self, name, value)`.

__setstate__ ()

__sizeof__ () → int
 size of object in memory, in bytes

`__str__()`

`__suppress_context__`

`__traceback__`

args

`with_traceback()`

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `Goulib.markup.OpeningError` (*tag*)

Bases: `Goulib.markup.MarkupError`

`__init__(tag)`

`__cause__`

exception cause

`__class__`

alias of `type`

`__context__`

exception context

`__delattr__`

Implement `delattr(self, name)`.

`__dir__()` → list

default `dir()` implementation

`__eq__`

Return `self==value`.

`__format__()`

default object formatter

`__ge__`

Return `self>=value`.

`__getattr__`

Return `getattr(self, name)`.

`__gt__`

Return `self>value`.

`__hash__`

Return `hash(self)`.

`__le__`

Return `self<=value`.

`__lt__`

Return `self<value`.

`__ne__`

Return `self!=value`.

`__new__()`

Create and return a new object. See `help(type)` for accurate signature.

`__reduce__()`

`__reduce_ex__()`

helper for pickle

```

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__setstate__ ()

__sizeof__ () → int
    size of object in memory, in bytes

__str__ ()

__suppress_context__

__traceback__

args

with_traceback ()
    Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception Goulib.markup.ArgumentError (tag)
    Bases: Goulib.markup.MarkupError

    __init__ (tag)

    __cause__
        exception cause

    __class__
        alias of type

    __context__
        exception context

    __delattr__
        Implement delattr(self, name).

    __dir__ () → list
        default dir() implementation

    __eq__
        Return self==value.

    __format__ ()
        default object formatter

    __ge__
        Return self>=value.

    __getattr__
        Return getattr(self, name).

    __gt__
        Return self>value.

    __hash__
        Return hash(self).

    __le__
        Return self<=value.

    __lt__
        Return self<value.

```

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__setstate__ ()

__sizeof__ () → int
size of object in memory, in bytes

__str__ ()

__suppress_context__

__traceback__

args

with_traceback ()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception Goulib.markup.InvalidElementError (tag, mode)

Bases: *Goulib.markup.MarkupError*

__cause__
exception cause

__class__
alias of `type`

__context__
exception context

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__init__ (*tag, mode*)

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__setstate__ ()

__sizeof__ () → int
size of object in memory, in bytes

__str__ ()

__suppress_context__

__traceback__

args

with_traceback ()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception Goulib.markup.**DeprecationError** (*tag*)
Bases: *Goulib.markup.MarkupError*

__cause__
exception cause

__class__
alias of *type*

__context__
exception context

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

`__format__()`
default object formatter

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__gt__`
Return self>value.

`__hash__`
Return hash(self).

`__le__`
Return self<=value.

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__()`
Create and return a new object. See help(type) for accurate signature.

`__reduce__()`

`__reduce_ex__()`
helper for pickle

`__repr__`
Return repr(self).

`__setattr__`
Implement setattr(self, name, value).

`__setstate__()`

`__sizeof__()` → int
size of object in memory, in bytes

`__str__()`

`__suppress_context__`

`__traceback__`

args

`with_traceback()`
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

`__init__(tag)`

exception `Goulib.markup.ModeError(mode)`
Bases: `Goulib.markup.MarkupError`

`__cause__`
exception cause

`__class__`
alias of `type`

__context__
exception context

__delattr__
Implement delattr(self, name).

__dir__ () → list
default dir() implementation

__eq__
Return self==value.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()

__reduce_ex__ ()
helper for pickle

__repr__
Return repr(self).

__setattr__
Implement setattr(self, name, value).

__setstate__ ()

__sizeof__ () → int
size of object in memory, in bytes

__str__ ()

__suppress_context__

__traceback__

args

with_traceback ()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

`__init__` (*mode*)

exception `Goulib.markup.CustomizationError`

Bases: `Goulib.markup.MarkupError`

`__cause__`

exception cause

`__class__`

alias of `type`

`__context__`

exception context

`__delattr__`

Implement `delattr(self, name)`.

`__dir__` () → list

default `dir()` implementation

`__eq__`

Return `self==value`.

`__format__` ()

default object formatter

`__ge__`

Return `self>=value`.

`__getattr__`

Return `getattr(self, name)`.

`__gt__`

Return `self>value`.

`__hash__`

Return `hash(self)`.

`__le__`

Return `self<=value`.

`__lt__`

Return `self<value`.

`__ne__`

Return `self!=value`.

`__new__` ()

Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()

`__reduce_ex__` ()

helper for pickle

`__repr__`

Return `repr(self)`.

`__setattr__`

Implement `setattr(self, name, value)`.

`__setstate__` ()

`__sizeof__` () → int

size of object in memory, in bytes

```

__str__()
__suppress_context__
__traceback__
args
with_traceback()
    Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.
__init__()

```

Goulib.math2 module

more math than `math` standard library, without numpy

`Goulib.math2.is_number(x)`

Returns True if x is a number of any type

`Goulib.math2.sign(number)`

Returns 1 if number is positive, -1 if negative, 0 if ==0

`Goulib.math2.cmp(x, y)`

Compare the two objects x and y and return an integer according to the outcome. The return value is negative if $x < y$, zero if $x == y$ and strictly positive if $x > y$.

`Goulib.math2.gcd(*args)`

greatest common divisor of an arbitrary number of args

`Goulib.math2.lcm(*args)`

least common multiple of any number of integers

`Goulib.math2.xgcd(a, b)`

Extended GCD

Returns (gcd, x, y) where gcd is the greatest common divisor of a and b

with the sign of b if b is nonzero, and with the sign of a if b is 0. The numbers x,y are such that $\text{gcd} = ax + by$.

`Goulib.math2.coprime(*args)`

Returns True if args are coprime to each other

`Goulib.math2.coprimes_gen(limit)`

generates coprime pairs using Farey sequence

`Goulib.math2.is_primitive_root(x, m, s={})`

returns True if x is a primitive root of m

Parameters s – set of coprimes to m, if already known

`Goulib.math2.primitive_root_gen(m)`

generate primitive roots modulo m

`Goulib.math2.primitive_roots(modulo)`

`Goulib.math2.quad(a, b, c, allow_complex=False)`

solves quadratic equations $aX^2 + bX + c = 0$

Parameters

- a, b, c – floats

- **allow_complex** – function returns complex roots if True

Returns x1,x2 real or complex solutions

Goulib.math2.**isclose** (*a, b, rel_tol=1e-09, abs_tol=0.0*)
approximately equal. Use this instead of `a==b` in floating point ops

implements <https://www.python.org/dev/peps/pep-0485/> :param a,b: the two values to be tested to relative closeness :param rel_tol: relative tolerance – it is the amount of error allowed, relative to the larger absolute value of a or b. For example, to set a tolerance of 5%, pass `tol=0.05`. The default tolerance is `1e-9`, which assures that the two values are the same within about 9 decimal digits. rel_tol must be greater than 0.0 :param abs_tol: minimum absolute tolerance level – useful for comparisons near zero.

Goulib.math2.**is_integer** (*x, epsilon=1e-06*)

Returns True if float x is almost an integer

Goulib.math2.**rint** (*v*)

Returns int value nearest to float v

Goulib.math2.**int_or_float** (*x, epsilon=1e-06*)

Parameters **x** – int or float

Returns int if x is (almost) an integer, otherwise float

Goulib.math2.**format** (*x, decimals=3*)
formats a float with given number of decimals, but not an int

Returns string repr of x with decimals if not int

Goulib.math2.**ceildiv** (*a, b*)

Goulib.math2.**isqrt** (*n*)
integer square root

Returns largest int x for which $x * x \leq n$

Goulib.math2.**sqrt** (*n*)
square root :return: float, or int if n is a perfect square

Goulib.math2.**is_square** (*n*)

Goulib.math2.**multiply** (*x, y*)
Karatsuba fast multiplication algorithm

https://en.wikipedia.org/wiki/Karatsuba_algorithm

Copyright (c) 2014 Project Nayuki <http://www.nayuki.io/page/karatsuba-multiplication>

Goulib.math2.**accsum** (*it*)
Yield accumulated sums of iterable: `accsum(count(1)) -> 1,3,6,10,...`

Goulib.math2.**cumsum** (*it*)
Yield accumulated sums of iterable: `accsum(count(1)) -> 1,3,6,10,...`

Goulib.math2.**mul** (*nums, init=1*)

Returns Product of nums

Goulib.math2.**dot_vv** (*a, b, default=0*)
dot product for vectors

Parameters

- **a** – vector (iterable)

- **b** – vector (iterable)
- **default** – default value of the multiplication operator

`Goulib.math2.dot_mv(a, b, default=0)`
dot product for vectors

Parameters

- **a** – matrix (iterable or iterables)
- **b** – vector (iterable)
- **default** – default value of the multiplication operator

`Goulib.math2.dot_mm(a, b, default=0)`
dot product for matrices

Parameters

- **a** – matrix (iterable or iterables)
- **b** – matrix (iterable or iterables)
- **default** – default value of the multiplication operator

`Goulib.math2.dot(a, b, default=0)`
dot product

general but slow : use `dot_vv`, `dot_mv` or `dot_mm` if you know a and b's dimensions

`Goulib.math2.zeros(shape)`

See <https://docs.scipy.org/doc/numpy/reference/generated/numpy.zeros.html>

`Goulib.math2.diag(v)`

Create a two-dimensional array with the flattened input as a diagonal.

Parameters **v** – If v is a 2-D array, return a copy of its diagonal. If v is a 1-D array, return a 2-D array with v on the diagonal

See <https://docs.scipy.org/doc/numpy/reference/generated/numpy.diag.html#numpy.diag>

`Goulib.math2.identity(n)`

`Goulib.math2.eye(n)`

`Goulib.math2.transpose(m)`

Returns matrix m transposed

`Goulib.math2.maximum(m)`

Compare N arrays and returns a new array containing the element-wise maxima

Parameters **m** – list of arrays (matrix)

Returns list of maximal values found in each column of m

See <http://docs.scipy.org/doc/numpy/reference/generated/numpy.maximum.html>

`Goulib.math2.minimum(m)`

Compare N arrays and returns a new array containing the element-wise minima

Parameters **m** – list of arrays (matrix)

Returns list of minimal values found in each column of m

See <http://docs.scipy.org/doc/numpy/reference/generated/numpy.minimum.html>

Goulib.math2.**vecadd** (*a*, *b*, *fillvalue*=0)
addition of vectors of unequal lengths

Goulib.math2.**vecsub** (*a*, *b*, *fillvalue*=0)
subtraction of vectors of unequal lengths

Goulib.math2.**vecneg** (*a*)
unary negation

Goulib.math2.**vecmul** (*a*, *b*)
product of vectors of unequal lengths

Goulib.math2.**vecdiv** (*a*, *b*)
quotient of vectors of unequal lengths

Goulib.math2.**veccompare** (*a*, *b*)
compare values in 2 lists. returns triple number of pairs where [a<b, a==b, a>b]

Goulib.math2.**sat** (*x*, *low*=0, *high*=None)
saturates x between low and high

Goulib.math2.**norm_2** (*v*)
Returns “normal” euclidian norm of vector v

Goulib.math2.**norm_1** (*v*)
Returns “manhattan” norm of vector v

Goulib.math2.**norm_inf** (*v*)
Returns infinite norm of vector v

Goulib.math2.**norm** (*v*, *order*=2)
See <http://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.norm.html>

Goulib.math2.**dist** (*a*, *b*, *norm*=<function norm_2>)

Goulib.math2.**vecunit** (*v*, *norm*=<function norm_2>)
Returns vector normalized

Goulib.math2.**hamming** (*s1*, *s2*)
Calculate the Hamming distance between two iterables

Goulib.math2.**sets_dist** (*a*, *b*)
See <http://stackoverflow.com/questions/11316539/calculating-the-distance-between-two-unordered-sets>

Goulib.math2.**sets levenshtein** (*a*, *b*)
levenshtein distance on sets
See http://en.wikipedia.org/wiki/Levenshtein_distance

Goulib.math2.**levenshtein** (*seq1*, *seq2*)
levenshtein distance
Returns distance between 2 iterables

See http://en.wikipedia.org/wiki/Levenshtein_distance

Goulib.math2.**recurrence** (*coefficients*, *values*, *cst*=0, *max*=None, *mod*=0)
general generator for recurrences

Parameters

- **values** – list of initial values

- **coefficients** – list of factors defining the recurrence

Goulib.math2.**fibonacci_gen** (*max=None, mod=0*)
Generate fibonacci serie

Goulib.math2.**fibonacci** (*n, mod=0*)
fibonacci series n-th element

Parameters

- **n** – int can be extremely high, like $1e19$!
- **mod** – int optional modulo

Goulib.math2.**is_fibonacci** (*n*)
returns True if n is in Fibonacci series

Goulib.math2.**pisano_cycle** (*mod*)

Goulib.math2.**pisano_period** (*mod*)

Goulib.math2.**pascal_gen** ()
Pascal's triangle read by rows: $C(n,k) = \text{binomial}(n,k) = n!/(k!*(n-k)!)$, $0 \leq k \leq n$.
<https://oeis.org/A007318>

Goulib.math2.**catalan** (*n*)
Catalan numbers: $C(n) = \text{binomial}(2n,n)/(n+1) = (2n)!/(n!(n+1)!)$.

Goulib.math2.**catalan_gen** ()
Generate Catalan numbers: $C(n) = \text{binomial}(2n,n)/(n+1) = (2n)!/(n!(n+1)!)$. Also called Segner numbers.

Goulib.math2.**is_pythagorean_triple** (*a, b, c*)

Goulib.math2.**primitive_triplets** ()
generates primitive Pythagorean triplets $x < y < z$

sorted by hypotenuse z, then longest side y through Berggren's matrices and breadth first traversal of ternary tree :see: https://en.wikipedia.org/wiki/Tree_of_primitive_Pythagorean_triplets

Goulib.math2.**triples** ()
generates all Pythagorean triplets triplets $x < y < z$ sorted by hypotenuse z, then longest side y

Goulib.math2.**divisors** (*n*)

Returns all divisors of n: `divisors(12) -> 1,2,3,6,12`

including 1 and n, except for 1 which returns a single 1 to avoid messing with sum of divisors...

Goulib.math2.**proper_divisors** (*n*)

Returns all divisors of n except n itself.

Goulib.math2.**sieve** (*n, oneisprime=False*)
prime numbers from 2 to a prime $< n$ Very fast ($n < 10,000,000$) in 0.4 sec.

Example: `>>>prime_sieve(25) [2, 3, 5, 7, 11, 13, 17, 19, 23]`

Algorithm & Python source: Robert William Hanks <http://stackoverflow.com/questions/17773352/python-sieve-prime-numbers>

Goulib.math2.**primes** (*n*)
memoized list of n first primes

Warning do not call with large n, use `prime_gen` instead

`Goulib.math2.is_prime` (*n*, *oneisprime=False*, *precision_for_huge_n=16*)
primality test. Uses Miller-Rabin for large *n*

Parameters

- **n** – int number to test
- **oneisprime** – bool True if 1 should be considered prime (it was, a long time ago)
- **precision_for_huge_n** – int number of primes to use in Miller

Returns True if *n* is a prime number

`Goulib.math2.primes_gen` (*start=2*, *stop=None*)
generate prime numbers from 'start'

`Goulib.math2.random_prime` (*bits*)
returns a random number of the specified bit length

`Goulib.math2.euclid_gen` ()
generates Euclid numbers: 1 + product of the first *n* primes

`Goulib.math2.prime_factors` (*num*, *start=2*)
generates all prime factors (ordered) of *num*

`Goulib.math2.prime_divisors` (*num*, *start=2*)
generates unique prime divisors (ordered) of *num*

`Goulib.math2.is_multiple` (*n*, *factors*)
return True if *n* has ONLY factors as prime factors

`Goulib.math2.factorize` (*n*)
find the prime factors of *n* along with their frequencies. Example:

```
>>> factor(786456)
[(2,3), (3,3), (11,1), (331,1)]
```

`Goulib.math2.factors` (*n*)

`Goulib.math2.number_of_divisors` (*n*)

`Goulib.math2.omega` (*n*)
Number of distinct primes dividing *n*

`Goulib.math2.bigomega` (*n*)
Number of prime divisors of *n* counted with multiplicity

`Goulib.math2.moebius` (*n*)
Möbius (or Moebius) function $\mu(n)$. $\mu(1) = 1$; $\mu(n) = (-1)^k$ if *n* is the product of *k* different primes; otherwise $\mu(n) = 0$.

`Goulib.math2.euler_phi` (*n*)
Euler totient function

See <http://stackoverflow.com/questions/1019040/how-many-numbers-below-n-are-coprimes-to-n>

`Goulib.math2.totient` (*n*)
Euler totient function

See <http://stackoverflow.com/questions/1019040/how-many-numbers-below-n-are-coprimes-to-n>

`Goulib.math2.prime_ktuple` (*constellation*)
generates tuples of primes with specified differences

Parameters **constellation** – iterable of int differences between primes to return

Note negative int means the difference must NOT be prime

See https://en.wikipedia.org/wiki/Prime_k-tuple

(0, 2) twin primes (0, 4) cousin primes (0, 6) sexy primes (0, 2, 6), (0, 4, 6) prime triplets (0, 6, 12, -18) sexy prime triplets (0, 2, 6, 8) prime quadruplets (0, 6, 12, 18) sexy prime quadruplets (0, 2, 6, 8, 12), (0, 4, 6, 10, 12) quintuplet primes (0, 4, 6, 10, 12, 16) sextuplet primes

Goulib.math2.**twin_primes** ()

Goulib.math2.**cousin_primes** ()

Goulib.math2.**sexy_primes** ()

Goulib.math2.**sexy_prime_triplets** ()

Goulib.math2.**sexy_prime_quadruplets** ()

Goulib.math2.**lucas_lehmer** (*p*)

Lucas Lehmer primality test for Mersenne exponent *p*

Parameters *p* – int

Returns True if 2^p-1 is prime

Goulib.math2.**digits_gen** (*num*, *base=10*)

generates int digits of *num* in base BACKWARDS

Goulib.math2.**digits** (*num*, *base=10*, *rev=False*)

Returns list of digits of *num* expressed in base, optionally reversed

Goulib.math2.**digsum** (*num*, *base=10*, *f=None*)

sum of digits

Parameters

- **num** – number
- **base** – optional base
- **f** – int power or function applied to each digit

Returns sum of $f(\text{digits})$ of *num*

$\text{digsum}(\text{num}) \rightarrow$ sum of digits $\text{digsum}(\text{num}, \text{base}=2) \rightarrow$ number of 1 bits in binary representation of *num* $\text{digsum}(\text{num}, \text{f}=2) \rightarrow$ sum of the squares of digits $\text{digsum}(\text{num}, \text{f}=\text{lambda } x:x**x) \rightarrow$ sum of the digits elevated to their own power

Goulib.math2.**integer_exponent** (*a*, *b=10*)

Returns int highest power of *b* that divides *a*.

See <https://reference.wolfram.com/language/ref/IntegerExponent.html>

Goulib.math2.**trailing_zeros** (*a*, *b=10*)

Returns int highest power of *b* that divides *a*.

See <https://reference.wolfram.com/language/ref/IntegerExponent.html>

Goulib.math2.**power_tower** (*v*)

Returns $v[0]**v[1]**v[2] \dots$

See <http://ajcr.net/Python-power-tower/>

Goulib.math2.**carries** (*a*, *b*, *base=10*, *pos=0*)

Returns int number of carries required to add a+b in base

`Goulib.math2.str_base(num, base=10, numerals='0123456789abcdefghijklmnopqrstuvwxyz')`

Returns string representation of num in base

Parameters

- **num** – int number (decimal)
- **base** – int base, 10 by default
- **numerals** – string with all chars representing numbers in base base. chars after the base-th are ignored

`Goulib.math2.num_from_digits(digits, base=10)`

Parameters

- **digits** – string or list of digits representing a number in given base
- **base** – int base, 10 by default

Returns int number

`Goulib.math2.reverse(i)`

`Goulib.math2.is_palindromic(num, base=10)`

Check if 'num' in base 'base' is a palindrome, that's it, if it can be read equally from left to right and right to left.

`Goulib.math2.is_permutation(num1, num2, base=10)`

Check if 'num1' and 'num2' have the same digits in base

`Goulib.math2.is_pandigital(num, base=10)`

Returns True if num contains all digits in specified base

`Goulib.math2.bouncy(n)`

`Goulib.math2.repunit_gen(digit=1)`

generate repunits

`Goulib.math2.repunit(n, digit=1)`

Returns nth repunit

`Goulib.math2.rational_form(numerator, denominator)`

information about the decimal representation of a rational number.

Returns 5 integer : integer, decimal, shift, repeat, cycle

•shift is the len of decimal with leading zeroes if any

•cycle is the len of repeat with leading zeroes if any

`Goulib.math2.rational_str(n, d)`

`Goulib.math2.rational_cycle(num, den)`

periodic part of the decimal expansion of num/den. Any initial 0's are placed at end of cycle.

See <https://oeis.org/A036275>

`Goulib.math2.tetrahedral(n)`

Returns int n-th tetrahedral number

See https://en.wikipedia.org/wiki/Tetrahedral_number

Goulib.math2.**sum_of_squares** (*n*)

Returns $1^2 + 2^2 + 3^2 + \dots + n^2$

See https://en.wikipedia.org/wiki/Square_pyramidal_number

Goulib.math2.**pyramidal** (*n*)

Returns $1^2 + 2^2 + 3^2 + \dots + n^2$

See https://en.wikipedia.org/wiki/Square_pyramidal_number

Goulib.math2.**sum_of_cubes** (*n*)

Returns $1^3 + 2^3 + 3^3 + \dots + n^3$

See https://en.wikipedia.org/wiki/Squared_triangular_number

Goulib.math2.**bernouilli_gen** (*init=1*)

generator of Bernouilli numbers

Parameters *init* – int -1 or +1.

- -1 for “first Bernoulli numbers” with $B_1 = -1/2$
- +1 for “second Bernoulli numbers” with $B_1 = +1/2$

https://en.wikipedia.org/wiki/Bernoulli_number
_Optimised_task_algorithm

https://rosettacode.org/wiki/Bernoulli_numbers#Python:

Goulib.math2.**bernouilli** (*n*, *init=1*)

Goulib.math2.**faulhaber** (*n*, *p*)

sum of the *p*-th powers of the first *n* positive integers

Returns $1^p + 2^p + 3^p + \dots + n^p$

See https://en.wikipedia.org/wiki/Faulhaber%27s_formula

Goulib.math2.**is_happy** (*n*)

Goulib.math2.**lychrel_seq** (*n*)

Goulib.math2.**lychrel_count** (*n*, *limit=96*)

number of lychrel iterations before *n* becomes palindromic

Parameters

- *n* – int number to test
- *limit* – int max number of loops. default 96 corresponds to the known most retarded non lychrel number

Warning there are palindrom lychrel numbers such as 4994

Goulib.math2.**is_lychrel** (*n*, *limit=96*)

Warning there are palindrom lychrel numbers such as 4994

Goulib.math2.**polygonal** (*s*, *n*)

Goulib.math2.**triangle** (*n*)

Returns *n*th triangle number, defined as the sum of [1,*n*] values.

See http://en.wikipedia.org/wiki/Triangular_number

Goulib.math2.**triangular** (*n*)

Returns nth triangle number, defined as the sum of [1,n] values.

See http://en.wikipedia.org/wiki/Triangular_number

Goulib.math2.**is_triangle**(*x*)

Returns True if *x* is a triangle number

Goulib.math2.**is_triangular**(*x*)

Returns True if *x* is a triangle number

Goulib.math2.**square**(*n*)

Goulib.math2.**pentagonal**(*n*)

Returns nth pentagonal number

See https://en.wikipedia.org/wiki/Pentagonal_number

Goulib.math2.**is_pentagonal**(*n*)

Returns True if *x* is a pentagonal number

Goulib.math2.**hexagonal**(*n*)

Returns nth hexagonal number

See https://en.wikipedia.org/wiki/Hexagonal_number

Goulib.math2.**is_hexagonal**(*n*)

Goulib.math2.**heptagonal**(*n*)

Goulib.math2.**is_heptagonal**(*n*)

Goulib.math2.**octagonal**(*n*)

Goulib.math2.**is_octagonal**(*n*)

Goulib.math2.**partition**(*n*)

The partition function $p(n)$

gives the number of partitions of a nonnegative integer *n* into positive integers. (There is one partition of zero into positive integers, i.e. the empty partition, since the empty sum is defined as 0.)

See http://oeis.org/wiki/Partition_function

Goulib.math2.**get_cardinal_name**(*num*)

Get cardinal name for number (0 to 1 million)

Goulib.math2.**abundance**(*n*)

Goulib.math2.**is_perfect**(*n*)

Returns -1 if *n* is deficient, 0 if perfect, 1 if abundant

See https://en.wikipedia.org/wiki/Perfect_number,

https://en.wikipedia.org/wiki/Abundant_number, https://en.wikipedia.org/wiki/Deficient_number

Goulib.math2.**number_of_digits**(*num*, *base=10*)

Return number of digits of *num* (expressed in base 'base')

Goulib.math2.**chakravala**(*n*)

solves $x^2 - n*y^2 = 1$ for *x*,*y* integers

https://en.wikipedia.org/wiki/Pell%27s_equation https://en.wikipedia.org/wiki/Chakravala_method

Goulib.math2.**factorial_gen**()
Generator of factorial

Goulib.math2.**binomial**(*n*, *k*)
<https://en.wikipedia.org/wiki/binomial>

Goulib.math2.**ncombinations**(*n*, *k*)
<https://en.wikipedia.org/wiki/binomial>

Goulib.math2.**binomial_exponent**(*n*, *k*, *p*)
Returns int largest power of *p* that divides binomial(*n*,*k*)

Goulib.math2.**log_factorial**(*n*)
Returns float approximation of ln(*n*!) by Ramanujan formula

Goulib.math2.**log_binomial**(*n*, *k*)
Returns float approximation of ln(binomial(*n*,*k*))

Goulib.math2.**ilog**(*a*, *b*, *upper_bound=False*)
discrete logarithm *x* such that $b^x=a$

Parameters

- **a**, **b** – integer
- **upper_bound** – bool. if True, returns smallest *x* such that $b^x \geq a$

Returns *x* integer such that $b^x=a$, or *upper_bound*, or None

https://en.wikipedia.org/wiki/Discrete_logarithm

Goulib.math2.**angle**(*u*, *v*, *unit=True*)

Parameters

- **u**, **v** – iterable vectors
- **unit** – bool True if vectors are unit vectors. False increases computations

Returns float angle *n* radians between *u* and *v* unit vectors *i*

Goulib.math2.**sin_over_x**(*x*)
numerically safe $\sin(x)/x$

Goulib.math2.**slerp**(*u*, *v*, *t*)
spherical linear interpolation

Parameters

- **u**, **v** – 3D unit vectors
- **t** – float in [0,1] interval

Returns vector interpolated between *u* and *v*

Goulib.math2.**proportional**(*nseats*, *votes*)
assign *n* seats proportionally to *votes* using the https://en.wikipedia.org/wiki/Hagenbach-Bischoff_quota method

Parameters

- **nseats** – int number of seats to assign
- **votes** – iterable of int or float weighting each party

Result list of ints seats allocated to each party

Goulib.math2.**triangular_repartition** (*x*, *n*)

divide 1 into *n* fractions such that:

- their sum is 1
- they follow a triangular linear repartition (sorry, no better name for now) where $x/1$ is the maximum

Goulib.math2.**rectangular_repartition** (*x*, *n*, *h*)

divide 1 into *n* fractions such that:

- their sum is 1
- they follow a repartition along a pulse of height $h < 1$

Goulib.math2.**de_bruijn** (*k*, *n*)

De Bruijn sequence for alphabet *k* and subsequences of length *n*.

https://en.wikipedia.org/wiki/De_Bruijn_sequence

Goulib.math2.**mod_pow** (*a*, *b*, *m*)

Returns $(a^b) \bmod m$

Goulib.math2.**mod_inv** (*a*, *b*)

Goulib.math2.**mod_div** (*a*, *b*, *m*)

Returns *x* such that $(b * x) \bmod m = a \bmod m$

Goulib.math2.**mod_fact** (*n*, *m*)

Returns $n! \bmod m$

Goulib.math2.**chinese_remainder** (*m*, *a*)

http://en.wikipedia.org/wiki/Chinese_remainder_theorem

Parameters

- **m** – list of int moduli
- **a** – list of int remainders

Returns smallest int *x* such that $x \bmod n_i = a_i$

Goulib.math2.**mod_binomial** (*n*, *k*, *m*, *q=None*)

calculates $C(n,k) \bmod m$ for large *n*,*k*,*m*

Parameters

- **n** – int total number of elements
- **k** – int number of elements to pick
- **m** – int modulo (or iterable of (m,p) tuples used internally)
- **q** – optional int power of *m* for prime *m*, used internally

Goulib.math2.**baby_step_giant_step** (*y*, *a*, *n*)

solves Discrete Logarithm Problem (DLP) $y = a^{**x} \bmod n$

Goulib.math2.**mod_matmul** (*A*, *B*, *mod=0*)

Goulib.math2.**mod_matpow** (*M*, *power*, *mod=0*)

Goulib.math2.**matrix_power** (*M*, *power*, *mod=0*)

`Goulib.math2.pi_digits_gen()`
 generates pi digits as a sequence of INTEGERS ! using Jeremy Gibbons spigot generator
 :see :<http://www.cs.ox.ac.uk/people/jeremy.gibbons/publications/spigot.pdf>

Goulib.motion module

motion simulation (kinematics)

class `Goulib.motion.PVA` (*funcs*)

Bases: `Goulib.plot.Plot`

represents a function of time returning position, velocity, and acceleration

`__init__` (*funcs*)

`__call__` (*t, t0=0*)

`__class__`
 alias of `type`

`__delattr__`
 Implement `delattr(self, name)`.

`__dir__` () → list
 default `dir()` implementation

`__eq__`
 Return `self==value`.

`__format__` ()
 default object formatter

`__ge__`
 Return `self>=value`.

`__getattr__`
 Return `getattr(self, name)`.

`__gt__`
 Return `self>value`.

`__hash__`
 Return `hash(self)`.

`__le__`
 Return `self<=value`.

`__lt__`
 Return `self<value`.

`__ne__`
 Return `self!=value`.

`__new__` ()
 Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
 helper for pickle

`__reduce_ex__` ()
 helper for pickle

__repr__

Return repr(self).

__setattr__

Implement setattr(self, name, value).

__sizeof__ () → int

size of object in memory, in bytes

__str__

Return str(self).

html (***kwargs*)

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

render (*fmt='svg', **kwargs*)

save (*filename, **kwargs*)

svg (***kwargs*)

class Goulib.motion.Segment (*t0, t1, funcs*)

Bases: *Goulib.motion.PVA*

a PVA defined between 2 times, null elsewhere

__init__ (*t0, t1, funcs*)

dt ()

start ()

startPos ()

startSpeed ()

startAcc ()

startJerk ()

startTime ()

end ()

endPos ()

endSpeed ()

endAcc ()

endJerk ()

endTime ()

timeWhenPosBiggerThan (*pos, resolution=0.01*)

search the first time when the position is bigger than pos :params pos: the pos that must at least be reached

:params resolution: the time resolution in sec

__call__ (*t*)

__class__

alias of *type*

`__delattr__`
 Implement `delattr(self, name)`.

`__dir__` () → list
 default `dir()` implementation

`__eq__`
 Return `self==value`.

`__format__` ()
 default object formatter

`__ge__`
 Return `self>=value`.

`__getattr__`
 Return `getattr(self, name)`.

`__gt__`
 Return `self>value`.

`__hash__`
 Return `hash(self)`.

`__le__`
 Return `self<=value`.

`__lt__`
 Return `self<value`.

`__ne__`
 Return `self!=value`.

`__new__` ()
 Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
 helper for pickle

`__reduce_ex__` ()
 helper for pickle

`__repr__`
 Return `repr(self)`.

`__setattr__`
 Implement `setattr(self, name, value)`.

`__sizeof__` () → int
 size of object in memory, in bytes

`__str__`
 Return `str(self)`.

`html` (***kwargs*)

`plot` (***kwargs*)
 renders on IPython Notebook (alias to make usage more straightforward)

`png` (***kwargs*)

`render` (*fmt='svg', **kwargs*)

`save` (*filename, **kwargs*)

svg (***kwargs*)

class `Goulib.motion.Segments` (*segments=[]*, *label='Segments'*)

Bases: `Goulib.motion.Segment`

can be initialized with a list of segment (that of course can also be a Segments) :param label: a label can be given

__init__ (*segments=[]*, *label='Segments'*)

can be initialized with a list of segment (that of course can also be a Segments) :param label: a label can be given

__str__ ()

html ()

update ()

yet only calculates t0 and t1

insert (*segment*, *autoJoin=True*)

insert a segment into Segments :param segment: the segment to add. must be in a range that is not already defined or it will rise a value error exception :param autoJoin: if True and the added segment has the same starting position as the last segment's end

and both velocity are 0 then a segment of (pos,v=0,a=0) is automatically added. this help describing movements only where there is currently a movement

add (*segments*, *autoJoin=True*)

add a segment or a list of segment to the segments

start ()

end ()

__call__ (*t*)

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__le__

Return `self<=value`.

```

__lt__
    Return self<value.

__ne__
    Return self!=value.

__new__ ()
    Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
    helper for pickle

__reduce_ex__ ()
    helper for pickle

__repr__
    Return repr(self).

__setattr__
    Implement setattr(self, name, value).

__sizeof__ () → int
    size of object in memory, in bytes

dt ()

endAcc ()

endJerk ()

endPos ()

endSpeed ()

endTime ()

plot (**kwargs)
    renders on IPython Notebook (alias to make usage more straightforward)

png (**kwargs)

render (fmt='svg', **kwargs)

save (filename, **kwargs)

startAcc ()

startJerk ()

startPos ()

startSpeed ()

startTime ()

svg (**kwargs)

timeWhenPosBiggerThan (pos, resolution=0.01)
    search the first time when the position is bigger than pos :params pos: the pos that must at least be reached
    :params resolution: the time resolution in sec

class Goulib.motion.SegmentPoly (t0, t1, p)
    Bases: Goulib.motion.Segment
    a segment defined by a polynomial position law

    __init__ (t0, t1, p)

```

`__call__` (*t*)

`__class__`
alias of `type`

`__delattr__`
Implement `delattr(self, name)`.

`__dir__` () → list
default `dir()` implementation

`__eq__`
Return `self==value`.

`__format__` ()
default object formatter

`__ge__`
Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self>value`.

`__hash__`
Return `hash(self)`.

`__le__`
Return `self<=value`.

`__lt__`
Return `self<value`.

`__ne__`
Return `self!=value`.

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return `repr(self)`.

`__setattr__`
Implement `setattr(self, name, value)`.

`__sizeof__` () → int
size of object in memory, in bytes

`__str__`
Return `str(self)`.

`dt` ()

`end` ()

`endAcc` ()

```

endJerk ()
endPos ()
endSpeed ()
endTime ()
html (**kwargs)
plot (**kwargs)
    renders on IPython Notebook (alias to make usage more straightforward)
png (**kwargs)
render (fmt='svg', **kwargs)
save (filename, **kwargs)
start ()
startAcc ()
startJerk ()
startPos ()
startSpeed ()
startTime ()
svg (**kwargs)
timeWhenPosBiggerThan (pos, resolution=0.01)
    search the first time when the position is bigger than pos :params pos: the pos that must at least be reached
    :params resolution: the time resolution in sec

```

class Goulib.motion.**Actuator** (stateMachine, vmax, acc, name='', pos=<Quantity(0, 'meter')>, distPerTurn=<Quantity(1, 'millimeter')>, angle=<Quantity(0, 'degree')>, mass=<Quantity(1, 'kilogram')>, friction=<Quantity(0, 'newton')>)

Bases: `object`

simulate an actuator. each movements are recorded in a Segments object the goal of this class is to simplify the writing in most common cases

Params stateMachine a stateMachine. the only requirement for the simulation is to have a .time as V(time,'s') and a .displayMove boolean

Params acc the default acceleration of the actuator

Params vmax the default vmax

Params name name of the actuator

Params pos the initial position

Params distPerTurn the distance of the actuator per motor (or reductor) turn

Params angle if 0, the mass is moving horizontally, if 90° vertically: CAREFULL in that case the bigger the position, the higher

Params mass the mass to move (for both inertia and lifting force)

Params friction the friction force TODO: currently no difference between u0 and udynamic

inertia of the pulley should be taken into consideration

WARNING: the maxForce simulation is minimalist: as we know nothing about the reversibility of the gears and where is the friction, friction is always added even if it might be compensated by the mass in the case we go down

`__init__` (*stateMachine*, *vmax*, *acc*, *name*=' ', *pos*=<Quantity(0, 'meter')>, *distPerTurn*=<Quantity(1, 'millimeter')>, *angle*=<Quantity(0, 'degree')>, *mass*=<Quantity(1, 'kilogram')>, *friction*=<Quantity(0, 'newton')>)

Params stateMachine a stateMachine. the only requirement for the simulation is to have a .time as V(time,'s') and a .displayMove boolean

Params acc the default acceleration of the actuator

Params vmax the default vmax

Params name name of the actuator

Params pos the initial position

Params distPerTurn the distance of the actuator per motor (or reductor) turn

Params angle if 0, the mass is moving horizontally, if 90° vertically: CAREFULL in that case the bigger the position, the higher

Params mass the mass to move (for both inertia and lifting force)

Params friction the friction force TODO: currently no difference between u0 and udynamic

inertia of the pulley should be taken into consideration

WARNING: the maxForce simulation is minimalist: as we know nothing about the reversibility of the gears and where is the friction, friction is always added even if it might be compensated by the mass in the case we go down

move (*newpos*, *relative*=False, *time*=None, *wait*=True, *vmax*=None, *acc*=None)

moves the actuator to newpos :params newpos: the new absolute position :params time: the starting time of the move. by default (None) the state machine time will be used but

one can force the starting poing in the past typically to do parallel moves of different actuators

Params vmax by default the values given at initialisation, but a value for this move can be given

Params acc by default the values given at initialisation, but a value for this move can be given

endTime ()

P (*t*)

maxAbsAcc ()

maxAbsSpeed ()

maxForce ()

maxTork ()

maxRpm ()

displayLast ()

display (*fromTime*=None, *toTime*=None)

varNames ()

returns a list of internal variables. intended to be used in the Table.__init__

```

varRowUnits ()
varDict ()
__class__
    alias of type
__delattr__
    Implement delattr(self, name).
__dir__ () → list
    default dir() implementation
__eq__
    Return self==value.
__format__ ()
    default object formatter
__ge__
    Return self>=value.
__getattr__
    Return getattr(self, name).
__gt__
    Return self>value.
__hash__
    Return hash(self).
__le__
    Return self<=value.
__lt__
    Return self<value.
__ne__
    Return self!=value.
__new__ ()
    Create and return a new object. See help(type) for accurate signature.
__reduce__ ()
    helper for pickle
__reduce_ex__ ()
    helper for pickle
__repr__
    Return repr(self).
__setattr__
    Implement setattr(self, name, value).
__sizeof__ () → int
    size of object in memory, in bytes
__str__
    Return str(self).

```

class `Goulib.motion.TimeDiagram` (*actuators, stateMachines=[], fromTime=None, toTime=None*)
 Bases: `Goulib.plot.Plot`

Params `stateMachines` [(stateMachine,pos,posShift),...]

__init__ (*actuators, stateMachines=[], fromTime=None, toTime=None*)
 Params *stateMachines* [(stateMachine,pos,posShift),...]

__repr__ ()

saveAsCsv (*filename*)

__class__
 alias of *type*

__delattr__
 Implement delattr(self, name).

__dir__ () → list
 default dir() implementation

__eq__
 Return self==value.

__format__ ()
 default object formatter

__ge__
 Return self>=value.

__getattr__
 Return getattr(self, name).

__gt__
 Return self>value.

__hash__
 Return hash(self).

__le__
 Return self<=value.

__lt__
 Return self<value.

__ne__
 Return self!=value.

__new__ ()
 Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
 helper for pickle

__reduce_ex__ ()
 helper for pickle

__setattr__
 Implement setattr(self, name, value).

__sizeof__ () → int
 size of object in memory, in bytes

__str__
 Return str(self).

html (***kwargs*)

plot (***kwargs*)
 renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

render (*fmt='svg', **kwargs*)

save (*filename, **kwargs*)

svg (***kwargs*)

Goulib.motion.**ramp** (*dp, v0, v1, a*)

Parameters

- **dp** – float delta position or None if unknown
- **v0** – float initial velocity or None if unknown
- **v1** – float final velocity or None if unknown
- **a** – float acceleration

Returns float shortest time to accelerate between constraints

Goulib.motion.**trapeze** (*dp, vmax, a, v0=0, v2=0*)

Parameters

- **dp** – float delta position
- **vmax** – float maximal velocity
- **a** – float acceleration
- **v0** – float initial velocity, 0 by default
- **v2** – float final velocity, 0 by default

Returns tuple of 6 values:

- time at end of acceleration
- position at end of acceleration
- velocity at end of acceleration
- time at begin of deceleration
- position at begin of deceleration
- total time

Goulib.motion.**Segment2ndDegree** (*t0, t1, start, end=None*)
 calculates a constant acceleration Segment between start and end

Parameters

- **t0, t1** – float start,end time. one of both may be None for undefined
- **start** – (position, velocity, acceleration) float tuple. some values may be None for undefined
- **end** – (position, velocity, acceleration) float tuple. some values may be None for undefined

Returns *SegmentPoly*

the function can cope with almost any combination of defined/undefined parameters, among others (see tests):

- `Segment2ndDegree(t0,t1,(p0,v0),p1)` # time interval and start + end positions + initial speed
- `Segment2ndDegree(t0,t1,(p0,v0,a))` # time interval and start with acceleration
- `Segment2ndDegree(t0,t1,None,(p1,v1,a))` # time interval and end pva
- `Segment2ndDegree(t0,None,(p0,v0),(p1,v1))` # start + end positions + velocities
- `Segment2ndDegree(t0,None,(p0,v0,a),(None,v1))` # start pva + end velocity
- `Segment2ndDegree(None,t1,p0,(p1,v1,a))` # end pva + start position

the function also accepts some combinations of overconstraining parameters:

- `Segment2ndDegree(t0,t1,(p0,v0,a),p1)` # time interval, start pva, end position => adjust t1
- `Segment2ndDegree(t0,t1,(p0,v0,a),(None,v1))` # time interval, start pva, v1=max vel => adjust t1

Raises `ValueError` – when not enough parameters are specified to define the Segment univoquely

`Goulib.motion.Segment4thDegree (t0, t1, start, end)`

smooth trajectory from an initial position and initial speed (p0,v0) to a final position and speed (p1,v1) * if t1<=t0, t1 is calculated

`Goulib.motion.SegmentsTrapezoidalSpeed (t0, p0, p3, a, T=0, vmax=inf, v0=0, v3=0)`

Parameters

- **t0** – float start time
- **p0** – float start position
- **p3** – float end position
- **a** – float specified acceleration. if =0, use specified time
- **T** – float specified time. if =0 (default), use specified acceleration
- **vmax** – float max speed. default is infinity (i.e. triangular speed)
- **v0** – initial speed
- **v3** – final speed if T <> 0 then $v3 = v0 \cdot v1 + \frac{v0 \cdot v1}{T} \cdot T$

v0 +

|||

t0 t1 t2 t3

Goulib.optim module

various optimization algorithms : knapsack, traveling salesman, simulated annealing, differential evolution

class `Goulib.optim.ObjectiveFunction (objective_function)`

Bases: `object`

class to wrap an objective function and keep track of the best solution evaluated

`__init__ (objective_function)`

`__call__ (solution)`

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__le__
Return `self<=value`.

__lt__
Return `self<value`.

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__
Return `repr(self)`.

