dk Documentation

Release 0.7.9

Bjorn Pettersen

Jun 15, 2017

Contents

Installation:

```
pip install dk
```

Contents:

CHAPTER 1

Project information

The project home page is at:

http://thebjorn.github.io/dk/

The github directory is at:

https://github.com/thebjorn/dk

Report any issues at:

https://github.com/thebjorn/dk/issues

CHAPTER 2

dk

dk package

Subpackages

dk.collections package

Submodules

dk.collections.OrderedSet module

Ordered Set. Items can only be added once, further additions have no effect. The iterator iterates over the items in insertion order.

class dk.collections.OrderedSet.**oset**(*iterable=()*)
Bases: set

Ordered Set. Items can only be added once, further additions have no effect. The iterator iterates over the items in insertion order.

add(*item*)

dk.collections.invdict module

Inversable dictionary.

class dk.collections.invdict.**invdict**
Bases: dict

Inversable dict:

```
>>> -invdict({'key': 'val'}) == {'val': 'key'}
```

dk.collections.mmap module

class dk.collections.mmap.**mmap**(**attrs*)
Bases: list

Multi Map class, ie. a key/value collection where each key can occur multiple times. Implemented as a list of key/value tuples.

add(*key*, *val*)

append(*kv*)

dk.collections.pset module

Mapping classes.

class dk.collections.pset.**defset**(*defval*)
Bases: *dk.collections.pset.pset*

pset with default value.

class dk.collections.pset.**keyval**(*key*, *val*)
Bases: tuple

key
Alias for field number 0

val
Alias for field number 1

class dk.collections.pset.**pset**(*items=()*, **attrs*)
Bases: dict

This code is placed in the Public Domain, or released under the wtfpl (http://sam.zoy.org/wtfpl/COPYING) wherever PD is problematic.

Property Set class. A property set is an object where values are attached to attributes, but can still be iterated over as key/value pairs. The order of assignment is maintained during iteration. Only one value allowed per key.

```
>>> x = pset()
>>> x.a = 42
>>> x.b = 'foo'
>>> x.a = 314
>>> x
pset(a=314, b='foo')
```

apply(*fn*)
Apply function fn to all values in self.

items()

keys()

pprint(*indent=0*, *tab=' '*, *seen=None*)
Pretty print the pset, indented.

remove(*key*)
Remove key from client vars.

values()

class dk.collections.pset.**record**(*items=()*, ***attrs*)
Bases: *dk.collections.pset.pset*

A property set with commit, rollback, and encoding translation.

changed()
Return list of fields that have changed since last commit.

commit()
Copy current state to self._history

decode(*encoding*)
Decode using encoding.

encode(*encoding*)
Encode using encoding.

fields
Verbose name of all fields.

rollback()
Copy snapshot from self._history into self.

strvals(*empty=''*, *none='NULL'*, *encoding='u8'*)
Return a list of all values, formatted for human consumption.

trans(*source='iso-8859-1'*, *dest='utf-8'*)
Translate encoding.

dk.collections.pset.**test_pset**()
Unit tests...

```
>>> request = pset(REQUEST={}, META={}, path='/', user=None, session={}, method=
↪'GET',
...                 COOKIES={}, LANGUAGE_CODE='no')
>>> p = page(request)
>>> p.forms = 'fruit'
>>> p.forms.foo = 'bar'
>>> print p.forms.foo
bar
>>> p.forms.fob = 'baz'
>>> print p.forms.fob
baz
>>> x = pset()
>>> x.a
Traceback (most recent call last):
  ...
AttributeError: a
>>> y = pset(a=1, b=2, c=3)
>>> y.a
1
>>> y.b
2
>>> y.c
3
>>> z = pset()
>>> z.a = 1
>>> z.b = 2
>>> z.c = 3
>>> z[1]
2
```

```
>>> z
pset(a=1, b=2, c=3)
>>> class Point(pset): pass
>>> p = Point(x=11, y=22)
>>> p
Point(y=22, x=11)
```

dk.collections.pset.**xmlrepr**(*v*, *toplevel=False*)
Return v as xml tag-soup.