__setattr__
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return `str(self)`.

`Goulib.optim.nelder_mead` (*f*, *x_start*, *step=0.1*, *no_improve_thr=1e-05*, *no_improv_break=10*,
max_iter=0, *alpha=1.0*, *gamma=2.0*, *rho=-0.5*, *sigma=0.5*)

Pure Python implementation of the Nelder-Mead algorithm. also called “downhill simplex method” taken from <https://github.com/fchollet/nelder-mead>

Reference: https://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method ;param *f*: function to optimize, must return a scalar score

and operate over a numpy array of the same dimensions as `x_start`

Parameters

- **x_start** – (numpy array) initial position
- **step** – (float) look-around radius in initial step
- **no_improv_break** (*no_improv_thr*) – (float,int): break after `no_improv_break` iterations with an improvement lower than `no_improv_thr`
- **max_iter** – (int): always break after this number of iterations. Set it to 0 to loop indefinitely.
- **gamma, rho, sigma** (*alpha*) – (floats): parameters of the algorithm (see Wikipedia page for reference)

class `Goulib.optim.BinDict` (*capacity, f=<function _Bin.<lambda>>*)

Bases: `Goulib.optim._Bin, dict`

a container with a limited capacity :param capacity: int,float,tuple of whatever defines the capacity of the Bin :param f: function f(x) returning the capacity used by item x. Must return the empty capacity when f(0) is called

`__isub__` (*key*)

`__delitem__` (*key*)

`__iadd__` (*key, item*)

`__setitem__` (*key, item*)

`__class__`

alias of `type`

`__contains__` ()

True if D has a key k, else False.

`__delattr__`

Implement `delattr(self, name)`.

`__dir__` () → list

default `dir()` implementation

`__eq__`

Return `self==value`.

`__format__` ()

default object formatter

`__ge__`

Return `self>=value`.

`__getattr__`

Return `getattr(self, name)`.

`__getitem__` ()

`x.__getitem__(y) <==> x[y]`

`__gt__`

Return `self>value`.

`__hash__` = None

__init__ (*capacity*, *f*=<function *_Bin*.<lambda>>)
 a container with a limited capacity :param capacity: int,float,tuple of whatever defines the capacity of the Bin :param f: function f(x) returning the capacity used by item x. Must return the empty capacity when f(0) is called

__iter__
 Implement iter(self).

__le__
 Return self<=value.

__len__
 Return len(self).

__lt__
 Return self<value.

__ne__
 Return self!=value.

__new__ ()
 Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
 helper for pickle

__reduce_ex__ ()
 helper for pickle

__repr__ ()

__setattr__
 Implement setattr(self, name, value).

__sizeof__ () → size of D in memory, in bytes

__str__
 Return str(self).

clear () → None. Remove all items from D.

copy () → a shallow copy of D

fits (*item*)
Returns bool True if item fits in bin without exceeding capacity

fromkeys ()
 Returns a new dict with keys from iterable and values equal to value.

get (*k*, *d*) → D[k] if k in D, else d. d defaults to None.

items () → a set-like object providing a view on D's items

keys () → a set-like object providing a view on D's keys

pop (*k*, *d*) → v, remove specified key and return the corresponding value.
 If key is not found, d is returned if given, otherwise KeyError is raised

popitem () → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise KeyError if D is empty.

setdefault (*k*, *d*) → D.get(k,d), also set D[k]=d if k not in D

size ()

update ($[E]$, $**F$) \rightarrow None. Update D from dict/iterable E and F.

If E is present and has a `.keys()` method, then does: for k in E: D[k] = E[k] If E is present and lacks a `.keys()` method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

values () \rightarrow an object providing a view on D's values

class Goulib.optim.**BinList** (*capacity, f=<function _Bin.<lambda>>*)

Bases: Goulib.optim._Bin, list

a container with a limited capacity :param capacity: int,float,tuple of whatever defines the capacity of the Bin
:param f: function f(x) returning the capacity used by item x. Must return the empty capacity when f(0) is called

__iadd__ (*item*)

append (*item*)

insert (*i, item*)

extend (*more*)

__isub__ (*item*)

remove (*item*)

pop (*i=-1*)

__setitem__ (*i, item*)

called when replacing a value in list

__add__

Return self+value.

__class__

alias of `type`

__contains__

Return key in self.

__delattr__

Implement `delattr(self, name)`.

__delitem__

Delete `self[key]`.

__dir__ () \rightarrow list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__getitem__ ()

`x.__getitem__(y) <==> x[y]`

__gt__

Return `self>value`.

`__hash__ = None`

`__imul__`

Implement `self*=value`.

`__init__` (*capacity*, *f*=<function `_Bin.<lambda>>`>)

a container with a limited capacity :param capacity: int,float,tuple of whatever defines the capacity of the Bin :param f: function f(x) returning the capacity used by item x. Must return the empty capacity when f(0) is called

`__iter__`

Implement `iter(self)`.

`__le__`

Return `self<=value`.

`__len__`

Return `len(self)`.

`__lt__`

Return `self<value`.

`__mul__`

Return `self*value.n`

`__ne__`

Return `self!=value`.

`__new__` ()

Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()

helper for pickle

`__reduce_ex__` ()

helper for pickle

`__repr__` ()

`__reversed__` ()

`L.__reversed__()` – return a reverse iterator over the list

`__rmul__`

Return `self*value`.

`__setattr__`

Implement `setattr(self, name, value)`.

`__sizeof__` ()

`L.__sizeof__()` – size of L in memory, in bytes

`__str__`

Return `str(self)`.

`clear` () → None – remove all items from L

`copy` () → list – a shallow copy of L

`count` (*value*) → integer – return number of occurrences of value

`fits` (*item*)

Returns bool True if item fits in bin without exceeding capacity

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

reverse ()
L.reverse() – reverse *IN PLACE*

size ()

sort (*key=None*, *reverse=False*) → None – stable sort **IN PLACE**

Goulib.optim.**first_fit_decreasing** (*items*, *bins*, *maxbins=0*)
fit items in bins using the “first fit decreasing” method :param items: iterable of items :param bins: iterable of Bin s. Must have at least one Bin :return: list of items that didn’t fit. (bins are filled by side-effect)

Goulib.optim.**hillclimb** (*init_function*, *move_operator*, *objective_function*, *max_evaluations*)
hillclimb until either max_evaluations is reached or we are at a local optima

Goulib.optim.**hillclimb_and_restart** (*init_function*, *move_operator*, *objective_function*,
max_evaluations)
repeatedly hillclimb until max_evaluations is reached

Goulib.optim.**P** (*prev_score*, *next_score*, *temperature*)

Goulib.optim.**kirkpatrick_cooling** (*start_temp*, *alpha*)

Goulib.optim.**anneal** (*init_function*, *move_operator*, *objective_function*, *max_evaluations*, *start_temp*,
alpha)

Goulib.optim.**reversed_sections** (*tour*)
generator to return all possible variations where the section between two cities are swapped

Goulib.optim.**swapped_cities** (*tour*)
generator to create all possible variations where two cities have been swapped

Goulib.optim.**tour_length** (*points*, *dist*, *tour=None*)
generator of point-to-point distances along a tour

Goulib.optim.**tsp** (*points*, *dist*, *max_iterations=100*, *start_temp=None*, *alpha=None*, *close=True*,
rand=True)
Traveling Salesman Problem @see http://en.wikipedia.org/wiki/Travelling_salesman_problem @param points : iterable containing all points @param dist : function returning the distance between 2 points : def dist(a,b): @param max_iterations :max number of optimization steps @param start_temp, alpha : params for the simulated annealing algorithm. if None, hill climbing is used @param close : computes closed TSP. if False, open TSP starting at points[0] @return iterations,score,best : number of iterations used, minimal length found, best path as list of indexes of points

class Goulib.optim.**DifferentialEvolution** (*evaluator*, *population_size=50*, *f=None*, *cr=0.9*,
eps=0.01, *n_cross=1*, *max_iter=10000*, *monitor_cycle=200*, *out=None*, *show_progress=False*,
show_progress_nth_cycle=1, *insert_solution_vector=None*, *dither_constant=0.4*)

Bases: `object`

This is a python implementation of differential evolution taken from http://cci.lbl.gov/cctbx_sources/scitbx/differential_evolution.py

It assumes an evaluator class is passed in that has the following functionality data members:

- n :: The number of parameters domain :: a list [(low,high)]*n
with approximate upper and lower limits for each parameter
- x :: a place holder for a final solution

also a function called ‘target’ is needed. This function should take a parameter vector as input and return a the function to be minimized.

The code below was implemented on the basis of the following sources of information: 1. <http://www.icsi.berkeley.edu/~storn/code.html> 2. http://www.daimi.au.dk/~krink/fec05/articles/JV_ComparativeStudy_CEC04.pdf 3. http://ocw.mit.edu/NR/rdonlyres/Sloan-School-of-Management/15-099Fall2003/A40397B9-E8FB-4B45-A41B-D1F69218901F/0/ses2_storn_price.pdf

The developers of the differential evolution method have this advice: (taken from ref. 1)

If you are going to optimize your own objective function with DE, you may try the following classical settings for the input file first: Choose method e.g. DE/rand/1/bin, set the number of parents NP to 10 times the number of parameters, select weighting factor F=0.8, and crossover constant CR=0.9. It has been found recently that selecting F from the interval [0.5, 1.0] randomly for each generation or for each difference vector, a technique called dither, improves convergence behaviour significantly, especially for noisy objective functions. It has also been found that setting CR to a low value, e.g. CR=0.2 helps optimizing separable functions since it fosters the search along the coordinate axes. On the contrary this choice is not effective if parameter dependence is encountered, something which is frequently occurring in real-world optimization problems rather than artificial test functions. So for parameter dependence the choice of CR=0.9 is more appropriate. Another interesting empirical finding is that raising NP above, say, 40 does not substantially improve the convergence, independent of the number of parameters. It is worthwhile to experiment with these suggestions. Make sure that you initialize your parameter vectors by exploiting their full numerical range, i.e. if a parameter is allowed to exhibit values in the range [-100, 100] it’s a good idea to pick the initial values from this range instead of unnecessarily restricting diversity.

Keep in mind that different problems often require different settings for NP, F and CR (have a look into the different papers to get a feeling for the settings). If you still get misconvergence you might want to try a different method. We mostly use DE/rand/1/... or DE/best/1/... . The crossover method is not so important although Ken Price claims that binomial is never worse than exponential. In case of misconvergence also check your choice of objective function. There might be a better one to describe your problem. Any knowledge that you have about the problem should be worked into the objective function. A good objective function can make all the difference.

Note: NP is called population size in the routine below.) Note: [0.5,1.0] dither is the default behavior unless f is set to a value other than None.

```
__init__(evaluator, population_size=50, f=None, cr=0.9, eps=0.01, n_cross=1, max_iter=10000,
         monitor_cycle=200, out=None, show_progress=False, show_progress_nth_cycle=1, insert_solution_vector=None, dither_constant=0.4)
```

```
optimize()
```

```
make_random_population()
```

```
score_population()
```

```
__class__
    alias of type
```

```
__delattr__
    Implement delattr(self, name).
```

```
__dir__() → list
    default dir() implementation
```

```
__eq__
    Return self==value.
```

```
__format__()
    default object formatter
```

`__ge__`
Return self>=value.

`__getattr__`
Return getattr(self, name).

`__gt__`
Return self>value.

`__hash__`
Return hash(self).

`__le__`
Return self<=value.

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return repr(self).

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__` () → int
size of object in memory, in bytes

`__str__`
Return str(self).

`evolve` ()

Goulib.piecewise module

piecewise-defined functions

class Goulib.piecewise.**Piecewise** (init=[], default=0, start=-inf)

Bases: *Goulib.expr.Expr*

piecewise function defined by a sorted list of (startx, Expr)

`__init__` (init=[], default=0, start=-inf)

`__call__` (x)

returns value of Expr at point x

index (x, v=None)

finds an existing point or insert one and returns its index

`__len__` ()

`__iter__()`
 iterators through discontinuities. take the opportunity to delete redundant tuples

`append(item)`
 appends a (x,y) item. In fact inserts it at correct position and returns the corresponding index

`extend(iterable)`
 appends an iterable of (x,y) values

`__getitem__(i)`

`__str__()`

`iapply(f, right)`
 apply function to self

`apply(f, right=None)`
 apply function to copy of self

`applx(f)`
 apply a function to each x value

`__lshift__(dx)`

`__rshift__(dx)`

`__add__(right)`

`__and__(right)`

`__class__`
 alias of `type`

`__delattr__`
 Implement `delattr(self, name)`.

`__dir__()` → list
 default `dir()` implementation

`__div__(right)`

`__eq__(other)`

`__format__()`
 default object formatter

`__ge__`
 Return `self >= value`.

`__getattr__`
 Return `getattr(self, name)`.

`__gt__`
 Return `self > value`.

`__hash__ = None`

`__invert__()`

`__le__`
 Return `self <= value`.

`__lt__(other)`

`__mul__(right)`

`__ne__`
Return self!=value.

`__neg__` ()

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__or__` (right)

`__pow__` (right)

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__` ()

`__rmul__` (right)

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__` () → int
size of object in memory, in bytes

`__sub__` (right)

`__truediv__` (right)

`__xor__` (right)

`html` (**kwargs)

`isconstant`
Returns True if Expr evaluates to a constant number of bool

`latex` ()
Returns string LaTeX formula

`plot` (**kwargs)
renders on IPython Notebook (alias to make usage more straightforward)

`png` (**kwargs)

`points` (min=0, xmax=None, eps=0)
@return x and y for a line plot

`render` (fmt='svg', **kwargs)

`save` (filename, **kwargs)

`svg` (**kwargs)

Goulib.plot module

plotable rich object display on IPython/Jupyter notebooks

class `Goulib.plot.Plot`

Bases: `object`

base class for plotable rich object display on IPython notebooks inspired from http://nbviewer.ipython.org/github/ipython/ipython/blob/3607712653c66d63e0d7f13f073bde8c0f209ba8/docs/examples/notebooks/display_protocol.ipynb

render (*fmt='svg', **kwargs*)

save (*filename, **kwargs*)

png (***kwargs*)

svg (***kwargs*)

html (***kwargs*)

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

__getattr__

Return `getattr(self, name)`.

__gt__

Return `self>value`.

__hash__

Return `hash(self)`.

__init__

Initialize self. See `help(type(self))` for accurate signature.

__le__

Return `self<=value`.

__lt__

Return `self<value`.

__ne__

Return `self!=value`.

__new__ ()

Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()

helper for pickle

`__reduce_ex__()`
helper for pickle

`__repr__`
Return repr(self).

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return str(self).

`Goulib.plot.render` (*plotables*, *fmt*=`'svg'`, ***kwargs*)
renders several Plot objects

`Goulib.plot.png` (*plotables*, ***kwargs*)

`Goulib.plot.svg` (*plotables*, ***kwargs*)

`Goulib.plot.plot` (*plotables*, ***kwargs*)

`Goulib.plot.save` (*plotables*, *filename*, ***kwargs*)

Goulib.polynomial module

simple manipulation of polynomials (without SymPy) see <http://docs.sympy.org/dev/modules/polys/reference.html> if you need more ...

class `Goulib.polynomial.Polynomial` (*val*)

Bases: `Goulib.expr.Expr`

Param *val* can be:

- an iterable of the factors in ascending powers order : `Polynomial([1,2,3])` holds $3*x^2+2*x+1$
- a string of the form “ $ax^n + b*x^m + \dots + c x + d$ ” where a,b,c,d, are floats and n,m ... are integers the ‘x’ variable name is fixed, and the spaces and ‘*’ chars are optional. terms can be in any order, and even “overlap” : `Polynomial('3x+x^2-x')` holds x^2+2*x

`__init__` (*val*)

Param *val* can be:

- an iterable of the factors in ascending powers order : `Polynomial([1,2,3])` holds $3*x^2+2*x+1$
- a string of the form “ $ax^n + b*x^m + \dots + c x + d$ ” where a,b,c,d, are floats and n,m ... are integers the ‘x’ variable name is fixed, and the spaces and ‘*’ chars are optional. terms can be in any order, and even “overlap” : `Polynomial('3x+x^2-x')` holds x^2+2*x

`__lt__` (*other*)

`__eq__` (*other*)

`__add__` (*other*)

`__radd__` (*other*)

`__sub__` (*other*)

```

__rsub__(other)
__mul__(other)
__rmul__(other)
__neg__()
__pow__(e)
integral()
derivative()
__and__(right)
__call__(x=None, **kwargs)
    evaluate the Expr at x OR compose self(x())
__class__
    alias of type
__delattr__
    Implement delattr(self, name).
__dir__() → list
    default dir() implementation
__div__(right)
__format__()
    default object formatter
__ge__
    Return self >= value.
__getattr__
    Return getattr(self, name).
__gt__
    Return self > value.
__hash__ = None
__invert__()
__le__
    Return self <= value.
__lshift__(dx)
__ne__
    Return self != value.
__new__()
    Create and return a new object. See help(type) for accurate signature.
__or__(right)
__reduce__()
    helper for pickle
__reduce_ex__()
    helper for pickle
__repr__()

```

`__rshift__` (*dx*)

`__setattr__`

Implement setattr(self, name, value).

`__sizeof__` () → int

size of object in memory, in bytes

`__str__` ()

`__truediv__` (*right*)

`__xor__` (*right*)

applx (*f*, *var='x'*)

function composition $f \circ \text{self} = \text{self}(f(x))$

apply (*f*, *right=None*)

function composition $\text{self} \circ f = f(\text{self}(x))$

html (***kwargs*)

isconstant

Returns True if Expr evaluates to a constant number of bool

latex ()

Returns string LaTeX formula

plot (***kwargs*)

renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

render (*fmt='svg'*, ***kwargs*)

save (*filename*, ***kwargs*)

svg (***kwargs*)

Goulib.polynomial.**plist** (*term*)

Force term to have the form of a polynomial list

Goulib.polynomial.**peval** (*plist*, *x*, *x2=None*)

Eval the plist at value *x*. If two values are given, the difference between the second and the first is returned. This latter feature is included for the purpose of evaluating definite integrals.

Goulib.polynomial.**integral** (*plist*)

Return a new plist corresponding to the integral of the input plist. This function uses zero as the constant term, which is okay when evaluating a definite integral, for example, but is otherwise ambiguous.

The math forces the coefficients to be turned into floats. Consider importing `__future__` division to simplify this.

Goulib.polynomial.**derivative** (*plist*)

Return a new plist corresponding to the derivative of the input plist.

Goulib.polynomial.**add** (*p1*, *p2*)

Return a new plist corresponding to the sum of the two input plists.

Goulib.polynomial.**sub** (*p1*, *p2*)

Goulib.polynomial.**mult_const** (*p*, *c*)

Return a new plist corresponding to the input plist multiplied by a const

`Goulib.polynomial.multiply(p1, p2)`

Return a new plist corresponding to the product of the two input plists

`Goulib.polynomial.mult_one(p, c, i)`

Return a new plist corresponding to the product of the input plist p with the single term $c \cdot x^i$

`Goulib.polynomial.power(p, e)`

Return a new plist corresponding to the e-th power of the input plist p

`Goulib.polynomial.parse_string(s)`

Do very, very primitive parsing of a string into a plist. 'x' is the only term considered for the polynomial, and this routine can only handle terms of the form: $7x^2 + 6x - 5$ and will choke on seemingly simple forms such as $x^2 \cdot 7 - 1$ or $x^{**2} - 1$

`Goulib.polynomial.tostring(p, **kwargs)`

Convert a plist into a string. This looks overly complex at first, but most of the complexity is caused by special cases.

Goulib.statemachine module

state machines with graph representation

class `Goulib.statemachine.StateDiagram` (*name=None, comment=None, filename=None, directory=None, format=None, engine=None, encoding=None, graph_attr=None, node_attr=None, edge_attr=None, body=None, strict=False*)

Bases: `graphviz.dot.Digraph`

helper to write State Diagrams graph in iPython notebook This library uses Graphviz that has to be installed separately (<http://www.graphviz.org/>)

state (*number, descr, actions, transitions*)

Parameters

- **number** – the state number (int)
- **descr** – a string describing the state
- **actions** – a string describing the various actions. use `
` to make a new line
- **transitions** – a array of tuple (`<new_state>,"condition"`)

`Goulib.statemachine.noPrint(*args)`

class `Goulib.statemachine.Simulation`

Bases: `object`

all simulation should derive from this class that has some helper

`__init__()`

setOutput (*h1=<function noPrint>, h2=<function noPrint>, h3=<function noPrint>, h=<function noPrint>, hinfo=<function noPrint>, hsuccess=<function noPrint>, hwarning=<function noPrint>, herror=<function noPrint>, displayState=<function noPrint>, displayObj=<function noPrint>, displayPlot=<function noPrint>*)

class `Goulib.statemachine.EventLog`

Bases: `object`

log (*stateMachine*)

```
class Goulib.statemachine.StateChangeLog (newState)
    Bases: Goulib.statemachine.EventLog

    __init__ (newState)

class Goulib.statemachine.WaitLog (untilTime, waitForWhat)
    Bases: Goulib.statemachine.EventLog

    __init__ (untilTime, waitForWhat)

class Goulib.statemachine.TooLateLog (pastTime, missedWhat)
    Bases: Goulib.statemachine.EventLog

    __init__ (pastTime, missedWhat)

class Goulib.statemachine.TimeMarker (name)
    Bases: object

    __init__ (name)

    set (time)

    __call__ ()

    __repr__ ()

class Goulib.statemachine.StateMachine (simulation=None, name=None, back-
                                         ground_color='#F5ECCE')
    Bases: object

    __init__ (simulation=None, name=None, background_color='#F5ECCE')

    reset ()
        where all actuators should be declared and other variables

    __reset__ ()

    parseDoc (state, f)

    __call__ (time)
        find the state at time. time must be in seconds

    displayGraph ()

    checkOnTimeAndWait (time, what)
        checks that the self.time < time if this is not the case an error will be logged with the message

    wait (time, cause='unknown cause')

    run (start=0, stops=[], startTime=None, maxSteps=100000, maxTime=<Quantity(1000, 'second')>)
        runs the behavioral simulation :params start: is the starting state of the simulation :params stops: a list of
        states that will stop the simulation (after having simulated this last state) :params startTime: a time to start
        this run if None takes self.time :params maxState: is the number of states being evaluated before the end
        of simulation :params maxTime: is the virtual time at which the simulation ends_in_comment_or_string
        :params displayStates: at every new state, display the state in Notebook as well as the time when entered
        :params displayMove: if True, every actuator.move displays the graph of the move

        returns the time when run finishes

    hinfo (*args)

    hsuccess (*args)

    hwarning (*args)

    herror (*args)
```

lastExitTime (*state*)
display (*fromTime=None, toTime=None*)

Goulib.stats module

very basic statistics functions

`Goulib.stats.mean_var` (*data*)
 mean and variance by stable algorithm :param :return: float (mean, variance) of data uses a stable algo by Knuth

`Goulib.stats.mean` (*data*)

Returns mean of data

`Goulib.stats.avg` (*data*)

Returns mean of data

`Goulib.stats.variance` (*data*)

Returns variance of data, faster (?) if mean is already available

`Goulib.stats.var` (*data*)

Returns variance of data, faster (?) if mean is already available

`Goulib.stats.stddev` (*data*)

Returns standard deviation of data

`Goulib.stats.confidence_interval` (*data, conf=0.95*)

Returns (low,high) bounds of 95% confidence interval of data

`Goulib.stats.median` (*data, is_sorted=False*)

Returns median of data

`Goulib.stats.mode` (*data, is_sorted=False*)

Returns mode (most frequent value) of data

`Goulib.stats.kurtosis` (*data*)

`Goulib.stats.covariance` (*data1, data2*)

`Goulib.stats.stats` (*l*)

Returns min,max,sum,sum2,avg,var of a list

class `Goulib.stats.Stats` (*data=[], mean=None, var=None*)

Bases: `object`

an object that computes mean, variance and modes of data that is appended to it as in a list (but actual values are not stored)

`__init__` (*data=[], mean=None, var=None*)

`__repr__` ()

`append` (*x*)

add data x to Stats

`extend` (*data*)

remove (*data*)
remove data from Stats :param data: value or iterable of values

sum

sum1

sum2

mean

avg

average

mu

variance

var

stddev

sigma

__add__ (*other*)

__sub__ (*other*)

__mul__ (*other*)

__neg__ ()

__pow__ (*n*)

covariance (*other*)

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__le__
Return `self<=value`.

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__()`
Create and return a new object. See help(type) for accurate signature.

`__reduce__()`
helper for pickle

`__reduce_ex__()`
helper for pickle

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__()` → int
size of object in memory, in bytes

`__str__`
Return str(self).

class `Goulib.stats.Discrete` (*data*)

Bases: `Goulib.stats.Stats`

discrete probability density function

Parameters `data` – can be:

- list of equiprobable values (uniform distribution)
- dict of x:p values:probability pairs

`__init__` (*data*)

Parameters `data` – can be:

- list of equiprobable values (uniform distribution)
- dict of x:p values:probability pairs

`__call__` (*x*)

`__add__` (*other*)

`__class__`
alias of `type`

`__delattr__`
Implement delattr(self, name).

`__dir__` () → list
default dir() implementation

`__eq__`
Return self==value.

`__format__` ()
default object formatter

`__ge__`
Return self>=value.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self > value`.

`__hash__`
Return `hash(self)`.

`__le__`
Return `self <= value`.

`__lt__`
Return `self < value`.

`__mul__` (*other*)

`__ne__`
Return `self != value`.

`__neg__` ()

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__pow__` (*n*)

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__` ()

`__setattr__`
Implement `setattr(self, name, value)`.

`__sizeof__` () → int
size of object in memory, in bytes

`__str__`
Return `str(self)`.

`__sub__` (*other*)

`append` (*x*)
add data *x* to Stats

`average`

`avg`

`covariance` (*other*)

`extend` (*data*)

`mean`

`mu`

`remove` (*data*)
remove data from Stats :param *data*: value or iterable of values

`sigma`

`stddev`

sum
sum1
sum2
var
variance

```
class Goulib.stats.PDF(pdf, data=[])
    Bases: Goulib.expr.Expr, Goulib.stats.Stats
    probability density function
    __init__(pdf, data=[])
    __call__(x=None, **kwargs)
    __add__(right)
    __and__(right)
    __class__
        alias of type
    __delattr__
        Implement delattr(self, name).
    __dir__() → list
        default dir() implementation
    __div__(right)
    __eq__(other)
    __format__()
        default object formatter
    __ge__
        Return self>=value.
    __getattr__
        Return getattr(self, name).
    __gt__
        Return self>value.
    __hash__ = None
    __invert__()
    __le__
        Return self<=value.
    __lshift__(dx)
    __lt__(other)
    __mul__(right)
    __ne__
        Return self!=value.
    __neg__()
    __new__()
        Create and return a new object. See help(type) for accurate signature.
```

`__or__` (*right*)

`__pow__` (*right*)

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__` ()

`__rmul__` (*right*)

`__rshift__` (*dx*)

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__` () → int
size of object in memory, in bytes

`__str__` ()

`__sub__` (*right*)

`__truediv__` (*right*)

`__xor__` (*right*)

append (*x*)
add data x to Stats

applx (*f*, *var='x'*)
function composition f o self = self(f(x))

apply (*f*, *right=None*)
function composition self o f = f(self(x))

average

avg

covariance (*other*)

extend (*data*)

html (***kwargs*)

isconstant
Returns True if Expr evaluates to a constant number of bool

latex ()
Returns string LaTeX formula

mean

mu

plot (***kwargs*)
renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

remove (*data*)
remove data from Stats :param data: value or iterable of values

render (*fmt='svg', **kwargs*)

save (*filename, **kwargs*)

sigma

stddev

sum

sum1

sum2

svg (***kwargs*)

var

variance

`Goulib.stats.normal_pdf(x, mu, sigma)`

Return the probability density function at x

class `Goulib.stats.Normal(data=[], mean=0, var=1)`

Bases: `Goulib.stats.PDF`

represents a normal distributed variable the base class (list) optionally contains data

if data is specified, it it used to fit a normal law

`__init__(data=[], mean=0, var=1)`

if data is specified, it it used to fit a normal law

sigma

`__str__()`

latex ()

linear (*a, b=0*)

Returns $a*\text{self}+b$

`__mul__(a)`

`__div__(a)`

`__truediv__(a)`

`__add__(other)`

`__radd__(other)`

`__neg__()`

`__sub__(other)`

`__rsub__(other)`

covariance (*other*)

cov (*other*)

pearson (*other*)

correlation (*other*)

corr (*other*)

`__and__(right)`

__call__ (*x=None, **kwargs*)

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__ (*other*)

__format__ ()
default object formatter

__ge__
Return `self >= value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self > value`.

__hash__ = None

__invert__ ()

__le__
Return `self <= value`.

__lshift__ (*dx*)

__lt__ (*other*)

__ne__
Return `self != value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__or__ (*right*)

__pow__ (*right*)

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__repr__ ()

__rmul__ (*right*)

__rshift__ (*dx*)

__setattr__
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

__xor__ (*right*)

append (*x*)
add data *x* to Stats

applyx (*f*, *var*='x')
function composition $f \circ \text{self} = \text{self}(f(x))$

apply (*f*, *right*=None)
function composition $\text{self} \circ f = f(\text{self}(x))$

average

avg

extend (*data*)

html (***kwargs*)

isconstant
Returns True if Expr evaluates to a constant number of bool

mean

mu

plot (***kwargs*)
renders on IPython Notebook (alias to make usage more straightforward)

png (***kwargs*)

remove (*data*)
remove data from Stats :param data: value or iterable of values

render (*fmt*='svg', ***kwargs*)

save (*filename*, ***kwargs*)

stddev

sum

sum1

sum2

svg (***kwargs*)

var

variance

Goulib.stats.linear_regression (*x*, *y*, *conf*=None)

Parameters

- **x**, **y** – iterable data
- **conf** – float confidence level [0..1]. If None, confidence intervals are not returned

Returns b0,b1,b2, (b0

Return the linear regression parameters and their <prob> confidence intervals.

ex: >>> linear_regression([1,2,3],[10,11,11.5],0.95)

Goulib.table module

“mini pandas.DataFrame” Table class with Excel + CSV I/O, easy access to columns, HTML output, and much more.

Goulib.table.**attr**(args)

class Goulib.table.**Cell**(data=None, align=None, fmt=None, tag=None, style={})

Bases: `object`

Table cell with HTML attributes

Parameters

- **data** – cell value(s) of any type
- **align** – string for HTML align attribute
- **fmt** – format string applied applied to data
- **tag** – called to build each cell. defaults to ‘td’
- **style** – dict or string for HTML style attribute

`__init__`(data=None, align=None, fmt=None, tag=None, style={})

Parameters

- **data** – cell value(s) of any type
- **align** – string for HTML align attribute
- **fmt** – format string applied applied to data
- **tag** – called to build each cell. defaults to ‘td’
- **style** – dict or string for HTML style attribute

`__repr__`()

static read(x)

interprets x as int, float, string or None

html(**kwargs)

Returns string HTML formatted cell:

- if data is int, default align=“right”
- if data is float, default align=“right” and fmt=‘%0.2f’
- if data is `timedelta`, align = “right” and formatting is done by `datetime2.strftimedelta()`

`__class__`

alias of `type`

`__delattr__`

Implement `delattr(self, name)`.

`__dir__`() → list

default `dir()` implementation

`__eq__`

Return `self==value`.

__format__ ()
default object formatter

__ge__
Return self>=value.

__getattr__
Return getattr(self, name).

__gt__
Return self>value.

__hash__
Return hash(self).

__le__
Return self<=value.

__lt__
Return self<value.

__ne__
Return self!=value.

__new__ ()
Create and return a new object. See help(type) for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement setattr(self, name, value).

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return str(self).

class Goulib.table.Row (*data, align=None, fmt=None, tag=None, style={}*)

Bases: `object`

Table row with HTML attributes

Parameters

- **data** – (list of) cell value(s) of any type
- **align** – (list of) string for HTML align attribute
- **fmt** – (list of) format string applied applied to data
- **tag** – (list of) tags called to build each cell. defaults to ‘td’
- **style** – (list of) dict or string for HTML style attribute

__init__ (*data, align=None, fmt=None, tag=None, style={}*)

Parameters

- **data** – (list of) cell value(s) of any type
- **align** – (list of) string for HTML align attribute

- **fmt** – (list of) format string applied applied to data
- **tag** – (list of) tags called to build each cell. defaults to ‘td’
- **style** – (list of) dict or string for HTML style attribute

__repr__ ()

html (*cell_args*={}, ***kwargs*)
return in HTML format

__class__
alias of `type`

__delattr__
Implement `delattr(self, name)`.

__dir__ () → list
default `dir()` implementation

__eq__
Return `self==value`.

__format__ ()
default object formatter

__ge__
Return `self>=value`.

__getattr__
Return `getattr(self, name)`.

__gt__
Return `self>value`.

__hash__
Return `hash(self)`.

__le__
Return `self<=value`.

__lt__
Return `self<value`.

__ne__
Return `self!=value`.

__new__ ()
Create and return a new object. See `help(type)` for accurate signature.

__reduce__ ()
helper for pickle

__reduce_ex__ ()
helper for pickle

__setattr__
Implement `setattr(self, name, value)`.

__sizeof__ () → int
size of object in memory, in bytes

__str__
Return `str(self)`.

class `Goulib.table.Table` (*data=[]*, ***kwargs*)

Bases: `list`

Table class with CSV I/O, easy access to columns, HTML output

inits a table, optionally by reading a Excel, csv or html file :param data: list of list of cells, or string as filename
:param titles: optional list of strings used as column id :param footer: optional list of functions used as column reducers

`__init__` (*data=[]*, ***kwargs*)

inits a table, optionally by reading a Excel, csv or html file :param data: list of list of cells, or string as filename :param titles: optional list of strings used as column id :param footer: optional list of functions used as column reducers

`__repr__` ()

Returns repr string of titles+5 first lines

`__str__` ()

Returns string of full tables with linefeeds

`html` (*head=None*, *foot=None*, *colstyle={}*, ***kwargs*)

HTML representation of table

Parameters

- **head** – optional column headers, .titles by default
- **foot** – optional column footers, .footer by default
- **style** – (list of) dict of style attributes
- **kwargs** – optional parameters passed along to tag('table'... except: * start=optional start row * stop=optional end row used to display a subset of lines. in this case rows with '...' cells are displayed before and/or after the lines

Returns string HTML representation of table

`load` (*filename*, ***kwargs*)

`read_element` (*element*, ***kwargs*)

read table from a DOM element. :Warning: drops all formatting

`read_html` (*filename*, ***kwargs*)

read first table in HTML file

`read_json` (*filename*, ***kwargs*)

appends a json file made of lines dictionaries

`read_xls` (*filename*, ***kwargs*)

appends an Excel table

`read_csv` (*filename*, ***kwargs*)

appends a .csv or similar file to the table

`save` (*filename*, ***kwargs*)

`write_xlsx` (*filename*, ***kwargs*)

`json` (***kwargs*)

Returns string JSON representation of table

`write_csv` (*filename*, ***kwargs*)

write the table in Excel csv format, optionally transposed

`__eq__` (*other*)
compare 2 Tables contents, mainly for tests

`ncols` ()
Returns number of columns, ignoring title

`find_col` (*title*)
finds a column from a part of the title

`icol` (*column*)
iterates a column

`col` (*column, title=False*)

`cols` (*title=False*)
iterator through columns

`transpose` (*titles_column=0*)
transpose table :param: *titles_column* :return: Table where rows are made from self's columns and vice-versa

`index` (*value, column=0*)
Returns int row number of first line where column contains value

`__getitem__` (*n*)

`get` (*row, col*)

`set` (*row, col, value*)

`setcol` (*col, value, i=0*)
set column values :param *col*: int or string column index :param *value*: single value assigned to whole column or iterable assigned to each cell :param *i*: optional int : index of first row to assign

`__add__`
Return self+value.

`__class__`
alias of `type`

`__contains__`
Return key in self.

`__delattr__`
Implement `delattr(self, name)`.

`__delitem__`
Delete `self[key]`.

`__dir__` () → list
default `dir()` implementation

`__format__` ()
default object formatter

`__ge__`
Return `self >= value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self > value`.

`__hash__` = None

`__iadd__`
Implement self+=value.

`__imul__`
Implement self*=value.

`__iter__`
Implement iter(self).

`__le__`
Return self<=value.

`__len__`
Return len(self).

`__lt__`
Return self<value.

`__mul__`
Return self*value.n

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__reversed__` ()
L.__reversed__() – return a reverse iterator over the list

`__rmul__`
Return self*value.

`__setattr__`
Implement setattr(self, name, value).

`__setitem__`
Set self[key] to value.

`__sizeof__` ()
L.__sizeof__() – size of L in memory, in bytes

`append` (*line*)
appends a line to table ;param line: can be either: * a list * a dict or column names:values

`clear` () → None – remove all items from L

`copy` () → list – a shallow copy of L

`count` (*value*) → integer – return number of occurrences of value

`extend` (*iterable*) → None – extend list by appending elements from the iterable

`insert` ()
L.insert(index, object) – insert object before index

pop (*[index]*) → item – remove and return item at index (default last).
Raises `IndexError` if list is empty or index is out of range.

remove (*value*) → None – remove first occurrence of value.
Raises `ValueError` if the value is not present.

reverse ()
L.reverse() – reverse *IN PLACE*

addcol (*title, val=None, i=0*)
add column to the right

sort (*by, reverse=False*)
sort by column

rowasdict (*i*)
returns a line as a dict

asdict ()

groupby_gen (*by, sort=True, removecol=True*)
generates subtables

groupby (*by, sort=True, removecol=True*)
ordered dictionary of subtables

hierarchy (*by='Level', factory=<function Table.<lambda>>, linkfct=<function Table.<lambda>>*)
builds a structure from a table containing a “level” column

applyf (*by, f, skiperrors=False*)
apply a function to a column :param by: column name of number :param f: function of the form lambda cell:content :param skiperrors: bool. if True, errors while running f are ignored :return: bool True if ok, False if skiperrors==True and conversion failed

to_datetime (*by, fmt='%Y-%m-%d %H:%M:%S', skiperrors=False*)
convert a column to datetime

to_date (*by, fmt='%Y-%m-%d', skiperrors=False*)
convert a column to date

to_time (*by, fmt='%H:%M:%S', skiperrors=False*)
convert a column to time

to_timedelta (*by, fmt=None, skiperrors=False*)
convert a column to time

total (*funcs*)
build a footer row by applying funcs to all columns

remove_lines_where (*f, value=(None, 0, '')*)
Parameters *f* – function of the form lambda line:bool returning True if line should be removed
Returns int number of lines removed

Goulib.tests module

utilities for unit tests (using nose)

Goulib.tests.pprint_gen (*iterable, indices=[0, 1, 2, -3, -2, -1], sep='...'*)
generates items at specified indices

Goulib.tests.pprint (*iterable, indices=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -3, -2, -1], timeout=1*)

class Goulib.tests.TestCase (*methodName='runTest'*)

Bases: unittest.case.TestCase

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

assertSequenceEqual (*seq1, seq2, msg=None, seq_type=None, places=7, delta=None, reltol=None*)

An equality assertion for ordered sequences (like lists and tuples). constraints on seq1,seq2 from unittest.TestCase.assertSequenceEqual are mostly removed

Parameters

- **seq2** (*seq1,*) – iterables to compare for (quasi) equality
- **msg** – optional string message to use on failure instead of a list of differences
- **places** – int number of digits to consider in float comparisons. If None, enforces strict equality
- **delta** – optional float absolute tolerance value
- **reltol** – optional float relative tolerance value

base_types = ((<class 'int'>), (<class 'str'>), <class 'str'>, <class 'bool'>, <class 'set'>, <class 'dict'>)

assertEqual (*first, second, places=7, msg=None, delta=None, reltol=None*)

automatically calls assertAlmostEqual when needed :param first, second: objects to compare for (quasi) equality :param places: int number of digits to consider in float comparisons.

If None, forces strict equality

Parameters

- **msg** – optional string error message to display in cas of failure
- **delta** – optional float absolute tolerance value
- **reltol** – optional float relative tolerance value

assertCountEqual (*seq1, seq2, msg=None*)

compare iterables converted to sets : order has no importance

__call__ (**args, **kwds*)

__class__

alias of `type`

__delattr__

Implement delattr(self, name).

__dir__ () → list

default dir() implementation

__eq__ (*other*)

__format__ ()

default object formatter

__ge__

Return self>=value.

__getattr__

Return getattr(self, name).

`__gt__`
Return self>value.

`__hash__` ()

`__init__` (*methodName='runTest'*)
Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`__le__`
Return self<=value.

`__lt__`
Return self<value.

`__ne__`
Return self!=value.

`__new__` ()
Create and return a new object. See help(type) for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__` ()

`__setattr__`
Implement setattr(self, name, value).

`__sizeof__` () → int
size of object in memory, in bytes

`__str__` ()

addCleanup (*function, *args, **kwargs*)
Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

addTypeEqualityFunc (*typeobj, function*)
Add a type specific assertEquals style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

- typeobj:** The data type to call this function on when both values are of the same type in assertEquals().
- function:** The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

assertAlmostEqual (*first, second, places=None, msg=None, delta=None*)
Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

If the two objects compare equal then they will automatically compare almost equal.

assertAlmostEquals (**args, **kwargs*)

assertDictContainsSubset (*subset, dictionary, msg=None*)

Checks whether dictionary is a superset of subset.

assertDictEqual (*d1, d2, msg=None*)

assertEquals (**args, **kwargs*)

assertFalse (*expr, msg=None*)

Check that the expression is false.

assertGreater (*a, b, msg=None*)

Just like self.assertTrue(a > b), but with a nicer default message.

assertGreaterEqual (*a, b, msg=None*)

Just like self.assertTrue(a >= b), but with a nicer default message.

assertIn (*member, container, msg=None*)

Just like self.assertTrue(a in b), but with a nicer default message.

assertIs (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is b), but with a nicer default message.

assertIsInstance (*obj, cls, msg=None*)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

assertIsNone (*obj, msg=None*)

Same as self.assertTrue(obj is None), with a nicer default message.

assertIsNot (*expr1, expr2, msg=None*)

Just like self.assertTrue(a is not b), but with a nicer default message.

assertIsNotNone (*obj, msg=None*)

Included for symmetry with assertIsNone.

assertLess (*a, b, msg=None*)

Just like self.assertTrue(a < b), but with a nicer default message.

assertLessEqual (*a, b, msg=None*)

Just like self.assertTrue(a <= b), but with a nicer default message.

assertListEqual (*list1, list2, msg=None*)

A list-specific equality assertion.

Args: list1: The first list to compare. list2: The second list to compare. msg: Optional message to use on failure instead of a list of

differences.

assertLogs (*logger=None, level=None*)

Fail unless a log message of level *level* or higher is emitted on *logger_name* or its children. If omitted, *level* defaults to INFO and *logger* defaults to the root logger.

This method must be used as a context manager, and will yield a recording object with two attributes: *output* and *records*. At the end of the context manager, the *output* attribute will be a list of the matching formatted log messages and the *records* attribute will be a list of the corresponding LogRecord objects.

Example:

```
with self.assertLogs('foo', level='INFO') as cm:
    logging.getLogger('foo').info('first message')
    logging.getLogger('foo.bar').error('second message')
```

```
self.assertEqual(cm.output, ['INFO:foo:first message',
                             'ERROR:foo.bar:second message'])
```

assertMultiLineEqual (*first, second, msg=None*)

Assert that two multi-line strings are equal.

assertNotAlmostEqual (*first, second, places=None, msg=None, delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most significant digit).

Objects that are equal automatically fail.

assertNotAlmostEqual (**args, **kwargs*)

assertNotEqual (*first, second, msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

assertNotEquals (**args, **kwargs*)

assertNotIn (*member, container, msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

assertNotIsInstance (*obj, cls, msg=None*)

Included for symmetry with assertIsInstance.

assertNotRegex (*text, unexpected_regex, msg=None*)

Fail the test if the text matches the regular expression.

assertNotRegexMatches (**args, **kwargs*)

assertRaises (*expected_exception, *args, **kwargs*)

Fail unless an exception of class `expected_exception` is raised by the callable when invoked with specified positional and keyword arguments. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with the callable and arguments omitted, will return a context object used like this:

```
with self.assertRaises(SomeException):
    do_something()
```

An optional keyword argument 'msg' can be provided when `assertRaises` is used as a context object.

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

```
with self.assertRaises(SomeException) as cm:
    do_something()
the_exception = cm.exception
self.assertEqual(the_exception.error_code, 3)
```

assertRaisesRegex (*expected_exception, expected_regex, *args, **kwargs*)

Asserts that the message in a raised exception matches a regex.

Args: `expected_exception`: Exception class expected to be raised. `expected_regex`: Regex (re pattern object or string) expected

to be found in error message.

args: Function to be called and extra positional args. kwargs: Extra kwargs. msg: Optional message used in case of failure. Can only be used

when `assertRaisesRegex` is used as a context manager.

assertRaisesRegex (*args, **kwargs)

assertRegex (text, expected_regex, msg=None)

Fail the test unless the text matches the regular expression.

assertRegexMatches (*args, **kwargs)

assertSetEqual (set1, set2, msg=None)

A set-specific equality assertion.

Args: set1: The first set to compare. set2: The second set to compare. msg: Optional message to use on failure instead of a list of

differences.

`assertSetEqual` uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

assertTrue (expr, msg=None)

Check that the expression is true.

assertTupleEqual (tuple1, tuple2, msg=None)

A tuple-specific equality assertion.

Args: tuple1: The first tuple to compare. tuple2: The second tuple to compare. msg: Optional message to use on failure instead of a list of

differences.

assertWarns (expected_warning, *args, **kwargs)

Fail unless a warning of class `warnClass` is triggered by the callable when invoked with specified positional and keyword arguments. If a different type of warning is triggered, it will not be handled: depending on the other warning filtering rules in effect, it might be silenced, printed out, or raised as an exception.

If called with the callable and arguments omitted, will return a context object used like this:

```
with self.assertWarns(SomeWarning):
    do_something()
```

An optional keyword argument 'msg' can be provided when `assertWarns` is used as a context object.

The context manager keeps a reference to the first matching warning as the 'warning' attribute; similarly, the 'filename' and 'lineno' attributes give you information about the line of Python code from which the warning was triggered. This allows you to inspect the warning after the assertion:

```
with self.assertWarns(SomeWarning) as cm:
    do_something()
the_warning = cm.warning
self.assertEqual(the_warning.some_attribute, 147)
```

assertWarnsRegex (expected_warning, expected_regex, *args, **kwargs)

Asserts that the message in a triggered warning matches a regex. Basic functioning is similar to `assertWarns()` with the addition that only warnings whose messages also match the regular expression are considered successful matches.

Args: expected_warning: Warning class expected to be triggered. expected_regex: Regex (re pattern object or string) expected

to be found in error message.

args: Function to be called and extra positional args. kwargs: Extra kwargs. msg: Optional message used in case of failure. Can only be used

when `assertWarnsRegex` is used as a context manager.

assert_ (*args, **kwargs)

countTestCases ()

debug ()

Run the test without collecting errors in a `TestResult`

defaultTestResult ()

doCleanups ()

Execute all cleanup functions. Normally called for you after `tearDown`.

fail (msg=None)

Fail immediately, with the given message.

failIf (*args, **kwargs)

failIfAlmostEqual (*args, **kwargs)

failIfEqual (*args, **kwargs)

failUnless (*args, **kwargs)

failUnlessAlmostEqual (*args, **kwargs)

failUnlessEqual (*args, **kwargs)

failUnlessRaises (*args, **kwargs)

failureException

alias of `AssertionError`

id ()

longMessage = True

maxDiff = 640

run (result=None)

setUp ()

Hook method for setting up the test fixture before exercising it.

setUpClass ()

Hook method for setting up class fixture before running tests in the class.

shortDescription ()

Returns a one-line description of the test, or `None` if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

skipTest (reason)

Skip this test.

subTest (msg=None, **params)

Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

tearDown()

Hook method for deconstructing the test fixture after testing it.

tearDownClass()

Hook method for deconstructing the class fixture after running all tests in the class.

Goulib.tests.**pep8**(*name*)

Goulib.tests.**setlog**(*level=20, fmt='%%(levelname)s:%(filename)s:%(funcName)s: %(message)s'*)
initializes logging :param level: logging level :param fmt: string

Goulib.tests.**runmodule**(*level=20, argv=[]*)

Parameters *argv* – optional list of string with additional options passed to nose.run

see <http://nose.readthedocs.org/en/latest/usage.html>

Goulib.tests.**runtests**(*level=20, argv=[]*)

Parameters *argv* – optional list of string with additional options passed to nose.run

see <http://nose.readthedocs.org/en/latest/usage.html>

Goulib.workdays module

WorkCalendar class with datetime operations on working hours, handling holidays merges and improves BusinessHours and workdays packages

class Goulib.workdays.**WorkCalendar**(*worktime=[datetime.time(0, 0), datetime.time(23, 59, 59, 999999)], parent=[], weekends=(5, 6), holidays=set()*)

Bases: `object`

WorkCalendar class with datetime operations on working hours

__init__(*worktime=[datetime.time(0, 0), datetime.time(23, 59, 59, 999999)], parent=[], weekends=(5, 6), holidays=set()*)

start

end

setworktime(*worktime*)

addholidays(*days*)

add day(s) to to known holidays. dates with year==4 (to allow Feb 29th) apply every year note : holidays set may contain weekends too.

isworkday(*day*)

@return True if day is a work day

isworktime(*time*)

@return True if you're supposed to work at that time

nextworkday(*day*)

@return next work day

prevworkday(*day*)

@return previous work day

range(*start, end*)

range of workdays between start (included) and end (not included)

workdays(*start_date, ndays*)

list of ndays workdays from start

workday (*start_date*, *ndays*)

Same as Excel WORKDAY function. Returns a date that is the indicated number of working days before or after the starting date. Working days exclude weekends and any dates identified as holidays. Use WORKDAY to exclude weekends or holidays when you calculate invoice due dates, expected delivery times, or the number of days of work performed.

cast (*time*, *retro=False*)

force time to be in workhours

worktime (*day*)

@return interval of time worked a given day

workdatetime (*day*)

@return interval of datetime worked a given day

diff (*t1*, *t2*)

@return timedelta worktime between t1 and t2 (= t2-t1)

gethours (*t1*, *t2*)

@return fractional work hours between t1 and t2 (= t2-t1)

plus (*start*, *t*)

@return start time + t work time (positive or negative)

minus (*start*, *t*)

@return start time - t work time (positive or negative)

networkdays (*start_date*, *end_date*)

Same as Excel NETWORKDAYS function. Returns the number of whole working days between start_date and end_date (inclusive of both start_date and end_date). Working days exclude weekends and any dates identified in holidays. Use NETWORKDAYS to calculate employee benefits that accrue based on the number of days worked during a specific term

FRI = 4

MON = 0

SAT = 5

SUN = 6

THU = 3

TUE = 1

WED = 2

__class__

alias of `type`

__delattr__

Implement `delattr(self, name)`.

__dir__ () → list

default `dir()` implementation

__eq__

Return `self==value`.

__format__ ()

default object formatter

__ge__

Return `self>=value`.

`__getattr__`
Return `getattr(self, name)`.

`__gt__`
Return `self > value`.

`__hash__`
Return `hash(self)`.

`__le__`
Return `self <= value`.

`__lt__`
Return `self < value`.

`__ne__`
Return `self != value`.

`__new__` ()
Create and return a new object. See `help(type)` for accurate signature.

`__reduce__` ()
helper for pickle

`__reduce_ex__` ()
helper for pickle

`__repr__`
Return `repr(self)`.

`__setattr__`
Implement `setattr(self, name, value)`.

`__sizeof__` () → int
size of object in memory, in bytes

`__str__`
Return `str(self)`.

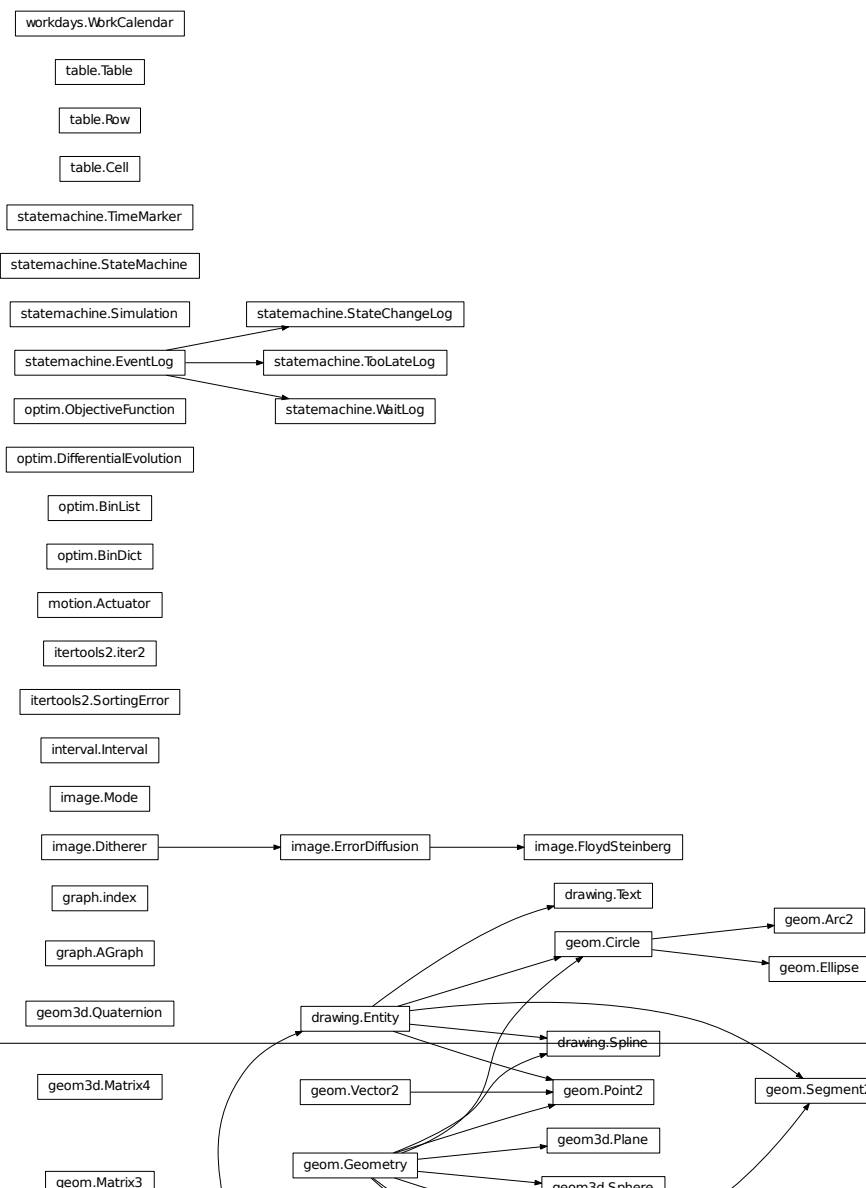
`Goulib.workdays.FullTime` = <**`Goulib.workdays.WorkCalendar` object**>
compatibility with <http://pypi.python.org/pypi/BusinessHours>

`Goulib.workdays.workday` (*start_date*, *ndays*, *holidays*=[])
Same as Excel WORKDAY function. Returns a date that is the indicated number of working days before or after the starting date. Working days exclude weekends and any dates identified as holidays. Use WORKDAY to exclude weekends or holidays when you calculate invoice due dates, expected delivery times, or the number of days of work performed.

`Goulib.workdays.networkdays` (*start_date*, *end_date*, *holidays*=[])
Same as Excel NETWORKDAYS function. Returns the number of whole working days between *start_date* and *end_date* (inclusive of both *start_date* and *end_date*). Working days exclude weekends and any dates identified in holidays. Use NETWORKDAYS to calculate employee benefits that accrue based on the number of days worked during a specific term

CHAPTER 3

Classes



CHAPTER 4

Indices and tables

- genindex
- modindex
- search
- changes

g

Goulib.colors, 6
Goulib.container, 11
Goulib.datetime2, 17
Goulib.decorators, 27
Goulib.drawing, 33
Goulib.expr, 62
Goulib.geom, 67
Goulib.geom3d, 98
Goulib.graph, 109
Goulib.image, 148
Goulib.interval, 158
Goulib.itertools2, 166
Goulib.markup, 175
Goulib.math2, 191
Goulib.motion, 203
Goulib.optim, 214
Goulib.piecewise, 222
Goulib.plot, 224
Goulib.polynomial, 226
Goulib.statemachine, 229
Goulib.stats, 231
Goulib.table, 240
Goulib.tests, 246
Goulib.workdays, 253

Symbols

- `__abs__` (Goulib.datetime2.timedelta2 attribute), 23
- `__abs__()` (Goulib.geom.Arc2 method), 87
- `__abs__()` (Goulib.geom.Circle method), 84
- `__abs__()` (Goulib.geom.Ellipse method), 90
- `__abs__()` (Goulib.geom.Matrix3 method), 98
- `__abs__()` (Goulib.geom.Point2 method), 74
- `__abs__()` (Goulib.geom.Segment2 method), 81
- `__abs__()` (Goulib.geom.Vector2 method), 73
- `__abs__()` (Goulib.geom3d.Quaternion method), 106
- `__abs__()` (Goulib.geom3d.Segment3 method), 101
- `__abs__()` (Goulib.geom3d.Vector3 method), 99
- `__abs__()` (Goulib.image.Image method), 152
- `__abstractmethods__` (Goulib.drawing.Chain attribute), 32, 49
- `__abstractmethods__` (Goulib.drawing.Drawing attribute), 33, 58
- `__abstractmethods__` (Goulib.drawing.Group attribute), 30, 42
- `__abstractmethods__` (Goulib.drawing.Instance attribute), 31, 46
- `__abstractmethods__` (Goulib.drawing.Rect attribute), 32, 52
- `__abstractmethods__` (Goulib.drawing.Spline attribute), 30, 39
- `__abstractmethods__` (Goulib.geom.Arc2 attribute), 87
- `__abstractmethods__` (Goulib.geom.Circle attribute), 84
- `__abstractmethods__` (Goulib.geom.Ellipse attribute), 90
- `__abstractmethods__` (Goulib.geom.Geometry attribute), 68
- `__abstractmethods__` (Goulib.geom.Line2 attribute), 79
- `__abstractmethods__` (Goulib.geom.Point2 attribute), 74
- `__abstractmethods__` (Goulib.geom.Ray2 attribute), 80
- `__abstractmethods__` (Goulib.geom.Segment2 attribute), 81
- `__abstractmethods__` (Goulib.geom3d.Line3 attribute), 101
- `__abstractmethods__` (Goulib.geom3d.Plane attribute), 103
- `__abstractmethods__` (Goulib.geom3d.Point3 attribute), 100
- `__abstractmethods__` (Goulib.geom3d.Ray3 attribute), 101
- `__abstractmethods__` (Goulib.geom3d.Segment3 attribute), 101
- `__abstractmethods__` (Goulib.geom3d.Sphere attribute), 102
- `__abstractmethods__` (Goulib.itertools2.keep attribute), 173
- `__add__` (Goulib.datetime2.date2 attribute), 20
- `__add__` (Goulib.datetime2.datetime2 attribute), 17
- `__add__` (Goulib.datetime2.timedelta2 attribute), 23
- `__add__` (Goulib.drawing.Chain attribute), 49
- `__add__` (Goulib.drawing.Drawing attribute), 58
- `__add__` (Goulib.drawing.Group attribute), 42
- `__add__` (Goulib.drawing.Rect attribute), 52
- `__add__` (Goulib.optim.BinList attribute), 218
- `__add__` (Goulib.table.Table attribute), 244
- `__add__()` (Goulib.colors.Color method), 7
- `__add__()` (Goulib.container.Sequence method), 16
- `__add__()` (Goulib.drawing.BBox method), 35
- `__add__()` (Goulib.expr.Expr method), 63
- `__add__()` (Goulib.geom.Point2 method), 74
- `__add__()` (Goulib.geom.Vector2 method), 72
- `__add__()` (Goulib.geom3d.Vector3 method), 98
- `__add__()` (Goulib.image.Image method), 153
- `__add__()` (Goulib.interval.Box method), 164
- `__add__()` (Goulib.interval.Interval method), 160
- `__add__()` (Goulib.interval.Intervals method), 162
- `__add__()` (Goulib.itertools2.iter2 method), 170
- `__add__()` (Goulib.piecewise.Piecewise method), 223
- `__add__()` (Goulib.polynomial.Polynomial method), 226
- `__add__()` (Goulib.stats.Discrete method), 233
- `__add__()` (Goulib.stats.Normal method), 237
- `__add__()` (Goulib.stats.PDF method), 235
- `__add__()` (Goulib.stats.Stats method), 232
- `__and__()` (Goulib.container.Sequence method), 16
- `__and__()` (Goulib.expr.Expr method), 63
- `__and__()` (Goulib.piecewise.Piecewise method), 223

- `__and__()` (Goulib.polynomial.Polynomial method), 227
- `__and__()` (Goulib.stats.Normal method), 237
- `__and__()` (Goulib.stats.PDF method), 235
- `__bool__` (Goulib.datetime2.timedelta2 attribute), 23
- `__bool__()` (Goulib.geom3d.Vector3 method), 98
- `__bool__()` (Goulib.graph.DiGraph method), 129
- `__bool__()` (Goulib.graph.GeoGraph method), 114
- `__call__()` (Goulib.drawing.BBox method), 28, 34
- `__call__()` (Goulib.expr.Expr method), 63
- `__call__()` (Goulib.geom.Matrix3 method), 96
- `__call__()` (Goulib.geom3d.Matrix4 method), 103
- `__call__()` (Goulib.image.Ditherer method), 155
- `__call__()` (Goulib.image.ErrorDiffusion method), 156
- `__call__()` (Goulib.image.FloydSteinberg method), 157
- `__call__()` (Goulib.interval.Box method), 163
- `__call__()` (Goulib.interval.Intervals method), 162
- `__call__()` (Goulib.markup.element method), 175
- `__call__()` (Goulib.markup.page method), 177
- `__call__()` (Goulib.motion.PVA method), 203
- `__call__()` (Goulib.motion.Segment method), 204
- `__call__()` (Goulib.motion.SegmentPoly method), 207
- `__call__()` (Goulib.motion.Segments method), 206
- `__call__()` (Goulib.optim.ObjectiveFunction method), 214
- `__call__()` (Goulib.piecewise.Piecewise method), 222
- `__call__()` (Goulib.polynomial.Polynomial method), 227
- `__call__()` (Goulib.statemachine.StateMachine method), 230
- `__call__()` (Goulib.statemachine.TimeMarker method), 230
- `__call__()` (Goulib.stats.Discrete method), 233
- `__call__()` (Goulib.stats.Normal method), 237
- `__call__()` (Goulib.stats.PDF method), 235
- `__call__()` (Goulib.tests.TestCase method), 247
- `__cause__` (Goulib.itertools2.SortingError attribute), 171
- `__cause__` (Goulib.markup.ArgumentError attribute), 185
- `__cause__` (Goulib.markup.ClosingError attribute), 183
- `__cause__` (Goulib.markup.CustomizationError attribute), 190
- `__cause__` (Goulib.markup.DeprecationError attribute), 187
- `__cause__` (Goulib.markup.InvalidElementError attribute), 186
- `__cause__` (Goulib.markup.MarkupError attribute), 181
- `__cause__` (Goulib.markup.ModeError attribute), 188
- `__cause__` (Goulib.markup.OpeningError attribute), 184
- `__class__` (Goulib.colors.Color attribute), 7
- `__class__` (Goulib.colors.Palette attribute), 9
- `__class__` (Goulib.container.Record attribute), 11
- `__class__` (Goulib.container.Sequence attribute), 16
- `__class__` (Goulib.container.SortedCollection attribute), 14
- `__class__` (Goulib.datetime2.date2 attribute), 20
- `__class__` (Goulib.datetime2.datetime2 attribute), 17
- `__class__` (Goulib.datetime2.time2 attribute), 22
- `__class__` (Goulib.datetime2.timedelta2 attribute), 23
- `__class__` (Goulib.drawing.BBox attribute), 35
- `__class__` (Goulib.drawing.Chain attribute), 49
- `__class__` (Goulib.drawing.Drawing attribute), 58
- `__class__` (Goulib.drawing.Entity attribute), 38
- `__class__` (Goulib.drawing.Group attribute), 42
- `__class__` (Goulib.drawing.Instance attribute), 46
- `__class__` (Goulib.drawing.Rect attribute), 52
- `__class__` (Goulib.drawing.Spline attribute), 39
- `__class__` (Goulib.drawing.Text attribute), 56
- `__class__` (Goulib.expr.Expr attribute), 63
- `__class__` (Goulib.expr.LatexVisitor attribute), 66
- `__class__` (Goulib.expr.TextVisitor attribute), 65
- `__class__` (Goulib.geom.Arc2 attribute), 87
- `__class__` (Goulib.geom.Circle attribute), 84
- `__class__` (Goulib.geom.Ellipse attribute), 90
- `__class__` (Goulib.geom.Geometry attribute), 68
- `__class__` (Goulib.geom.Line2 attribute), 79
- `__class__` (Goulib.geom.Matrix3 attribute), 96
- `__class__` (Goulib.geom.Point2 attribute), 74
- `__class__` (Goulib.geom.Ray2 attribute), 80
- `__class__` (Goulib.geom.Segment2 attribute), 81
- `__class__` (Goulib.geom.Vector2 attribute), 73
- `__class__` (Goulib.graph.AGraph attribute), 113
- `__class__` (Goulib.graph.DiGraph attribute), 129
- `__class__` (Goulib.graph.GeoGraph attribute), 114
- `__class__` (Goulib.graph.index attribute), 112
- `__class__` (Goulib.graph.index.Index attribute), 110
- `__class__` (Goulib.graph.index.Property attribute), 109
- `__class__` (Goulib.image.Ditherer attribute), 155
- `__class__` (Goulib.image.ErrorDiffusion attribute), 156
- `__class__` (Goulib.image.FloydSteinberg attribute), 157
- `__class__` (Goulib.image.Image attribute), 153
- `__class__` (Goulib.image.Mode attribute), 148
- `__class__` (Goulib.interval.Box attribute), 164
- `__class__` (Goulib.interval.Interval attribute), 160
- `__class__` (Goulib.interval.Intervals attribute), 162
- `__class__` (Goulib.itertools2.SortingError attribute), 171
- `__class__` (Goulib.itertools2.iter2 attribute), 170
- `__class__` (Goulib.itertools2.keep attribute), 173
- `__class__` (Goulib.markup.ArgumentError attribute), 185
- `__class__` (Goulib.markup.ClosingError attribute), 183
- `__class__` (Goulib.markup.CustomizationError attribute), 190
- `__class__` (Goulib.markup.DeprecationError attribute), 187
- `__class__` (Goulib.markup.InvalidElementError attribute), 186
- `__class__` (Goulib.markup.MarkupError attribute), 181
- `__class__` (Goulib.markup.ModeError attribute), 188
- `__class__` (Goulib.markup.OpeningError attribute), 184
- `__class__` (Goulib.markup.dummy attribute), 179
- `__class__` (Goulib.markup.element attribute), 175

- `__class__` (Goulib.markup.page attribute), 178
- `__class__` (Goulib.markup.russell attribute), 180
- `__class__` (Goulib.motion.Actuator attribute), 211
- `__class__` (Goulib.motion.PVA attribute), 203
- `__class__` (Goulib.motion.Segment attribute), 204
- `__class__` (Goulib.motion.SegmentPoly attribute), 208
- `__class__` (Goulib.motion.Segments attribute), 206
- `__class__` (Goulib.motion.TimeDiagram attribute), 212
- `__class__` (Goulib.optim.BinDict attribute), 216
- `__class__` (Goulib.optim.BinList attribute), 218
- `__class__` (Goulib.optim.DifferentialEvolution attribute), 221
- `__class__` (Goulib.optim.ObjectiveFunction attribute), 214
- `__class__` (Goulib.piecewise.Piecewise attribute), 223
- `__class__` (Goulib.plot.Plot attribute), 225
- `__class__` (Goulib.polynomial.Polynomial attribute), 227
- `__class__` (Goulib.stats.Discrete attribute), 233
- `__class__` (Goulib.stats.Normal attribute), 238
- `__class__` (Goulib.stats.PDF attribute), 235
- `__class__` (Goulib.stats.Stats attribute), 232
- `__class__` (Goulib.table.Cell attribute), 240
- `__class__` (Goulib.table.Row attribute), 242
- `__class__` (Goulib.table.Table attribute), 244
- `__class__` (Goulib.tests.TestCase attribute), 247
- `__class__` (Goulib.workdays.WorkCalendar attribute), 254
- `__contains__` (Goulib.drawing.Chain attribute), 49
- `__contains__` (Goulib.drawing.Drawing attribute), 58
- `__contains__` (Goulib.drawing.Group attribute), 42
- `__contains__` (Goulib.drawing.Rect attribute), 52
- `__contains__` (Goulib.optim.BinList attribute), 218
- `__contains__` (Goulib.table.Table attribute), 244
- `__contains__` (Goulib.colors.Palette method), 9
- `__contains__` (Goulib.container.Record method), 11
- `__contains__` (Goulib.container.Sequence method), 16
- `__contains__` (Goulib.container.SortedCollection method), 14
- `__contains__` (Goulib.drawing.BBox method), 28, 34
- `__contains__` (Goulib.drawing.Instance method), 46
- `__contains__` (Goulib.drawing.Spline method), 39
- `__contains__` (Goulib.geom.Arc2 method), 87
- `__contains__` (Goulib.geom.Circle method), 84
- `__contains__` (Goulib.geom.Ellipse method), 90
- `__contains__` (Goulib.geom.Geometry method), 68
- `__contains__` (Goulib.geom.Line2 method), 79
- `__contains__` (Goulib.geom.Point2 method), 74
- `__contains__` (Goulib.geom.Ray2 method), 80
- `__contains__` (Goulib.geom.Segment2 method), 81
- `__contains__` (Goulib.geom3d.Sphere method), 102
- `__contains__` (Goulib.graph.DiGraph method), 129
- `__contains__` (Goulib.graph.GeoGraph method), 114
- `__contains__` (Goulib.graph.index.Index method), 110
- `__contains__` (Goulib.interval.Box method), 164
- `__contains__` (Goulib.interval.Interval method), 160
- `__contains__` (Goulib.interval.Intervals method), 162
- `__contains__` (Goulib.markup.russell method), 180
- `__contains__` (Goulib.optim.BinDict method), 216
- `__context__` (Goulib.itertools2.SortingError attribute), 171
- `__context__` (Goulib.markup.ArgumentError attribute), 185
- `__context__` (Goulib.markup.ClosingError attribute), 183
- `__context__` (Goulib.markup.CustomizationError attribute), 190
- `__context__` (Goulib.markup.DeprecationError attribute), 187
- `__context__` (Goulib.markup.InvalidElementError attribute), 186
- `__context__` (Goulib.markup.MarkupError attribute), 181
- `__context__` (Goulib.markup.ModeError attribute), 188
- `__context__` (Goulib.markup.OpeningError attribute), 184
- `__copy__` (Goulib.drawing.Chain method), 49
- `__copy__` (Goulib.drawing.Drawing method), 58
- `__copy__` (Goulib.drawing.Group method), 30, 42
- `__copy__` (Goulib.drawing.Rect method), 52
- `__delattr__` (Goulib.colors.Color attribute), 7
- `__delattr__` (Goulib.colors.Palette attribute), 9
- `__delattr__` (Goulib.container.Record attribute), 11
- `__delattr__` (Goulib.container.Sequence attribute), 16
- `__delattr__` (Goulib.container.SortedCollection attribute), 14
- `__delattr__` (Goulib.datetime2.date2 attribute), 20
- `__delattr__` (Goulib.datetime2.datetime2 attribute), 17
- `__delattr__` (Goulib.datetime2.time2 attribute), 22
- `__delattr__` (Goulib.datetime2.timedelta2 attribute), 24
- `__delattr__` (Goulib.drawing.BBox attribute), 35
- `__delattr__` (Goulib.drawing.Chain attribute), 49
- `__delattr__` (Goulib.drawing.Drawing attribute), 58
- `__delattr__` (Goulib.drawing.Entity attribute), 38
- `__delattr__` (Goulib.drawing.Group attribute), 42
- `__delattr__` (Goulib.drawing.Instance attribute), 46
- `__delattr__` (Goulib.drawing.Rect attribute), 52
- `__delattr__` (Goulib.drawing.Spline attribute), 39
- `__delattr__` (Goulib.drawing.Text attribute), 56
- `__delattr__` (Goulib.expr.Expr attribute), 63
- `__delattr__` (Goulib.expr.LatexVisitor attribute), 66
- `__delattr__` (Goulib.expr.TextVisitor attribute), 65
- `__delattr__` (Goulib.geom.Arc2 attribute), 87
- `__delattr__` (Goulib.geom.Circle attribute), 85
- `__delattr__` (Goulib.geom.Ellipse attribute), 90
- `__delattr__` (Goulib.geom.Geometry attribute), 68
- `__delattr__` (Goulib.geom.Line2 attribute), 79
- `__delattr__` (Goulib.geom.Matrix3 attribute), 96
- `__delattr__` (Goulib.geom.Point2 attribute), 74
- `__delattr__` (Goulib.geom.Ray2 attribute), 80
- `__delattr__` (Goulib.geom.Segment2 attribute), 81

- `__delattr__` (Goulib.geom.Vector2 attribute), 73
- `__delattr__` (Goulib.graph.AGraph attribute), 113
- `__delattr__` (Goulib.graph.DiGraph attribute), 130
- `__delattr__` (Goulib.graph.GeoGraph attribute), 114
- `__delattr__` (Goulib.graph.index attribute), 112
- `__delattr__` (Goulib.graph.index.Index attribute), 110
- `__delattr__` (Goulib.graph.index.Property attribute), 109
- `__delattr__` (Goulib.image.Ditherer attribute), 155
- `__delattr__` (Goulib.image.ErrorDiffusion attribute), 156
- `__delattr__` (Goulib.image.FloydSteinberg attribute), 157
- `__delattr__` (Goulib.image.Image attribute), 153
- `__delattr__` (Goulib.image.Mode attribute), 148
- `__delattr__` (Goulib.interval.Box attribute), 164
- `__delattr__` (Goulib.interval.Interval attribute), 160
- `__delattr__` (Goulib.interval.Intervals attribute), 162
- `__delattr__` (Goulib.itertools2.SortingError attribute), 172
- `__delattr__` (Goulib.itertools2.iter2 attribute), 170
- `__delattr__` (Goulib.itertools2.keep attribute), 173
- `__delattr__` (Goulib.markup.ArgumentError attribute), 185
- `__delattr__` (Goulib.markup.ClosingError attribute), 183
- `__delattr__` (Goulib.markup.CustomizationError attribute), 190
- `__delattr__` (Goulib.markup.DeprecationError attribute), 187
- `__delattr__` (Goulib.markup.InvalidElementError attribute), 186
- `__delattr__` (Goulib.markup.MarkupError attribute), 181
- `__delattr__` (Goulib.markup.ModeError attribute), 189
- `__delattr__` (Goulib.markup.OpeningError attribute), 184
- `__delattr__` (Goulib.markup.dummy attribute), 179
- `__delattr__` (Goulib.markup.element attribute), 175
- `__delattr__` (Goulib.markup.page attribute), 178
- `__delattr__` (Goulib.markup.russell attribute), 180
- `__delattr__` (Goulib.motion.Actuator attribute), 211
- `__delattr__` (Goulib.motion.PVA attribute), 203
- `__delattr__` (Goulib.motion.Segment attribute), 204
- `__delattr__` (Goulib.motion.SegmentPoly attribute), 208
- `__delattr__` (Goulib.motion.Segments attribute), 206
- `__delattr__` (Goulib.motion.TimeDiagram attribute), 212
- `__delattr__` (Goulib.optim.BinDict attribute), 216
- `__delattr__` (Goulib.optim.BinList attribute), 218
- `__delattr__` (Goulib.optim.DifferentialEvolution attribute), 221
- `__delattr__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__delattr__` (Goulib.piecewise.Piecewise attribute), 223
- `__delattr__` (Goulib.plot.Plot attribute), 225
- `__delattr__` (Goulib.polynomial.Polynomial attribute), 227
- `__delattr__` (Goulib.stats.Discrete attribute), 233
- `__delattr__` (Goulib.stats.Normal attribute), 238
- `__delattr__` (Goulib.stats.PDF attribute), 235
- `__delattr__` (Goulib.stats.Stats attribute), 232
- `__delattr__` (Goulib.table.Cell attribute), 240
- `__delattr__` (Goulib.table.Row attribute), 242
- `__delattr__` (Goulib.table.Table attribute), 244
- `__delattr__` (Goulib.tests.TestCase attribute), 247
- `__delattr__` (Goulib.workdays.WorkCalendar attribute), 254
- `__delitem__` (Goulib.colors.Palette attribute), 9
- `__delitem__` (Goulib.container.Record attribute), 11
- `__delitem__` (Goulib.drawing.BBox attribute), 35
- `__delitem__` (Goulib.drawing.Chain attribute), 49
- `__delitem__` (Goulib.drawing.Drawing attribute), 58
- `__delitem__` (Goulib.drawing.Group attribute), 42
- `__delitem__` (Goulib.drawing.Rect attribute), 52
- `__delitem__` (Goulib.graph.index.Index attribute), 110
- `__delitem__` (Goulib.interval.Box attribute), 164
- `__delitem__` (Goulib.interval.Interval attribute), 160
- `__delitem__` (Goulib.optim.BinList attribute), 218
- `__delitem__` (Goulib.table.Table attribute), 244
- `__delitem__` (Goulib.optim.BinDict method), 216
- `__dir__` (Goulib.colors.Color method), 7
- `__dir__` (Goulib.colors.Palette method), 9
- `__dir__` (Goulib.container.Record method), 11
- `__dir__` (Goulib.container.Sequence method), 16
- `__dir__` (Goulib.container.SortedCollection method), 14
- `__dir__` (Goulib.datetime2.date2 method), 20
- `__dir__` (Goulib.datetime2.datetime2 method), 17
- `__dir__` (Goulib.datetime2.time2 method), 22
- `__dir__` (Goulib.datetime2.timedelta2 method), 24
- `__dir__` (Goulib.drawing.BBox method), 35
- `__dir__` (Goulib.drawing.Chain method), 49
- `__dir__` (Goulib.drawing.Drawing method), 58
- `__dir__` (Goulib.drawing.Entity method), 38
- `__dir__` (Goulib.drawing.Group method), 42
- `__dir__` (Goulib.drawing.Instance method), 46
- `__dir__` (Goulib.drawing.Rect method), 52
- `__dir__` (Goulib.drawing.Spline method), 39
- `__dir__` (Goulib.drawing.Text method), 56
- `__dir__` (Goulib.expr.Expr method), 64
- `__dir__` (Goulib.expr.LatexVisitor method), 66
- `__dir__` (Goulib.expr.TextVisitor method), 65
- `__dir__` (Goulib.geom.Arc2 method), 88
- `__dir__` (Goulib.geom.Circle method), 85
- `__dir__` (Goulib.geom.Ellipse method), 90
- `__dir__` (Goulib.geom.Geometry method), 68
- `__dir__` (Goulib.geom.Line2 method), 79
- `__dir__` (Goulib.geom.Matrix3 method), 97
- `__dir__` (Goulib.geom.Point2 method), 74
- `__dir__` (Goulib.geom.Ray2 method), 80
- `__dir__` (Goulib.geom.Segment2 method), 81
- `__dir__` (Goulib.geom.Vector2 method), 73
- `__dir__` (Goulib.graph.AGraph method), 113
- `__dir__` (Goulib.graph.DiGraph method), 130
- `__dir__` (Goulib.graph.GeoGraph method), 114
- `__dir__` (Goulib.graph.index method), 112

- `__dir__()` (Goulib.graph.index.Index method), 111
- `__dir__()` (Goulib.graph.index.Property method), 109
- `__dir__()` (Goulib.image.Ditherer method), 155
- `__dir__()` (Goulib.image.ErrorDiffusion method), 156
- `__dir__()` (Goulib.image.FloydSteinberg method), 157
- `__dir__()` (Goulib.image.Image method), 153
- `__dir__()` (Goulib.image.Mode method), 148
- `__dir__()` (Goulib.interval.Box method), 164
- `__dir__()` (Goulib.interval.Interval method), 160
- `__dir__()` (Goulib.interval.Intervals method), 162
- `__dir__()` (Goulib.itertools2.SortingError method), 172
- `__dir__()` (Goulib.itertools2.iter2 method), 170
- `__dir__()` (Goulib.itertools2.keep method), 173
- `__dir__()` (Goulib.markup.ArgumentError method), 185
- `__dir__()` (Goulib.markup.ClosingError method), 183
- `__dir__()` (Goulib.markup.CustomizationError method), 190
- `__dir__()` (Goulib.markup.DeprecationError method), 187
- `__dir__()` (Goulib.markup.InvalidElementError method), 186
- `__dir__()` (Goulib.markup.MarkupError method), 182
- `__dir__()` (Goulib.markup.ModeError method), 189
- `__dir__()` (Goulib.markup.OpeningError method), 184
- `__dir__()` (Goulib.markup.dummy method), 179
- `__dir__()` (Goulib.markup.element method), 175
- `__dir__()` (Goulib.markup.page method), 178
- `__dir__()` (Goulib.markup.russell method), 180
- `__dir__()` (Goulib.motion.Actuator method), 211
- `__dir__()` (Goulib.motion.PVA method), 203
- `__dir__()` (Goulib.motion.Segment method), 205
- `__dir__()` (Goulib.motion.SegmentPoly method), 208
- `__dir__()` (Goulib.motion.Segments method), 206
- `__dir__()` (Goulib.motion.TimeDiagram method), 212
- `__dir__()` (Goulib.optim.BinDict method), 216
- `__dir__()` (Goulib.optim.BinList method), 218
- `__dir__()` (Goulib.optim.DifferentialEvolution method), 221
- `__dir__()` (Goulib.optim.ObjectiveFunction method), 215
- `__dir__()` (Goulib.piecewise.Piecewise method), 223
- `__dir__()` (Goulib.plot.Plot method), 225
- `__dir__()` (Goulib.polynomial.Polynomial method), 227
- `__dir__()` (Goulib.stats.Discrete method), 233
- `__dir__()` (Goulib.stats.Normal method), 238
- `__dir__()` (Goulib.stats.PDF method), 235
- `__dir__()` (Goulib.stats.Stats method), 232
- `__dir__()` (Goulib.table.Cell method), 240
- `__dir__()` (Goulib.table.Row method), 242
- `__dir__()` (Goulib.table.Table method), 244
- `__dir__()` (Goulib.tests.TestCase method), 247
- `__dir__()` (Goulib.workdays.WorkCalendar method), 254
- `__div__()` (Goulib.expr.Expr method), 63
- `__div__()` (Goulib.geom.Point2 method), 74
- `__div__()` (Goulib.geom.Vector2 method), 72
- `__div__()` (Goulib.geom3d.Vector3 method), 99
- `__div__()` (Goulib.image.Image method), 153
- `__div__()` (Goulib.piecewise.Piecewise method), 223
- `__div__()` (Goulib.polynomial.Polynomial method), 227
- `__div__()` (Goulib.stats.Normal method), 237
- `__div__()` (Goulib.stats.PDF method), 235
- `__divmod__` (Goulib.datetime2.timedelta2 attribute), 24
- `__eq__` (Goulib.colors.Palette attribute), 9
- `__eq__` (Goulib.container.Record attribute), 11
- `__eq__` (Goulib.container.Sequence attribute), 16
- `__eq__` (Goulib.container.SortedCollection attribute), 15
- `__eq__` (Goulib.datetime2.date2 attribute), 20
- `__eq__` (Goulib.datetime2.datetime2 attribute), 17
- `__eq__` (Goulib.datetime2.time2 attribute), 22
- `__eq__` (Goulib.datetime2.timedelta2 attribute), 24
- `__eq__` (Goulib.drawing.BBox attribute), 35
- `__eq__` (Goulib.drawing.Chain attribute), 49
- `__eq__` (Goulib.drawing.Drawing attribute), 59
- `__eq__` (Goulib.drawing.Entity attribute), 38
- `__eq__` (Goulib.drawing.Group attribute), 42
- `__eq__` (Goulib.drawing.Instance attribute), 46
- `__eq__` (Goulib.drawing.Rect attribute), 52
- `__eq__` (Goulib.drawing.Spline attribute), 39
- `__eq__` (Goulib.drawing.Text attribute), 56
- `__eq__` (Goulib.expr.LatexVisitor attribute), 66
- `__eq__` (Goulib.expr.TextVisitor attribute), 65
- `__eq__` (Goulib.geom.Geometry attribute), 68
- `__eq__` (Goulib.graph.AGraph attribute), 113
- `__eq__` (Goulib.graph.index attribute), 112
- `__eq__` (Goulib.graph.index.Index attribute), 111
- `__eq__` (Goulib.graph.index.Property attribute), 109
- `__eq__` (Goulib.image.Ditherer attribute), 155
- `__eq__` (Goulib.image.ErrorDiffusion attribute), 156
- `__eq__` (Goulib.image.FloydSteinberg attribute), 157
- `__eq__` (Goulib.image.Image attribute), 153
- `__eq__` (Goulib.image.Mode attribute), 148
- `__eq__` (Goulib.interval.Box attribute), 164
- `__eq__` (Goulib.interval.Intervals attribute), 162
- `__eq__` (Goulib.itertools2.SortingError attribute), 172
- `__eq__` (Goulib.itertools2.iter2 attribute), 171
- `__eq__` (Goulib.itertools2.keep attribute), 173
- `__eq__` (Goulib.markup.ArgumentError attribute), 185
- `__eq__` (Goulib.markup.ClosingError attribute), 183
- `__eq__` (Goulib.markup.CustomizationError attribute), 190
- `__eq__` (Goulib.markup.DeprecationError attribute), 187
- `__eq__` (Goulib.markup.InvalidElementError attribute), 186
- `__eq__` (Goulib.markup.MarkupError attribute), 182
- `__eq__` (Goulib.markup.ModeError attribute), 189
- `__eq__` (Goulib.markup.OpeningError attribute), 184
- `__eq__` (Goulib.markup.dummy attribute), 179
- `__eq__` (Goulib.markup.element attribute), 175
- `__eq__` (Goulib.markup.page attribute), 178