dk.collections.sdict module

class dk.collections.sdict.**sdict**(**attrs*)
Bases: dict

Sorted Dictionary class. Iterating over the sdict will give back the key/value pairs in order of insertion. A key can only be in a sdict once.

keys()

values()

dk.collections.xmlrec module

dk.collections.xmlrec.**Boolean**(*s*)

dk.collections.xmlrec.**Date**(*s*)

dk.collections.xmlrec.**Datetime**(*s*)

dk.collections.xmlrec.**NOK**(*s*)

class dk.collections.xmlrec.**xmlrec**(*soup*, **types*)
Bases: *dk.collections.pset.pset*

convert = {'date': <function Date>, 'int': <type 'int'>, 'NOK': <function NOK>, 'bool': <function Boolean>, 'datetime

Module contents

Abstract Data Types – mostly record types with different semantics.

dk.html package

Submodules

dk.html.css module

class dk.html.css.**css**(**attrs*)
Bases: *dk.collections.pset.pset*

dk.html.html module

HTML helper file.

class dk.html.html.**a**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**abbr**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**acronym**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**address**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**applet**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**area**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**b**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**base**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**bdo**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**big**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**blockquote**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**body**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**bokmaal**
alias of *link*

class dk.html.html.**br**(***kw*)
Bases: *dk.html.html.xtag*

class dk.html.html.**bsefont**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**button**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**caption**(**content*, ***kw*)
Bases: *dk.html.html.dtag*

class dk.html.html.**center**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**checkbox_input**
alias of *input*

class dk.html.html.**cite**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**closetag**(*tag_name*, **content*, ***kw*)
Bases: *dk.html.html.tag*

flatten(*lst=None*)

class dk.html.html.**code**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**col**(***kw*)
Bases: *dk.html.html.xtag*

class dk.html.html.**colgroup**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**color**

aqua = '"#00FFFF"'

black = '"#000000"'

blue = '"#0000FF"'

fuchsia = '"#FF00FF"'

gray = '"#808080"'

green = '"#008000"'

lime = '"#00FF00"'

maroon = '"#800000"'

navy = '"#000080"'

olive = '"#808000"'

purple = '"#800080"'

red = '"#FF0000"'

silver = '"#COCOCO"'

teal = '"#008080"'

white = '"#FFFFFF"'

yellow = '"#FFFF00"'

class dk.html.html.**dd**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**del_**
alias of del

class dk.html.html.**dfn**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**dir_**
alias of dir

class dk.html.html.**div**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**dl**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**doctype**
alias of DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"

dk.html.html.**doctype401frameset**
alias of DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd"

dk.html.html.**doctype401strict**
alias of DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"

dk.html.html.**doctype401transitional**
alias of DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"

class dk.html.html.**dt**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**dtag**(*tag_name*, **content*, ***kw*)
Bases: *dk.html.html.tag*

d(issappearing)tag: if the content is empty, i.e. self.content == (‘’,) this tag doesn’t output anything at all. Useful for legends, table captions, etc.

class dk.html.html.**em**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**english**
alias of *link*

dk.html.html.**escape**(*strval*, *enc=None*)
Convert string s (potentially unicode) to a ascii string with entitydefs like ø æ etc.

dk.html.html.**escape_char**(*unichar*)

dk.html.html.**escaped_array**(*strval*)
Convert unicode string to list of ascii characters or entitydefs like ø etc.

class dk.html.html.**fieldset**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**font**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**form**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**frame**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**frameset**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**h1**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**h2**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**h3**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**h4**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**h5**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**h6**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**head**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**hidden_input**
alias of *input*

class dk.html.html.**hr**(***kw*)
Bases: *dk.html.html.xtag*

class dk.html.html.**html**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**i**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**iframe**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**img**(***kw*)
Bases: *dk.html.html.xtag*

class dk.html.html.**input**(***kw*)
Bases: *dk.html.html.xtag*

class dk.html.html.**ins**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**kbd**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**label**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**legend**(**content*, ***kw*)
Bases: *dk.html.html.dtag*

class dk.html.html.**li**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**lines**(**content*)
Bases: *dk.html.html.text*

like text, except each item in content is separated with a
 tag.

flatten()

class dk.html.html.**link**(***kw*)
Bases: *dk.html.html.xtag*

class dk.html.html.**map**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**menu**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**meta**(**kw*)
Bases: *dk.html.html.xtag*

dk.html.html.**mkdtag**(*name*, ***attrs*)

dk.html.html.**mkstag**(*name*)

dk.html.html.**mktag**(*name*, *_parent=<class 'dk.html.html.tag'>*, *_nlafter=False*, ***attrs*)

dk.html.html.**mkxtag**(*name*, ***attrs*)

dk.html.html.**next**
alias of *link*

class dk.html.html.**nobr**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**noframes**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**norm_attr_name**(*attr*)

dk.html.html.**normalize**(*v*)
returns a stringified unicode version of v

dk.html.html.**norsk**
alias of *link*

class dk.html.html.**noscript**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**nynorsk**
alias of *link*

dk.html.html.**object_**
alias of object

class dk.html.html.**ol**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**opentag**(*tag_name*, **content*, ***kw*)
Bases: *dk.html.html.tag*

flatten(*lst=None*)

class dk.html.html.**optgroup**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**option**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**p**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**page**(*xtitle*, *abody*)
Shortcut to get a page up quickly.

class dk.html.html.**param**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**password_input**
alias of *input*

dk.html.html.**pdf**
alias of *link*

dk.html.html.**plain_attribute**(*strval*, *legal='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ012345*
._:')

class dk.html.html.**pre**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**prev**
alias of *link*

class dk.html.html.**q**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**quote**(*strval*)

dk.html.html.**quote_if_needed**(*strval*)

dk.html.html.**quote_smart**(*strval*)

dk.html.html.**quote_xhtml**(*v*)

dk.html.html.**radio_input**
alias of *input*

dk.html.html.**rawstr2unicode**(*strval*)

class dk.html.html.**s**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**samp**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**script**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**select**(*options*, *selected=None*, ***kw*)
Bases: *dk.html.html.tag*

options

selected

values

class dk.html.html.**small**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**span**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**sqlresult**(*res*, *desc=None*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**stag**(*tag_name*, ***kw*)
Bases: *dk.html.html.xtag*

s(ingle)tag

dk.html.html.**start**
alias of *link*

class dk.html.html.**strike**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**strong**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**style**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**stylesheet**
alias of *link*

class dk.html.html.**sub**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**submit_button**
alias of *input*

class dk.html.html.**sup**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**table**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**tabledesc**(**cols*)
Bases: object

class dk.html.html.**tag**(*tag_name*, **content*, ***kw*)
Bases: *dk.html.html.xtag*

Regular tag: outputs an open tag with attributes, followed by its contents, followed by a closing tag.

Attributes can be set either as keyword arguments in the constructor or by assigning to attributes of the object.

Content can be any combination of items, iterables, and generators:

```
>>> table(tr(td(i) for i in range(5)), tr(td(i**i) for i in range(5)))
<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr><td>1</td><td>1</td><td>4</td><td>27</td><td>256</td></tr>
</table>
```

NB: Attributes that conflict with Python keywords have an underline appended, e.g.: mytag.class_ = ...

close_tag()

flatten(*lst=None*)

open_tag()

xcontent

class dk.html.html.**tbody**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**td**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**text**(**content*)
Bases: *dk.html.html.tag*

text tag: outputs its contents without any tags around it. Useful for grouping at the top level.

flatten()

dk.html.html.**text_input**
alias of *input*

class dk.html.html.**textarea**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**tfoot**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**th**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**thead**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**title**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**tr**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**tt**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**u**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**u8escape**(*strval*)

class dk.html.html.**ul**(**content*, ***kw*)
Bases: *dk.html.html.tag*

dk.html.html.**unescape**(*txt*)
Convert text containing entitydefs into Unicode.

class dk.html.html.**var**(**content*, ***kw*)
Bases: *dk.html.html.tag*

class dk.html.html.**xtag**(*tag_name*, ***kw*)
Bases: object

x(ml-style)tag: a tag without content or a closing tag.