- `__eq__` (Goulib.markup.russell attribute), 180
- `__eq__` (Goulib.motion.Actuator attribute), 211
- `__eq__` (Goulib.motion.PVA attribute), 203
- `__eq__` (Goulib.motion.Segment attribute), 205
- `__eq__` (Goulib.motion.SegmentPoly attribute), 208
- `__eq__` (Goulib.motion.Segments attribute), 206
- `__eq__` (Goulib.motion.TimeDiagram attribute), 212
- `__eq__` (Goulib.optim.BinDict attribute), 216
- `__eq__` (Goulib.optim.BinList attribute), 218
- `__eq__` (Goulib.optim.DifferentialEvolution attribute), 221
- `__eq__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__eq__` (Goulib.plot.Plot attribute), 225
- `__eq__` (Goulib.stats.Discrete attribute), 233
- `__eq__` (Goulib.stats.Stats attribute), 232
- `__eq__` (Goulib.table.Cell attribute), 240
- `__eq__` (Goulib.table.Row attribute), 242
- `__eq__` (Goulib.workdays.WorkCalendar attribute), 254
- `__eq__()` (Goulib.colors.Color method), 7
- `__eq__()` (Goulib.expr.Expr method), 63
- `__eq__()` (Goulib.geom.Arc2 method), 87
- `__eq__()` (Goulib.geom.Circle method), 84
- `__eq__()` (Goulib.geom.Ellipse method), 90
- `__eq__()` (Goulib.geom.Line2 method), 78
- `__eq__()` (Goulib.geom.Matrix3 method), 96
- `__eq__()` (Goulib.geom.Point2 method), 74
- `__eq__()` (Goulib.geom.Ray2 method), 80
- `__eq__()` (Goulib.geom.Segment2 method), 81
- `__eq__()` (Goulib.geom.Vector2 method), 72
- `__eq__()` (Goulib.geom3d.Vector3 method), 98
- `__eq__()` (Goulib.graph.DiGraph method), 130
- `__eq__()` (Goulib.graph.GeoGraph method), 115
- `__eq__()` (Goulib.interval.Interval method), 159
- `__eq__()` (Goulib.piecewise.Piecewise method), 223
- `__eq__()` (Goulib.polynomial.Polynomial method), 226
- `__eq__()` (Goulib.stats.Normal method), 238
- `__eq__()` (Goulib.stats.PDF method), 235
- `__eq__()` (Goulib.table.Table method), 243
- `__eq__()` (Goulib.tests.TestCase method), 247
- `__floordiv__` (Goulib.datetime2.timedelta2 attribute), 24
- `__floordiv__()` (Goulib.geom.Point2 method), 75
- `__floordiv__()` (Goulib.geom.Vector2 method), 72
- `__floordiv__()` (Goulib.geom3d.Vector3 method), 99
- `__format__` (Goulib.colors.Color method), 8
- `__format__` (Goulib.colors.Palette method), 9
- `__format__` (Goulib.container.Record method), 11
- `__format__` (Goulib.container.Sequence method), 16
- `__format__` (Goulib.container.SortedCollection method), 15
- `__format__` (Goulib.datetime2.date2 method), 20
- `__format__` (Goulib.datetime2.datetime2 method), 18
- `__format__` (Goulib.datetime2.time2 method), 22
- `__format__` (Goulib.datetime2.timedelta2 method), 24
- `__format__` (Goulib.drawing.BBox method), 35
- `__format__` (Goulib.drawing.Chain method), 49
- `__format__` (Goulib.drawing.Drawing method), 59
- `__format__` (Goulib.drawing.Entity method), 38
- `__format__` (Goulib.drawing.Group method), 42
- `__format__` (Goulib.drawing.Instance method), 46
- `__format__` (Goulib.drawing.Rect method), 52
- `__format__` (Goulib.drawing.Spline method), 39
- `__format__` (Goulib.drawing.Text method), 56
- `__format__` (Goulib.expr.Expr method), 64
- `__format__` (Goulib.expr.LatexVisitor method), 66
- `__format__` (Goulib.expr.TextVisitor method), 65
- `__format__` (Goulib.geom.Arc2 method), 88
- `__format__` (Goulib.geom.Circle method), 85
- `__format__` (Goulib.geom.Ellipse method), 90
- `__format__` (Goulib.geom.Geometry method), 68
- `__format__` (Goulib.geom.Line2 method), 79
- `__format__` (Goulib.geom.Matrix3 method), 97
- `__format__` (Goulib.geom.Point2 method), 75
- `__format__` (Goulib.geom.Ray2 method), 80
- `__format__` (Goulib.geom.Segment2 method), 81
- `__format__` (Goulib.geom.Vector2 method), 73
- `__format__` (Goulib.graph.AGraph method), 113
- `__format__` (Goulib.graph.DiGraph method), 130
- `__format__` (Goulib.graph.GeoGraph method), 115
- `__format__` (Goulib.graph.index method), 112
- `__format__` (Goulib.graph.index.Index method), 111
- `__format__` (Goulib.graph.index.Property method), 109
- `__format__` (Goulib.image.Ditherer method), 155
- `__format__` (Goulib.image.ErrorDiffusion method), 156
- `__format__` (Goulib.image.FloydSteinberg method), 157
- `__format__` (Goulib.image.Image method), 153
- `__format__` (Goulib.image.Mode method), 148
- `__format__` (Goulib.interval.Box method), 164
- `__format__` (Goulib.interval.Interval method), 160
- `__format__` (Goulib.interval.Intervals method), 162
- `__format__` (Goulib.itertools2.SortingError method), 172
- `__format__` (Goulib.itertools2.iter2 method), 171
- `__format__` (Goulib.itertools2.keep method), 173
- `__format__` (Goulib.markup.ArgumentError method), 185
- `__format__` (Goulib.markup.ClosingError method), 183
- `__format__` (Goulib.markup.CustomizationError method), 190
- `__format__` (Goulib.markup.DeprecationError method), 187
- `__format__` (Goulib.markup.InvalidElementError method), 186
- `__format__` (Goulib.markup.MarkupError method), 182
- `__format__` (Goulib.markup.ModeError method), 189
- `__format__` (Goulib.markup.OpeningError method), 184
- `__format__` (Goulib.markup.dummy method), 179

- `__format__` () (Goulib.markup.element method), 175
- `__format__` () (Goulib.markup.page method), 178
- `__format__` () (Goulib.markup.russell method), 180
- `__format__` () (Goulib.motion.Actuator method), 211
- `__format__` () (Goulib.motion.PVA method), 203
- `__format__` () (Goulib.motion.Segment method), 205
- `__format__` () (Goulib.motion.SegmentPoly method), 208
- `__format__` () (Goulib.motion.Segments method), 206
- `__format__` () (Goulib.motion.TimeDiagram method), 212
- `__format__` () (Goulib.optim.BinDict method), 216
- `__format__` () (Goulib.optim.BinList method), 218
- `__format__` () (Goulib.optim.DifferentialEvolution method), 221
- `__format__` () (Goulib.optim.ObjectiveFunction method), 215
- `__format__` () (Goulib.piecewise.Piecewise method), 223
- `__format__` () (Goulib.plot.Plot method), 225
- `__format__` () (Goulib.polynomial.Polynomial method), 227
- `__format__` () (Goulib.stats.Discrete method), 233
- `__format__` () (Goulib.stats.Normal method), 238
- `__format__` () (Goulib.stats.PDF method), 235
- `__format__` () (Goulib.stats.Stats method), 232
- `__format__` () (Goulib.table.Cell method), 240
- `__format__` () (Goulib.table.Row method), 242
- `__format__` () (Goulib.table.Table method), 244
- `__format__` () (Goulib.tests.TestCase method), 247
- `__format__` () (Goulib.workdays.WorkCalendar method), 254
- `__ge__` (Goulib.colors.Color attribute), 8
- `__ge__` (Goulib.colors.Palette attribute), 9
- `__ge__` (Goulib.container.Record attribute), 11
- `__ge__` (Goulib.container.Sequence attribute), 16
- `__ge__` (Goulib.container.SortedCollection attribute), 15
- `__ge__` (Goulib.datetime2.date2 attribute), 20
- `__ge__` (Goulib.datetime2.datetime2 attribute), 18
- `__ge__` (Goulib.datetime2.time2 attribute), 22
- `__ge__` (Goulib.datetime2.timedelta2 attribute), 24
- `__ge__` (Goulib.drawing.BBox attribute), 35
- `__ge__` (Goulib.drawing.Chain attribute), 49
- `__ge__` (Goulib.drawing.Drawing attribute), 59
- `__ge__` (Goulib.drawing.Entity attribute), 38
- `__ge__` (Goulib.drawing.Group attribute), 43
- `__ge__` (Goulib.drawing.Instance attribute), 46
- `__ge__` (Goulib.drawing.Rect attribute), 52
- `__ge__` (Goulib.drawing.Spline attribute), 40
- `__ge__` (Goulib.drawing.Text attribute), 57
- `__ge__` (Goulib.expr.Expr attribute), 64
- `__ge__` (Goulib.expr.LatexVisitor attribute), 66
- `__ge__` (Goulib.expr.TextVisitor attribute), 65
- `__ge__` (Goulib.geom.Arc2 attribute), 88
- `__ge__` (Goulib.geom.Circle attribute), 85
- `__ge__` (Goulib.geom.Ellipse attribute), 90
- `__ge__` (Goulib.geom.Geometry attribute), 68
- `__ge__` (Goulib.geom.Line2 attribute), 79
- `__ge__` (Goulib.geom.Matrix3 attribute), 97
- `__ge__` (Goulib.geom.Point2 attribute), 75
- `__ge__` (Goulib.geom.Ray2 attribute), 80
- `__ge__` (Goulib.geom.Segment2 attribute), 81
- `__ge__` (Goulib.geom.Vector2 attribute), 73
- `__ge__` (Goulib.graph.AGraph attribute), 113
- `__ge__` (Goulib.graph.DiGraph attribute), 130
- `__ge__` (Goulib.graph.GeoGraph attribute), 115
- `__ge__` (Goulib.graph.index attribute), 112
- `__ge__` (Goulib.graph.index.Index attribute), 111
- `__ge__` (Goulib.graph.index.Property attribute), 109
- `__ge__` (Goulib.image.Ditherer attribute), 155
- `__ge__` (Goulib.image.ErrorDiffusion attribute), 156
- `__ge__` (Goulib.image.FloydSteinberg attribute), 157
- `__ge__` (Goulib.image.Image attribute), 153
- `__ge__` (Goulib.image.Mode attribute), 149
- `__ge__` (Goulib.interval.Box attribute), 164
- `__ge__` (Goulib.interval.Interval attribute), 160
- `__ge__` (Goulib.interval.Intervals attribute), 162
- `__ge__` (Goulib.itertools2.SortingError attribute), 172
- `__ge__` (Goulib.itertools2.iter2 attribute), 171
- `__ge__` (Goulib.itertools2.keep attribute), 173
- `__ge__` (Goulib.markup.ArgumentError attribute), 185
- `__ge__` (Goulib.markup.ClosingError attribute), 183
- `__ge__` (Goulib.markup.CustomizationError attribute), 190
- `__ge__` (Goulib.markup.DeprecationError attribute), 188
- `__ge__` (Goulib.markup.InvalidElementError attribute), 186
- `__ge__` (Goulib.markup.MarkupError attribute), 182
- `__ge__` (Goulib.markup.ModeError attribute), 189
- `__ge__` (Goulib.markup.OpeningError attribute), 184
- `__ge__` (Goulib.markup.dummy attribute), 179
- `__ge__` (Goulib.markup.element attribute), 175
- `__ge__` (Goulib.markup.page attribute), 178
- `__ge__` (Goulib.markup.russell attribute), 181
- `__ge__` (Goulib.motion.Actuator attribute), 211
- `__ge__` (Goulib.motion.PVA attribute), 203
- `__ge__` (Goulib.motion.Segment attribute), 205
- `__ge__` (Goulib.motion.SegmentPoly attribute), 208
- `__ge__` (Goulib.motion.Segments attribute), 206
- `__ge__` (Goulib.motion.TimeDiagram attribute), 212
- `__ge__` (Goulib.optim.BinDict attribute), 216
- `__ge__` (Goulib.optim.BinList attribute), 218
- `__ge__` (Goulib.optim.DifferentialEvolution attribute), 221
- `__ge__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__ge__` (Goulib.piecewise.Piecewise attribute), 223
- `__ge__` (Goulib.plot.Plot attribute), 225
- `__ge__` (Goulib.polynomial.Polynomial attribute), 227
- `__ge__` (Goulib.stats.Discrete attribute), 233
- `__ge__` (Goulib.stats.Normal attribute), 238
- `__ge__` (Goulib.stats.PDF attribute), 235

- `__ge__` (Goulib.stats.Stats attribute), 232
- `__ge__` (Goulib.table.Cell attribute), 241
- `__ge__` (Goulib.table.Row attribute), 242
- `__ge__` (Goulib.table.Table attribute), 244
- `__ge__` (Goulib.tests.TestCase attribute), 247
- `__ge__` (Goulib.workdays.WorkCalendar attribute), 254
- `__getattr__()` (Goulib.container.Record method), 11
- `__getattr__()` (Goulib.markup.page method), 177
- `__getattrattribute__` (Goulib.colors.Color attribute), 8
- `__getattrattribute__` (Goulib.colors.Palette attribute), 9
- `__getattrattribute__` (Goulib.container.Record attribute), 11
- `__getattrattribute__` (Goulib.container.Sequence attribute), 16
- `__getattrattribute__` (Goulib.container.SortedCollection attribute), 15
- `__getattrattribute__` (Goulib.datetime2.date2 attribute), 20
- `__getattrattribute__` (Goulib.datetime2.datetime2 attribute), 18
- `__getattrattribute__` (Goulib.datetime2.time2 attribute), 22
- `__getattrattribute__` (Goulib.datetime2.timedelta2 attribute), 24
- `__getattrattribute__` (Goulib.drawing.BBox attribute), 35
- `__getattrattribute__` (Goulib.drawing.Chain attribute), 49
- `__getattrattribute__` (Goulib.drawing.Drawing attribute), 59
- `__getattrattribute__` (Goulib.drawing.Entity attribute), 38
- `__getattrattribute__` (Goulib.drawing.Group attribute), 43
- `__getattrattribute__` (Goulib.drawing.Instance attribute), 46
- `__getattrattribute__` (Goulib.drawing.Rect attribute), 52
- `__getattrattribute__` (Goulib.drawing.Spline attribute), 40
- `__getattrattribute__` (Goulib.drawing.Text attribute), 57
- `__getattrattribute__` (Goulib.expr.Expr attribute), 64
- `__getattrattribute__` (Goulib.expr.LatexVisitor attribute), 66
- `__getattrattribute__` (Goulib.expr.TextVisitor attribute), 65
- `__getattrattribute__` (Goulib.geom.Arc2 attribute), 88
- `__getattrattribute__` (Goulib.geom.Circle attribute), 85
- `__getattrattribute__` (Goulib.geom.Ellipse attribute), 90
- `__getattrattribute__` (Goulib.geom.Geometry attribute), 68
- `__getattrattribute__` (Goulib.geom.Line2 attribute), 79
- `__getattrattribute__` (Goulib.geom.Matrix3 attribute), 97
- `__getattrattribute__` (Goulib.geom.Point2 attribute), 75
- `__getattrattribute__` (Goulib.geom.Ray2 attribute), 80
- `__getattrattribute__` (Goulib.geom.Segment2 attribute), 81
- `__getattrattribute__` (Goulib.geom.Vector2 attribute), 73
- `__getattrattribute__` (Goulib.graph.AGraph attribute), 113
- `__getattrattribute__` (Goulib.graph.DiGraph attribute), 130
- `__getattrattribute__` (Goulib.graph.GeoGraph attribute), 115
- `__getattrattribute__` (Goulib.graph.index attribute), 112
- `__getattrattribute__` (Goulib.graph.index.Index attribute), 111
- `__getattrattribute__` (Goulib.graph.index.Property attribute), 109
- `__getattrattribute__` (Goulib.image.Ditherer attribute), 155
- `__getattrattribute__` (Goulib.image.ErrorDiffusion attribute), 156
- `__getattrattribute__` (Goulib.image.FloydSteinberg attribute), 157
- `__getattrattribute__` (Goulib.image.Image attribute), 153
- `__getattrattribute__` (Goulib.image.Mode attribute), 149
- `__getattrattribute__` (Goulib.interval.Box attribute), 164
- `__getattrattribute__` (Goulib.interval.Interval attribute), 160
- `__getattrattribute__` (Goulib.interval.Intervals attribute), 162
- `__getattrattribute__` (Goulib.itertools2.SortingError attribute), 172
- `__getattrattribute__` (Goulib.itertools2.iter2 attribute), 171
- `__getattrattribute__` (Goulib.itertools2.keep attribute), 173
- `__getattrattribute__` (Goulib.markup.ArgumentError attribute), 185
- `__getattrattribute__` (Goulib.markup.ClosingError attribute), 183
- `__getattrattribute__` (Goulib.markup.CustomizationError attribute), 190
- `__getattrattribute__` (Goulib.markup.DeprecationError attribute), 188
- `__getattrattribute__` (Goulib.markup.InvalidElementError attribute), 186
- `__getattrattribute__` (Goulib.markup.MarkupError attribute), 182
- `__getattrattribute__` (Goulib.markup.ModeError attribute), 189
- `__getattrattribute__` (Goulib.markup.OpeningError attribute), 184
- `__getattrattribute__` (Goulib.markup.dummy attribute), 180
- `__getattrattribute__` (Goulib.markup.element attribute), 175
- `__getattrattribute__` (Goulib.markup.page attribute), 179
- `__getattrattribute__` (Goulib.markup.russell attribute), 181
- `__getattrattribute__` (Goulib.motion.Actuator attribute), 211
- `__getattrattribute__` (Goulib.motion.PVA attribute), 203
- `__getattrattribute__` (Goulib.motion.Segment attribute), 205
- `__getattrattribute__` (Goulib.motion.SegmentPoly attribute), 208
- `__getattrattribute__` (Goulib.motion.Segments attribute), 206
- `__getattrattribute__` (Goulib.motion.TimeDiagram attribute), 212
- `__getattrattribute__` (Goulib.optim.BinDict attribute), 216
- `__getattrattribute__` (Goulib.optim.BinList attribute), 218
- `__getattrattribute__` (Goulib.optim.DifferentialEvolution attribute), 222
- `__getattrattribute__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__getattrattribute__` (Goulib.piecewise.Piecewise attribute), 223
- `__getattrattribute__` (Goulib.plot.Plot attribute), 225
- `__getattrattribute__` (Goulib.polynomial.Polynomial attribute), 227
- `__getattrattribute__` (Goulib.stats.Discrete attribute), 233
- `__getattrattribute__` (Goulib.stats.Normal attribute), 238
- `__getattrattribute__` (Goulib.stats.PDF attribute), 235
- `__getattrattribute__` (Goulib.stats.Stats attribute), 232

- `__getattr__` (Goulib.table.Cell attribute), 241
- `__getattr__` (Goulib.table.Row attribute), 242
- `__getattr__` (Goulib.table.Table attribute), 244
- `__getattr__` (Goulib.tests.TestCase attribute), 247
- `__getattr__` (Goulib.workdays.WorkCalendar attribute), 254
- `__getitem__` (Goulib.colors.Palette method), 9
- `__getitem__` (Goulib.container.Record method), 11
- `__getitem__` (Goulib.container.Sequence method), 16
- `__getitem__` (Goulib.container.SortedCollection method), 14
- `__getitem__` (Goulib.drawing.BBox method), 35
- `__getitem__` (Goulib.drawing.Chain method), 49
- `__getitem__` (Goulib.drawing.Drawing method), 59
- `__getitem__` (Goulib.drawing.Group method), 43
- `__getitem__` (Goulib.drawing.Rect method), 53
- `__getitem__` (Goulib.geom.Matrix3 method), 96
- `__getitem__` (Goulib.geom3d.Matrix4 method), 103
- `__getitem__` (Goulib.graph.DiGraph method), 130
- `__getitem__` (Goulib.graph.GeoGraph method), 115
- `__getitem__` (Goulib.graph.index.Index method), 111
- `__getitem__` (Goulib.image.Image method), 151
- `__getitem__` (Goulib.interval.Box method), 164
- `__getitem__` (Goulib.interval.Interval method), 160
- `__getitem__` (Goulib.interval.Intervals method), 162
- `__getitem__` (Goulib.optim.BinDict method), 216
- `__getitem__` (Goulib.optim.BinList method), 218
- `__getitem__` (Goulib.piecewise.Piecewise method), 223
- `__getitem__` (Goulib.table.Table method), 244
- `__gt__` (Goulib.colors.Color attribute), 8
- `__gt__` (Goulib.colors.Palette attribute), 9
- `__gt__` (Goulib.container.Record attribute), 11
- `__gt__` (Goulib.container.Sequence attribute), 16
- `__gt__` (Goulib.container.SortedCollection attribute), 15
- `__gt__` (Goulib.datetime2.date2 attribute), 20
- `__gt__` (Goulib.datetime2.datetime2 attribute), 18
- `__gt__` (Goulib.datetime2.time2 attribute), 22
- `__gt__` (Goulib.datetime2.timedelta2 attribute), 24
- `__gt__` (Goulib.drawing.BBox attribute), 35
- `__gt__` (Goulib.drawing.Chain attribute), 49
- `__gt__` (Goulib.drawing.Drawing attribute), 59
- `__gt__` (Goulib.drawing.Entity attribute), 38
- `__gt__` (Goulib.drawing.Group attribute), 43
- `__gt__` (Goulib.drawing.Instance attribute), 46
- `__gt__` (Goulib.drawing.Rect attribute), 53
- `__gt__` (Goulib.drawing.Spline attribute), 40
- `__gt__` (Goulib.drawing.Text attribute), 57
- `__gt__` (Goulib.expr.Expr attribute), 64
- `__gt__` (Goulib.expr.LatexVisitor attribute), 66
- `__gt__` (Goulib.expr.TextVisitor attribute), 65
- `__gt__` (Goulib.geom.Arc2 attribute), 88
- `__gt__` (Goulib.geom.Circle attribute), 85
- `__gt__` (Goulib.geom.Ellipse attribute), 90
- `__gt__` (Goulib.geom.Geometry attribute), 68
- `__gt__` (Goulib.geom.Line2 attribute), 79
- `__gt__` (Goulib.geom.Matrix3 attribute), 97
- `__gt__` (Goulib.geom.Point2 attribute), 75
- `__gt__` (Goulib.geom.Ray2 attribute), 80
- `__gt__` (Goulib.geom.Segment2 attribute), 81
- `__gt__` (Goulib.geom.Vector2 attribute), 73
- `__gt__` (Goulib.graph.AGraph attribute), 113
- `__gt__` (Goulib.graph.DiGraph attribute), 130
- `__gt__` (Goulib.graph.GeoGraph attribute), 115
- `__gt__` (Goulib.graph.index attribute), 112
- `__gt__` (Goulib.graph.index.Index attribute), 111
- `__gt__` (Goulib.graph.index.Property attribute), 109
- `__gt__` (Goulib.image.Ditherer attribute), 155
- `__gt__` (Goulib.image.ErrorDiffusion attribute), 156
- `__gt__` (Goulib.image.FloydSteinberg attribute), 157
- `__gt__` (Goulib.image.Image attribute), 153
- `__gt__` (Goulib.image.Mode attribute), 149
- `__gt__` (Goulib.interval.Box attribute), 164
- `__gt__` (Goulib.interval.Interval attribute), 160
- `__gt__` (Goulib.interval.Intervals attribute), 162
- `__gt__` (Goulib.itertools2.SortingError attribute), 172
- `__gt__` (Goulib.itertools2.iter2 attribute), 171
- `__gt__` (Goulib.itertools2.keep attribute), 174
- `__gt__` (Goulib.markup.ArgumentError attribute), 185
- `__gt__` (Goulib.markup.ClosingError attribute), 183
- `__gt__` (Goulib.markup.CustomizationError attribute), 190
- `__gt__` (Goulib.markup.DeprecationError attribute), 188
- `__gt__` (Goulib.markup.InvalidElementError attribute), 186
- `__gt__` (Goulib.markup.MarkupError attribute), 182
- `__gt__` (Goulib.markup.ModeError attribute), 189
- `__gt__` (Goulib.markup.OpeningError attribute), 184
- `__gt__` (Goulib.markup.dummy attribute), 180
- `__gt__` (Goulib.markup.element attribute), 175
- `__gt__` (Goulib.markup.page attribute), 179
- `__gt__` (Goulib.markup.russell attribute), 181
- `__gt__` (Goulib.motion.Actuator attribute), 211
- `__gt__` (Goulib.motion.PVA attribute), 203
- `__gt__` (Goulib.motion.Segment attribute), 205
- `__gt__` (Goulib.motion.SegmentPoly attribute), 208
- `__gt__` (Goulib.motion.Segments attribute), 206
- `__gt__` (Goulib.motion.TimeDiagram attribute), 212
- `__gt__` (Goulib.optim.BinDict attribute), 216
- `__gt__` (Goulib.optim.BinList attribute), 218
- `__gt__` (Goulib.optim.DifferentialEvolution attribute), 222
- `__gt__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__gt__` (Goulib.piecewise.Piecewise attribute), 223
- `__gt__` (Goulib.plot.Plot attribute), 225
- `__gt__` (Goulib.polynomial.Polynomial attribute), 227
- `__gt__` (Goulib.stats.Discrete attribute), 234
- `__gt__` (Goulib.stats.Normal attribute), 238
- `__gt__` (Goulib.stats.PDF attribute), 235

- `__gt__` (Goulib.stats.Stats attribute), 232
- `__gt__` (Goulib.table.Cell attribute), 241
- `__gt__` (Goulib.table.Row attribute), 242
- `__gt__` (Goulib.table.Table attribute), 244
- `__gt__` (Goulib.tests.TestCase attribute), 248
- `__gt__` (Goulib.workdays.WorkCalendar attribute), 255
- `__hash__` (Goulib.colors.Palette attribute), 9
- `__hash__` (Goulib.container.Record attribute), 11
- `__hash__` (Goulib.container.Sequence attribute), 16
- `__hash__` (Goulib.container.SortedCollection attribute), 15
- `__hash__` (Goulib.datetime2.date2 attribute), 20
- `__hash__` (Goulib.datetime2.datetime2 attribute), 18
- `__hash__` (Goulib.datetime2.time2 attribute), 22
- `__hash__` (Goulib.datetime2.timedelta2 attribute), 24
- `__hash__` (Goulib.drawing.BBox attribute), 35
- `__hash__` (Goulib.drawing.Chain attribute), 49
- `__hash__` (Goulib.drawing.Drawing attribute), 59
- `__hash__` (Goulib.drawing.Entity attribute), 38
- `__hash__` (Goulib.drawing.Group attribute), 43
- `__hash__` (Goulib.drawing.Instance attribute), 46
- `__hash__` (Goulib.drawing.Rect attribute), 53
- `__hash__` (Goulib.drawing.Spline attribute), 40
- `__hash__` (Goulib.drawing.Text attribute), 57
- `__hash__` (Goulib.expr.Expr attribute), 64
- `__hash__` (Goulib.expr.LatexVisitor attribute), 66
- `__hash__` (Goulib.expr.TextVisitor attribute), 65
- `__hash__` (Goulib.geom.Arc2 attribute), 88
- `__hash__` (Goulib.geom.Circle attribute), 85
- `__hash__` (Goulib.geom.Ellipse attribute), 90
- `__hash__` (Goulib.geom.Geometry attribute), 69
- `__hash__` (Goulib.geom.Line2 attribute), 79
- `__hash__` (Goulib.geom.Matrix3 attribute), 97
- `__hash__` (Goulib.geom.Ray2 attribute), 80
- `__hash__` (Goulib.geom.Segment2 attribute), 82
- `__hash__` (Goulib.geom3d.Vector3 attribute), 99
- `__hash__` (Goulib.graph.AGraph attribute), 113
- `__hash__` (Goulib.graph.DiGraph attribute), 130
- `__hash__` (Goulib.graph.GeoGraph attribute), 115
- `__hash__` (Goulib.graph.index attribute), 112
- `__hash__` (Goulib.graph.index.Index attribute), 111
- `__hash__` (Goulib.graph.index.Property attribute), 110
- `__hash__` (Goulib.image.Ditherer attribute), 155
- `__hash__` (Goulib.image.ErrorDiffusion attribute), 156
- `__hash__` (Goulib.image.FloydSteinberg attribute), 157
- `__hash__` (Goulib.image.Mode attribute), 149
- `__hash__` (Goulib.interval.Box attribute), 164
- `__hash__` (Goulib.interval.Intervals attribute), 162
- `__hash__` (Goulib.itertools2.SortingError attribute), 172
- `__hash__` (Goulib.itertools2.iter2 attribute), 171
- `__hash__` (Goulib.itertools2.keep attribute), 174
- `__hash__` (Goulib.markup.ArgumentError attribute), 185
- `__hash__` (Goulib.markup.ClosingError attribute), 183
- `__hash__` (Goulib.markup.CustomizationError attribute), 190
- `__hash__` (Goulib.markup.DeprecationError attribute), 188
- `__hash__` (Goulib.markup.InvalidElementError attribute), 186
- `__hash__` (Goulib.markup.MarkupError attribute), 182
- `__hash__` (Goulib.markup.ModeError attribute), 189
- `__hash__` (Goulib.markup.OpeningError attribute), 184
- `__hash__` (Goulib.markup.dummy attribute), 180
- `__hash__` (Goulib.markup.element attribute), 175
- `__hash__` (Goulib.markup.page attribute), 179
- `__hash__` (Goulib.markup.russell attribute), 181
- `__hash__` (Goulib.motion.Actuator attribute), 211
- `__hash__` (Goulib.motion.PVA attribute), 203
- `__hash__` (Goulib.motion.Segment attribute), 205
- `__hash__` (Goulib.motion.SegmentPoly attribute), 208
- `__hash__` (Goulib.motion.Segments attribute), 206
- `__hash__` (Goulib.motion.TimeDiagram attribute), 212
- `__hash__` (Goulib.optim.BinDict attribute), 216
- `__hash__` (Goulib.optim.BinList attribute), 218
- `__hash__` (Goulib.optim.DifferentialEvolution attribute), 222
- `__hash__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__hash__` (Goulib.piecewise.Piecewise attribute), 223
- `__hash__` (Goulib.plot.Plot attribute), 225
- `__hash__` (Goulib.polynomial.Polynomial attribute), 227
- `__hash__` (Goulib.stats.Discrete attribute), 234
- `__hash__` (Goulib.stats.Normal attribute), 238
- `__hash__` (Goulib.stats.PDF attribute), 235
- `__hash__` (Goulib.stats.Stats attribute), 232
- `__hash__` (Goulib.table.Cell attribute), 241
- `__hash__` (Goulib.table.Row attribute), 242
- `__hash__` (Goulib.table.Table attribute), 244
- `__hash__` (Goulib.workdays.WorkCalendar attribute), 255
- `__hash__()` (Goulib.colors.Color method), 7
- `__hash__()` (Goulib.geom.Point2 method), 75
- `__hash__()` (Goulib.geom.Vector2 method), 72
- `__hash__()` (Goulib.image.Image method), 152
- `__hash__()` (Goulib.interval.Interval method), 159
- `__hash__()` (Goulib.tests.TestCase method), 248
- `__iadd__` (Goulib.drawing.Chain attribute), 49
- `__iadd__` (Goulib.drawing.Drawing attribute), 59
- `__iadd__` (Goulib.drawing.Group attribute), 43
- `__iadd__` (Goulib.drawing.Rect attribute), 53
- `__iadd__` (Goulib.table.Table attribute), 245
- `__iadd__()` (Goulib.drawing.BBox method), 28, 34
- `__iadd__()` (Goulib.geom.Point2 method), 75
- `__iadd__()` (Goulib.geom.Vector2 method), 72
- `__iadd__()` (Goulib.geom3d.Vector3 method), 98
- `__iadd__()` (Goulib.interval.Box method), 164
- `__iadd__()` (Goulib.interval.Interval method), 160

- `__iadd__()` (Goulib.interval.Intervals method), 162
- `__iadd__()` (Goulib.optim.BinDict method), 216
- `__iadd__()` (Goulib.optim.BinList method), 218
- `__imul__` (Goulib.drawing.BBox attribute), 35
- `__imul__` (Goulib.drawing.Chain attribute), 49
- `__imul__` (Goulib.drawing.Drawing attribute), 59
- `__imul__` (Goulib.drawing.Group attribute), 43
- `__imul__` (Goulib.drawing.Rect attribute), 53
- `__imul__` (Goulib.interval.Box attribute), 164
- `__imul__` (Goulib.interval.Interval attribute), 160
- `__imul__` (Goulib.optim.BinList attribute), 219
- `__imul__` (Goulib.table.Table attribute), 245
- `__imul__()` (Goulib.geom.Matrix3 method), 96
- `__imul__()` (Goulib.geom.Point2 method), 75
- `__imul__()` (Goulib.geom.Vector2 method), 72
- `__imul__()` (Goulib.geom3d.Matrix4 method), 103
- `__imul__()` (Goulib.geom3d.Quaternion method), 106
- `__imul__()` (Goulib.geom3d.Vector3 method), 99
- `__init__` (Goulib.datetime2.date2 attribute), 20
- `__init__` (Goulib.drawing.Entity attribute), 38
- `__init__` (Goulib.drawing.Group attribute), 43
- `__init__` (Goulib.graph.AGraph attribute), 113
- `__init__` (Goulib.graph.index attribute), 112
- `__init__` (Goulib.graph.index.Property attribute), 110
- `__init__` (Goulib.itertools2.SortingError attribute), 172
- `__init__` (Goulib.markup.MarkupError attribute), 182
- `__init__` (Goulib.markup.dummy attribute), 180
- `__init__` (Goulib.markup.russell attribute), 181
- `__init__` (Goulib.plot.Plot attribute), 225
- `__init__()` (Goulib.colors.Color method), 6
- `__init__()` (Goulib.colors.Palette method), 8
- `__init__()` (Goulib.container.Record method), 11
- `__init__()` (Goulib.container.Sequence method), 15
- `__init__()` (Goulib.container.SortedCollection method), 14
- `__init__()` (Goulib.datetime2.datetime2 method), 17
- `__init__()` (Goulib.datetime2.time2 method), 22
- `__init__()` (Goulib.datetime2.timedelta2 method), 23
- `__init__()` (Goulib.drawing.BBox method), 28, 34
- `__init__()` (Goulib.drawing.Chain method), 31, 48
- `__init__()` (Goulib.drawing.Drawing method), 33, 59
- `__init__()` (Goulib.drawing.Instance method), 31, 45
- `__init__()` (Goulib.drawing.Rect method), 32, 52
- `__init__()` (Goulib.drawing.Spline method), 29, 39
- `__init__()` (Goulib.drawing.Text method), 32, 56
- `__init__()` (Goulib.expr.Expr method), 63
- `__init__()` (Goulib.expr.LatexVisitor method), 67
- `__init__()` (Goulib.expr.TextVisitor method), 64
- `__init__()` (Goulib.geom.Arc2 method), 87
- `__init__()` (Goulib.geom.Circle method), 84
- `__init__()` (Goulib.geom.Ellipse method), 90
- `__init__()` (Goulib.geom.Geometry method), 68
- `__init__()` (Goulib.geom.Line2 method), 78
- `__init__()` (Goulib.geom.Matrix3 method), 96
- `__init__()` (Goulib.geom.Point2 method), 75
- `__init__()` (Goulib.geom.Ray2 method), 80
- `__init__()` (Goulib.geom.Segment2 method), 82
- `__init__()` (Goulib.geom.Vector2 method), 72
- `__init__()` (Goulib.geom3d.Line3 method), 100
- `__init__()` (Goulib.geom3d.Matrix4 method), 103
- `__init__()` (Goulib.geom3d.Plane method), 103
- `__init__()` (Goulib.geom3d.Quaternion method), 106
- `__init__()` (Goulib.geom3d.Sphere method), 101
- `__init__()` (Goulib.geom3d.Vector3 method), 98
- `__init__()` (Goulib.graph.DiGraph method), 108, 129
- `__init__()` (Goulib.graph.GeoGraph method), 107, 114
- `__init__()` (Goulib.graph.index.Index method), 107, 110
- `__init__()` (Goulib.image.Ditherer method), 155
- `__init__()` (Goulib.image.ErrorDiffusion method), 156
- `__init__()` (Goulib.image.FloydSteinberg method), 157
- `__init__()` (Goulib.image.Image method), 150
- `__init__()` (Goulib.image.Mode method), 148
- `__init__()` (Goulib.interval.Box method), 163
- `__init__()` (Goulib.interval.Interval method), 159
- `__init__()` (Goulib.interval.Intervals method), 162
- `__init__()` (Goulib.itertools2.iter2 method), 170
- `__init__()` (Goulib.itertools2.keep method), 173
- `__init__()` (Goulib.markup.ArgumentError method), 185
- `__init__()` (Goulib.markup.ClosingError method), 183
- `__init__()` (Goulib.markup.CustomizationError method), 191
- `__init__()` (Goulib.markup.DeprecationError method), 188
- `__init__()` (Goulib.markup.InvalidElementError method), 187
- `__init__()` (Goulib.markup.ModeError method), 189
- `__init__()` (Goulib.markup.OpeningError method), 184
- `__init__()` (Goulib.markup.element method), 175
- `__init__()` (Goulib.markup.page method), 177
- `__init__()` (Goulib.motion.Actuator method), 210
- `__init__()` (Goulib.motion.PVA method), 203
- `__init__()` (Goulib.motion.Segment method), 204
- `__init__()` (Goulib.motion.SegmentPoly method), 207
- `__init__()` (Goulib.motion.Segments method), 206
- `__init__()` (Goulib.motion.TimeDiagram method), 211
- `__init__()` (Goulib.optim.BinDict method), 216
- `__init__()` (Goulib.optim.BinList method), 219
- `__init__()` (Goulib.optim.DifferentialEvolution method), 221
- `__init__()` (Goulib.optim.ObjectiveFunction method), 214
- `__init__()` (Goulib.piecewise.Piecewise method), 222
- `__init__()` (Goulib.polynomial.Polynomial method), 226
- `__init__()` (Goulib.statemachine.Simulation method), 229
- `__init__()` (Goulib.statemachine.StateChangeLog method), 230
- `__init__()` (Goulib.statemachine.StateMachine method), 230

- `__init__()` (Goulib.statemachine.TimeMarker method), 230
- `__init__()` (Goulib.statemachine.TooLateLog method), 230
- `__init__()` (Goulib.statemachine.WaitLog method), 230
- `__init__()` (Goulib.stats.Discrete method), 233
- `__init__()` (Goulib.stats.Normal method), 237
- `__init__()` (Goulib.stats.PDF method), 235
- `__init__()` (Goulib.stats.Stats method), 231
- `__init__()` (Goulib.table.Cell method), 240
- `__init__()` (Goulib.table.Row method), 241
- `__init__()` (Goulib.table.Table method), 243
- `__init__()` (Goulib.tests.TestCase method), 248
- `__init__()` (Goulib.workdays.WorkCalendar method), 253
- `__inv__()` (Goulib.image.Image method), 152
- `__invert__()` (Goulib.expr.Expr method), 63
- `__invert__()` (Goulib.piecewise.Piecewise method), 223
- `__invert__()` (Goulib.polynomial.Polynomial method), 227
- `__invert__()` (Goulib.stats.Normal method), 238
- `__invert__()` (Goulib.stats.PDF method), 235
- `__isub__()` (Goulib.optim.BinDict method), 216
- `__isub__()` (Goulib.optim.BinList method), 218
- `__iter__` (Goulib.colors.Palette attribute), 9
- `__iter__` (Goulib.container.Record attribute), 11
- `__iter__` (Goulib.drawing.BBox attribute), 35
- `__iter__` (Goulib.drawing.Chain attribute), 50
- `__iter__` (Goulib.drawing.Drawing attribute), 59
- `__iter__` (Goulib.drawing.Group attribute), 43
- `__iter__` (Goulib.drawing.Rect attribute), 53
- `__iter__` (Goulib.graph.index.Index attribute), 111
- `__iter__` (Goulib.interval.Box attribute), 164
- `__iter__` (Goulib.interval.Interval attribute), 160
- `__iter__` (Goulib.optim.BinDict attribute), 217
- `__iter__` (Goulib.optim.BinList attribute), 219
- `__iter__` (Goulib.table.Table attribute), 245
- `__iter__()` (Goulib.container.Sequence method), 16
- `__iter__()` (Goulib.container.SortedCollection method), 14
- `__iter__()` (Goulib.drawing.Instance method), 31, 46
- `__iter__()` (Goulib.geom.Matrix3 method), 96
- `__iter__()` (Goulib.geom.Point2 method), 75
- `__iter__()` (Goulib.geom.Vector2 method), 72
- `__iter__()` (Goulib.geom3d.Matrix4 method), 103
- `__iter__()` (Goulib.geom3d.Vector3 method), 98
- `__iter__()` (Goulib.graph.DiGraph method), 130
- `__iter__()` (Goulib.graph.GeoGraph method), 115
- `__iter__()` (Goulib.interval.Intervals method), 162
- `__iter__()` (Goulib.itertools2.iter2 method), 170
- `__iter__()` (Goulib.itertools2.keep method), 174
- `__iter__()` (Goulib.piecewise.Piecewise method), 222
- `__le__` (Goulib.colors.Color attribute), 8
- `__le__` (Goulib.colors.Palette attribute), 9
- `__le__` (Goulib.container.Record attribute), 11
- `__le__` (Goulib.container.Sequence attribute), 16
- `__le__` (Goulib.container.SortedCollection attribute), 15
- `__le__` (Goulib.datetime2.date2 attribute), 20
- `__le__` (Goulib.datetime2.datetime2 attribute), 18
- `__le__` (Goulib.datetime2.time2 attribute), 22
- `__le__` (Goulib.datetime2.timedelta2 attribute), 24
- `__le__` (Goulib.drawing.BBox attribute), 35
- `__le__` (Goulib.drawing.Chain attribute), 50
- `__le__` (Goulib.drawing.Drawing attribute), 59
- `__le__` (Goulib.drawing.Entity attribute), 38
- `__le__` (Goulib.drawing.Group attribute), 43
- `__le__` (Goulib.drawing.Instance attribute), 46
- `__le__` (Goulib.drawing.Rect attribute), 53
- `__le__` (Goulib.drawing.Spline attribute), 40
- `__le__` (Goulib.drawing.Text attribute), 57
- `__le__` (Goulib.expr.Expr attribute), 64
- `__le__` (Goulib.expr.LatexVisitor attribute), 66
- `__le__` (Goulib.expr.TextVisitor attribute), 65
- `__le__` (Goulib.geom.Arc2 attribute), 88
- `__le__` (Goulib.geom.Circle attribute), 85
- `__le__` (Goulib.geom.Ellipse attribute), 90
- `__le__` (Goulib.geom.Geometry attribute), 69
- `__le__` (Goulib.geom.Line2 attribute), 79
- `__le__` (Goulib.geom.Matrix3 attribute), 97
- `__le__` (Goulib.geom.Point2 attribute), 75
- `__le__` (Goulib.geom.Ray2 attribute), 80
- `__le__` (Goulib.geom.Segment2 attribute), 82
- `__le__` (Goulib.geom.Vector2 attribute), 73
- `__le__` (Goulib.graph.AGraph attribute), 113
- `__le__` (Goulib.graph.DiGraph attribute), 130
- `__le__` (Goulib.graph.GeoGraph attribute), 115
- `__le__` (Goulib.graph.index attribute), 112
- `__le__` (Goulib.graph.index.Index attribute), 111
- `__le__` (Goulib.graph.index.Property attribute), 110
- `__le__` (Goulib.image.Ditherer attribute), 155
- `__le__` (Goulib.image.ErrorDiffusion attribute), 156
- `__le__` (Goulib.image.FloydSteinberg attribute), 157
- `__le__` (Goulib.image.Image attribute), 153
- `__le__` (Goulib.image.Mode attribute), 149
- `__le__` (Goulib.interval.Box attribute), 164
- `__le__` (Goulib.interval.Interval attribute), 160
- `__le__` (Goulib.interval.Intervals attribute), 162
- `__le__` (Goulib.itertools2.SortingError attribute), 172
- `__le__` (Goulib.itertools2.iter2 attribute), 171
- `__le__` (Goulib.itertools2.keep attribute), 174
- `__le__` (Goulib.markup.ArgumentError attribute), 185
- `__le__` (Goulib.markup.ClosingError attribute), 183
- `__le__` (Goulib.markup.CustomizationError attribute), 190
- `__le__` (Goulib.markup.DeprecationError attribute), 188
- `__le__` (Goulib.markup.InvalidElementError attribute), 187
- `__le__` (Goulib.markup.MarkupError attribute), 182
- `__le__` (Goulib.markup.ModeError attribute), 189