E.g.
 would be xtag('br')

[2009-03-11] w3 validator complains that 4.01 loose should not use <foo /> but <foo>.

attributes()
return a string like key="val".

flatten()

dk.html.theme module

Attempt at defining themes from Python (probably better to do this elsewhere).

dk.html.theme.**mkpalette**(*_source*, *base*, *colorlist*)
Convenience function to define a palette.

class dk.html.theme.**palette**
Color palettes from http://colormatch.dk. Colors are named with a lower case x in front of the nearest defined colorvalue.

class LightSkyBlue2

Gray7 = '#121212'

Gray93 = '#ededed'

LightSkyBlue2 = '#a4d3ee'

get = ['#a4d3ee', '#6f8ea1', '#edd7a4', '#a1896f', '#121212', '#ededed']

xLemonChiffon4 = '#a1896f'

xLightSlateGray = '#6f8ea1'

xNavajoWhite2 = '#edd7a4'

class palette.**xForrestGreen**

Gray48 = '#7a7a7a'

Gray52 = '#858585'

get = ['#257b24', '#3cc73a', '#5f7a23', '#9ac73a', '#858585', '#7a7a7a']

xForrestGreen = '#257b24'

xLimeGreen = '#3cc73a'

xOliveDrab4 = '#5f7a23'

xYellowGreen = '#9ac73a'

dk.html.uhtml module

New version of html.py module that works on/with Unicode.

class dk.html.uhtml.**EscapedString**
Bases: unicode

class dk.html.uhtml.**a**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**abbr**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**acronym**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**address**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**applet**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**area**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**b**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**base**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**bdo**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**big**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**blockquote**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**body**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**bokmaal**
alias of *link*

class dk.html.uhtml.**br**(***kw*)
Bases: *dk.html.uhtml.xtag*

class dk.html.uhtml.**bsefont**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**button**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**caption**(**content*, ***kw*)
Bases: *dk.html.uhtml.dtag*

class dk.html.uhtml.**center**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**checkbox_input**
alias of *input*

class dk.html.uhtml.**cite**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**closetag**(*tag_name*, **content*, ***kw*)
Bases: *dk.html.uhtml.tag*

flatten(*lst=None*)

class dk.html.uhtml.**code**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**col**(***kw*)
Bases: *dk.html.uhtml.xtag*

class dk.html.uhtml.**colgroup**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**dd**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**del_**
alias of `del`

class dk.html.uhtml.**dfn**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**dir_**
alias of `dir`

class dk.html.uhtml.**div**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**dl**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**doctype**
alias of DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"

dk.html.uhtml.**doctype401frameset**
alias of DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd"

dk.html.uhtml.**doctype401strict**
alias of DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"

dk.html.uhtml.**doctype401transitional**
alias of DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"

class dk.html.uhtml.**dt**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**dtag**(*tag_name*, **content*, ***kw*)
Bases: *dk.html.uhtml.tag*

d(issappearing)tag: if the content is empty, i.e. self.content == (‘’,) this tag doesn’t output anything at all. Useful for legends, table captions, etc.

class dk.html.uhtml.**em**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**english**
alias of *link*

dk.html.uhtml.**escape**(*s*, *enc=None*)
Convert string s (potentially unicode) to a ascii string with entitydefs like ø æ etc.

dk.html.uhtml.**escape_char**(*unichar*)

dk.html.uhtml.**escaped_array**(*s*)
Convert unicode string to list of ascii characters or entitydefs like ø etc.

class dk.html.uhtml.**fieldset**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**font**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**form**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**frame**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**frameset**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**h1**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**h2**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**h3**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**h4**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**h5**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**h6**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**head**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**hidden_input**
alias of *input*

class dk.html.uhtml.**hr**(***kw*)
Bases: *dk.html.uhtml.xtag*

class dk.html.uhtml.**html**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**i**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**iframe**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**img**(***kw*)
Bases: *dk.html.uhtml.xtag*

class dk.html.uhtml.**input**(***kw*)
Bases: *dk.html.uhtml.xtag*

class dk.html.uhtml.**ins**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**kbd**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**label**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**legend**(**content*, ***kw*)
Bases: *dk.html.uhtml.dtag*

class dk.html.uhtml.**li**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**lines**(**content*)
Bases: *dk.html.uhtml.text*

like text, except each item in content is separated with a
 tag.

flatten()

class dk.html.uhtml.**link**(***kw*)
Bases: *dk.html.uhtml.xtag*

class dk.html.uhtml.**map**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**menu**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**meta**(**kw)
Bases: *dk.html.uhtml.xtag*

dk.html.uhtml.**mkdtag**(*name*, ***attrs*)

dk.html.uhtml.**mkstag**(*name*)

dk.html.uhtml.**mktag**(*name*, *_parent=<class 'dk.html.uhtml.tag'>*, *_nlafter=False*, ***attrs*)

dk.html.uhtml.**mkxtag**(*name*, ***attrs*)

dk.html.uhtml.**next**
alias of *link*

class dk.html.uhtml.**nobr**(*content, **kw)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**noframes**(*content, **kw)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**norm_attr_name**(*a*)
`_foo_bar => _foo_bar, class_ => class, max_height => max-height`

```
>>> norm_attr_name(u'class_')
u'class'
>>> norm_attr_name(u'z_index')
u'z-index'
```

dk.html.uhtml.**normalize**(*v*)
returns a stringified unicode version of v

dk.html.uhtml.**norsk**
alias of *link*

class dk.html.uhtml.**noscript**(*content, **kw)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**nynorsk**
alias of *link*

dk.html.uhtml.**object_**
alias of `object`

class dk.html.uhtml.**ol**(*content, **kw)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**opentag**(*tag_name*, *content, **kw)
Bases: *dk.html.uhtml.tag*

flatten(*lst=None*)

class dk.html.uhtml.**optgroup**(*content, **kw)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**option**(*content, **kw)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**p**(*content, **kw)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**param**(*content, **kw)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**password_input**
alias of *input*

dk.html.uhtml.**pdf**
alias of *link*

class dk.html.uhtml.**pre**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**prev**
alias of *link*

class dk.html.uhtml.**q**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**quote**(*v*)

```
>>> quote(u"Bjorn's")
u'"Bjorn\'s"'
>>> quote(u'the "best"')
u'"the &quot;best&quot;"'
```

dk.html.uhtml.**radio_input**
alias of *input*

dk.html.uhtml.**rawstr2unicode**(*s*)

class dk.html.uhtml.**s**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**samp**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**script**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**select**(*options*, *selected=None*, ***kw*)
Bases: *dk.html.uhtml.tag*

options

selected

values

class dk.html.uhtml.**small**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**span**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**stag**(*tag_name*, ***kw*)
Bases: *dk.html.uhtml.xtag*

s(ingle)tag

dk.html.uhtml.**start**
alias of *link*

class dk.html.uhtml.**strike**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**strong**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**style**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**stylesheet**
alias of *link*

class dk.html.uhtml.**sub**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**submit_button**
alias of *input*

class dk.html.uhtml.**sup**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**table**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**tabledesc**(**cols*)
Bases: object

class dk.html.uhtml.**tag**(*tag_name*, **content*, ***kw*)
Bases: *dk.html.uhtml.xtag*

Regular tag: outputs an open tag with attributes, followed by its contents, followed by a closing tag.

Attributes can be set either as keyword arguments in the constructor or by assigning to attributes of the object.