- `__le__` (Goulib.markup.OpeningError attribute), 184
- `__le__` (Goulib.markup.dummy attribute), 180
- `__le__` (Goulib.markup.element attribute), 175
- `__le__` (Goulib.markup.page attribute), 179
- `__le__` (Goulib.markup.russell attribute), 181
- `__le__` (Goulib.motion.Actuator attribute), 211
- `__le__` (Goulib.motion.PVA attribute), 203
- `__le__` (Goulib.motion.Segment attribute), 205
- `__le__` (Goulib.motion.SegmentPoly attribute), 208
- `__le__` (Goulib.motion.Segments attribute), 206
- `__le__` (Goulib.motion.TimeDiagram attribute), 212
- `__le__` (Goulib.optim.BinDict attribute), 217
- `__le__` (Goulib.optim.BinList attribute), 219
- `__le__` (Goulib.optim.DifferentialEvolution attribute), 222
- `__le__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__le__` (Goulib.piecewise.Piecewise attribute), 223
- `__le__` (Goulib.plot.Plot attribute), 225
- `__le__` (Goulib.polynomial.Polynomial attribute), 227
- `__le__` (Goulib.stats.Discrete attribute), 234
- `__le__` (Goulib.stats.Normal attribute), 238
- `__le__` (Goulib.stats.PDF attribute), 235
- `__le__` (Goulib.stats.Stats attribute), 232
- `__le__` (Goulib.table.Cell attribute), 241
- `__le__` (Goulib.table.Row attribute), 242
- `__le__` (Goulib.table.Table attribute), 245
- `__le__` (Goulib.tests.TestCase attribute), 248
- `__le__` (Goulib.workdays.WorkCalendar attribute), 255
- `__len__` (Goulib.colors.Palette attribute), 9
- `__len__` (Goulib.container.Record attribute), 11
- `__len__` (Goulib.drawing.BBox attribute), 35
- `__len__` (Goulib.drawing.Chain attribute), 50
- `__len__` (Goulib.drawing.Drawing attribute), 59
- `__len__` (Goulib.drawing.Group attribute), 43
- `__len__` (Goulib.drawing.Rect attribute), 53
- `__len__` (Goulib.graph.index.Index attribute), 111
- `__len__` (Goulib.interval.Box attribute), 164
- `__len__` (Goulib.interval.Interval attribute), 161
- `__len__` (Goulib.optim.BinDict attribute), 217
- `__len__` (Goulib.optim.BinList attribute), 219
- `__len__` (Goulib.table.Table attribute), 245
- `__len__()` (Goulib.container.SortedCollection method), 14
- `__len__()` (Goulib.geom.Point2 method), 75
- `__len__()` (Goulib.geom.Vector2 method), 72
- `__len__()` (Goulib.geom3d.Vector3 method), 98
- `__len__()` (Goulib.graph.DiGraph method), 130
- `__len__()` (Goulib.graph.GeoGraph method), 115
- `__len__()` (Goulib.interval.Intervals method), 162
- `__len__()` (Goulib.piecewise.Piecewise method), 222
- `__lshift__()` (Goulib.expr.Expr method), 63
- `__lshift__()` (Goulib.piecewise.Piecewise method), 223
- `__lshift__()` (Goulib.polynomial.Polynomial method), 227
- `__lshift__()` (Goulib.stats.Normal method), 238
- `__lshift__()` (Goulib.stats.PDF method), 235
- `__lt__` (Goulib.colors.Color attribute), 8
- `__lt__` (Goulib.colors.Palette attribute), 9
- `__lt__` (Goulib.container.Record attribute), 12
- `__lt__` (Goulib.container.Sequence attribute), 16
- `__lt__` (Goulib.container.SortedCollection attribute), 15
- `__lt__` (Goulib.datetime2.date2 attribute), 21
- `__lt__` (Goulib.datetime2.datetime2 attribute), 18
- `__lt__` (Goulib.datetime2.time2 attribute), 22
- `__lt__` (Goulib.datetime2.timedelta2 attribute), 24
- `__lt__` (Goulib.drawing.BBox attribute), 35
- `__lt__` (Goulib.drawing.Chain attribute), 50
- `__lt__` (Goulib.drawing.Drawing attribute), 59
- `__lt__` (Goulib.drawing.Entity attribute), 38
- `__lt__` (Goulib.drawing.Group attribute), 43
- `__lt__` (Goulib.drawing.Instance attribute), 46
- `__lt__` (Goulib.drawing.Rect attribute), 53
- `__lt__` (Goulib.drawing.Spline attribute), 40
- `__lt__` (Goulib.drawing.Text attribute), 57
- `__lt__` (Goulib.expr.LatexVisitor attribute), 66
- `__lt__` (Goulib.expr.TextVisitor attribute), 65
- `__lt__` (Goulib.geom.Arc2 attribute), 88
- `__lt__` (Goulib.geom.Circle attribute), 85
- `__lt__` (Goulib.geom.Ellipse attribute), 90
- `__lt__` (Goulib.geom.Geometry attribute), 69
- `__lt__` (Goulib.geom.Line2 attribute), 79
- `__lt__` (Goulib.geom.Matrix3 attribute), 97
- `__lt__` (Goulib.geom.Point2 attribute), 75
- `__lt__` (Goulib.geom.Ray2 attribute), 80
- `__lt__` (Goulib.geom.Segment2 attribute), 82
- `__lt__` (Goulib.geom.Vector2 attribute), 73
- `__lt__` (Goulib.graph.AGraph attribute), 113
- `__lt__` (Goulib.graph.DiGraph attribute), 131
- `__lt__` (Goulib.graph.GeoGraph attribute), 115
- `__lt__` (Goulib.graph.index attribute), 112
- `__lt__` (Goulib.graph.index.Index attribute), 111
- `__lt__` (Goulib.graph.index.Property attribute), 110
- `__lt__` (Goulib.image.Ditherer attribute), 155
- `__lt__` (Goulib.image.ErrorDiffusion attribute), 156
- `__lt__` (Goulib.image.FloydSteinberg attribute), 157
- `__lt__` (Goulib.image.Mode attribute), 149
- `__lt__` (Goulib.interval.Box attribute), 164
- `__lt__` (Goulib.interval.Intervals attribute), 162
- `__lt__` (Goulib.itertools2.SortingError attribute), 172
- `__lt__` (Goulib.itertools2.iter2 attribute), 171
- `__lt__` (Goulib.itertools2.keep attribute), 174
- `__lt__` (Goulib.markup.ArgumentError attribute), 185
- `__lt__` (Goulib.markup.ClosingError attribute), 183
- `__lt__` (Goulib.markup.CustomizationError attribute), 190
- `__lt__` (Goulib.markup.DeprecationError attribute), 188
- `__lt__` (Goulib.markup.InvalidElementError attribute), 187

- `__lt__` (Goulib.markup.MarkupError attribute), 182
- `__lt__` (Goulib.markup.ModeError attribute), 189
- `__lt__` (Goulib.markup.OpeningError attribute), 184
- `__lt__` (Goulib.markup.dummy attribute), 180
- `__lt__` (Goulib.markup.element attribute), 175
- `__lt__` (Goulib.markup.page attribute), 179
- `__lt__` (Goulib.markup.russell attribute), 181
- `__lt__` (Goulib.motion.Actuator attribute), 211
- `__lt__` (Goulib.motion.PVA attribute), 203
- `__lt__` (Goulib.motion.Segment attribute), 205
- `__lt__` (Goulib.motion.SegmentPoly attribute), 208
- `__lt__` (Goulib.motion.Segments attribute), 206
- `__lt__` (Goulib.motion.TimeDiagram attribute), 212
- `__lt__` (Goulib.optim.BinDict attribute), 217
- `__lt__` (Goulib.optim.BinList attribute), 219
- `__lt__` (Goulib.optim.DifferentialEvolution attribute), 222
- `__lt__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__lt__` (Goulib.plot.Plot attribute), 225
- `__lt__` (Goulib.stats.Discrete attribute), 234
- `__lt__` (Goulib.stats.Stats attribute), 232
- `__lt__` (Goulib.table.Cell attribute), 241
- `__lt__` (Goulib.table.Row attribute), 242
- `__lt__` (Goulib.table.Table attribute), 245
- `__lt__` (Goulib.tests.TestCase attribute), 248
- `__lt__` (Goulib.workdays.WorkCalendar attribute), 255
- `__lt__()` (Goulib.expr.Expr method), 63
- `__lt__()` (Goulib.image.Image method), 150
- `__lt__()` (Goulib.interval.Interval method), 159
- `__lt__()` (Goulib.piecewise.Piecewise method), 223
- `__lt__()` (Goulib.polynomial.Polynomial method), 226
- `__lt__()` (Goulib.stats.Normal method), 238
- `__lt__()` (Goulib.stats.PDF method), 235
- `__mod__` (Goulib.datetime2.timedelta2 attribute), 24
- `__mod__()` (Goulib.container.Sequence method), 16
- `__mul__` (Goulib.datetime2.timedelta2 attribute), 24
- `__mul__` (Goulib.drawing.BBox attribute), 35
- `__mul__` (Goulib.drawing.Chain attribute), 50
- `__mul__` (Goulib.drawing.Drawing attribute), 59
- `__mul__` (Goulib.drawing.Group attribute), 43
- `__mul__` (Goulib.drawing.Rect attribute), 53
- `__mul__` (Goulib.interval.Box attribute), 165
- `__mul__` (Goulib.interval.Interval attribute), 161
- `__mul__` (Goulib.optim.BinList attribute), 219
- `__mul__` (Goulib.table.Table attribute), 245
- `__mul__()` (Goulib.colors.Color method), 7
- `__mul__()` (Goulib.expr.Expr method), 63
- `__mul__()` (Goulib.geom.Matrix3 method), 96
- `__mul__()` (Goulib.geom.Point2 method), 75
- `__mul__()` (Goulib.geom.Vector2 method), 72
- `__mul__()` (Goulib.geom3d.Matrix4 method), 103
- `__mul__()` (Goulib.geom3d.Quaternion method), 106
- `__mul__()` (Goulib.geom3d.Vector3 method), 98
- `__mul__()` (Goulib.image.Image method), 153
- `__mul__()` (Goulib.piecewise.Piecewise method), 223
- `__mul__()` (Goulib.polynomial.Polynomial method), 227
- `__mul__()` (Goulib.stats.Discrete method), 234
- `__mul__()` (Goulib.stats.Normal method), 237
- `__mul__()` (Goulib.stats.PDF method), 235
- `__mul__()` (Goulib.stats.Stats method), 232
- `__ne__` (Goulib.colors.Color attribute), 8
- `__ne__` (Goulib.colors.Palette attribute), 9
- `__ne__` (Goulib.container.Record attribute), 12
- `__ne__` (Goulib.container.Sequence attribute), 17
- `__ne__` (Goulib.container.SortedCollection attribute), 15
- `__ne__` (Goulib.datetime2.date2 attribute), 21
- `__ne__` (Goulib.datetime2.datetime2 attribute), 18
- `__ne__` (Goulib.datetime2.time2 attribute), 22
- `__ne__` (Goulib.datetime2.timedelta2 attribute), 24
- `__ne__` (Goulib.drawing.BBox attribute), 36
- `__ne__` (Goulib.drawing.Chain attribute), 50
- `__ne__` (Goulib.drawing.Drawing attribute), 59
- `__ne__` (Goulib.drawing.Entity attribute), 38
- `__ne__` (Goulib.drawing.Group attribute), 43
- `__ne__` (Goulib.drawing.Instance attribute), 46
- `__ne__` (Goulib.drawing.Rect attribute), 53
- `__ne__` (Goulib.drawing.Spline attribute), 40
- `__ne__` (Goulib.drawing.Text attribute), 57
- `__ne__` (Goulib.expr.Expr attribute), 64
- `__ne__` (Goulib.expr.LatexVisitor attribute), 66
- `__ne__` (Goulib.expr.TextVisitor attribute), 65
- `__ne__` (Goulib.geom.Arc2 attribute), 88
- `__ne__` (Goulib.geom.Circle attribute), 85
- `__ne__` (Goulib.geom.Ellipse attribute), 90
- `__ne__` (Goulib.geom.Geometry attribute), 69
- `__ne__` (Goulib.geom.Line2 attribute), 79
- `__ne__` (Goulib.geom.Matrix3 attribute), 97
- `__ne__` (Goulib.geom.Point2 attribute), 75
- `__ne__` (Goulib.geom.Ray2 attribute), 80
- `__ne__` (Goulib.geom.Segment2 attribute), 82
- `__ne__` (Goulib.geom.Vector2 attribute), 73
- `__ne__` (Goulib.graph.AGraph attribute), 113
- `__ne__` (Goulib.graph.DiGraph attribute), 131
- `__ne__` (Goulib.graph.GeoGraph attribute), 115
- `__ne__` (Goulib.graph.index attribute), 112
- `__ne__` (Goulib.graph.index.Index attribute), 111
- `__ne__` (Goulib.graph.index.Property attribute), 110
- `__ne__` (Goulib.image.Ditherer attribute), 155
- `__ne__` (Goulib.image.ErrorDiffusion attribute), 156
- `__ne__` (Goulib.image.FloydSteinberg attribute), 157
- `__ne__` (Goulib.image.Image attribute), 153
- `__ne__` (Goulib.image.Mode attribute), 149
- `__ne__` (Goulib.interval.Box attribute), 165
- `__ne__` (Goulib.interval.Interval attribute), 161
- `__ne__` (Goulib.interval.Intervals attribute), 162
- `__ne__` (Goulib.itertools2.SortingError attribute), 172
- `__ne__` (Goulib.itertools2.iter2 attribute), 171
- `__ne__` (Goulib.itertools2.keep attribute), 174

- `__ne__` (Goulib.markup.ArgumentError attribute), 185
- `__ne__` (Goulib.markup.ClosingError attribute), 183
- `__ne__` (Goulib.markup.CustomizationError attribute), 190
- `__ne__` (Goulib.markup.DeprecationError attribute), 188
- `__ne__` (Goulib.markup.InvalidElementError attribute), 187
- `__ne__` (Goulib.markup.MarkupError attribute), 182
- `__ne__` (Goulib.markup.ModeError attribute), 189
- `__ne__` (Goulib.markup.OpeningError attribute), 184
- `__ne__` (Goulib.markup.dummy attribute), 180
- `__ne__` (Goulib.markup.element attribute), 176
- `__ne__` (Goulib.markup.page attribute), 179
- `__ne__` (Goulib.markup.russell attribute), 181
- `__ne__` (Goulib.motion.Actuator attribute), 211
- `__ne__` (Goulib.motion.PVA attribute), 203
- `__ne__` (Goulib.motion.Segment attribute), 205
- `__ne__` (Goulib.motion.SegmentPoly attribute), 208
- `__ne__` (Goulib.motion.Segments attribute), 207
- `__ne__` (Goulib.motion.TimeDiagram attribute), 212
- `__ne__` (Goulib.optim.BinDict attribute), 217
- `__ne__` (Goulib.optim.BinList attribute), 219
- `__ne__` (Goulib.optim.DifferentialEvolution attribute), 222
- `__ne__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__ne__` (Goulib.piecewise.Piecewise attribute), 223
- `__ne__` (Goulib.plot.Plot attribute), 225
- `__ne__` (Goulib.polynomial.Polynomial attribute), 227
- `__ne__` (Goulib.stats.Discrete attribute), 234
- `__ne__` (Goulib.stats.Normal attribute), 238
- `__ne__` (Goulib.stats.PDF attribute), 235
- `__ne__` (Goulib.stats.Stats attribute), 233
- `__ne__` (Goulib.table.Cell attribute), 241
- `__ne__` (Goulib.table.Row attribute), 242
- `__ne__` (Goulib.table.Table attribute), 245
- `__ne__` (Goulib.tests.TestCase attribute), 248
- `__ne__` (Goulib.workdays.WorkCalendar attribute), 255
- `__new__` (Goulib.geom3d.Vector3 method), 98
- `__neg__` (Goulib.datetime2.timedelta2 attribute), 24
- `__neg__` (Goulib.colors.Color method), 7
- `__neg__` (Goulib.expr.Expr method), 63
- `__neg__` (Goulib.geom.Point2 method), 75
- `__neg__` (Goulib.geom.Vector2 method), 73
- `__neg__` (Goulib.geom3d.Vector3 method), 99
- `__neg__` (Goulib.image.Image method), 152
- `__neg__` (Goulib.piecewise.Piecewise method), 224
- `__neg__` (Goulib.polynomial.Polynomial method), 227
- `__neg__` (Goulib.stats.Discrete method), 234
- `__neg__` (Goulib.stats.Normal method), 237
- `__neg__` (Goulib.stats.PDF method), 235
- `__neg__` (Goulib.stats.Stats method), 232
- `__new__` (Goulib.colors.Color method), 8
- `__new__` (Goulib.colors.Palette method), 9
- `__new__` (Goulib.container.Record method), 12
- `__new__` (Goulib.container.Sequence method), 17
- `__new__` (Goulib.container.SortedCollection method), 15
- `__new__` (Goulib.datetime2.date2 method), 21
- `__new__` (Goulib.datetime2.datetime2 method), 18
- `__new__` (Goulib.datetime2.time2 method), 22
- `__new__` (Goulib.datetime2.timedelta2 method), 24
- `__new__` (Goulib.drawing.BBox method), 36
- `__new__` (Goulib.drawing.Chain method), 50
- `__new__` (Goulib.drawing.Drawing method), 59
- `__new__` (Goulib.drawing.Entity method), 38
- `__new__` (Goulib.drawing.Group method), 43
- `__new__` (Goulib.drawing.Instance method), 46
- `__new__` (Goulib.drawing.Rect method), 53
- `__new__` (Goulib.drawing.Spline method), 40
- `__new__` (Goulib.drawing.Text method), 57
- `__new__` (Goulib.expr.Expr method), 64
- `__new__` (Goulib.expr.LatexVisitor method), 66
- `__new__` (Goulib.expr.TextVisitor method), 65
- `__new__` (Goulib.geom.Arc2 method), 88
- `__new__` (Goulib.geom.Circle method), 85
- `__new__` (Goulib.geom.Ellipse method), 91
- `__new__` (Goulib.geom.Geometry method), 69
- `__new__` (Goulib.geom.Line2 method), 79
- `__new__` (Goulib.geom.Matrix3 method), 97
- `__new__` (Goulib.geom.Point2 method), 75
- `__new__` (Goulib.geom.Ray2 method), 80
- `__new__` (Goulib.geom.Segment2 method), 82
- `__new__` (Goulib.geom.Vector2 method), 73
- `__new__` (Goulib.graph.AGraph method), 113
- `__new__` (Goulib.graph.DiGraph method), 131
- `__new__` (Goulib.graph.GeoGraph method), 116
- `__new__` (Goulib.graph.index method), 112
- `__new__` (Goulib.graph.index.Index method), 111
- `__new__` (Goulib.graph.index.Property method), 110
- `__new__` (Goulib.image.Ditherer method), 155
- `__new__` (Goulib.image.ErrorDiffusion method), 156
- `__new__` (Goulib.image.FloydSteinberg method), 157
- `__new__` (Goulib.image.Image method), 153
- `__new__` (Goulib.image.Mode method), 149
- `__new__` (Goulib.interval.Box method), 165
- `__new__` (Goulib.interval.Interval method), 161
- `__new__` (Goulib.interval.Intervals method), 162
- `__new__` (Goulib.itertools2.SortingError method), 172
- `__new__` (Goulib.itertools2.iter2 method), 171
- `__new__` (Goulib.itertools2.keep method), 174
- `__new__` (Goulib.markup.ArgumentError method), 186
- `__new__` (Goulib.markup.ClosingError method), 183
- `__new__` (Goulib.markup.CustomizationError method), 190
- `__new__` (Goulib.markup.DeprecationError method), 188
- `__new__` (Goulib.markup.InvalidElementError method), 187

- `__new__()` (Goulib.markup.MarkupError method), 182
- `__new__()` (Goulib.markup.ModeError method), 189
- `__new__()` (Goulib.markup.OpeningError method), 184
- `__new__()` (Goulib.markup.dummy method), 180
- `__new__()` (Goulib.markup.element method), 176
- `__new__()` (Goulib.markup.page method), 179
- `__new__()` (Goulib.markup.russell method), 181
- `__new__()` (Goulib.motion.Actuator method), 211
- `__new__()` (Goulib.motion.PVA method), 203
- `__new__()` (Goulib.motion.Segment method), 205
- `__new__()` (Goulib.motion.SegmentPoly method), 208
- `__new__()` (Goulib.motion.Segments method), 207
- `__new__()` (Goulib.motion.TimeDiagram method), 212
- `__new__()` (Goulib.optim.BinDict method), 217
- `__new__()` (Goulib.optim.BinList method), 219
- `__new__()` (Goulib.optim.DifferentialEvolution method), 222
- `__new__()` (Goulib.optim.ObjectiveFunction method), 215
- `__new__()` (Goulib.piecewise.Piecewise method), 224
- `__new__()` (Goulib.plot.Plot method), 225
- `__new__()` (Goulib.polynomial.Polynomial method), 227
- `__new__()` (Goulib.stats.Discrete method), 234
- `__new__()` (Goulib.stats.Normal method), 238
- `__new__()` (Goulib.stats.PDF method), 235
- `__new__()` (Goulib.stats.Stats method), 233
- `__new__()` (Goulib.table.Cell method), 241
- `__new__()` (Goulib.table.Row method), 242
- `__new__()` (Goulib.table.Table method), 245
- `__new__()` (Goulib.tests.TestCase method), 248
- `__new__()` (Goulib.workdays.WorkCalendar method), 255
- `__next__()` (Goulib.itertools2.iter2 method), 170
- `__next__()` (Goulib.itertools2.keep method), 173
- `__nonzero__()` (Goulib.drawing.BBox method), 36
- `__nonzero__()` (Goulib.geom3d.Vector3 method), 98
- `__nonzero__()` (Goulib.graph.DiGraph method), 131
- `__nonzero__()` (Goulib.graph.GeoGraph method), 116
- `__nonzero__()` (Goulib.image.Image method), 150
- `__nonzero__()` (Goulib.interval.Box method), 164
- `__nonzero__()` (Goulib.interval.Interval method), 160
- `__or__()` (Goulib.container.Sequence method), 16
- `__or__()` (Goulib.expr.Expr method), 63
- `__or__()` (Goulib.piecewise.Piecewise method), 224
- `__or__()` (Goulib.polynomial.Polynomial method), 227
- `__or__()` (Goulib.stats.Normal method), 238
- `__or__()` (Goulib.stats.PDF method), 235
- `__pos__()` (Goulib.datetime2.timedelta2 attribute), 24
- `__pos__()` (Goulib.geom.Point2 method), 75
- `__pos__()` (Goulib.geom.Vector2 method), 73
- `__pos__()` (Goulib.geom3d.Vector3 method), 99
- `__pow__()` (Goulib.expr.Expr method), 63
- `__pow__()` (Goulib.piecewise.Piecewise method), 224
- `__pow__()` (Goulib.polynomial.Polynomial method), 227
- `__pow__()` (Goulib.stats.Discrete method), 234
- `__pow__()` (Goulib.stats.Normal method), 238
- `__pow__()` (Goulib.stats.PDF method), 236
- `__pow__()` (Goulib.stats.Stats method), 232
- `__radd__()` (Goulib.datetime2.date2 attribute), 21
- `__radd__()` (Goulib.datetime2.datetime2 attribute), 18
- `__radd__()` (Goulib.datetime2.timedelta2 attribute), 24
- `__radd__()` (Goulib.colors.Color method), 7
- `__radd__()` (Goulib.geom.Point2 method), 75
- `__radd__()` (Goulib.geom.Vector2 method), 72
- `__radd__()` (Goulib.geom3d.Vector3 method), 98
- `__radd__()` (Goulib.image.Image method), 153
- `__radd__()` (Goulib.polynomial.Polynomial method), 226
- `__radd__()` (Goulib.stats.Normal method), 237
- `__rdiv__()` (Goulib.geom.Point2 method), 75
- `__rdiv__()` (Goulib.geom.Vector2 method), 72
- `__rdiv__()` (Goulib.geom3d.Vector3 method), 99
- `__rdivmod__()` (Goulib.datetime2.timedelta2 attribute), 24
- `__reduce__()` (Goulib.colors.Color method), 8
- `__reduce__()` (Goulib.colors.Palette method), 9
- `__reduce__()` (Goulib.container.Record method), 12
- `__reduce__()` (Goulib.container.Sequence method), 17
- `__reduce__()` (Goulib.container.SortedCollection method), 14
- `__reduce__()` (Goulib.datetime2.date2 method), 21
- `__reduce__()` (Goulib.datetime2.datetime2 method), 18
- `__reduce__()` (Goulib.datetime2.time2 method), 22
- `__reduce__()` (Goulib.datetime2.timedelta2 method), 24
- `__reduce__()` (Goulib.drawing.BBox method), 36
- `__reduce__()` (Goulib.drawing.Chain method), 50
- `__reduce__()` (Goulib.drawing.Drawing method), 59
- `__reduce__()` (Goulib.drawing.Entity method), 38
- `__reduce__()` (Goulib.drawing.Group method), 43
- `__reduce__()` (Goulib.drawing.Instance method), 46
- `__reduce__()` (Goulib.drawing.Rect method), 53
- `__reduce__()` (Goulib.drawing.Spline method), 40
- `__reduce__()` (Goulib.drawing.Text method), 57
- `__reduce__()` (Goulib.expr.Expr method), 64
- `__reduce__()` (Goulib.expr.LatexVisitor method), 66
- `__reduce__()` (Goulib.expr.TextVisitor method), 65
- `__reduce__()` (Goulib.geom.Arc2 method), 88
- `__reduce__()` (Goulib.geom.Circle method), 85
- `__reduce__()` (Goulib.geom.Ellipse method), 91
- `__reduce__()` (Goulib.geom.Geometry method), 69
- `__reduce__()` (Goulib.geom.Line2 method), 79
- `__reduce__()` (Goulib.geom.Matrix3 method), 97
- `__reduce__()` (Goulib.geom.Point2 method), 75
- `__reduce__()` (Goulib.geom.Ray2 method), 80
- `__reduce__()` (Goulib.geom.Segment2 method), 82
- `__reduce__()` (Goulib.geom.Vector2 method), 73
- `__reduce__()` (Goulib.graph.AGraph method), 114
- `__reduce__()` (Goulib.graph.DiGraph method), 131
- `__reduce__()` (Goulib.graph.GeoGraph method), 116
- `__reduce__()` (Goulib.graph.index method), 113

- `__reduce__()` (Goulib.graph.index.Index method), 111
- `__reduce__()` (Goulib.graph.index.Property method), 110
- `__reduce__()` (Goulib.image.Ditherer method), 155
- `__reduce__()` (Goulib.image.ErrorDiffusion method), 156
- `__reduce__()` (Goulib.image.FloydSteinberg method), 157
- `__reduce__()` (Goulib.image.Image method), 153
- `__reduce__()` (Goulib.image.Mode method), 149
- `__reduce__()` (Goulib.interval.Box method), 165
- `__reduce__()` (Goulib.interval.Interval method), 161
- `__reduce__()` (Goulib.interval.Intervals method), 162
- `__reduce__()` (Goulib.itertools2.SortingError method), 172
- `__reduce__()` (Goulib.itertools2.iter2 method), 171
- `__reduce__()` (Goulib.itertools2.keep method), 174
- `__reduce__()` (Goulib.markup.ArgumentError method), 186
- `__reduce__()` (Goulib.markup.ClosingError method), 183
- `__reduce__()` (Goulib.markup.CustomizationError method), 190
- `__reduce__()` (Goulib.markup.DeprecationError method), 188
- `__reduce__()` (Goulib.markup.InvalidElementError method), 187
- `__reduce__()` (Goulib.markup.MarkupError method), 182
- `__reduce__()` (Goulib.markup.ModeError method), 189
- `__reduce__()` (Goulib.markup.OpeningError method), 184
- `__reduce__()` (Goulib.markup.dummy method), 180
- `__reduce__()` (Goulib.markup.element method), 176
- `__reduce__()` (Goulib.markup.page method), 179
- `__reduce__()` (Goulib.markup.russell method), 181
- `__reduce__()` (Goulib.motion.Actuator method), 211
- `__reduce__()` (Goulib.motion.PVA method), 203
- `__reduce__()` (Goulib.motion.Segment method), 205
- `__reduce__()` (Goulib.motion.SegmentPoly method), 208
- `__reduce__()` (Goulib.motion.Segments method), 207
- `__reduce__()` (Goulib.motion.TimeDiagram method), 212
- `__reduce__()` (Goulib.optim.BinDict method), 217
- `__reduce__()` (Goulib.optim.BinList method), 219
- `__reduce__()` (Goulib.optim.DifferentialEvolution method), 222
- `__reduce__()` (Goulib.optim.ObjectiveFunction method), 215
- `__reduce__()` (Goulib.piecewise.Piecewise method), 224
- `__reduce__()` (Goulib.plot.Plot method), 225
- `__reduce__()` (Goulib.polynomial.Polynomial method), 227
- `__reduce__()` (Goulib.stats.Discrete method), 234
- `__reduce__()` (Goulib.stats.Normal method), 238
- `__reduce__()` (Goulib.stats.PDF method), 236
- `__reduce__()` (Goulib.stats.Stats method), 233
- `__reduce__()` (Goulib.table.Cell method), 241
- `__reduce__()` (Goulib.table.Row method), 242
- `__reduce__()` (Goulib.table.Table method), 245
- `__reduce__()` (Goulib.tests.TestCase method), 248
- `__reduce__()` (Goulib.workdays.WorkCalendar method), 255
- `__reduce_ex__()` (Goulib.colors.Color method), 8
- `__reduce_ex__()` (Goulib.colors.Palette method), 9
- `__reduce_ex__()` (Goulib.container.Record method), 12
- `__reduce_ex__()` (Goulib.container.Sequence method), 17
- `__reduce_ex__()` (Goulib.container.SortedCollection method), 15
- `__reduce_ex__()` (Goulib.datetime2.date2 method), 21
- `__reduce_ex__()` (Goulib.datetime2.datetime2 method), 18
- `__reduce_ex__()` (Goulib.datetime2.time2 method), 23
- `__reduce_ex__()` (Goulib.datetime2.timedelta2 method), 24
- `__reduce_ex__()` (Goulib.drawing.BBox method), 36
- `__reduce_ex__()` (Goulib.drawing.Chain method), 50
- `__reduce_ex__()` (Goulib.drawing.Drawing method), 59
- `__reduce_ex__()` (Goulib.drawing.Entity method), 39
- `__reduce_ex__()` (Goulib.drawing.Group method), 43
- `__reduce_ex__()` (Goulib.drawing.Instance method), 46
- `__reduce_ex__()` (Goulib.drawing.Rect method), 53
- `__reduce_ex__()` (Goulib.drawing.Spline method), 40
- `__reduce_ex__()` (Goulib.drawing.Text method), 57
- `__reduce_ex__()` (Goulib.expr.Expr method), 64
- `__reduce_ex__()` (Goulib.expr.LatexVisitor method), 66
- `__reduce_ex__()` (Goulib.expr.TextVisitor method), 65
- `__reduce_ex__()` (Goulib.geom.Arc2 method), 88
- `__reduce_ex__()` (Goulib.geom.Circle method), 85
- `__reduce_ex__()` (Goulib.geom.Ellipse method), 91
- `__reduce_ex__()` (Goulib.geom.Geometry method), 69
- `__reduce_ex__()` (Goulib.geom.Line2 method), 79
- `__reduce_ex__()` (Goulib.geom.Matrix3 method), 97
- `__reduce_ex__()` (Goulib.geom.Point2 method), 76
- `__reduce_ex__()` (Goulib.geom.Ray2 method), 80
- `__reduce_ex__()` (Goulib.geom.Segment2 method), 82
- `__reduce_ex__()` (Goulib.geom.Vector2 method), 73
- `__reduce_ex__()` (Goulib.graph.AGraph method), 114
- `__reduce_ex__()` (Goulib.graph.DiGraph method), 131
- `__reduce_ex__()` (Goulib.graph.GeoGraph method), 116
- `__reduce_ex__()` (Goulib.graph.index method), 113
- `__reduce_ex__()` (Goulib.graph.index.Index method), 111
- `__reduce_ex__()` (Goulib.graph.index.Property method), 110
- `__reduce_ex__()` (Goulib.image.Ditherer method), 156
- `__reduce_ex__()` (Goulib.image.ErrorDiffusion method), 156
- `__reduce_ex__()` (Goulib.image.FloydSteinberg method), 157
- `__reduce_ex__()` (Goulib.image.Image method), 154
- `__reduce_ex__()` (Goulib.image.Mode method), 149

- `__reduce_ex__()` (Goulib.interval.Box method), 165
- `__reduce_ex__()` (Goulib.interval.Interval method), 161
- `__reduce_ex__()` (Goulib.interval.Intervals method), 162
- `__reduce_ex__()` (Goulib.itertools2.SortingError method), 172
- `__reduce_ex__()` (Goulib.itertools2.iter2 method), 171
- `__reduce_ex__()` (Goulib.itertools2.keep method), 174
- `__reduce_ex__()` (Goulib.markup.ArgumentError method), 186
- `__reduce_ex__()` (Goulib.markup.ClosingError method), 183
- `__reduce_ex__()` (Goulib.markup.CustomizationError method), 190
- `__reduce_ex__()` (Goulib.markup.DeprecationError method), 188
- `__reduce_ex__()` (Goulib.markup.InvalidElementError method), 187
- `__reduce_ex__()` (Goulib.markup.MarkupError method), 182
- `__reduce_ex__()` (Goulib.markup.ModeError method), 189
- `__reduce_ex__()` (Goulib.markup.OpeningError method), 184
- `__reduce_ex__()` (Goulib.markup.dummy method), 180
- `__reduce_ex__()` (Goulib.markup.element method), 176
- `__reduce_ex__()` (Goulib.markup.page method), 179
- `__reduce_ex__()` (Goulib.markup.russell method), 181
- `__reduce_ex__()` (Goulib.motion.Actuator method), 211
- `__reduce_ex__()` (Goulib.motion.PVA method), 203
- `__reduce_ex__()` (Goulib.motion.Segment method), 205
- `__reduce_ex__()` (Goulib.motion.SegmentPoly method), 208
- `__reduce_ex__()` (Goulib.motion.Segments method), 207
- `__reduce_ex__()` (Goulib.motion.TimeDiagram method), 212
- `__reduce_ex__()` (Goulib.optim.BinDict method), 217
- `__reduce_ex__()` (Goulib.optim.BinList method), 219
- `__reduce_ex__()` (Goulib.optim.DifferentialEvolution method), 222
- `__reduce_ex__()` (Goulib.optim.ObjectiveFunction method), 215
- `__reduce_ex__()` (Goulib.piecewise.Piecewise method), 224
- `__reduce_ex__()` (Goulib.plot.Plot method), 225
- `__reduce_ex__()` (Goulib.polynomial.Polynomial method), 227
- `__reduce_ex__()` (Goulib.stats.Discrete method), 234
- `__reduce_ex__()` (Goulib.stats.Normal method), 238
- `__reduce_ex__()` (Goulib.stats.PDF method), 236
- `__reduce_ex__()` (Goulib.stats.Stats method), 233
- `__reduce_ex__()` (Goulib.table.Cell method), 241
- `__reduce_ex__()` (Goulib.table.Row method), 242
- `__reduce_ex__()` (Goulib.table.Table method), 245
- `__reduce_ex__()` (Goulib.tests.TestCase method), 248
- `__reduce_ex__()` (Goulib.workdays.WorkCalendar method), 255
- `__repr__` (Goulib.container.Record attribute), 12
- `__repr__` (Goulib.datetime2.date2 attribute), 21
- `__repr__` (Goulib.datetime2.datetime2 attribute), 18
- `__repr__` (Goulib.datetime2.time2 attribute), 23
- `__repr__` (Goulib.datetime2.timedelta2 attribute), 25
- `__repr__` (Goulib.drawing.BBox attribute), 36
- `__repr__` (Goulib.drawing.Drawing attribute), 59
- `__repr__` (Goulib.drawing.Group attribute), 43
- `__repr__` (Goulib.expr.LatexVisitor attribute), 66
- `__repr__` (Goulib.expr.TextVisitor attribute), 65
- `__repr__` (Goulib.geom.Geometry attribute), 69
- `__repr__` (Goulib.graph.AGraph attribute), 114
- `__repr__` (Goulib.graph.DiGraph attribute), 131
- `__repr__` (Goulib.graph.GeoGraph attribute), 116
- `__repr__` (Goulib.graph.index attribute), 113
- `__repr__` (Goulib.graph.index.Index attribute), 111
- `__repr__` (Goulib.graph.index.Property attribute), 110
- `__repr__` (Goulib.interval.Box attribute), 165
- `__repr__` (Goulib.itertools2.SortingError attribute), 172
- `__repr__` (Goulib.itertools2.iter2 attribute), 171
- `__repr__` (Goulib.itertools2.keep attribute), 174
- `__repr__` (Goulib.markup.ArgumentError attribute), 186
- `__repr__` (Goulib.markup.ClosingError attribute), 183
- `__repr__` (Goulib.markup.CustomizationError attribute), 190
- `__repr__` (Goulib.markup.DeprecationError attribute), 188
- `__repr__` (Goulib.markup.InvalidElementError attribute), 187
- `__repr__` (Goulib.markup.MarkupError attribute), 182
- `__repr__` (Goulib.markup.ModeError attribute), 189
- `__repr__` (Goulib.markup.OpeningError attribute), 184
- `__repr__` (Goulib.markup.dummy attribute), 180
- `__repr__` (Goulib.markup.element attribute), 176
- `__repr__` (Goulib.markup.page attribute), 179
- `__repr__` (Goulib.markup.russell attribute), 181
- `__repr__` (Goulib.motion.Actuator attribute), 211
- `__repr__` (Goulib.motion.PVA attribute), 203
- `__repr__` (Goulib.motion.Segment attribute), 205
- `__repr__` (Goulib.motion.SegmentPoly attribute), 208
- `__repr__` (Goulib.motion.Segments attribute), 207
- `__repr__` (Goulib.optim.DifferentialEvolution attribute), 222
- `__repr__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__repr__` (Goulib.plot.Plot attribute), 226
- `__repr__` (Goulib.workdays.WorkCalendar attribute), 255
- `__repr__()` (Goulib.colors.Color method), 7
- `__repr__()` (Goulib.colors.Palette method), 8
- `__repr__()` (Goulib.container.Sequence method), 16
- `__repr__()` (Goulib.container.SortedCollection method), 14
- `__repr__()` (Goulib.drawing.Chain method), 31, 48

- `__repr__()` (Goulib.drawing.Entity method), 28, 37
- `__repr__()` (Goulib.drawing.Instance method), 31, 46
- `__repr__()` (Goulib.drawing.Rect method), 32, 52
- `__repr__()` (Goulib.drawing.Spline method), 40
- `__repr__()` (Goulib.drawing.Text method), 57
- `__repr__()` (Goulib.expr.Expr method), 63
- `__repr__()` (Goulib.geom.Arc2 method), 87
- `__repr__()` (Goulib.geom.Circle method), 84
- `__repr__()` (Goulib.geom.Ellipse method), 90
- `__repr__()` (Goulib.geom.Line2 method), 78
- `__repr__()` (Goulib.geom.Matrix3 method), 96
- `__repr__()` (Goulib.geom.Point2 method), 76
- `__repr__()` (Goulib.geom.Ray2 method), 80
- `__repr__()` (Goulib.geom.Segment2 method), 81
- `__repr__()` (Goulib.geom.Vector2 method), 72
- `__repr__()` (Goulib.geom3d.Line3 method), 100
- `__repr__()` (Goulib.geom3d.Matrix4 method), 103
- `__repr__()` (Goulib.geom3d.Plane method), 103
- `__repr__()` (Goulib.geom3d.Quaternion method), 106
- `__repr__()` (Goulib.geom3d.Segment3 method), 101
- `__repr__()` (Goulib.geom3d.Sphere method), 102
- `__repr__()` (Goulib.geom3d.Vector3 method), 98
- `__repr__()` (Goulib.image.Ditherer method), 155
- `__repr__()` (Goulib.image.ErrorDiffusion method), 157
- `__repr__()` (Goulib.image.FloydSteinberg method), 157
- `__repr__()` (Goulib.image.Image method), 152
- `__repr__()` (Goulib.image.Mode method), 148
- `__repr__()` (Goulib.interval.Interval method), 159
- `__repr__()` (Goulib.interval.Intervals method), 162
- `__repr__()` (Goulib.motion.TimeDiagram method), 212
- `__repr__()` (Goulib.optim.BinDict method), 217
- `__repr__()` (Goulib.optim.BinList method), 219
- `__repr__()` (Goulib.piecewise.Piecewise method), 224
- `__repr__()` (Goulib.polynomial.Polynomial method), 227
- `__repr__()` (Goulib.statemachine.TimeMarker method), 230
- `__repr__()` (Goulib.stats.Discrete method), 234
- `__repr__()` (Goulib.stats.Normal method), 238
- `__repr__()` (Goulib.stats.PDF method), 236
- `__repr__()` (Goulib.stats.Stats method), 231
- `__repr__()` (Goulib.table.Cell method), 240
- `__repr__()` (Goulib.table.Row method), 242
- `__repr__()` (Goulib.table.Table method), 243
- `__repr__()` (Goulib.tests.TestCase method), 248
- `__reset__()` (Goulib.statemachine.StateMachine method), 230
- `__reversed__()` (Goulib.colors.Palette method), 9
- `__reversed__()` (Goulib.container.Record method), 12
- `__reversed__()` (Goulib.container.SortedCollection method), 14
- `__reversed__()` (Goulib.drawing.BBox method), 36
- `__reversed__()` (Goulib.drawing.Chain method), 50
- `__reversed__()` (Goulib.drawing.Drawing method), 59
- `__reversed__()` (Goulib.drawing.Group method), 43
- `__reversed__()` (Goulib.drawing.Rect method), 53
- `__reversed__()` (Goulib.interval.Box method), 165
- `__reversed__()` (Goulib.interval.Interval method), 161
- `__reversed__()` (Goulib.interval.Intervals method), 162
- `__reversed__()` (Goulib.optim.BinList method), 219
- `__reversed__()` (Goulib.table.Table method), 245
- `__rfloordiv__()` (Goulib.datetime2.timedelta2 attribute), 25
- `__rfloordiv__()` (Goulib.geom.Point2 method), 76
- `__rfloordiv__()` (Goulib.geom.Vector2 method), 72
- `__rfloordiv__()` (Goulib.geom3d.Vector3 method), 99
- `__rmod__()` (Goulib.datetime2.timedelta2 attribute), 25
- `__rmul__()` (Goulib.datetime2.timedelta2 attribute), 25
- `__rmul__()` (Goulib.drawing.BBox attribute), 36
- `__rmul__()` (Goulib.drawing.Chain attribute), 50
- `__rmul__()` (Goulib.drawing.Drawing attribute), 59
- `__rmul__()` (Goulib.drawing.Group attribute), 43
- `__rmul__()` (Goulib.drawing.Rect attribute), 53
- `__rmul__()` (Goulib.interval.Box attribute), 165
- `__rmul__()` (Goulib.interval.Interval attribute), 161
- `__rmul__()` (Goulib.optim.BinList attribute), 219
- `__rmul__()` (Goulib.table.Table attribute), 245
- `__rmul__()` (Goulib.expr.Expr method), 63
- `__rmul__()` (Goulib.geom.Point2 method), 76
- `__rmul__()` (Goulib.geom.Vector2 method), 72
- `__rmul__()` (Goulib.geom3d.Vector3 method), 98
- `__rmul__()` (Goulib.piecewise.Piecewise method), 224
- `__rmul__()` (Goulib.polynomial.Polynomial method), 227
- `__rmul__()` (Goulib.stats.Normal method), 238
- `__rmul__()` (Goulib.stats.PDF method), 236
- `__rshift__()` (Goulib.expr.Expr method), 63
- `__rshift__()` (Goulib.piecewise.Piecewise method), 223
- `__rshift__()` (Goulib.polynomial.Polynomial method), 227
- `__rshift__()` (Goulib.stats.Normal method), 238
- `__rshift__()` (Goulib.stats.PDF method), 236
- `__rsub__()` (Goulib.datetime2.date2 attribute), 21
- `__rsub__()` (Goulib.datetime2.datetime2 attribute), 18
- `__rsub__()` (Goulib.datetime2.timedelta2 attribute), 25
- `__rsub__()` (Goulib.geom.Point2 method), 76
- `__rsub__()` (Goulib.geom.Vector2 method), 72
- `__rsub__()` (Goulib.geom3d.Vector3 method), 98
- `__rsub__()` (Goulib.polynomial.Polynomial method), 226
- `__rsub__()` (Goulib.stats.Normal method), 237
- `__rtruediv__()` (Goulib.datetime2.timedelta2 attribute), 25
- `__rtruediv__()` (Goulib.geom.Point2 method), 76
- `__rtruediv__()` (Goulib.geom.Vector2 method), 73
- `__rtruediv__()` (Goulib.geom3d.Vector3 method), 99
- `__setattr__()` (Goulib.colors.Color attribute), 8
- `__setattr__()` (Goulib.colors.Palette attribute), 10
- `__setattr__()` (Goulib.container.Sequence attribute), 17
- `__setattr__()` (Goulib.container.SortedCollection attribute), 15
- `__setattr__()` (Goulib.datetime2.date2 attribute), 21
- `__setattr__()` (Goulib.datetime2.datetime2 attribute), 18

- __setattr__ (Goulib.datetime2.time2 attribute), 23
- __setattr__ (Goulib.datetime2.timedelta2 attribute), 25
- __setattr__ (Goulib.drawing.BBox attribute), 36
- __setattr__ (Goulib.drawing.Chain attribute), 50
- __setattr__ (Goulib.drawing.Drawing attribute), 60
- __setattr__ (Goulib.drawing.Entity attribute), 39
- __setattr__ (Goulib.drawing.Group attribute), 43
- __setattr__ (Goulib.drawing.Instance attribute), 46
- __setattr__ (Goulib.drawing.Rect attribute), 53
- __setattr__ (Goulib.drawing.Spline attribute), 40
- __setattr__ (Goulib.drawing.Text attribute), 57
- __setattr__ (Goulib.expr.Expr attribute), 64
- __setattr__ (Goulib.expr.LatexVisitor attribute), 66
- __setattr__ (Goulib.expr.TextVisitor attribute), 65
- __setattr__ (Goulib.geom.Arc2 attribute), 88
- __setattr__ (Goulib.geom.Circle attribute), 85
- __setattr__ (Goulib.geom.Ellipse attribute), 91
- __setattr__ (Goulib.geom.Geometry attribute), 69
- __setattr__ (Goulib.geom.Line2 attribute), 79
- __setattr__ (Goulib.geom.Matrix3 attribute), 97
- __setattr__ (Goulib.geom.Point2 attribute), 76
- __setattr__ (Goulib.geom.Ray2 attribute), 80
- __setattr__ (Goulib.geom.Segment2 attribute), 82
- __setattr__ (Goulib.geom.Vector2 attribute), 74
- __setattr__ (Goulib.graph.AGraph attribute), 114
- __setattr__ (Goulib.graph.DiGraph attribute), 131
- __setattr__ (Goulib.graph.GeoGraph attribute), 116
- __setattr__ (Goulib.graph.index attribute), 113
- __setattr__ (Goulib.graph.index.Index attribute), 111
- __setattr__ (Goulib.graph.index.Property attribute), 110
- __setattr__ (Goulib.image.Ditherer attribute), 156
- __setattr__ (Goulib.image.ErrorDiffusion attribute), 157
- __setattr__ (Goulib.image.FloydSteinberg attribute), 158
- __setattr__ (Goulib.image.Image attribute), 154
- __setattr__ (Goulib.image.Mode attribute), 149
- __setattr__ (Goulib.interval.Box attribute), 165
- __setattr__ (Goulib.interval.Interval attribute), 161
- __setattr__ (Goulib.interval.Intervals attribute), 163
- __setattr__ (Goulib.itertools2.SortingError attribute), 172
- __setattr__ (Goulib.itertools2.iter2 attribute), 171
- __setattr__ (Goulib.itertools2.keep attribute), 174
- __setattr__ (Goulib.markup.ArgumentError attribute), 186
- __setattr__ (Goulib.markup.ClosingError attribute), 183
- __setattr__ (Goulib.markup.CustomizationError attribute), 190
- __setattr__ (Goulib.markup.DeprecationError attribute), 188
- __setattr__ (Goulib.markup.InvalidElementError attribute), 187
- __setattr__ (Goulib.markup.MarkupError attribute), 182
- __setattr__ (Goulib.markup.ModeError attribute), 189
- __setattr__ (Goulib.markup.OpeningError attribute), 185
- __setattr__ (Goulib.markup.dummy attribute), 180
- __setattr__ (Goulib.markup.element attribute), 176
- __setattr__ (Goulib.markup.page attribute), 179
- __setattr__ (Goulib.markup.russell attribute), 181
- __setattr__ (Goulib.motion.Actuator attribute), 211
- __setattr__ (Goulib.motion.PVA attribute), 204
- __setattr__ (Goulib.motion.Segment attribute), 205
- __setattr__ (Goulib.motion.SegmentPoly attribute), 208
- __setattr__ (Goulib.motion.Segments attribute), 207
- __setattr__ (Goulib.motion.TimeDiagram attribute), 212
- __setattr__ (Goulib.optim.BinDict attribute), 217
- __setattr__ (Goulib.optim.BinList attribute), 219
- __setattr__ (Goulib.optim.DifferentialEvolution attribute), 222
- __setattr__ (Goulib.optim.ObjectiveFunction attribute), 215
- __setattr__ (Goulib.piecewise.Piecewise attribute), 224
- __setattr__ (Goulib.plot.Plot attribute), 226
- __setattr__ (Goulib.polynomial.Polynomial attribute), 228
- __setattr__ (Goulib.stats.Discrete attribute), 234
- __setattr__ (Goulib.stats.Normal attribute), 238
- __setattr__ (Goulib.stats.PDF attribute), 236
- __setattr__ (Goulib.stats.Stats attribute), 233
- __setattr__ (Goulib.table.Cell attribute), 241
- __setattr__ (Goulib.table.Row attribute), 242
- __setattr__ (Goulib.table.Table attribute), 245
- __setattr__ (Goulib.tests.TestCase attribute), 248
- __setattr__ (Goulib.workdays.WorkCalendar attribute), 255
- __setattr__ () (Goulib.container.Record method), 11
- __setitem__ (Goulib.colors.Palette attribute), 10
- __setitem__ (Goulib.container.Record attribute), 12
- __setitem__ (Goulib.drawing.BBox attribute), 36
- __setitem__ (Goulib.drawing.Chain attribute), 50
- __setitem__ (Goulib.drawing.Drawing attribute), 60
- __setitem__ (Goulib.drawing.Group attribute), 43
- __setitem__ (Goulib.drawing.Rect attribute), 53
- __setitem__ (Goulib.graph.index.Index attribute), 111
- __setitem__ (Goulib.interval.Box attribute), 165
- __setitem__ (Goulib.interval.Interval attribute), 161
- __setitem__ (Goulib.table.Table attribute), 245
- __setitem__ () (Goulib.geom.Matrix3 method), 96
- __setitem__ () (Goulib.geom3d.Matrix4 method), 103
- __setitem__ () (Goulib.optim.BinDict method), 216
- __setitem__ () (Goulib.optim.BinList method), 218
- __setstate__ () (Goulib.itertools2.SortingError method), 172
- __setstate__ () (Goulib.markup.ArgumentError method), 186
- __setstate__ () (Goulib.markup.ClosingError method), 183
- __setstate__ () (Goulib.markup.CustomizationError method), 190

- `__setstate__()` (Goulib.markup.DeprecationError method), 188
- `__setstate__()` (Goulib.markup.InvalidElementError method), 187
- `__setstate__()` (Goulib.markup.MarkupError method), 182
- `__setstate__()` (Goulib.markup.ModeError method), 189
- `__setstate__()` (Goulib.markup.OpeningError method), 185
- `__sizeof__()` (Goulib.colors.Color method), 8
- `__sizeof__()` (Goulib.colors.Palette method), 10
- `__sizeof__()` (Goulib.container.Record method), 12
- `__sizeof__()` (Goulib.container.Sequence method), 17
- `__sizeof__()` (Goulib.container.SortedCollection method), 15
- `__sizeof__()` (Goulib.datetime2.date2 method), 21
- `__sizeof__()` (Goulib.datetime2.datetime2 method), 18
- `__sizeof__()` (Goulib.datetime2.time2 method), 23
- `__sizeof__()` (Goulib.datetime2.timedelta2 method), 25
- `__sizeof__()` (Goulib.drawing.BBox method), 36
- `__sizeof__()` (Goulib.drawing.Chain method), 50
- `__sizeof__()` (Goulib.drawing.Drawing method), 60
- `__sizeof__()` (Goulib.drawing.Entity method), 39
- `__sizeof__()` (Goulib.drawing.Group method), 44
- `__sizeof__()` (Goulib.drawing.Instance method), 47
- `__sizeof__()` (Goulib.drawing.Rect method), 53
- `__sizeof__()` (Goulib.drawing.Spline method), 40
- `__sizeof__()` (Goulib.drawing.Text method), 57
- `__sizeof__()` (Goulib.expr.Expr method), 64
- `__sizeof__()` (Goulib.expr.LatexVisitor method), 66
- `__sizeof__()` (Goulib.expr.TextVisitor method), 65
- `__sizeof__()` (Goulib.geom.Arc2 method), 88
- `__sizeof__()` (Goulib.geom.Circle method), 85
- `__sizeof__()` (Goulib.geom.Ellipse method), 91
- `__sizeof__()` (Goulib.geom.Geometry method), 69
- `__sizeof__()` (Goulib.geom.Line2 method), 79
- `__sizeof__()` (Goulib.geom.Matrix3 method), 97
- `__sizeof__()` (Goulib.geom.Point2 method), 76
- `__sizeof__()` (Goulib.geom.Ray2 method), 80
- `__sizeof__()` (Goulib.geom.Segment2 method), 82
- `__sizeof__()` (Goulib.geom.Vector2 method), 74
- `__sizeof__()` (Goulib.graph.AGraph method), 114
- `__sizeof__()` (Goulib.graph.DiGraph method), 131
- `__sizeof__()` (Goulib.graph.Graph method), 116
- `__sizeof__()` (Goulib.graph.index method), 113
- `__sizeof__()` (Goulib.graph.index.Index method), 111
- `__sizeof__()` (Goulib.graph.index.Property method), 110
- `__sizeof__()` (Goulib.image.Ditherer method), 156
- `__sizeof__()` (Goulib.image.ErrorDiffusion method), 157
- `__sizeof__()` (Goulib.image.FloydSteinberg method), 158
- `__sizeof__()` (Goulib.image.Image method), 154
- `__sizeof__()` (Goulib.image.Mode method), 149
- `__sizeof__()` (Goulib.interval.Box method), 165
- `__sizeof__()` (Goulib.interval.Interval method), 161
- `__sizeof__()` (Goulib.interval.Intervals method), 163
- `__sizeof__()` (Goulib.itertools2.SortingError method), 172
- `__sizeof__()` (Goulib.itertools2.iter2 method), 171
- `__sizeof__()` (Goulib.itertools2.keep method), 174
- `__sizeof__()` (Goulib.markup.ArgumentError method), 186
- `__sizeof__()` (Goulib.markup.ClosingError method), 183
- `__sizeof__()` (Goulib.markup.CustomizationError method), 190
- `__sizeof__()` (Goulib.markup.DeprecationError method), 188
- `__sizeof__()` (Goulib.markup.InvalidElementError method), 187
- `__sizeof__()` (Goulib.markup.MarkupError method), 182
- `__sizeof__()` (Goulib.markup.ModeError method), 189
- `__sizeof__()` (Goulib.markup.OpeningError method), 185
- `__sizeof__()` (Goulib.markup.dummy method), 180
- `__sizeof__()` (Goulib.markup.element method), 176
- `__sizeof__()` (Goulib.markup.page method), 179
- `__sizeof__()` (Goulib.markup.russell method), 181
- `__sizeof__()` (Goulib.motion.Actuator method), 211
- `__sizeof__()` (Goulib.motion.PVA method), 204
- `__sizeof__()` (Goulib.motion.Segment method), 205
- `__sizeof__()` (Goulib.motion.SegmentPoly method), 208
- `__sizeof__()` (Goulib.motion.Segments method), 207
- `__sizeof__()` (Goulib.motion.TimeDiagram method), 212
- `__sizeof__()` (Goulib.optim.BinDict method), 217
- `__sizeof__()` (Goulib.optim.BinList method), 219
- `__sizeof__()` (Goulib.optim.DifferentialEvolution method), 222
- `__sizeof__()` (Goulib.optim.ObjectiveFunction method), 215
- `__sizeof__()` (Goulib.piecewise.Piecewise method), 224
- `__sizeof__()` (Goulib.plot.Plot method), 226
- `__sizeof__()` (Goulib.polynomial.Polynomial method), 228
- `__sizeof__()` (Goulib.stats.Discrete method), 234
- `__sizeof__()` (Goulib.stats.Normal method), 238
- `__sizeof__()` (Goulib.stats.PDF method), 236
- `__sizeof__()` (Goulib.stats.Stats method), 233
- `__sizeof__()` (Goulib.table.Cell method), 241
- `__sizeof__()` (Goulib.table.Row method), 242
- `__sizeof__()` (Goulib.table.Table method), 245
- `__sizeof__()` (Goulib.tests.TestCase method), 248
- `__sizeof__()` (Goulib.workdays.WorkCalendar method), 255
- `__slotnames__` (Goulib.geom.Matrix3 attribute), 97
- `__slots__` (Goulib.itertools2.keep attribute), 174
- `__str__` (Goulib.colors.Color attribute), 8
- `__str__` (Goulib.colors.Palette attribute), 10
- `__str__` (Goulib.container.Sequence attribute), 17
- `__str__` (Goulib.container.SortedCollection attribute), 15
- `__str__` (Goulib.datetime2.date2 attribute), 21

- `__str__` (Goulib.datetime2.datetime2 attribute), 18
- `__str__` (Goulib.datetime2.time2 attribute), 23
- `__str__` (Goulib.datetime2.timedelta2 attribute), 25
- `__str__` (Goulib.drawing.BBox attribute), 36
- `__str__` (Goulib.drawing.Chain attribute), 50
- `__str__` (Goulib.drawing.Drawing attribute), 60
- `__str__` (Goulib.drawing.Entity attribute), 39
- `__str__` (Goulib.drawing.Group attribute), 44
- `__str__` (Goulib.drawing.Instance attribute), 47
- `__str__` (Goulib.drawing.Rect attribute), 53
- `__str__` (Goulib.drawing.Spline attribute), 40
- `__str__` (Goulib.drawing.Text attribute), 57
- `__str__` (Goulib.expr.LatexVisitor attribute), 67
- `__str__` (Goulib.expr.TextVisitor attribute), 65
- `__str__` (Goulib.geom.Arc2 attribute), 88
- `__str__` (Goulib.geom.Circle attribute), 85
- `__str__` (Goulib.geom.Ellipse attribute), 91
- `__str__` (Goulib.geom.Geometry attribute), 69
- `__str__` (Goulib.geom.Line2 attribute), 79
- `__str__` (Goulib.geom.Matrix3 attribute), 97
- `__str__` (Goulib.geom.Point2 attribute), 76
- `__str__` (Goulib.geom.Ray2 attribute), 81
- `__str__` (Goulib.geom.Segment2 attribute), 82
- `__str__` (Goulib.geom.Vector2 attribute), 74
- `__str__` (Goulib.graph.AGraph attribute), 114
- `__str__` (Goulib.graph.index attribute), 113
- `__str__` (Goulib.graph.index.Index attribute), 111
- `__str__` (Goulib.graph.index.Property attribute), 110
- `__str__` (Goulib.image.Ditherer attribute), 156
- `__str__` (Goulib.image.ErrorDiffusion attribute), 157
- `__str__` (Goulib.image.FloydSteinberg attribute), 158
- `__str__` (Goulib.image.Image attribute), 154
- `__str__` (Goulib.image.Mode attribute), 149
- `__str__` (Goulib.interval.Box attribute), 165
- `__str__` (Goulib.interval.Intervals attribute), 163
- `__str__` (Goulib.itertools2.SortingError attribute), 172
- `__str__` (Goulib.itertools2.iter2 attribute), 171
- `__str__` (Goulib.itertools2.keep attribute), 174
- `__str__` (Goulib.markup.dummy attribute), 180
- `__str__` (Goulib.markup.element attribute), 176
- `__str__` (Goulib.markup.russell attribute), 181
- `__str__` (Goulib.motion.Actuator attribute), 211
- `__str__` (Goulib.motion.PVA attribute), 204
- `__str__` (Goulib.motion.Segment attribute), 205
- `__str__` (Goulib.motion.SegmentPoly attribute), 208
- `__str__` (Goulib.motion.TimeDiagram attribute), 212
- `__str__` (Goulib.optim.BinDict attribute), 217
- `__str__` (Goulib.optim.BinList attribute), 219
- `__str__` (Goulib.optim.DifferentialEvolution attribute), 222
- `__str__` (Goulib.optim.ObjectiveFunction attribute), 215
- `__str__` (Goulib.plot.Plot attribute), 226
- `__str__` (Goulib.stats.Discrete attribute), 234
- `__str__` (Goulib.stats.Stats attribute), 233
- `__str__` (Goulib.table.Cell attribute), 241
- `__str__` (Goulib.table.Row attribute), 242
- `__str__` (Goulib.workdays.WorkCalendar attribute), 255
- `__str__()` (Goulib.container.Record method), 11
- `__str__()` (Goulib.expr.Expr method), 63
- `__str__()` (Goulib.graph.DiGraph method), 131
- `__str__()` (Goulib.graph.GeoGraph method), 116
- `__str__()` (Goulib.interval.Interval method), 159
- `__str__()` (Goulib.markup.ArgumentError method), 186
- `__str__()` (Goulib.markup.ClosingError method), 183
- `__str__()` (Goulib.markup.CustomizationError method), 190
- `__str__()` (Goulib.markup.DeprecationError method), 188
- `__str__()` (Goulib.markup.InvalidElementError method), 187
- `__str__()` (Goulib.markup.MarkupError method), 181
- `__str__()` (Goulib.markup.ModeError method), 189
- `__str__()` (Goulib.markup.OpeningError method), 185
- `__str__()` (Goulib.markup.page method), 177
- `__str__()` (Goulib.motion.Segments method), 206
- `__str__()` (Goulib.piecewise.Piecewise method), 223
- `__str__()` (Goulib.polynomial.Polynomial method), 228
- `__str__()` (Goulib.stats.Normal method), 237
- `__str__()` (Goulib.stats.PDF method), 236
- `__str__()` (Goulib.table.Table method), 243
- `__str__()` (Goulib.tests.TestCase method), 248
- `__sub__` (Goulib.datetime2.date2 attribute), 21
- `__sub__` (Goulib.datetime2.timedelta2 attribute), 25
- `__sub__()` (Goulib.colors.Color method), 7
- `__sub__()` (Goulib.container.Sequence method), 16
- `__sub__()` (Goulib.datetime2.datetime2 method), 17
- `__sub__()` (Goulib.expr.Expr method), 63
- `__sub__()` (Goulib.geom.Matrix3 method), 96
- `__sub__()` (Goulib.geom.Point2 method), 76
- `__sub__()` (Goulib.geom.Vector2 method), 72
- `__sub__()` (Goulib.geom3d.Vector3 method), 98
- `__sub__()` (Goulib.image.Image method), 153
- `__sub__()` (Goulib.piecewise.Piecewise method), 224
- `__sub__()` (Goulib.polynomial.Polynomial method), 226
- `__sub__()` (Goulib.stats.Discrete method), 234
- `__sub__()` (Goulib.stats.Normal method), 237
- `__sub__()` (Goulib.stats.PDF method), 236
- `__sub__()` (Goulib.stats.Stats method), 232
- `__suppress_context__` (Goulib.itertools2.SortingError attribute), 172
- `__suppress_context__` (Goulib.markup.ArgumentError attribute), 186
- `__suppress_context__` (Goulib.markup.ClosingError attribute), 184
- `__suppress_context__` (Goulib.markup.CustomizationError attribute), 191
- `__suppress_context__` (Goulib.markup.DeprecationError attribute), 188

- `__suppress_context__` (Goulib.markup.InvalidElementError attribute), 187
 - `__suppress_context__` (Goulib.markup.MarkupError attribute), 182
 - `__suppress_context__` (Goulib.markup.ModeError attribute), 189
 - `__suppress_context__` (Goulib.markup.OpeningError attribute), 185
 - `__traceback__` (Goulib.itertools2.SortingError attribute), 172
 - `__traceback__` (Goulib.markup.ArgumentError attribute), 186
 - `__traceback__` (Goulib.markup.ClosingError attribute), 184
 - `__traceback__` (Goulib.markup.CustomizationError attribute), 191
 - `__traceback__` (Goulib.markup.DeprecationError attribute), 188
 - `__traceback__` (Goulib.markup.InvalidElementError attribute), 187
 - `__traceback__` (Goulib.markup.MarkupError attribute), 182
 - `__traceback__` (Goulib.markup.ModeError attribute), 189
 - `__traceback__` (Goulib.markup.OpeningError attribute), 185
 - `__truediv__` (Goulib.datetime2.timedelta2 attribute), 25
 - `__truediv__()` (Goulib.expr.Expr method), 63
 - `__truediv__()` (Goulib.geom.Point2 method), 76
 - `__truediv__()` (Goulib.geom.Vector2 method), 72
 - `__truediv__()` (Goulib.geom3d.Vector3 method), 99
 - `__truediv__()` (Goulib.image.Image method), 153
 - `__truediv__()` (Goulib.piecewise.Piecewise method), 224
 - `__truediv__()` (Goulib.polynomial.Polynomial method), 228
 - `__truediv__()` (Goulib.stats.Normal method), 237
 - `__truediv__()` (Goulib.stats.PDF method), 236
 - `__xor__()` (Goulib.expr.Expr method), 63
 - `__xor__()` (Goulib.piecewise.Piecewise method), 224
 - `__xor__()` (Goulib.polynomial.Polynomial method), 228
 - `__xor__()` (Goulib.stats.Normal method), 238
 - `__xor__()` (Goulib.stats.PDF method), 236
- ## A
- `abundance()` (in module Goulib.math2), 200
 - `accsum()` (in module Goulib.math2), 192
 - `accumulate()` (Goulib.container.Sequence method), 17
 - `accumulate()` (in module Goulib.itertools2), 168
 - `aci_to_color()` (in module Goulib.colors), 10
 - `Actuator` (class in Goulib.motion), 209
 - `adapt_rgb()` (in module Goulib.image), 149
 - `add()` (Goulib.image.Image method), 153
 - `add()` (Goulib.markup.page method), 177
 - `add()` (Goulib.motion.Segments method), 206
 - `add()` (in module Goulib.polynomial), 228
 - `add_cycle()` (Goulib.graph.DiGraph method), 131
 - `add_cycle()` (Goulib.graph.GeoGraph method), 116
 - `add_edge()` (Goulib.graph.DiGraph method), 131
 - `add_edge()` (Goulib.graph.GeoGraph method), 116
 - `add_edges_from()` (Goulib.graph.DiGraph method), 131
 - `add_edges_from()` (Goulib.graph.GeoGraph method), 116
 - `add_months()` (in module Goulib.datetime2), 26
 - `add_node()` (Goulib.graph.DiGraph method), 132
 - `add_node()` (Goulib.graph.GeoGraph method), 117
 - `add_nodes_from()` (Goulib.graph.DiGraph method), 132
 - `add_nodes_from()` (Goulib.graph.GeoGraph method), 117
 - `add_path()` (Goulib.graph.DiGraph method), 132
 - `add_path()` (Goulib.graph.GeoGraph method), 117
 - `add_star()` (Goulib.graph.DiGraph method), 132
 - `add_star()` (Goulib.graph.GeoGraph method), 117
 - `add_weighted_edges_from()` (Goulib.graph.DiGraph method), 132
 - `add_weighted_edges_from()` (Goulib.graph.GeoGraph method), 117
 - `addCleanup()` (Goulib.tests.TestCase method), 248
 - `addcol()` (Goulib.table.Table method), 246
 - `addcontent()` (Goulib.markup.page method), 177
 - `addfooter()` (Goulib.markup.page method), 177
 - `addheader()` (Goulib.markup.page method), 177
 - `addholidays()` (Goulib.workdays.WorkCalendar method), 253
 - `addTypeEqualityFunc()` (Goulib.tests.TestCase method), 248
 - `adjacency_iter()` (Goulib.graph.DiGraph method), 133
 - `adjacency_iter()` (Goulib.graph.GeoGraph method), 118
 - `adjacency_list()` (Goulib.graph.DiGraph method), 133
 - `adjacency_list()` (Goulib.graph.GeoGraph method), 118
 - `adjlist_dict_factory` (Goulib.graph.DiGraph attribute), 133
 - `adjlist_dict_factory` (Goulib.graph.GeoGraph attribute), 118
 - `AGraph` (class in Goulib.graph), 107, 113
 - `all()` (in module Goulib.itertools2), 169
 - `all_pairs()` (in module Goulib.itertools2), 169
 - `alpha_composite()` (in module Goulib.image), 154
 - `alpha_composite_with_color()` (in module Goulib.image), 154
 - `angle()` (Goulib.geom.Arc2 method), 87
 - `angle()` (Goulib.geom.Matrix3 method), 97
 - `angle()` (Goulib.geom.Point2 method), 76
 - `angle()` (Goulib.geom.Vector2 method), 73
 - `angle()` (Goulib.geom3d.Vector3 method), 99
 - `angle()` (in module Goulib.math2), 201
 - `anneal()` (in module Goulib.optim), 220
 - `any()` (in module Goulib.itertools2), 168
 - `append()` (Goulib.drawing.BBox method), 36
 - `append()` (Goulib.drawing.Chain method), 31, 48

- append() (Goulib.drawing.Drawing method), 60
 - append() (Goulib.drawing.Group method), 30, 42
 - append() (Goulib.drawing.Rect method), 53
 - append() (Goulib.interval.Box method), 165
 - append() (Goulib.interval.Interval method), 161
 - append() (Goulib.itertools2.iter2 method), 170
 - append() (Goulib.optim.BinList method), 218
 - append() (Goulib.piecewise.Piecewise method), 223
 - append() (Goulib.stats.Discrete method), 234
 - append() (Goulib.stats.Normal method), 238
 - append() (Goulib.stats.PDF method), 236
 - append() (Goulib.stats.Stats method), 231
 - append() (Goulib.table.Table method), 245
 - applx() (Goulib.expr.Expr method), 63
 - applx() (Goulib.piecewise.Piecewise method), 223
 - applx() (Goulib.polynomial.Polynomial method), 228
 - applx() (Goulib.stats.Normal method), 239
 - applx() (Goulib.stats.PDF method), 236
 - apply() (Goulib.container.Sequence method), 16
 - apply() (Goulib.expr.Expr method), 63
 - apply() (Goulib.piecewise.Piecewise method), 223
 - apply() (Goulib.polynomial.Polynomial method), 228
 - apply() (Goulib.stats.Normal method), 239
 - apply() (Goulib.stats.PDF method), 236
 - applyf() (Goulib.table.Table method), 246
 - arange() (in module Goulib.itertools2), 166
 - Arc2 (class in Goulib.geom), 87
 - arc_from_3_points() (in module Goulib.geom), 87
 - area (Goulib.drawing.BBox attribute), 28, 34
 - argPair() (in module Goulib.geom), 69
 - args (Goulib.itertools2.SortingError attribute), 172
 - args (Goulib.markup.ArgumentError attribute), 186
 - args (Goulib.markup.ClosingError attribute), 184
 - args (Goulib.markup.CustomizationError attribute), 191
 - args (Goulib.markup.DeprecationError attribute), 188
 - args (Goulib.markup.InvalidElementError attribute), 187
 - args (Goulib.markup.MarkupError attribute), 182
 - args (Goulib.markup.ModeError attribute), 189
 - args (Goulib.markup.OpeningError attribute), 185
 - ArgumentError, 185
 - asdict() (Goulib.table.Table method), 246
 - assert_() (Goulib.tests.TestCase method), 252
 - assertAlmostEqual() (Goulib.tests.TestCase method), 248
 - assertAlmostEquals() (Goulib.tests.TestCase method), 249
 - assertCountEqual() (Goulib.tests.TestCase method), 247
 - assertDictContainsSubset() (Goulib.tests.TestCase method), 249
 - assertDictEqual() (Goulib.tests.TestCase method), 249
 - assertEqual() (Goulib.tests.TestCase method), 247
 - assertEquals() (Goulib.tests.TestCase method), 249
 - assertFalse() (Goulib.tests.TestCase method), 249
 - assertGreater() (Goulib.tests.TestCase method), 249
 - assertGreaterEqual() (Goulib.tests.TestCase method), 249
 - assertIn() (Goulib.tests.TestCase method), 249
 - assertIs() (Goulib.tests.TestCase method), 249
 - assertIsInstance() (Goulib.tests.TestCase method), 249
 - assertIsNone() (Goulib.tests.TestCase method), 249
 - assertIsNot() (Goulib.tests.TestCase method), 249
 - assertIsNotNone() (Goulib.tests.TestCase method), 249
 - assertLess() (Goulib.tests.TestCase method), 249
 - assertLessEqual() (Goulib.tests.TestCase method), 249
 - assertListEqual() (Goulib.tests.TestCase method), 249
 - assertLogs() (Goulib.tests.TestCase method), 249
 - assertMultiLineEqual() (Goulib.tests.TestCase method), 250
 - assertNotAlmostEqual() (Goulib.tests.TestCase method), 250
 - assertNotAlmostEquals() (Goulib.tests.TestCase method), 250
 - assertNotEqual() (Goulib.tests.TestCase method), 250
 - assertNotEquals() (Goulib.tests.TestCase method), 250
 - assertNotIn() (Goulib.tests.TestCase method), 250
 - assertNotIsInstance() (Goulib.tests.TestCase method), 250
 - assertNotRegex() (Goulib.tests.TestCase method), 250
 - assertNotRegexMatches() (Goulib.tests.TestCase method), 250
 - assertRaises() (Goulib.tests.TestCase method), 250
 - assertRaisesRegex() (Goulib.tests.TestCase method), 250
 - assertRaisesRegex() (Goulib.tests.TestCase method), 251
 - assertRegex() (Goulib.tests.TestCase method), 251
 - assertRegexMatches() (Goulib.tests.TestCase method), 251
 - assertSequenceEqual() (Goulib.tests.TestCase method), 247
 - assertSetEqual() (Goulib.tests.TestCase method), 251
 - assertTrue() (Goulib.tests.TestCase method), 251
 - assertTupleEqual() (Goulib.tests.TestCase method), 251
 - assertWarns() (Goulib.tests.TestCase method), 251
 - assertWarnsRegex() (Goulib.tests.TestCase method), 251
 - astimezone() (Goulib.datetime2.datetime2 method), 18
 - attr() (in module Goulib.table), 240
 - average (Goulib.stats.Discrete attribute), 234
 - average (Goulib.stats.Normal attribute), 239
 - average (Goulib.stats.PDF attribute), 236
 - average (Goulib.stats.Stats attribute), 232
 - average_hash() (Goulib.image.Image method), 152
 - avg (Goulib.stats.Discrete attribute), 234
 - avg (Goulib.stats.Normal attribute), 239
 - avg (Goulib.stats.PDF attribute), 236
 - avg (Goulib.stats.Stats attribute), 232
 - avg() (in module Goulib.stats), 231
- ## B
- baby_step_giant_step() (in module Goulib.math2), 202
 - base_types (Goulib.tests.TestCase attribute), 247

- BBox (class in Goulib.drawing), 27, 34
bbox() (Goulib.drawing.Chain method), 50
bbox() (Goulib.drawing.Drawing method), 60
bbox() (Goulib.drawing.Entity method), 28, 37
bbox() (Goulib.drawing.Group method), 44
bbox() (Goulib.drawing.Instance method), 47
bbox() (Goulib.drawing.Rect method), 54
bbox() (Goulib.drawing.Spline method), 30, 39
bbox() (Goulib.drawing.Text method), 32, 56
bbox() (Goulib.geom.Arc2 method), 88
bbox() (Goulib.geom.Circle method), 85
bbox() (Goulib.geom.Ellipse method), 91
bbox() (Goulib.geom.Point2 method), 76
bbox() (Goulib.geom.Segment2 method), 82
bernouilli() (in module Goulib.math2), 199
bernouilli_gen() (in module Goulib.math2), 199
best() (in module Goulib.itertools2), 169
bigomega() (in module Goulib.math2), 196
BinDict (class in Goulib.optim), 216
BinList (class in Goulib.optim), 218
binomial() (in module Goulib.math2), 201
binomial_exponent() (in module Goulib.math2), 201
bisect() (Goulib.geom.Segment2 method), 81
bool2gray() (in module Goulib.image), 158
bool2rgb() (in module Goulib.image), 158
bouncy() (in module Goulib.math2), 198
Box (class in Goulib.interval), 163
box() (Goulib.graph.DiGraph method), 133
box() (Goulib.graph.GeoGraph method), 118
box_size() (Goulib.graph.DiGraph method), 133
box_size() (Goulib.graph.GeoGraph method), 118
- ## C
- carries() (in module Goulib.math2), 197
cartesian_product() (in module Goulib.itertools2), 168
cast() (Goulib.workdays.WorkCalendar method), 254
catalan() (in module Goulib.math2), 195
catalan_gen() (in module Goulib.math2), 195
ceildiv() (in module Goulib.math2), 192
Cell (class in Goulib.table), 240
center (Goulib.drawing.Chain attribute), 50
center (Goulib.drawing.Drawing attribute), 60
center (Goulib.drawing.Entity attribute), 28, 37
center (Goulib.drawing.Group attribute), 44
center (Goulib.drawing.Instance attribute), 47
center (Goulib.drawing.Rect attribute), 54
center (Goulib.drawing.Spline attribute), 40
center (Goulib.drawing.Text attribute), 57
center (Goulib.geom.Arc2 attribute), 88
center (Goulib.geom.Circle attribute), 85
center (Goulib.geom.Ellipse attribute), 91
center (Goulib.geom.Point2 attribute), 76
center (Goulib.geom.Segment2 attribute), 82
center (Goulib.interval.Box attribute), 163
center (Goulib.interval.Interval attribute), 159
center() (Goulib.drawing.BBox method), 28, 35
cgiprint() (in module Goulib.markup), 175
Chain (class in Goulib.drawing), 31, 48
chainify() (Goulib.drawing.Chain method), 50
chainify() (Goulib.drawing.Drawing method), 60
chainify() (Goulib.drawing.Group method), 30, 42
chainify() (Goulib.drawing.Rect method), 54
chains() (in module Goulib.drawing), 32, 52
chakravala() (in module Goulib.math2), 200
checkOnTimeAndWait() (Goulib.statemachine.StateMachine method), 230
chinese_remainder() (in module Goulib.math2), 202
Circle (class in Goulib.geom), 83
circle_from_3_points() (in module Goulib.geom), 86
clear() (Goulib.colors.Palette method), 10
clear() (Goulib.container.Record method), 12
clear() (Goulib.container.SortedCollection method), 14
clear() (Goulib.drawing.BBox method), 36
clear() (Goulib.drawing.Chain method), 50
clear() (Goulib.drawing.Drawing method), 60
clear() (Goulib.drawing.Group method), 44
clear() (Goulib.drawing.Rect method), 54
clear() (Goulib.graph.DiGraph method), 133
clear() (Goulib.graph.GeoGraph method), 118
clear() (Goulib.graph.index.Index method), 111
clear() (Goulib.interval.Box method), 165
clear() (Goulib.interval.Interval method), 161
clear() (Goulib.interval.Intervals method), 163
clear() (Goulib.optim.BinDict method), 217
clear() (Goulib.optim.BinList method), 219
clear() (Goulib.table.Table method), 245
close() (Goulib.markup.element method), 175
closest_edges() (Goulib.graph.DiGraph method), 133
closest_edges() (Goulib.graph.GeoGraph method), 118
closest_nodes() (Goulib.graph.DiGraph method), 133
closest_nodes() (Goulib.graph.GeoGraph method), 118
ClosingError, 182
cmp() (in module Goulib.math2), 191
cmyk (Goulib.colors.Color attribute), 7
cmyk2rgb() (in module Goulib.colors), 6
cmyk2rgb() (in module Goulib.image), 158
col() (Goulib.table.Table method), 244
Color (class in Goulib.colors), 6
color (Goulib.drawing.Chain attribute), 50
color (Goulib.drawing.Drawing attribute), 60
color (Goulib.drawing.Entity attribute), 28, 37
color (Goulib.drawing.Group attribute), 30, 42
color (Goulib.drawing.Instance attribute), 47
color (Goulib.drawing.Rect attribute), 54
color (Goulib.drawing.Spline attribute), 40
color (Goulib.drawing.Text attribute), 57
color (Goulib.geom.Arc2 attribute), 88
color (Goulib.geom.Circle attribute), 85

- color (Goulib.geom.Ellipse attribute), 91
 - color (Goulib.geom.Point2 attribute), 76
 - color (Goulib.geom.Segment2 attribute), 82
 - color_range() (in module Goulib.colors), 10
 - color_to_aci() (in module Goulib.colors), 10
 - colorize() (Goulib.image.Image method), 152
 - ColorTable() (in module Goulib.colors), 10
 - cols() (Goulib.table.Table method), 244
 - combinations_with_replacement() (in module Goulib.itertools2), 168
 - combine() (Goulib.datetime2.datetime2 method), 18
 - compact() (in module Goulib.itertools2), 167
 - compose() (Goulib.colors.Color method), 7
 - compose() (Goulib.image.Image method), 153
 - compose() (in module Goulib.itertools2), 168
 - compress() (in module Goulib.itertools2), 167
 - confidence_interval() (in module Goulib.stats), 231
 - conjugated() (Goulib.geom3d.Quaternion method), 106
 - connect() (Goulib.drawing.Chain method), 50
 - connect() (Goulib.drawing.Drawing method), 60
 - connect() (Goulib.drawing.Group method), 44
 - connect() (Goulib.drawing.Instance method), 47
 - connect() (Goulib.drawing.Rect method), 54
 - connect() (Goulib.drawing.Spline method), 40
 - connect() (Goulib.geom.Arc2 method), 88
 - connect() (Goulib.geom.Circle method), 84
 - connect() (Goulib.geom.Ellipse method), 91
 - connect() (Goulib.geom.Geometry method), 68
 - connect() (Goulib.geom.Line2 method), 79
 - connect() (Goulib.geom.Point2 method), 74
 - connect() (Goulib.geom.Ray2 method), 81
 - connect() (Goulib.geom.Segment2 method), 82
 - connect() (Goulib.geom3d.Line3 method), 101
 - connect() (Goulib.geom3d.Plane method), 103
 - connect() (Goulib.geom3d.Point3 method), 100
 - connect() (Goulib.geom3d.Sphere method), 102
 - contiguity() (Goulib.graph.DiGraph method), 133
 - contiguity() (Goulib.graph.GeoGraph method), 118
 - contiguous() (Goulib.drawing.Chain method), 31, 48
 - contiguous() (Goulib.drawing.Rect method), 54
 - convert() (Goulib.colors.Color method), 7
 - convert() (Goulib.image.Image method), 152
 - convert() (in module Goulib.colors), 6
 - convert() (in module Goulib.image), 158
 - converter() (in module Goulib.colors), 6
 - coprime() (in module Goulib.math2), 191
 - coprimes_gen() (in module Goulib.math2), 191
 - copy() (Goulib.colors.Palette method), 10
 - copy() (Goulib.container.Record method), 12
 - copy() (Goulib.container.SortedCollection method), 14
 - copy() (Goulib.drawing.BBox method), 36
 - copy() (Goulib.drawing.Chain method), 50
 - copy() (Goulib.drawing.Drawing method), 60
 - copy() (Goulib.drawing.Group method), 44
 - copy() (Goulib.drawing.Rect method), 54
 - copy() (Goulib.drawing.Chain method), 51
 - copy() (Goulib.drawing.Drawing method), 60
 - copy() (Goulib.drawing.Group method), 44
 - copy() (Goulib.drawing.Rect method), 54
 - copy() (Goulib.graph.index.Index method), 107, 110
 - copy() (Goulib.interval.Box method), 165
 - copy() (Goulib.interval.Interval method), 161
 - copy() (Goulib.interval.Intervals method), 163
 - copy() (Goulib.optim.BinDict method), 217
 - copy() (Goulib.optim.BinList method), 219
 - copy() (Goulib.table.Table method), 245
 - corner() (Goulib.drawing.BBox method), 36
 - corner() (Goulib.interval.Box method), 163
 - corr() (Goulib.stats.Normal method), 237
 - correlation() (Goulib.image.Image method), 152
 - correlation() (Goulib.stats.Normal method), 237
 - count() (Goulib.container.SortedCollection method), 14
 - count() (Goulib.drawing.BBox method), 36
 - count() (Goulib.drawing.Chain method), 51
 - count() (Goulib.drawing.Drawing method), 60
 - count() (Goulib.drawing.Group method), 44
 - count() (Goulib.drawing.Rect method), 54
 - count() (Goulib.graph.index.Index method), 107, 110
 - count() (Goulib.interval.Box method), 165
 - count() (Goulib.interval.Interval method), 161
 - count() (Goulib.interval.Intervals method), 163
 - count() (Goulib.optim.BinList method), 219
 - count() (Goulib.table.Table method), 245
 - count_unique() (in module Goulib.itertools2), 168
 - countTestCases() (Goulib.tests.TestCase method), 252
 - cousin_primes() (in module Goulib.math2), 197
 - cov() (Goulib.stats.Normal method), 237
 - covariance() (Goulib.stats.Discrete method), 234
 - covariance() (Goulib.stats.Normal method), 237
 - covariance() (Goulib.stats.PDF method), 236
 - covariance() (Goulib.stats.Stats method), 232
 - covariance() (in module Goulib.stats), 231
 - crop() (Goulib.image.Image method), 151
 - cross() (Goulib.geom.Point2 method), 76
 - cross() (Goulib.geom.Vector2 method), 73
 - cross() (Goulib.geom3d.Vector3 method), 99
 - css() (Goulib.markup.page method), 178
 - ctime() (Goulib.datetime2.date2 method), 21
 - ctime() (Goulib.datetime2.datetime2 method), 18
 - cumsum() (in module Goulib.math2), 192
 - CustomizationError, 190
- ## D
- date() (Goulib.datetime2.datetime2 method), 18
 - date2 (class in Goulib.datetime2), 20
 - date_add() (in module Goulib.datetime2), 26
 - datef() (in module Goulib.datetime2), 26
 - datetime2 (class in Goulib.datetime2), 17
 - datetime_intersect() (in module Goulib.datetime2), 26
 - datetimef() (in module Goulib.datetime2), 25

- day (Goulib.datetime2.date2 attribute), 21
 day (Goulib.datetime2.datetime2 attribute), 18
 days (Goulib.datetime2.timedelta2 attribute), 25
 days() (in module Goulib.datetime2), 26
 daysgen() (in module Goulib.datetime2), 26
 de_bruijn() (in module Goulib.math2), 202
 debug() (Goulib.tests.TestCase method), 252
 debug() (in module Goulib.decorators), 27
 defaultTestResult() (Goulib.tests.TestCase method), 252
 degree() (Goulib.graph.DiGraph method), 134
 degree() (Goulib.graph.GeoGraph method), 118
 degree_iter() (Goulib.graph.DiGraph method), 134
 degree_iter() (Goulib.graph.GeoGraph method), 119
 delauney_triangulation() (in module Goulib.graph), 108, 148
 delete() (Goulib.graph.index.Index method), 107, 110
 deltaE() (Goulib.colors.Color method), 7
 deltaE() (Goulib.image.Image method), 153
 deltaE() (in module Goulib.colors), 10
 DeprecationError, 187
 derivative() (Goulib.polynomial.Polynomial method), 227
 derivative() (in module Goulib.polynomial), 228
 detect_cycle() (in module Goulib.itertools2), 174
 determinant() (Goulib.geom.Matrix3 method), 98
 determinant() (Goulib.geom3d.Matrix4 method), 104
 diag() (in module Goulib.math2), 193
 dictsplit() (in module Goulib.itertools2), 170
 diff() (Goulib.workdays.WorkCalendar method), 254
 diff() (in module Goulib.itertools2), 173
 DifferentialEvolution (class in Goulib.optim), 220
 digits() (in module Goulib.math2), 197
 digits_gen() (in module Goulib.math2), 197
 DiGraph (class in Goulib.graph), 108, 129
 digsum() (in module Goulib.math2), 197
 Discrete (class in Goulib.stats), 233
 disk() (in module Goulib.image), 154
 display() (Goulib.motion.Actuator method), 210
 display() (Goulib.statemachine.StateMachine method), 231
 displayGraph() (Goulib.statemachine.StateMachine method), 230
 displayLast() (Goulib.motion.Actuator method), 210
 dist() (Goulib.graph.DiGraph method), 134
 dist() (Goulib.graph.GeoGraph method), 119
 dist() (Goulib.image.Image method), 152
 dist() (in module Goulib.math2), 194
 distance() (Goulib.drawing.Chain method), 51
 distance() (Goulib.drawing.Drawing method), 60
 distance() (Goulib.drawing.Group method), 44
 distance() (Goulib.drawing.Instance method), 47
 distance() (Goulib.drawing.Rect method), 54
 distance() (Goulib.drawing.Spline method), 40
 distance() (Goulib.geom.Arc2 method), 88
 distance() (Goulib.geom.Circle method), 85
 distance() (Goulib.geom.Ellipse method), 91
 distance() (Goulib.geom.Geometry method), 68
 distance() (Goulib.geom.Line2 method), 80
 distance() (Goulib.geom.Point2 method), 74
 distance() (Goulib.geom.Ray2 method), 81
 distance() (Goulib.geom.Segment2 method), 82
 distance_on_sphere() (Goulib.geom3d.Sphere method), 102
 dither() (Goulib.image.Image method), 152
 dither() (in module Goulib.image), 158
 Ditherer (class in Goulib.image), 155
 divisors() (in module Goulib.math2), 195
 doCleanups() (Goulib.tests.TestCase method), 252
 dot() (Goulib.geom.Point2 method), 76
 dot() (Goulib.geom.Vector2 method), 73
 dot() (Goulib.geom3d.Vector3 method), 99
 dot() (in module Goulib.math2), 193
 dot_mm() (in module Goulib.math2), 193
 dot_mv() (in module Goulib.math2), 193
 dot_vv() (in module Goulib.math2), 192
 draw() (Goulib.drawing.Chain method), 51
 draw() (Goulib.drawing.Drawing method), 60
 draw() (Goulib.drawing.Entity method), 29, 38
 draw() (Goulib.drawing.Group method), 44
 draw() (Goulib.drawing.Instance method), 47
 draw() (Goulib.drawing.Rect method), 54
 draw() (Goulib.drawing.Spline method), 40
 draw() (Goulib.drawing.Text method), 57
 draw() (Goulib.geom.Arc2 method), 88
 draw() (Goulib.geom.Circle method), 85
 draw() (Goulib.geom.Ellipse method), 91
 draw() (Goulib.geom.Point2 method), 76
 draw() (Goulib.geom.Segment2 method), 82
 draw() (Goulib.graph.DiGraph method), 134
 draw() (Goulib.graph.GeoGraph method), 119
 draw_networkx() (in module Goulib.graph), 108, 147
 Drawing (class in Goulib.drawing), 33, 58
 drop() (in module Goulib.itertools2), 166
 dst() (Goulib.datetime2.datetime2 method), 19
 dst() (Goulib.datetime2.time2 method), 23
 dt() (Goulib.motion.Segment method), 204
 dt() (Goulib.motion.SegmentPoly method), 208
 dt() (Goulib.motion.Segments method), 207
 dummy (class in Goulib.markup), 179
- ## E
- edge_attr_dict_factory (Goulib.graph.DiGraph attribute), 134
 edge_attr_dict_factory (Goulib.graph.GeoGraph attribute), 119
 edge_key_dict_factory (Goulib.graph.DiGraph attribute), 135
 edge_key_dict_factory (Goulib.graph.GeoGraph attribute), 119

edges() (Goulib.graph.DiGraph method), 135
edges() (Goulib.graph.GeoGraph method), 119
edges_iter() (Goulib.graph.DiGraph method), 135
edges_iter() (Goulib.graph.GeoGraph method), 120
element (class in Goulib.markup), 175
Ellipse (class in Goulib.geom), 90
empty() (Goulib.drawing.BBox method), 36
empty() (Goulib.interval.Box method), 164
empty() (Goulib.interval.Interval method), 160
end (Goulib.drawing.BBox attribute), 36
end (Goulib.drawing.Chain attribute), 31, 48
end (Goulib.drawing.Drawing attribute), 60
end (Goulib.drawing.Entity attribute), 28, 37
end (Goulib.drawing.Group attribute), 44
end (Goulib.drawing.Instance attribute), 47
end (Goulib.drawing.Rect attribute), 54
end (Goulib.drawing.Spline attribute), 30, 39
end (Goulib.drawing.Text attribute), 57
end (Goulib.geom.Arc2 attribute), 88
end (Goulib.geom.Circle attribute), 85
end (Goulib.geom.Ellipse attribute), 91
end (Goulib.geom.Point2 attribute), 76
end (Goulib.geom.Segment2 attribute), 82
end (Goulib.interval.Box attribute), 163
end (Goulib.interval.Interval attribute), 159
end (Goulib.workdays.WorkCalendar attribute), 253
end() (Goulib.motion.Segment method), 204
end() (Goulib.motion.SegmentPoly method), 208
end() (Goulib.motion.Segments method), 206
endAcc() (Goulib.motion.Segment method), 204
endAcc() (Goulib.motion.SegmentPoly method), 208
endAcc() (Goulib.motion.Segments method), 207
endJerk() (Goulib.motion.Segment method), 204
endJerk() (Goulib.motion.SegmentPoly method), 208
endJerk() (Goulib.motion.Segments method), 207
endPos() (Goulib.motion.Segment method), 204
endPos() (Goulib.motion.SegmentPoly method), 209
endPos() (Goulib.motion.Segments method), 207
endSpeed() (Goulib.motion.Segment method), 204
endSpeed() (Goulib.motion.SegmentPoly method), 209
endSpeed() (Goulib.motion.Segments method), 207
endTime() (Goulib.motion.Actuator method), 210
endTime() (Goulib.motion.Segment method), 204
endTime() (Goulib.motion.SegmentPoly method), 209
endTime() (Goulib.motion.Segments method), 207
ensure_sorted() (in module Goulib.itertools2), 173
Entity (class in Goulib.drawing), 28, 37
enumerates() (in module Goulib.itertools2), 166
equal() (in module Goulib.datetime2), 26
ErrorDiffusion (class in Goulib.image), 156
escape() (in module Goulib.markup), 179
euclid_gen() (in module Goulib.math2), 196
euclidean_minimum_spanning_tree() (in module Goulib.graph), 109, 148

euler_phi() (in module Goulib.math2), 196
eval() (in module Goulib.expr), 62
EventLog (class in Goulib.statemachine), 229
every() (in module Goulib.itertools2), 166
evolve() (Goulib.optim.DifferentialEvolution method), 222
expand() (Goulib.image.Image method), 153
Expr (class in Goulib.expr), 62
extend() (Goulib.drawing.BBox method), 36
extend() (Goulib.drawing.Chain method), 51
extend() (Goulib.drawing.Drawing method), 60
extend() (Goulib.drawing.Group method), 30, 42
extend() (Goulib.drawing.Rect method), 54
extend() (Goulib.interval.Box method), 165
extend() (Goulib.interval.Interval method), 161
extend() (Goulib.optim.BinList method), 218
extend() (Goulib.piecewise.Piecewise method), 223
extend() (Goulib.stats.Discrete method), 234
extend() (Goulib.stats.Normal method), 239
extend() (Goulib.stats.PDF method), 236
extend() (Goulib.stats.Stats method), 231
extend() (Goulib.table.Table method), 245
eye() (in module Goulib.math2), 193

F

factorial_gen() (in module Goulib.math2), 200
factorize() (in module Goulib.math2), 196
factors() (in module Goulib.math2), 196
fail() (Goulib.tests.TestCase method), 252
failIf() (Goulib.tests.TestCase method), 252
failIfAlmostEqual() (Goulib.tests.TestCase method), 252
failIfEqual() (Goulib.tests.TestCase method), 252
failUnless() (Goulib.tests.TestCase method), 252
failUnlessAlmostEqual() (Goulib.tests.TestCase method), 252
failUnlessEqual() (Goulib.tests.TestCase method), 252
failUnlessRaises() (Goulib.tests.TestCase method), 252
failureException (Goulib.tests.TestCase attribute), 252
faulhaber() (in module Goulib.math2), 199
fibonacci() (in module Goulib.math2), 195
fibonacci_gen() (in module Goulib.math2), 195
fig2img() (in module Goulib.image), 155
figure() (Goulib.drawing.Chain method), 51
figure() (Goulib.drawing.Drawing method), 60
figure() (Goulib.drawing.Entity static method), 29, 38
figure() (Goulib.drawing.Group method), 44
figure() (Goulib.drawing.Instance method), 47
figure() (Goulib.drawing.Rect method), 54
figure() (Goulib.drawing.Spline method), 40
figure() (Goulib.drawing.Text method), 57
figure() (Goulib.geom.Arc2 method), 88
figure() (Goulib.geom.Circle method), 85
figure() (Goulib.geom.Ellipse method), 91
figure() (Goulib.geom.Point2 method), 76

- figure() (Goulib.geom.Segment2 method), 82
figure() (in module Goulib.graph), 108, 147
filter() (Goulib.container.Sequence method), 17
filter() (Goulib.image.Image method), 152
filter2() (in module Goulib.itertools2), 169
find() (Goulib.container.SortedCollection method), 14
find() (Goulib.interval.Intervals method), 163
find() (in module Goulib.itertools2), 170
find_col() (Goulib.table.Table method), 244
find_ge() (Goulib.container.SortedCollection method), 14
find_ge() (Goulib.interval.Intervals method), 163
find_gt() (Goulib.container.SortedCollection method), 14
find_gt() (Goulib.interval.Intervals method), 163
find_le() (Goulib.container.SortedCollection method), 14
find_le() (Goulib.interval.Intervals method), 163
find_lt() (Goulib.container.SortedCollection method), 14
find_lt() (Goulib.interval.Intervals method), 163
first() (in module Goulib.itertools2), 166
first_fit_decreasing() (in module Goulib.optim), 220
first_match() (in module Goulib.itertools2), 174
fits() (Goulib.optim.BinDict method), 217
fits() (Goulib.optim.BinList method), 219
flatten() (in module Goulib.itertools2), 167
floyd() (in module Goulib.itertools2), 174
FloydSteinberg (class in Goulib.image), 157
format() (in module Goulib.math2), 192
FRI (Goulib.workdays.WorkCalendar attribute), 254
from_dxf() (Goulib.drawing.Chain static method), 31, 49
from_dxf() (Goulib.drawing.Drawing method), 60
from_dxf() (Goulib.drawing.Entity static method), 29, 37
from_dxf() (Goulib.drawing.Group method), 30, 42
from_dxf() (Goulib.drawing.Instance static method), 31, 46
from_dxf() (Goulib.drawing.Rect method), 54
from_dxf() (Goulib.drawing.Spline method), 40
from_dxf() (Goulib.drawing.Text method), 57
from_dxf() (Goulib.geom.Arc2 method), 89
from_dxf() (Goulib.geom.Circle method), 86
from_dxf() (Goulib.geom.Ellipse method), 91
from_dxf() (Goulib.geom.Point2 method), 76
from_dxf() (Goulib.geom.Segment2 method), 82
from_pdf() (Goulib.drawing.Chain static method), 31, 48
from_pdf() (Goulib.drawing.Drawing method), 61
from_pdf() (Goulib.drawing.Entity static method), 29, 37
from_pdf() (Goulib.drawing.Group method), 44
from_pdf() (Goulib.drawing.Instance method), 47
from_pdf() (Goulib.drawing.Rect method), 54
from_pdf() (Goulib.drawing.Spline method), 41
from_pdf() (Goulib.drawing.Text method), 58
from_pdf() (Goulib.geom.Arc2 method), 89
from_pdf() (Goulib.geom.Circle method), 86
from_pdf() (Goulib.geom.Ellipse method), 91
from_pdf() (Goulib.geom.Point2 method), 77
from_pdf() (Goulib.geom.Segment2 method), 83
from_svg() (Goulib.drawing.Chain static method), 31, 48
from_svg() (Goulib.drawing.Drawing method), 61
from_svg() (Goulib.drawing.Entity static method), 29, 37
from_svg() (Goulib.drawing.Group method), 44
from_svg() (Goulib.drawing.Instance method), 47
from_svg() (Goulib.drawing.Rect method), 54
from_svg() (Goulib.drawing.Spline method), 41
from_svg() (Goulib.drawing.Text method), 58
from_svg() (Goulib.geom.Arc2 method), 89
from_svg() (Goulib.geom.Circle method), 86
from_svg() (Goulib.geom.Ellipse method), 91
from_svg() (Goulib.geom.Point2 method), 77
from_svg() (Goulib.geom.Segment2 method), 83
fromkeys() (Goulib.colors.Palette method), 10
fromkeys() (Goulib.container.Record method), 12
fromkeys() (Goulib.graph.index.Index method), 112
fromkeys() (Goulib.optim.BinDict method), 217
fromordinal() (Goulib.datetime2.date2 method), 21
fromordinal() (Goulib.datetime2.datetime2 method), 19
fromtimestamp() (Goulib.datetime2.date2 method), 21
fromtimestamp() (Goulib.datetime2.datetime2 method), 19
fspecial() (in module Goulib.image), 154
FullTime (in module Goulib.workdays), 255
- ## G
- gcd() (in module Goulib.math2), 191
generic_visit() (Goulib.expr.LatexVisitor method), 67
generic_visit() (Goulib.expr.TextVisitor method), 66
GeoGraph (class in Goulib.graph), 107, 114
Geometry (class in Goulib.geom), 67
get() (Goulib.colors.Palette method), 10
get() (Goulib.container.Record method), 12
get() (Goulib.graph.index.Index method), 112
get() (Goulib.optim.BinDict method), 217
get() (Goulib.table.Table method), 244
get_angle_axis() (Goulib.geom3d.Quaternion method), 106
get_cardinal_name() (in module Goulib.math2), 200
get_edge_data() (Goulib.graph.DiGraph method), 136
get_edge_data() (Goulib.graph.GeoGraph method), 121
get_euler() (Goulib.geom3d.Quaternion method), 106
get_function_source() (in module Goulib.expr), 62
get_matrix() (Goulib.geom3d.Quaternion method), 106
get_thread_pool() (in module Goulib.decorators), 27
getcolors() (Goulib.image.Image method), 150
getdata() (Goulib.image.Image method), 150
gethours() (Goulib.workdays.WorkCalendar method), 254
getpalette() (Goulib.image.Image method), 150
getpixel() (Goulib.image.Image method), 150
Goulib.colors (module), 6
Goulib.container (module), 11
Goulib.datetime2 (module), 17

- Goulib.decorators (module), 27
 - Goulib.drawing (module), 27, 33
 - Goulib.expr (module), 62
 - Goulib.geom (module), 67
 - Goulib.geom3d (module), 98
 - Goulib.graph (module), 107, 109
 - Goulib.image (module), 148
 - Goulib.interval (module), 158
 - Goulib.itertools2 (module), 166
 - Goulib.markup (module), 175
 - Goulib.math2 (module), 191
 - Goulib.motion (module), 203
 - Goulib.optim (module), 214
 - Goulib.piecewise (module), 222
 - Goulib.plot (module), 224
 - Goulib.polynomial (module), 226
 - Goulib.statemachine (module), 229
 - Goulib.stats (module), 231
 - Goulib.table (module), 240
 - Goulib.tests (module), 246
 - Goulib.workdays (module), 253
 - gray2bool() (in module Goulib.image), 158
 - gray2rgb() (in module Goulib.image), 158
 - grayscale() (Goulib.image.Image method), 152
 - Group (class in Goulib.drawing), 30, 41
 - groupby() (Goulib.table.Table method), 246
 - groupby_gen() (Goulib.table.Table method), 246
 - groups() (in module Goulib.itertools2), 167
 - guessmode() (in module Goulib.image), 149
- ## H
- hamming() (in module Goulib.math2), 194
 - has_edge() (Goulib.graph.DiGraph method), 137
 - has_edge() (Goulib.graph.GeoGraph method), 122
 - has_node() (Goulib.graph.DiGraph method), 137
 - has_node() (Goulib.graph.GeoGraph method), 122
 - has_predecessor() (Goulib.graph.DiGraph method), 137
 - has_successor() (Goulib.graph.DiGraph method), 137
 - height (Goulib.drawing.BBox attribute), 28, 34
 - heptagonal() (in module Goulib.math2), 200
 - herror() (Goulib.statemachine.StateMachine method), 230
 - hex (Goulib.colors.Color attribute), 7
 - hex2rgb() (in module Goulib.colors), 6
 - hexagonal() (in module Goulib.math2), 200
 - hierarchy() (Goulib.table.Table method), 246
 - hillclimb() (in module Goulib.optim), 220
 - hillclimb_and_restart() (in module Goulib.optim), 220
 - hinfo() (Goulib.statemachine.StateMachine method), 230
 - hour (Goulib.datetime2.datetime2 attribute), 19
 - hour (Goulib.datetime2.time2 attribute), 23
 - hours() (in module Goulib.datetime2), 26
 - hsuccess() (Goulib.statemachine.StateMachine method), 230
 - hsv (Goulib.colors.Color attribute), 7
 - html() (Goulib.drawing.Chain method), 51
 - html() (Goulib.drawing.Drawing method), 61
 - html() (Goulib.drawing.Entity method), 39
 - html() (Goulib.drawing.Group method), 44
 - html() (Goulib.drawing.Instance method), 47
 - html() (Goulib.drawing.Rect method), 55
 - html() (Goulib.drawing.Spline method), 41
 - html() (Goulib.drawing.Text method), 58
 - html() (Goulib.expr.Expr method), 64
 - html() (Goulib.geom.Arc2 method), 89
 - html() (Goulib.geom.Circle method), 86
 - html() (Goulib.geom.Ellipse method), 91
 - html() (Goulib.geom.Point2 method), 77
 - html() (Goulib.geom.Segment2 method), 83
 - html() (Goulib.graph.DiGraph method), 138
 - html() (Goulib.graph.GeoGraph method), 122
 - html() (Goulib.image.Image method), 154
 - html() (Goulib.motion.PVA method), 204
 - html() (Goulib.motion.Segment method), 205
 - html() (Goulib.motion.SegmentPoly method), 209
 - html() (Goulib.motion.Segments method), 206
 - html() (Goulib.motion.TimeDiagram method), 212
 - html() (Goulib.piecewise.Piecewise method), 224
 - html() (Goulib.plot.Plot method), 225
 - html() (Goulib.polynomial.Polynomial method), 228
 - html() (Goulib.stats.Normal method), 239
 - html() (Goulib.stats.PDF method), 236
 - html() (Goulib.table.Cell method), 240
 - html() (Goulib.table.Row method), 242
 - html() (Goulib.table.Table method), 243
 - hull() (Goulib.interval.Interval method), 160
 - hwarning() (Goulib.statemachine.StateMachine method), 230
- ## I
- iapply() (Goulib.piecewise.Piecewise method), 223
 - icol() (Goulib.table.Table method), 244
 - icross() (in module Goulib.itertools2), 169
 - id() (Goulib.tests.TestCase method), 252
 - identity() (Goulib.geom.Matrix3 method), 96
 - identity() (Goulib.geom3d.Matrix4 method), 103
 - identity() (Goulib.geom3d.Quaternion method), 106
 - identity() (in module Goulib.itertools2), 168
 - identity() (in module Goulib.math2), 193
 - ifind() (in module Goulib.itertools2), 169
 - ilen() (in module Goulib.itertools2), 166
 - ilog() (in module Goulib.math2), 201
 - Image (class in Goulib.image), 149
 - in_degree() (Goulib.graph.DiGraph method), 138
 - in_degree_iter() (Goulib.graph.DiGraph method), 138
 - in_edges() (Goulib.graph.DiGraph method), 138
 - in_edges_iter() (Goulib.graph.DiGraph method), 139
 - in_interval() (in module Goulib.interval), 158

- ind2any() (in module Goulib.image), 158
- ind2rgb() (in module Goulib.image), 158
- index (class in Goulib.graph), 107, 109
- index() (Goulib.colors.Palette method), 8
- index() (Goulib.container.Sequence method), 16
- index() (Goulib.container.SortedCollection method), 14
- index() (Goulib.drawing.BBox method), 36
- index() (Goulib.drawing.Chain method), 51
- index() (Goulib.drawing.Drawing method), 61
- index() (Goulib.drawing.Group method), 44
- index() (Goulib.drawing.Rect method), 55
- index() (Goulib.interval.Box method), 165
- index() (Goulib.interval.Interval method), 161
- index() (Goulib.interval.Intervals method), 163
- index() (Goulib.optim.BinList method), 219
- index() (Goulib.piecewise.Piecewise method), 222
- index() (Goulib.table.Table method), 244
- index() (in module Goulib.itertools2), 166
- index.Index (class in Goulib.graph), 107, 110
- index.Property (class in Goulib.graph), 107, 109
- index_max() (in module Goulib.itertools2), 169
- index_min() (in module Goulib.itertools2), 169
- init() (Goulib.markup.page method), 177
- init__() (Goulib.datetime2.date2 method), 20
- insert() (Goulib.container.SortedCollection method), 14
- insert() (Goulib.drawing.BBox method), 36
- insert() (Goulib.drawing.Chain method), 51
- insert() (Goulib.drawing.Drawing method), 61
- insert() (Goulib.drawing.Group method), 44
- insert() (Goulib.drawing.Rect method), 55
- insert() (Goulib.graph.index.Index method), 107, 110
- insert() (Goulib.interval.Box method), 165
- insert() (Goulib.interval.Interval method), 161
- insert() (Goulib.interval.Intervals method), 162
- insert() (Goulib.itertools2.iter2 method), 170
- insert() (Goulib.motion.Segments method), 206
- insert() (Goulib.optim.BinList method), 218
- insert() (Goulib.table.Table method), 245
- insert_right() (Goulib.container.SortedCollection method), 14
- insert_right() (Goulib.interval.Intervals method), 163
- Instance (class in Goulib.drawing), 30, 45
- int_or_float() (in module Goulib.math2), 192
- integer_exponent() (in module Goulib.math2), 197
- integral() (Goulib.polynomial.Polynomial method), 227
- integral() (in module Goulib.polynomial), 228
- interleave() (in module Goulib.itertools2), 169
- intersect() (Goulib.drawing.Chain method), 51
- intersect() (Goulib.drawing.Drawing method), 61
- intersect() (Goulib.drawing.Group method), 44
- intersect() (Goulib.drawing.Instance method), 47
- intersect() (Goulib.drawing.Rect method), 55
- intersect() (Goulib.drawing.Spline method), 41
- intersect() (Goulib.drawing.Text method), 33, 56
- intersect() (Goulib.geom.Arc2 method), 87
- intersect() (Goulib.geom.Circle method), 84
- intersect() (Goulib.geom.Ellipse method), 92
- intersect() (Goulib.geom.Geometry method), 68
- intersect() (Goulib.geom.Line2 method), 79
- intersect() (Goulib.geom.Point2 method), 74
- intersect() (Goulib.geom.Ray2 method), 81
- intersect() (Goulib.geom.Segment2 method), 83
- intersect() (Goulib.geom3d.Line3 method), 101
- intersect() (Goulib.geom3d.Plane method), 103
- intersect() (Goulib.geom3d.Point3 method), 100
- intersect() (Goulib.geom3d.Sphere method), 102
- intersect() (in module Goulib.interval), 158
- intersect() (in module Goulib.itertools2), 173
- intersection() (Goulib.interval.Interval method), 159
- intersection() (in module Goulib.interval), 159
- intersectlen() (in module Goulib.interval), 159
- Interval (class in Goulib.interval), 159
- Intervals (class in Goulib.interval), 161
- InvalidElementError, 186
- inverse() (Goulib.geom.Matrix3 method), 98
- inverse() (Goulib.geom3d.Matrix4 method), 104
- invert() (Goulib.image.Image method), 152
- irange() (in module Goulib.itertools2), 166
- ireduce() (in module Goulib.itertools2), 168
- iremove() (in module Goulib.itertools2), 170
- is_directed() (Goulib.graph.DiGraph method), 139
- is_directed() (Goulib.graph.GeoGraph method), 122
- is_fibonacci() (in module Goulib.math2), 195
- is_happy() (in module Goulib.math2), 199
- is_heptagonal() (in module Goulib.math2), 200
- is_hexagonal() (in module Goulib.math2), 200
- is_integer() (in module Goulib.math2), 192
- is_lychrel() (in module Goulib.math2), 199
- is_multigraph() (Goulib.graph.DiGraph method), 139
- is_multigraph() (Goulib.graph.GeoGraph method), 122
- is_multiple() (in module Goulib.math2), 196
- is_number() (in module Goulib.math2), 191
- is_octagonal() (in module Goulib.math2), 200
- is_palindromic() (in module Goulib.math2), 198
- is_pandigital() (in module Goulib.math2), 198
- is_pentagonal() (in module Goulib.math2), 200
- is_perfect() (in module Goulib.math2), 200
- is_permutation() (in module Goulib.math2), 198
- is_prime() (in module Goulib.math2), 195
- is_primitive_root() (in module Goulib.math2), 191
- is_pythagorean_triple() (in module Goulib.math2), 195
- is_square() (in module Goulib.math2), 192
- is_triangle() (in module Goulib.math2), 200
- is_triangular() (in module Goulib.math2), 200
- iscallable() (in module Goulib.itertools2), 166
- isclose() (Goulib.colors.Color method), 7
- isclose() (in module Goulib.math2), 192
- isclosed() (Goulib.drawing.Chain method), 51

- isclosed() (Goulib.drawing.Drawing method), 61
 - isclosed() (Goulib.drawing.Entity method), 29, 37
 - isclosed() (Goulib.drawing.Group method), 44
 - isclosed() (Goulib.drawing.Instance method), 47
 - isclosed() (Goulib.drawing.Rect method), 55
 - isclosed() (Goulib.drawing.Spline method), 41
 - isclosed() (Goulib.drawing.Text method), 58
 - isclosed() (Goulib.geom.Arc2 method), 89
 - isclosed() (Goulib.geom.Circle method), 86
 - isclosed() (Goulib.geom.Ellipse method), 92
 - isclosed() (Goulib.geom.Point2 method), 77
 - isclosed() (Goulib.geom.Segment2 method), 83
 - isconstant (Goulib.expr.Expr attribute), 63
 - isconstant (Goulib.piecewise.Piecewise attribute), 224
 - isconstant (Goulib.polynomial.Polynomial attribute), 228
 - isconstant (Goulib.stats.Normal attribute), 239
 - isconstant (Goulib.stats.PDF attribute), 236
 - ishorizontal() (Goulib.drawing.Chain method), 51
 - ishorizontal() (Goulib.drawing.Drawing method), 61
 - ishorizontal() (Goulib.drawing.Entity method), 29, 37
 - ishorizontal() (Goulib.drawing.Group method), 45
 - ishorizontal() (Goulib.drawing.Instance method), 47
 - ishorizontal() (Goulib.drawing.Rect method), 55
 - ishorizontal() (Goulib.drawing.Spline method), 41
 - ishorizontal() (Goulib.drawing.Text method), 58
 - ishorizontal() (Goulib.geom.Arc2 method), 89
 - ishorizontal() (Goulib.geom.Circle method), 86
 - ishorizontal() (Goulib.geom.Ellipse method), 92
 - ishorizontal() (Goulib.geom.Point2 method), 77
 - ishorizontal() (Goulib.geom.Segment2 method), 83
 - isiterable() (in module Goulib.itertools2), 166
 - isline() (Goulib.drawing.Chain method), 51
 - isline() (Goulib.drawing.Drawing method), 61
 - isline() (Goulib.drawing.Entity method), 29, 37
 - isline() (Goulib.drawing.Group method), 45
 - isline() (Goulib.drawing.Instance method), 47
 - isline() (Goulib.drawing.Rect method), 55
 - isline() (Goulib.drawing.Spline method), 41
 - isline() (Goulib.drawing.Text method), 58
 - isline() (Goulib.geom.Arc2 method), 89
 - isline() (Goulib.geom.Circle method), 86
 - isline() (Goulib.geom.Ellipse method), 92
 - isline() (Goulib.geom.Point2 method), 77
 - isline() (Goulib.geom.Segment2 method), 83
 - isocalendar() (Goulib.datetime2.date2 method), 21
 - isocalendar() (Goulib.datetime2.datetime2 method), 19
 - isoformat() (Goulib.datetime2.date2 method), 21
 - isoformat() (Goulib.datetime2.datetime2 method), 19
 - isoformat() (Goulib.datetime2.time2 method), 23
 - isoformat() (Goulib.datetime2.timedelta2 method), 23
 - isoweekday() (Goulib.datetime2.date2 method), 21
 - isoweekday() (Goulib.datetime2.datetime2 method), 19
 - isplit() (in module Goulib.itertools2), 170
 - isqrt() (in module Goulib.math2), 192
 - isvertical() (Goulib.drawing.Chain method), 51
 - isvertical() (Goulib.drawing.Drawing method), 61
 - isvertical() (Goulib.drawing.Entity method), 29, 37
 - isvertical() (Goulib.drawing.Group method), 45
 - isvertical() (Goulib.drawing.Instance method), 47
 - isvertical() (Goulib.drawing.Rect method), 55
 - isvertical() (Goulib.drawing.Spline method), 41
 - isvertical() (Goulib.drawing.Text method), 58
 - isvertical() (Goulib.geom.Arc2 method), 89
 - isvertical() (Goulib.geom.Circle method), 86
 - isvertical() (Goulib.geom.Ellipse method), 92
 - isvertical() (Goulib.geom.Point2 method), 77
 - isvertical() (Goulib.geom.Segment2 method), 83
 - isworkday() (Goulib.workdays.WorkCalendar method), 253
 - isworktime() (Goulib.workdays.WorkCalendar method), 253
 - itemgetter() (in module Goulib.itertools2), 167
 - items() (Goulib.colors.Palette method), 10
 - items() (Goulib.container.Record method), 12
 - items() (Goulib.graph.index.Index method), 112
 - items() (Goulib.optim.BinDict method), 217
 - iter2 (class in Goulib.itertools2), 170
 - iterate() (in module Goulib.itertools2), 168
 - itimerout() (in module Goulib.decorators), 27
- ## J
- json() (Goulib.table.Table method), 243
- ## K
- keep (class in Goulib.itertools2), 173
 - key (Goulib.container.SortedCollection attribute), 14
 - key (Goulib.interval.Intervals attribute), 163
 - keys() (Goulib.colors.Palette method), 10
 - keys() (Goulib.container.Record method), 12
 - keys() (Goulib.graph.index.Index method), 112
 - keys() (Goulib.optim.BinDict method), 217
 - kirkpatrick_cooling() (in module Goulib.optim), 220
 - kurtosis() (in module Goulib.stats), 231
- ## L
- lab (Goulib.colors.Color attribute), 7
 - lab2ind() (in module Goulib.image), 158
 - last() (in module Goulib.itertools2), 166
 - lastExitTime() (Goulib.statemachine.StateMachine method), 230
 - latex() (Goulib.expr.Expr method), 63
 - latex() (Goulib.piecewise.Piecewise method), 224
 - latex() (Goulib.polynomial.Polynomial method), 228
 - latex() (Goulib.stats.Normal method), 237
 - latex() (Goulib.stats.PDF method), 236
 - LatexVisitor (class in Goulib.expr), 66
 - layer (Goulib.drawing.Chain attribute), 51
 - layer (Goulib.drawing.Drawing attribute), 61

layer (Goulib.drawing.Group attribute), 30, 42
 layer (Goulib.drawing.Rect attribute), 55
 lcm() (in module Goulib.math2), 191
 length (Goulib.drawing.Chain attribute), 51
 length (Goulib.drawing.Drawing attribute), 61
 length (Goulib.drawing.Group attribute), 45
 length (Goulib.drawing.Instance attribute), 47
 length (Goulib.drawing.Rect attribute), 55
 length (Goulib.drawing.Spline attribute), 30, 39
 length (Goulib.drawing.Text attribute), 32, 56
 length (Goulib.geom.Arc2 attribute), 89
 length (Goulib.geom.Circle attribute), 84
 length (Goulib.geom.Ellipse attribute), 92
 length (Goulib.geom.Point2 attribute), 77
 length (Goulib.geom.Segment2 attribute), 81
 length (Goulib.geom.Vector2 attribute), 73
 length (Goulib.geom3d.Segment3 attribute), 101
 length() (Goulib.graph.DiGraph method), 139
 length() (Goulib.graph.GeoGraph method), 122
 levenshtein() (in module Goulib.math2), 194
 Line2 (class in Goulib.geom), 78
 Line3 (class in Goulib.geom3d), 100
 linear() (Goulib.stats.Normal method), 237
 linear_regression() (in module Goulib.stats), 239
 linspace() (in module Goulib.itertools2), 167
 load() (Goulib.drawing.Drawing method), 33, 62
 load() (Goulib.image.Image method), 150
 load() (Goulib.table.Table method), 243
 log() (Goulib.statemachine.EventLog method), 229
 log_binomial() (in module Goulib.math2), 201
 log_factorial() (in module Goulib.math2), 201
 longMessage (Goulib.tests.TestCase attribute), 252
 lucas_lehmer() (in module Goulib.math2), 197
 luv (Goulib.colors.Color attribute), 7
 lychrel_count() (in module Goulib.math2), 199
 lychrel_seq() (in module Goulib.math2), 199

M

mag() (Goulib.geom.Matrix3 method), 97
 mag() (Goulib.geom.Point2 method), 77
 mag() (Goulib.geom.Vector2 method), 73
 mag() (Goulib.geom3d.Quaternion method), 106
 mag() (Goulib.geom3d.Vector3 method), 99
 mag2() (Goulib.geom.Matrix3 method), 98
 mag2() (Goulib.geom.Point2 method), 77
 mag2() (Goulib.geom.Segment2 method), 81
 mag2() (Goulib.geom.Vector2 method), 73
 mag2() (Goulib.geom3d.Quaternion method), 106
 mag2() (Goulib.geom3d.Segment3 method), 101
 mag2() (Goulib.geom3d.Vector3 method), 99
 make_random_population()
 (Goulib.optim.DifferentialEvolution method),
 221
 MarkupError, 181

Matrix3 (class in Goulib.geom), 92
 Matrix4 (class in Goulib.geom3d), 103
 matrix_power() (in module Goulib.math2), 202
 max (Goulib.datetime2.date2 attribute), 21
 max (Goulib.datetime2.datetime2 attribute), 19
 max (Goulib.datetime2.time2 attribute), 23
 max (Goulib.datetime2.timedelta2 attribute), 25
 max (Goulib.drawing.BBox attribute), 36
 max (Goulib.interval.Box attribute), 165
 maxAbsAcc() (Goulib.motion.Actuator method), 210
 maxAbsSpeed() (Goulib.motion.Actuator method), 210
 maxDiff (Goulib.tests.TestCase attribute), 252
 maxForce() (Goulib.motion.Actuator method), 210
 maximum() (in module Goulib.math2), 193
 maxRpm() (Goulib.motion.Actuator method), 210
 maxTork() (Goulib.motion.Actuator method), 210
 mean (Goulib.stats.Discrete attribute), 234
 mean (Goulib.stats.Normal attribute), 239
 mean (Goulib.stats.PDF attribute), 236
 mean (Goulib.stats.Stats attribute), 232
 mean() (in module Goulib.stats), 231
 mean_var() (in module Goulib.stats), 231
 median() (in module Goulib.stats), 231
 memoize() (in module Goulib.decorators), 27
 metainfo() (Goulib.markup.page method), 178
 microsecond (Goulib.datetime2.datetime2 attribute), 19
 microsecond (Goulib.datetime2.time2 attribute), 23
 microseconds (Goulib.datetime2.timedelta2 attribute), 25
 midpoint() (Goulib.geom.Segment2 method), 81
 min (Goulib.datetime2.date2 attribute), 21
 min (Goulib.datetime2.datetime2 attribute), 19
 min (Goulib.datetime2.time2 attribute), 23
 min (Goulib.datetime2.timedelta2 attribute), 25
 min (Goulib.drawing.BBox attribute), 36
 min (Goulib.interval.Box attribute), 165
 minimum() (in module Goulib.math2), 193
 minus() (Goulib.workdays.WorkCalendar method), 254
 minute (Goulib.datetime2.datetime2 attribute), 19
 minute (Goulib.datetime2.time2 attribute), 23
 minutes() (in module Goulib.datetime2), 26
 mod_binomial() (in module Goulib.math2), 202
 mod_div() (in module Goulib.math2), 202
 mod_fact() (in module Goulib.math2), 202
 mod_inv() (in module Goulib.math2), 202
 mod_matmul() (in module Goulib.math2), 202
 mod_matpow() (in module Goulib.math2), 202
 mod_pow() (in module Goulib.math2), 202
 Mode (class in Goulib.image), 148
 mode() (in module Goulib.stats), 231
 ModeError, 188
 moebius() (in module Goulib.math2), 196
 MON (Goulib.workdays.WorkCalendar attribute), 254
 month (Goulib.datetime2.date2 attribute), 21
 month (Goulib.datetime2.datetime2 attribute), 19

move() (Goulib.motion.Actuator method), 210
 move_to_end() (Goulib.colors.Palette method), 10
 move_to_end() (Goulib.container.Record method), 12
 mu (Goulib.stats.Discrete attribute), 234
 mu (Goulib.stats.Normal attribute), 239
 mu (Goulib.stats.PDF attribute), 236
 mu (Goulib.stats.Stats attribute), 232
 mul() (in module Goulib.math2), 192
 mult_const() (in module Goulib.polynomial), 228
 mult_one() (in module Goulib.polynomial), 229
 multi (Goulib.graph.DiGraph attribute), 139
 multi (Goulib.graph.GeoGraph attribute), 123
 multiply() (in module Goulib.math2), 192
 multiply() (in module Goulib.polynomial), 228

N

name (Goulib.colors.Color attribute), 7
 name (Goulib.graph.DiGraph attribute), 139
 name (Goulib.graph.GeoGraph attribute), 123
 native (Goulib.colors.Color attribute), 7
 nbunch_iter() (Goulib.graph.DiGraph method), 139
 nbunch_iter() (Goulib.graph.GeoGraph method), 123
 nchannels (Goulib.image.Image attribute), 150
 nchannels() (in module Goulib.image), 149
 ncols() (Goulib.table.Table method), 244
 ncombinations() (in module Goulib.math2), 201
 ndim() (in module Goulib.itertools2), 168
 nearest() (Goulib.graph.index.Index method), 107, 110
 nearest_color() (in module Goulib.colors), 10
 neighbors() (Goulib.graph.DiGraph method), 139
 neighbors() (Goulib.graph.GeoGraph method), 123
 neighbors_iter() (Goulib.graph.DiGraph method), 140
 neighbors_iter() (Goulib.graph.GeoGraph method), 123
 nelder_mead() (in module Goulib.optim), 215
 networkdays() (Goulib.workdays.WorkCalendar method), 254
 networkdays() (in module Goulib.workdays), 255
 new() (Goulib.geom3d.Matrix4 class method), 103
 new() (Goulib.image.Image static method), 150
 new_identity() (Goulib.geom.Matrix3 class method), 97
 new_identity() (Goulib.geom3d.Matrix4 class method), 103
 new_identity() (Goulib.geom3d.Quaternion class method), 106
 new_interpolate() (Goulib.geom3d.Quaternion class method), 106
 new_look_at() (Goulib.geom3d.Matrix4 class method), 104
 new_perspective() (Goulib.geom3d.Matrix4 class method), 104
 new_rotate() (Goulib.geom.Matrix3 class method), 98
 new_rotate_axis() (Goulib.geom3d.Matrix4 class method), 103

new_rotate_axis() (Goulib.geom3d.Quaternion class method), 106
 new_rotate_euler() (Goulib.geom3d.Matrix4 class method), 103
 new_rotate_euler() (Goulib.geom3d.Quaternion class method), 106
 new_rotate_matrix() (Goulib.geom3d.Quaternion class method), 106
 new_rotate_triple_axis() (Goulib.geom3d.Matrix4 class method), 104
 new_rotatex() (Goulib.geom3d.Matrix4 class method), 103
 new_rotatey() (Goulib.geom3d.Matrix4 class method), 103
 new_rotatez() (Goulib.geom3d.Matrix4 class method), 103
 new_scale() (Goulib.geom.Matrix3 class method), 98
 new_scale() (Goulib.geom3d.Matrix4 class method), 103
 new_translate() (Goulib.geom.Matrix3 class method), 98
 new_translate() (Goulib.geom3d.Matrix4 class method), 103
 next() (Goulib.itertools2.iter2 method), 170
 next() (Goulib.itertools2.keep method), 173
 next_permutation() (in module Goulib.itertools2), 170
 nextworkday() (Goulib.workdays.WorkCalendar method), 253
 no() (in module Goulib.itertools2), 169
 node_dict_factory (Goulib.graph.DiGraph attribute), 140
 node_dict_factory (Goulib.graph.GeoGraph attribute), 124
 nodebug() (in module Goulib.decorators), 27
 nodes() (Goulib.graph.DiGraph method), 140
 nodes() (Goulib.graph.GeoGraph method), 124
 nodes_iter() (Goulib.graph.DiGraph method), 140
 nodes_iter() (Goulib.graph.GeoGraph method), 124
 nodes_with_selfloops() (Goulib.graph.DiGraph method), 140
 nodes_with_selfloops() (Goulib.graph.GeoGraph method), 124
 noPrint() (in module Goulib.statemachine), 229
 norm() (in module Goulib.math2), 194
 norm_1() (in module Goulib.math2), 194
 norm_2() (in module Goulib.math2), 194
 norm_inf() (in module Goulib.math2), 194
 Normal (class in Goulib.stats), 237
 normal_pdf() (in module Goulib.stats), 237
 normalize() (Goulib.geom.Point2 method), 77
 normalize() (Goulib.geom.Vector2 method), 73
 normalize() (Goulib.geom3d.Quaternion method), 106
 normalize() (Goulib.geom3d.Vector3 method), 99
 normalize() (Goulib.image.Image method), 152
 normalize() (in module Goulib.image), 154
 normalized() (Goulib.geom.Point2 method), 77
 normalized() (Goulib.geom.Vector2 method), 73

- normalized() (Goulib.geom3d.Quaternion method), 106
normalized() (Goulib.geom3d.Vector3 method), 99
now() (Goulib.datetime2.datetime2 method), 19
npixels (Goulib.image.Image attribute), 150
nth() (in module Goulib.itertools2), 169
num_from_digits() (in module Goulib.math2), 198
number_of_digits() (in module Goulib.math2), 200
number_of_divisors() (in module Goulib.math2), 196
number_of_edges() (Goulib.graph.DiGraph method), 141
number_of_edges() (Goulib.graph.GeoGraph method), 125
number_of_nodes() (Goulib.graph.DiGraph method), 141
number_of_nodes() (Goulib.graph.GeoGraph method), 125
number_of_selfloops() (Goulib.graph.DiGraph method), 141
number_of_selfloops() (Goulib.graph.GeoGraph method), 125
- ## O
- ObjectiveFunction (class in Goulib.optim), 214
occurrences() (in module Goulib.itertools2), 168
octagonal() (in module Goulib.math2), 200
offset() (Goulib.geom.Matrix3 method), 97
omega() (in module Goulib.math2), 196
open() (Goulib.image.Image static method), 150
open() (Goulib.markup.element method), 175
OpeningError, 184
optimize() (Goulib.image.Image method), 151
optimize() (Goulib.optim.DifferentialEvolution method), 221
order() (Goulib.graph.DiGraph method), 141
order() (Goulib.graph.GeoGraph method), 125
orientation() (Goulib.geom.Matrix3 method), 98
out_degree() (Goulib.graph.DiGraph method), 141
out_degree_iter() (Goulib.graph.DiGraph method), 142
out_edges() (Goulib.graph.DiGraph method), 142
out_edges_iter() (Goulib.graph.DiGraph method), 142
overlap() (Goulib.interval.Interval method), 159
- ## P
- P() (Goulib.motion.Actuator method), 210
P() (in module Goulib.optim), 220
p1 (Goulib.drawing.Rect attribute), 32, 52
p1 (Goulib.geom.Segment2 attribute), 81
p1 (Goulib.geom3d.Line3 attribute), 100
p2 (Goulib.drawing.Rect attribute), 32, 52
p2 (Goulib.geom.Segment2 attribute), 81
p2 (Goulib.geom3d.Line3 attribute), 101
page (class in Goulib.markup), 176
pairwise() (Goulib.container.Sequence method), 17
pairwise() (in module Goulib.itertools2), 167
Palette (class in Goulib.colors), 8
palette() (in module Goulib.image), 158
parse_string() (in module Goulib.polynomial), 229
parseDoc() (Goulib.statemachine.StateMachine method), 230
partition() (in module Goulib.math2), 200
pascal_gen() (in module Goulib.math2), 195
paste() (Goulib.image.Image method), 151
patches() (Goulib.colors.Palette method), 8
patches() (Goulib.drawing.Chain method), 51
patches() (Goulib.drawing.Drawing method), 61
patches() (Goulib.drawing.Entity method), 29, 38
patches() (Goulib.drawing.Group method), 45
patches() (Goulib.drawing.Instance method), 47
patches() (Goulib.drawing.Rect method), 55
patches() (Goulib.drawing.Spline method), 41
patches() (Goulib.drawing.Text method), 33, 56
patches() (Goulib.geom.Arc2 method), 89
patches() (Goulib.geom.Circle method), 86
patches() (Goulib.geom.Ellipse method), 92
patches() (Goulib.geom.Point2 method), 77
patches() (Goulib.geom.Segment2 method), 83
PDF (class in Goulib.stats), 235
pearson() (Goulib.stats.Normal method), 237
pentagonal() (in module Goulib.math2), 200
pep8() (in module Goulib.tests), 253
peval() (in module Goulib.polynomial), 228
pi_digits_gen() (in module Goulib.math2), 202
Piecewise (class in Goulib.piecewise), 222
pil (Goulib.colors.Palette attribute), 8
pil (Goulib.image.Image attribute), 150
pisano_cycle() (in module Goulib.math2), 195
pisano_period() (in module Goulib.math2), 195
Plane (class in Goulib.geom3d), 102
plist() (in module Goulib.polynomial), 228
Plot (class in Goulib.plot), 224
plot() (Goulib.drawing.Chain method), 51
plot() (Goulib.drawing.Drawing method), 61
plot() (Goulib.drawing.Entity method), 39
plot() (Goulib.drawing.Group method), 45
plot() (Goulib.drawing.Instance method), 48
plot() (Goulib.drawing.Rect method), 55
plot() (Goulib.drawing.Spline method), 41
plot() (Goulib.drawing.Text method), 58
plot() (Goulib.expr.Expr method), 64
plot() (Goulib.geom.Arc2 method), 89
plot() (Goulib.geom.Circle method), 86
plot() (Goulib.geom.Ellipse method), 92
plot() (Goulib.geom.Point2 method), 77
plot() (Goulib.geom.Segment2 method), 83
plot() (Goulib.graph.DiGraph method), 143
plot() (Goulib.graph.GeoGraph method), 125
plot() (Goulib.image.Image method), 154
plot() (Goulib.motion.PVA method), 204
plot() (Goulib.motion.Segment method), 205
plot() (Goulib.motion.SegmentPoly method), 209

- plot() (Goulib.motion.Segments method), 207
- plot() (Goulib.motion.TimeDiagram method), 212
- plot() (Goulib.piecewise.Piecewise method), 224
- plot() (Goulib.plot.Plot method), 225
- plot() (Goulib.polynomial.Polynomial method), 228
- plot() (Goulib.stats.Normal method), 239
- plot() (Goulib.stats.PDF method), 236
- plot() (in module Goulib.plot), 226
- plus() (Goulib.workdays.WorkCalendar method), 254
- png() (Goulib.drawing.Chain method), 51
- png() (Goulib.drawing.Drawing method), 61
- png() (Goulib.drawing.Entity method), 39
- png() (Goulib.drawing.Group method), 45
- png() (Goulib.drawing.Instance method), 48
- png() (Goulib.drawing.Rect method), 55
- png() (Goulib.drawing.Spline method), 41
- png() (Goulib.drawing.Text method), 58
- png() (Goulib.expr.Expr method), 64
- png() (Goulib.geom.Arc2 method), 89
- png() (Goulib.geom.Circle method), 86
- png() (Goulib.geom.Ellipse method), 92
- png() (Goulib.geom.Point2 method), 77
- png() (Goulib.geom.Segment2 method), 83
- png() (Goulib.graph.DiGraph method), 143
- png() (Goulib.graph.GeoGraph method), 125
- png() (Goulib.image.Image method), 154
- png() (Goulib.motion.PVA method), 204
- png() (Goulib.motion.Segment method), 205
- png() (Goulib.motion.SegmentPoly method), 209
- png() (Goulib.motion.Segments method), 207
- png() (Goulib.motion.TimeDiagram method), 213
- png() (Goulib.piecewise.Piecewise method), 224
- png() (Goulib.plot.Plot method), 225
- png() (Goulib.polynomial.Polynomial method), 228
- png() (Goulib.stats.Normal method), 239
- png() (Goulib.stats.PDF method), 236
- png() (in module Goulib.plot), 226
- point() (Goulib.drawing.Chain method), 51
- point() (Goulib.drawing.Drawing method), 61
- point() (Goulib.drawing.Group method), 45
- point() (Goulib.drawing.Instance method), 48
- point() (Goulib.drawing.Rect method), 55
- point() (Goulib.drawing.Spline method), 41
- point() (Goulib.geom.Arc2 method), 87
- point() (Goulib.geom.Circle method), 84
- point() (Goulib.geom.Ellipse method), 92
- point() (Goulib.geom.Geometry method), 68
- point() (Goulib.geom.Line2 method), 79
- point() (Goulib.geom.Point2 method), 77
- point() (Goulib.geom.Ray2 method), 81
- point() (Goulib.geom.Segment2 method), 83
- point() (Goulib.geom3d.Line3 method), 101
- point() (Goulib.geom3d.Sphere method), 102
- Point2 (class in Goulib.geom), 74
- Point3 (class in Goulib.geom3d), 99
- points() (Goulib.piecewise.Piecewise method), 224
- points_on_sphere() (in module Goulib.graph), 109, 148
- Polar() (in module Goulib.geom), 78
- polygonal() (in module Goulib.math2), 199
- Polynomial (class in Goulib.polynomial), 226
- pop() (Goulib.colors.Palette method), 10
- pop() (Goulib.container.Record method), 12
- pop() (Goulib.container.SortedCollection method), 14
- pop() (Goulib.drawing.BBox method), 37
- pop() (Goulib.drawing.Chain method), 51
- pop() (Goulib.drawing.Drawing method), 61
- pop() (Goulib.drawing.Group method), 45
- pop() (Goulib.drawing.Rect method), 55
- pop() (Goulib.graph.index.Index method), 112
- pop() (Goulib.interval.Box method), 165
- pop() (Goulib.interval.Interval method), 161
- pop() (Goulib.interval.Intervals method), 163
- pop() (Goulib.optim.BinDict method), 217
- pop() (Goulib.optim.BinList method), 218
- pop() (Goulib.table.Table method), 245
- popitem() (Goulib.colors.Palette method), 10
- popitem() (Goulib.container.Record method), 12
- popitem() (Goulib.graph.index.Index method), 112
- popitem() (Goulib.optim.BinDict method), 217
- pos() (Goulib.graph.DiGraph method), 143
- pos() (Goulib.graph.GeoGraph method), 125
- power() (in module Goulib.polynomial), 229
- power_tower() (in module Goulib.math2), 197
- pprint() (in module Goulib.tests), 246
- pprint_gen() (in module Goulib.tests), 246
- prec() (Goulib.expr.LatexVisitor method), 67
- prec() (Goulib.expr.TextVisitor method), 64
- prec_BinOp() (Goulib.expr.LatexVisitor method), 67
- prec_BinOp() (Goulib.expr.TextVisitor method), 64
- prec_UnaryOp() (Goulib.expr.LatexVisitor method), 67
- prec_UnaryOp() (Goulib.expr.TextVisitor method), 64
- predecessors() (Goulib.graph.DiGraph method), 143
- predecessors_iter() (Goulib.graph.DiGraph method), 143
- prevworkday() (Goulib.workdays.WorkCalendar method), 253
- prime_divisors() (in module Goulib.math2), 196
- prime_factors() (in module Goulib.math2), 196
- prime_ktuple() (in module Goulib.math2), 196
- primes() (in module Goulib.math2), 195
- primes_gen() (in module Goulib.math2), 196
- primitive_root_gen() (in module Goulib.math2), 191
- primitive_roots() (in module Goulib.math2), 191
- primitive_triples() (in module Goulib.math2), 195
- project() (Goulib.geom.Point2 method), 77
- project() (Goulib.geom.Vector2 method), 73
- project() (Goulib.geom3d.Vector3 method), 99
- proper_divisors() (in module Goulib.math2), 195
- proper_subset() (Goulib.interval.Interval method), 160

proportional() (in module Goulib.math2), 201
 pure_pil_alpha_to_color_v1() (in module Goulib.image), 154
 pure_pil_alpha_to_color_v2() (in module Goulib.image), 154
 putpixel() (Goulib.image.Image method), 150
 PVA (class in Goulib.motion), 203
 pyramidal() (in module Goulib.math2), 199

Q

quad() (in module Goulib.math2), 191
 quantify() (in module Goulib.itertools2), 169
 quantize() (Goulib.image.Image method), 151
 quantize() (in module Goulib.image), 155
 Quaternion (class in Goulib.geom3d), 104

R

ramp() (in module Goulib.motion), 213
 rand_seq() (in module Goulib.itertools2), 169
 random_prime() (in module Goulib.math2), 196
 randomize() (in module Goulib.image), 155
 range() (Goulib.workdays.WorkCalendar method), 253
 rational_cycle() (in module Goulib.math2), 198
 rational_form() (in module Goulib.math2), 198
 rational_str() (in module Goulib.math2), 198
 Ray2 (class in Goulib.geom), 80
 Ray3 (class in Goulib.geom3d), 101
 read() (Goulib.table.Cell static method), 240
 read_csv() (Goulib.table.Table method), 243
 read_dxf() (Goulib.drawing.Drawing method), 33, 62
 read_element() (Goulib.table.Table method), 243
 read_html() (Goulib.table.Table method), 243
 read_json() (Goulib.table.Table method), 243
 read_pdf() (Goulib.drawing.Drawing method), 33, 62
 read_pdf() (in module Goulib.image), 154
 read_svg() (Goulib.drawing.Drawing method), 33, 62
 read_xls() (Goulib.table.Table method), 243
 Record (class in Goulib.container), 11
 Rect (class in Goulib.drawing), 32, 52
 rectangular_repartition() (in module Goulib.math2), 202
 recurrence() (in module Goulib.itertools2), 167
 recurrence() (in module Goulib.math2), 194
 reflect() (Goulib.geom.Point2 method), 77
 reflect() (Goulib.geom.Vector2 method), 73
 reflect() (Goulib.geom3d.Vector3 method), 99
 remove() (Goulib.container.SortedCollection method), 14
 remove() (Goulib.drawing.BBox method), 37
 remove() (Goulib.drawing.Chain method), 51
 remove() (Goulib.drawing.Drawing method), 61
 remove() (Goulib.drawing.Group method), 45
 remove() (Goulib.drawing.Rect method), 55
 remove() (Goulib.interval.Box method), 165
 remove() (Goulib.interval.Interval method), 161
 remove() (Goulib.interval.Intervals method), 163

remove() (Goulib.optim.BinList method), 218
 remove() (Goulib.stats.Discrete method), 234
 remove() (Goulib.stats.Normal method), 239
 remove() (Goulib.stats.PDF method), 236
 remove() (Goulib.stats.Stats method), 231
 remove() (Goulib.table.Table method), 246
 remove_edge() (Goulib.graph.DiGraph method), 143
 remove_edge() (Goulib.graph.GeoGraph method), 125
 remove_edges_from() (Goulib.graph.DiGraph method), 144
 remove_edges_from() (Goulib.graph.GeoGraph method), 126
 remove_lines_where() (Goulib.table.Table method), 246
 remove_node() (Goulib.graph.DiGraph method), 144
 remove_node() (Goulib.graph.GeoGraph method), 126
 remove_nodes_from() (Goulib.graph.DiGraph method), 144
 remove_nodes_from() (Goulib.graph.GeoGraph method), 126
 removef() (in module Goulib.itertools2), 170
 render() (Goulib.drawing.Chain method), 51
 render() (Goulib.drawing.Drawing method), 62
 render() (Goulib.drawing.Entity method), 29, 38
 render() (Goulib.drawing.Group method), 45
 render() (Goulib.drawing.Instance method), 48
 render() (Goulib.drawing.Rect method), 55
 render() (Goulib.drawing.Spline method), 41
 render() (Goulib.drawing.Text method), 58
 render() (Goulib.expr.Expr method), 64
 render() (Goulib.geom.Arc2 method), 89
 render() (Goulib.geom.Circle method), 86
 render() (Goulib.geom.Ellipse method), 92
 render() (Goulib.geom.Point2 method), 77
 render() (Goulib.geom.Segment2 method), 83
 render() (Goulib.graph.DiGraph method), 144
 render() (Goulib.graph.GeoGraph method), 127
 render() (Goulib.image.Image method), 150
 render() (Goulib.markup.element method), 175
 render() (Goulib.motion.PVA method), 204
 render() (Goulib.motion.Segment method), 205
 render() (Goulib.motion.SegmentPoly method), 209
 render() (Goulib.motion.Segments method), 207
 render() (Goulib.motion.TimeDiagram method), 213
 render() (Goulib.piecewise.Piecewise method), 224
 render() (Goulib.plot.Plot method), 225
 render() (Goulib.polynomial.Polynomial method), 228
 render() (Goulib.stats.Normal method), 239
 render() (Goulib.stats.PDF method), 236
 render() (in module Goulib.plot), 226
 replace() (Goulib.datetime2.date2 method), 21
 replace() (Goulib.datetime2.datetime2 method), 19
 replace() (Goulib.datetime2.time2 method), 23
 replace() (Goulib.image.Image method), 151
 repunit() (in module Goulib.math2), 198

- repunit_gen() (in module Goulib.math2), 198
- reset() (Goulib.statemachine.StateMachine method), 230
- reshape() (in module Goulib.itertools2), 168
- resize() (Goulib.image.Image method), 151
- resolution (Goulib.datetime2.date2 attribute), 22
- resolution (Goulib.datetime2.datetime2 attribute), 19
- resolution (Goulib.datetime2.time2 attribute), 23
- resolution (Goulib.datetime2.timedelta2 attribute), 25
- reverse() (Goulib.drawing.BBox method), 37
- reverse() (Goulib.drawing.Chain method), 51
- reverse() (Goulib.drawing.Drawing method), 62
- reverse() (Goulib.drawing.Group method), 45
- reverse() (Goulib.drawing.Rect method), 55
- reverse() (Goulib.graph.DiGraph method), 145
- reverse() (Goulib.interval.Box method), 165
- reverse() (Goulib.interval.Interval method), 161
- reverse() (Goulib.optim.BinList method), 220
- reverse() (Goulib.table.Table method), 246
- reverse() (in module Goulib.math2), 198
- reversed_sections() (in module Goulib.optim), 220
- rgb (Goulib.colors.Color attribute), 7
- rgb2cmyk() (in module Goulib.colors), 6
- rgb2cmyk() (in module Goulib.image), 158
- rgb2hex() (in module Goulib.colors), 6
- rgb2rgba() (in module Goulib.image), 158
- rint() (in module Goulib.math2), 192
- rotate() (Goulib.geom.Matrix3 method), 97
- rotate_around() (Goulib.geom3d.Vector3 method), 99
- rotate_axis() (Goulib.geom3d.Matrix4 method), 103
- rotate_axis() (Goulib.geom3d.Quaternion method), 106
- rotate_euler() (Goulib.geom3d.Matrix4 method), 103
- rotate_euler() (Goulib.geom3d.Quaternion method), 106
- rotate_matrix() (Goulib.geom3d.Quaternion method), 106
- rotate_triple_axis() (Goulib.geom3d.Matrix4 method), 103
- rotatex() (Goulib.geom3d.Matrix4 method), 103
- rotatey() (Goulib.geom3d.Matrix4 method), 103
- rotatez() (Goulib.geom3d.Matrix4 method), 103
- Row (class in Goulib.table), 241
- rowasdict() (Goulib.table.Table method), 246
- run() (Goulib.statemachine.StateMachine method), 230
- run() (Goulib.tests.TestCase method), 252
- runmodule() (in module Goulib.tests), 253
- runtests() (in module Goulib.tests), 253
- russell (class in Goulib.markup), 180
- S**
- SAT (Goulib.workdays.WorkCalendar attribute), 254
- sat() (in module Goulib.math2), 194
- save() (Goulib.container.Sequence method), 16
- save() (Goulib.drawing.Chain method), 52
- save() (Goulib.drawing.Drawing method), 33, 62
- save() (Goulib.drawing.Entity method), 39
- save() (Goulib.drawing.Group method), 45
- save() (Goulib.drawing.Instance method), 48
- save() (Goulib.drawing.Rect method), 55
- save() (Goulib.drawing.Spline method), 41
- save() (Goulib.drawing.Text method), 58
- save() (Goulib.expr.Expr method), 64
- save() (Goulib.geom.Arc2 method), 89
- save() (Goulib.geom.Circle method), 86
- save() (Goulib.geom.Ellipse method), 92
- save() (Goulib.geom.Point2 method), 77
- save() (Goulib.geom.Segment2 method), 83
- save() (Goulib.graph.DiGraph method), 145
- save() (Goulib.graph.GeoGraph method), 127
- save() (Goulib.image.Image method), 150
- save() (Goulib.motion.PVA method), 204
- save() (Goulib.motion.Segment method), 205
- save() (Goulib.motion.SegmentPoly method), 209
- save() (Goulib.motion.Segments method), 207
- save() (Goulib.motion.TimeDiagram method), 213
- save() (Goulib.piecewise.Piecewise method), 224
- save() (Goulib.plot.Plot method), 225
- save() (Goulib.polynomial.Polynomial method), 228
- save() (Goulib.stats.Normal method), 239
- save() (Goulib.stats.PDF method), 237
- save() (Goulib.table.Table method), 243
- save() (in module Goulib.plot), 226
- saveAsCsv() (Goulib.motion.TimeDiagram method), 212
- scale() (Goulib.geom.Matrix3 method), 96
- scale() (Goulib.geom3d.Matrix4 method), 103
- scale() (Goulib.image.Image method), 152
- score_population() (Goulib.optim.DifferentialEvolution method), 221
- scripts() (Goulib.markup.page method), 178
- second (Goulib.datetime2.datetime2 attribute), 19
- second (Goulib.datetime2.time2 attribute), 23
- seconds (Goulib.datetime2.timedelta2 attribute), 25
- Segment (class in Goulib.motion), 204
- Segment2 (class in Goulib.geom), 81
- Segment2ndDegree() (in module Goulib.motion), 213
- Segment3 (class in Goulib.geom3d), 101
- Segment4thDegree() (in module Goulib.motion), 214
- SegmentPoly (class in Goulib.motion), 207
- Segments (class in Goulib.motion), 206
- SegmentsTrapezoidalSpeed() (in module Goulib.motion), 214
- selfloop_edges() (Goulib.graph.DiGraph method), 145
- selfloop_edges() (Goulib.graph.GeoGraph method), 127
- separation() (Goulib.interval.Interval method), 159
- Sequence (class in Goulib.container), 15
- set() (Goulib.statemachine.TimeMarker method), 230
- set() (Goulib.table.Table method), 244
- set_dimension() (Goulib.graph.index.Property method), 107, 109
- setattr() (Goulib.drawing.Chain method), 52

- setattr() (Goulib.drawing.Drawing method), 62
 setattr() (Goulib.drawing.Entity method), 28, 37
 setattr() (Goulib.drawing.Group method), 45
 setattr() (Goulib.drawing.Instance method), 48
 setattr() (Goulib.drawing.Rect method), 55
 setattr() (Goulib.drawing.Spline method), 41
 setattr() (Goulib.drawing.Text method), 58
 setattr() (Goulib.geom.Arc2 method), 89
 setattr() (Goulib.geom.Circle method), 86
 setattr() (Goulib.geom.Ellipse method), 92
 setattr() (Goulib.geom.Point2 method), 77
 setattr() (Goulib.geom.Segment2 method), 83
 setcol() (Goulib.table.Table method), 244
 setdefault() (Goulib.colors.Palette method), 10
 setdefault() (Goulib.container.Record method), 12
 setdefault() (Goulib.graph.index.Index method), 112
 setdefault() (Goulib.optim.BinDict method), 217
 setlog() (in module Goulib.tests), 253
 setOutput() (Goulib.statemachine.Simulation method), 229
 setpalette() (Goulib.image.Image method), 150
 sets_dist() (in module Goulib.math2), 194
 sets_levenshtein() (in module Goulib.math2), 194
 setUp() (Goulib.tests.TestCase method), 252
 setUpClass() (Goulib.tests.TestCase method), 252
 setworktime() (Goulib.workdays.WorkCalendar method), 253
 sexy_prime_quadruplets() (in module Goulib.math2), 197
 sexy_prime_triplets() (in module Goulib.math2), 197
 sexy_primes() (in module Goulib.math2), 197
 shape (Goulib.image.Image attribute), 150
 shape() (in module Goulib.itertools2), 167
 shift() (Goulib.image.Image method), 153
 shortDescription() (Goulib.tests.TestCase method), 252
 shortest_path() (Goulib.graph.DiGraph method), 145
 shortest_path() (Goulib.graph.GeoGraph method), 127
 shuffle() (in module Goulib.itertools2), 169
 sieve() (in module Goulib.math2), 195
 sigma (Goulib.stats.Discrete attribute), 234
 sigma (Goulib.stats.Normal attribute), 237
 sigma (Goulib.stats.PDF attribute), 237
 sigma (Goulib.stats.Stats attribute), 232
 sign() (in module Goulib.math2), 191
 Simulation (class in Goulib.statemachine), 229
 sin_over_x() (in module Goulib.math2), 201
 singleton() (Goulib.interval.Interval method), 160
 size (Goulib.image.Image attribute), 150
 size (Goulib.interval.Box attribute), 163
 size (Goulib.interval.Interval attribute), 159
 size() (Goulib.drawing.BBox method), 28, 34
 size() (Goulib.graph.DiGraph method), 145
 size() (Goulib.graph.GeoGraph method), 127
 size() (Goulib.optim.BinDict method), 217
 size() (Goulib.optim.BinList method), 220
 skipTest() (Goulib.tests.TestCase method), 252
 slerp() (in module Goulib.math2), 201
 sort() (Goulib.container.Sequence method), 17
 sort() (Goulib.drawing.BBox method), 37
 sort() (Goulib.drawing.Chain method), 52
 sort() (Goulib.drawing.Drawing method), 62
 sort() (Goulib.drawing.Group method), 45
 sort() (Goulib.drawing.Rect method), 55
 sort() (Goulib.interval.Box method), 166
 sort() (Goulib.interval.Interval method), 161
 sort() (Goulib.optim.BinList method), 220
 sort() (Goulib.table.Table method), 246
 sort_indexes() (in module Goulib.itertools2), 169
 sorted() (Goulib.colors.Palette method), 9
 sorted_iterable() (in module Goulib.itertools2), 173
 SortedCollection (class in Goulib.container), 12
 SortingError, 171
 Sphere (class in Goulib.geom3d), 101
 Spherical() (in module Goulib.geom3d), 101
 Spline (class in Goulib.drawing), 29, 39
 split() (Goulib.image.Image method), 150
 split() (in module Goulib.itertools2), 170
 sqrt() (in module Goulib.math2), 192
 square() (in module Goulib.math2), 200
 start (Goulib.drawing.BBox attribute), 37
 start (Goulib.drawing.Chain attribute), 31, 48
 start (Goulib.drawing.Drawing attribute), 62
 start (Goulib.drawing.Entity attribute), 28, 37
 start (Goulib.drawing.Group attribute), 45
 start (Goulib.drawing.Instance attribute), 48
 start (Goulib.drawing.Rect attribute), 55
 start (Goulib.drawing.Spline attribute), 30, 39
 start (Goulib.drawing.Text attribute), 58
 start (Goulib.geom.Arc2 attribute), 89
 start (Goulib.geom.Circle attribute), 86
 start (Goulib.geom.Ellipse attribute), 92
 start (Goulib.geom.Point2 attribute), 77
 start (Goulib.geom.Segment2 attribute), 83
 start (Goulib.interval.Box attribute), 163
 start (Goulib.interval.Interval attribute), 159
 start (Goulib.workdays.WorkCalendar attribute), 253
 start() (Goulib.motion.Segment method), 204
 start() (Goulib.motion.SegmentPoly method), 209
 start() (Goulib.motion.Segments method), 206
 startAcc() (Goulib.motion.Segment method), 204
 startAcc() (Goulib.motion.SegmentPoly method), 209
 startAcc() (Goulib.motion.Segments method), 207
 startJerk() (Goulib.motion.Segment method), 204
 startJerk() (Goulib.motion.SegmentPoly method), 209
 startJerk() (Goulib.motion.Segments method), 207
 startPos() (Goulib.motion.Segment method), 204
 startPos() (Goulib.motion.SegmentPoly method), 209
 startPos() (Goulib.motion.Segments method), 207
 startSpeed() (Goulib.motion.Segment method), 204

startSpeed() (Goulib.motion.SegmentPoly method), 209
startSpeed() (Goulib.motion.Segments method), 207
startTime() (Goulib.motion.Segment method), 204
startTime() (Goulib.motion.SegmentPoly method), 209
startTime() (Goulib.motion.Segments method), 207
state() (Goulib.statemachine.StateDiagram method), 229
StateChangeLog (class in Goulib.statemachine), 229
StateDiagram (class in Goulib.statemachine), 229
StateMachine (class in Goulib.statemachine), 230
Stats (class in Goulib.stats), 231
stats() (Goulib.graph.DiGraph method), 146
stats() (Goulib.graph.GeoGraph method), 128
stats() (in module Goulib.stats), 231
stddev (Goulib.stats.Discrete attribute), 234
stddev (Goulib.stats.Normal attribute), 239
stddev (Goulib.stats.PDF attribute), 237
stddev (Goulib.stats.Stats attribute), 232
stddev() (in module Goulib.stats), 231
str() (Goulib.colors.Color method), 7
str_base() (in module Goulib.math2), 198
strftime() (Goulib.datetime2.date2 method), 22
strftime() (Goulib.datetime2.datetime2 method), 19
strftime() (Goulib.datetime2.time2 method), 23
strtimedelta() (in module Goulib.datetime2), 26
strptime() (Goulib.datetime2.datetime2 method), 19
style_dict2str() (in module Goulib.markup), 175
style_str2dict() (in module Goulib.markup), 175
sub() (Goulib.image.Image method), 153
sub() (in module Goulib.polynomial), 228
subdict() (in module Goulib.itertools2), 171
subgraph() (Goulib.graph.DiGraph method), 146
subgraph() (Goulib.graph.GeoGraph method), 128
subset() (Goulib.interval.Interval method), 160
subTest() (Goulib.tests.TestCase method), 252
successors() (Goulib.graph.DiGraph method), 146
successors_iter() (Goulib.graph.DiGraph method), 146
sum (Goulib.stats.Discrete attribute), 234
sum (Goulib.stats.Normal attribute), 239
sum (Goulib.stats.PDF attribute), 237
sum (Goulib.stats.Stats attribute), 232
sum1 (Goulib.stats.Discrete attribute), 235
sum1 (Goulib.stats.Normal attribute), 239
sum1 (Goulib.stats.PDF attribute), 237
sum1 (Goulib.stats.Stats attribute), 232
sum2 (Goulib.stats.Discrete attribute), 235
sum2 (Goulib.stats.Normal attribute), 239
sum2 (Goulib.stats.PDF attribute), 237
sum2 (Goulib.stats.Stats attribute), 232
sum_of_cubes() (in module Goulib.math2), 199
sum_of_squares() (in module Goulib.math2), 198
SUN (Goulib.workdays.WorkCalendar attribute), 254
svg() (Goulib.drawing.Chain method), 52
svg() (Goulib.drawing.Drawing method), 62
svg() (Goulib.drawing.Entity method), 39
svg() (Goulib.drawing.Group method), 45
svg() (Goulib.drawing.Instance method), 48
svg() (Goulib.drawing.Rect method), 55
svg() (Goulib.drawing.Spline method), 41
svg() (Goulib.drawing.Text method), 58
svg() (Goulib.expr.Expr method), 64
svg() (Goulib.geom.Arc2 method), 89
svg() (Goulib.geom.Circle method), 86
svg() (Goulib.geom.Ellipse method), 92
svg() (Goulib.geom.Point2 method), 77
svg() (Goulib.geom.Segment2 method), 83
svg() (Goulib.graph.DiGraph method), 146
svg() (Goulib.graph.GeoGraph method), 128
svg() (Goulib.image.Image method), 154
svg() (Goulib.motion.PVA method), 204
svg() (Goulib.motion.Segment method), 205
svg() (Goulib.motion.SegmentPoly method), 209
svg() (Goulib.motion.Segments method), 207
svg() (Goulib.motion.TimeDiagram method), 213
svg() (Goulib.piecewise.Piecewise method), 224
svg() (Goulib.plot.Plot method), 225
svg() (Goulib.polynomial.Polynomial method), 228
svg() (Goulib.stats.Normal method), 239
svg() (Goulib.stats.PDF method), 237
svg() (in module Goulib.plot), 226
swap() (Goulib.drawing.Chain method), 52
swap() (Goulib.drawing.Drawing method), 62
swap() (Goulib.drawing.Group method), 30, 42
swap() (Goulib.drawing.Rect method), 55
swap() (Goulib.drawing.Spline method), 30, 39
swap() (Goulib.geom.Arc2 method), 87
swap() (Goulib.geom.Circle method), 84
swap() (Goulib.geom.Ellipse method), 92
swap() (Goulib.geom.Segment2 method), 81
swap() (Goulib.geom3d.Segment3 method), 101
swap() (in module Goulib.itertools2), 167
swapped_cities() (in module Goulib.optim), 220

T

Table (class in Goulib.table), 242
tag() (in module Goulib.markup), 175
tails() (in module Goulib.itertools2), 168
take() (in module Goulib.itertools2), 166
takeevery() (in module Goulib.itertools2), 166
takenth() (in module Goulib.itertools2), 169
tangent() (Goulib.drawing.Chain method), 52
tangent() (Goulib.drawing.Drawing method), 62
tangent() (Goulib.drawing.Group method), 45
tangent() (Goulib.drawing.Instance method), 48
tangent() (Goulib.drawing.Rect method), 56
tangent() (Goulib.drawing.Spline method), 41
tangent() (Goulib.geom.Arc2 method), 87
tangent() (Goulib.geom.Circle method), 84
tangent() (Goulib.geom.Ellipse method), 92

- tangent() (Goulib.geom.Geometry method), 68
 tangent() (Goulib.geom.Line2 method), 79
 tangent() (Goulib.geom.Point2 method), 77
 tangent() (Goulib.geom.Ray2 method), 81
 tangent() (Goulib.geom.Segment2 method), 83
 tdround() (in module Goulib.datetime2), 26
 tearDown() (Goulib.tests.TestCase method), 252
 tearDownClass() (Goulib.tests.TestCase method), 253
 tee() (in module Goulib.itertools2), 167
 TestCase (class in Goulib.tests), 247
 tetrahedral() (in module Goulib.math2), 198
 Text (class in Goulib.drawing), 32, 56
 TextVisitor (class in Goulib.expr), 64
 threshold() (Goulib.image.Image method), 151
 THU (Goulib.workdays.WorkCalendar attribute), 254
 time() (Goulib.datetime2.datetime2 method), 19
 time2 (class in Goulib.datetime2), 22
 time_add() (in module Goulib.datetime2), 26
 time_intersect() (in module Goulib.datetime2), 27
 time_sub() (in module Goulib.datetime2), 26
 timedelta2 (class in Goulib.datetime2), 23
 timedelta_div() (in module Goulib.datetime2), 26
 timedelta_mul() (in module Goulib.datetime2), 26
 timedelta_sum() (in module Goulib.datetime2), 26
 timedeltaf() (in module Goulib.datetime2), 26
 TimeDiagram (class in Goulib.motion), 211
 timef() (in module Goulib.datetime2), 26
 TimeMarker (class in Goulib.statemachine), 230
 timeout() (in module Goulib.decorators), 27
 timestamp() (Goulib.datetime2.datetime2 method), 19
 timetuple() (Goulib.datetime2.date2 method), 22
 timetuple() (Goulib.datetime2.datetime2 method), 19
 timetz() (Goulib.datetime2.datetime2 method), 19
 timeWhenPosBiggerThan() (Goulib.motion.Segment method), 204
 timeWhenPosBiggerThan() (Goulib.motion.SegmentPoly method), 209
 timeWhenPosBiggerThan() (Goulib.motion.Segments method), 207
 to_date() (Goulib.table.Table method), 246
 to_datetime() (Goulib.table.Table method), 246
 to_directed() (Goulib.graph.DiGraph method), 146
 to_directed() (Goulib.graph.GeoGraph method), 128
 to_drawing() (in module Goulib.graph), 108, 148
 to_dxf() (Goulib.drawing.Chain method), 32, 49
 to_dxf() (Goulib.drawing.Drawing method), 62
 to_dxf() (Goulib.drawing.Entity method), 29, 37
 to_dxf() (Goulib.drawing.Group method), 45
 to_dxf() (Goulib.drawing.Instance method), 48
 to_dxf() (Goulib.drawing.Rect method), 56
 to_dxf() (Goulib.drawing.Spline method), 41
 to_dxf() (Goulib.drawing.Text method), 33, 56
 to_dxf() (Goulib.geom.Arc2 method), 89
 to_dxf() (Goulib.geom.Circle method), 86
 to_dxf() (Goulib.geom.Ellipse method), 92
 to_dxf() (Goulib.geom.Point2 method), 77
 to_dxf() (Goulib.geom.Segment2 method), 83
 to_json() (in module Goulib.graph), 108, 148
 to_networkx_graph() (in module Goulib.graph), 107, 114
 to_time() (Goulib.table.Table method), 246
 to_timedelta() (Goulib.table.Table method), 246
 to_undirected() (Goulib.graph.DiGraph method), 147
 to_undirected() (Goulib.graph.GeoGraph method), 129
 today() (Goulib.datetime2.date2 method), 22
 today() (Goulib.datetime2.datetime2 method), 19
 tol (Goulib.graph.DiGraph attribute), 147
 tol (Goulib.graph.GeoGraph attribute), 129
 TooLateLog (class in Goulib.statemachine), 230
 toordinal() (Goulib.datetime2.date2 method), 22
 toordinal() (Goulib.datetime2.datetime2 method), 20
 tostring() (in module Goulib.polynomial), 229
 total() (Goulib.table.Table method), 246
 total_seconds() (Goulib.datetime2.timedelta2 method), 25
 totient() (in module Goulib.math2), 196
 tour_length() (in module Goulib.optim), 220
 trailing_zeros() (in module Goulib.math2), 197
 trans() (Goulib.drawing.BBox method), 28, 35
 Trans() (in module Goulib.drawing), 27, 34
 transform() (Goulib.geom3d.Matrix4 method), 103
 translate() (Goulib.geom.Matrix3 method), 97
 translate() (Goulib.geom3d.Matrix4 method), 103
 transpose() (Goulib.geom.Matrix3 method), 98
 transpose() (Goulib.geom3d.Matrix4 method), 103
 transpose() (Goulib.table.Table method), 244
 transpose() (in module Goulib.math2), 193
 transposed() (Goulib.geom.Matrix3 method), 98
 transposed() (Goulib.geom3d.Matrix4 method), 103
 trapeze() (in module Goulib.motion), 213
 triangle() (in module Goulib.math2), 199
 triangular() (in module Goulib.math2), 199
 triangular_repartition() (in module Goulib.math2), 201
 triples() (in module Goulib.math2), 195
 tsp() (in module Goulib.optim), 220
 TUE (Goulib.workdays.WorkCalendar attribute), 254
 twin_primes() (in module Goulib.math2), 197
 tzinfo (Goulib.datetime2.datetime2 attribute), 20
 tzinfo (Goulib.datetime2.time2 attribute), 23
 tzname() (Goulib.datetime2.datetime2 method), 20
 tzname() (Goulib.datetime2.time2 method), 23
- ## U
- unescape() (in module Goulib.markup), 179
 unique() (Goulib.container.Sequence method), 17
 unique() (in module Goulib.itertools2), 168
 update() (Goulib.colors.Palette method), 8
 update() (Goulib.container.Record method), 12
 update() (Goulib.graph.index.Index method), 112

update() (Goulib.motion.Segments method), 206
update() (Goulib.optim.BinDict method), 217
utcfromtimestamp() (Goulib.datetime2.datetime2 method), 20
utcnow() (Goulib.datetime2.datetime2 method), 20
utcoffset() (Goulib.datetime2.datetime2 method), 20
utcoffset() (Goulib.datetime2.time2 method), 23
utctimetuple() (Goulib.datetime2.datetime2 method), 20

V

values() (Goulib.colors.Palette method), 10
values() (Goulib.container.Record method), 12
values() (Goulib.graph.index.Index method), 112
values() (Goulib.optim.BinDict method), 218
var (Goulib.stats.Discrete attribute), 235
var (Goulib.stats.Normal attribute), 239
var (Goulib.stats.PDF attribute), 237
var (Goulib.stats.Stats attribute), 232
var() (in module Goulib.stats), 231
varDict() (Goulib.motion.Actuator method), 211
variance (Goulib.stats.Discrete attribute), 235
variance (Goulib.stats.Normal attribute), 239
variance (Goulib.stats.PDF attribute), 237
variance (Goulib.stats.Stats attribute), 232
variance() (in module Goulib.stats), 231
varNames() (Goulib.motion.Actuator method), 210
varRowUnits() (Goulib.motion.Actuator method), 210
vecadd() (in module Goulib.math2), 193
veccompare() (in module Goulib.math2), 194
vecdiv() (in module Goulib.math2), 194
vecmul() (in module Goulib.math2), 194
vecneg() (in module Goulib.math2), 194
vecsub() (in module Goulib.math2), 194
Vector2 (class in Goulib.geom), 69
Vector3 (class in Goulib.geom3d), 98
vecunit() (in module Goulib.math2), 194
visit() (Goulib.expr.LatexVisitor method), 67
visit() (Goulib.expr.TextVisitor method), 66
visit_BinOp() (Goulib.expr.LatexVisitor method), 67
visit_BinOp() (Goulib.expr.TextVisitor method), 65
visit_Call() (Goulib.expr.LatexVisitor method), 67
visit_Call() (Goulib.expr.TextVisitor method), 65
visit_Compare() (Goulib.expr.LatexVisitor method), 67
visit_Compare() (Goulib.expr.TextVisitor method), 66
visit_Name() (Goulib.expr.LatexVisitor method), 67
visit_Name() (Goulib.expr.TextVisitor method), 65
visit_NameConstant() (Goulib.expr.LatexVisitor method), 67
visit_NameConstant() (Goulib.expr.TextVisitor method), 65
visit_Num() (Goulib.expr.LatexVisitor method), 67
visit_Num() (Goulib.expr.TextVisitor method), 66
visit_UnaryOp() (Goulib.expr.LatexVisitor method), 67
visit_UnaryOp() (Goulib.expr.TextVisitor method), 65

W

wait() (Goulib.statemachine.StateMachine method), 230
WaitLog (class in Goulib.statemachine), 230
WED (Goulib.workdays.WorkCalendar attribute), 254
weekday() (Goulib.datetime2.date2 method), 22
weekday() (Goulib.datetime2.datetime2 method), 20
width (Goulib.drawing.BBox attribute), 28, 34
with_traceback() (Goulib.itertools2.SortingError method), 173
with_traceback() (Goulib.markup.ArgumentError method), 186
with_traceback() (Goulib.markup.ClosingError method), 184
with_traceback() (Goulib.markup.CustomizationError method), 191
with_traceback() (Goulib.markup.DeprecationError method), 188
with_traceback() (Goulib.markup.InvalidElementError method), 187
with_traceback() (Goulib.markup.MarkupError method), 182
with_traceback() (Goulib.markup.ModeError method), 189
with_traceback() (Goulib.markup.OpeningError method), 185
WorkCalendar (class in Goulib.workdays), 253
workdatetime() (Goulib.workdays.WorkCalendar method), 254
workday() (Goulib.workdays.WorkCalendar method), 253
workday() (in module Goulib.workdays), 255
workdays() (Goulib.workdays.WorkCalendar method), 253
worktime() (Goulib.workdays.WorkCalendar method), 254
write_csv() (Goulib.table.Table method), 243
write_dot() (in module Goulib.graph), 108, 148
write_dxf() (in module Goulib.graph), 108, 148
write_json() (in module Goulib.graph), 108, 148
write_xlsx() (Goulib.table.Table method), 243

X

xgcd() (in module Goulib.math2), 191
xmax (Goulib.drawing.BBox attribute), 28, 34
xmed (Goulib.drawing.BBox attribute), 28, 34
xmin (Goulib.drawing.BBox attribute), 28, 34
xy (Goulib.drawing.Spline attribute), 30, 39
xy (Goulib.geom.Point2 attribute), 78
xy (Goulib.geom.Vector2 attribute), 72
xyY (Goulib.colors.Color attribute), 7
xyy2xyz() (in module Goulib.colors), 6
xyz (Goulib.colors.Color attribute), 7
xyz (Goulib.geom3d.Vector3 attribute), 98
xyz2xyy() (in module Goulib.colors), 6

Y

year (Goulib.datetime2.date2 attribute), 22
year (Goulib.datetime2.datetime2 attribute), 20
ymax (Goulib.drawing.BBox attribute), 28, 34
ymed (Goulib.drawing.BBox attribute), 28, 34
ymin (Goulib.drawing.BBox attribute), 28, 34

Z

zeros() (in module Goulib.math2), 193