Content can be any combination of items, iterables, and generators:

>> table(tr(td(i) for i in range(5)), tr(td(i**i) for i in range(5)))

NB: Attributes that conflict with Python keywords have an underline appended, e.g.: `mytag.class_ = ...`

close_tag()

flatten(*lst=None*)

open_tag()

xcontent

class dk.html.uhtml.**tbody**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**td**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**test_doctest**()

```
>>> br()
u'<br>'
>>> div('hello', b('world'))
u'<div>hello<b>world</b></div>\n'
>>> print select(options=[u'a', u'b'], name='foo')
u'<select name="foo" id="id_foo"><option value="a">a</option>\n<option value="b">b
↪</option>\n</select>'
```

class dk.html.uhtml.**text**(**content*)
Bases: *dk.html.uhtml.tag*

text tag: outputs its contents without any tags around it. Useful for grouping at the top level.

flatten()

dk.html.uhtml.**text_input**
alias of *input*

class dk.html.uhtml.**textarea**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**tfoot**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**th**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**thead**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**title**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**tr**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**tt**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**u**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

dk.html.uhtml.**u8escape**(*s*)

class dk.html.uhtml.**ul**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**var**(**content*, ***kw*)
Bases: *dk.html.uhtml.tag*

class dk.html.uhtml.**xtag**(*tag_name*, ***kw*)
Bases: object

x(ml-style)tag: a tag without content or a closing tag. E.g.
 would be xtag('br')

Note: [2009-03-11] w3 validator complains that 4.01 loose should not use <foo /> but <foo>.

attributes()
return a string like key="val".

flatten()

Module contents

dk.identifiers package

Submodules

dk.identifiers.kid module

Strukturen på våre kid-nummer: sensornr = '0123' # fire siffer kandidatnr = '01234' # fem siffer

kiddata = sensornr + kandidatnr kid = kiddata + kontroll_10(kiddata)

`dk.identifiers.kid.`**`create_kids`**(*tctr*, *sequencenum*)
BFP-8

`dk.identifiers.kid.`**`generate_kids`**(*sensornr*, *count*, *start=0*)
Generate `count` kid numbers for sensor with `sensornr` (prefix), starting at `start` (for the counting part of the kid).

`dk.identifiers.kid.`**`is_valid_kid`**(*s*)
Test if s is a valid kid number.

`dk.identifiers.kid.`**`kontroll_10`**(*s*)
MOD10 algoritmen.

MOD10 er forkortelse for Modulus 10 algoritmen, også kalt Luhn-algoritmen etter oppfinneren Hans Peter Luhn. Modulus 10 algoritmen benyttes bl.a. som beregningsmetode for et kontrollsiffer i KID-numre på bankenes innbetalingsblanketter. (http://no.wikipedia.org/wiki/MOD10)

`dk.identifiers.kid.`**`kontroll_11`**(*s*)
MOD11 algoritmen.

MOD11 er forkortelse for Modulus11. Modulus11 benyttes blant annet som beregningsmetode for et kontrollsiffer i kontonumre i norske banker, organisasjonsnummer og for det siste sifferet i norske fødselsnummer. (Norske fødselsnummer har to kontrollsifre, det nest siste er også et modulo 11 kontrollsiffer, men ved beregningen av dette benyttes det multiplikatorer i en annen og uregelmessig rekkefølge). (http://no.wikipedia.org/wiki/MOD11)

`dk.identifiers.kid.`**`vekt`**(*n*, *base*)
Funksjon som gir en repeterende serie med siffer fra `base`, som har lengde `n`.

For kontroll_10 er vekttall-serien: 2,1,2,1,2,1... for kontroll_11 er vekttall-serien: 2,3,4,5,6,7,2,3,4,5,...

dk.identifiers.navn module

Operations on names: shortening, normalizing, ...

TODO: This module should be connected to the names database.

exception `dk.identifiers.navn.`**`Name2Long`**
Bases: `exceptions.ValueError`

The name is too long and cannot be shortened.

`dk.identifiers.navn.`**`combine`**(*fornavn*, *etternavn*)
Combine *fornavn* and *etternavn* with a space in-between. Remove any double spaces and fix capitalization.

`dk.identifiers.navn.`**`forkort_navn`**(*lengde*, *fornavn*, *etternavn*)
Shorten names and return the result as a string.

`dk.identifiers.navn.`**`forkort_navn_u8`**(*lengde*, *fornavn_u8*, *etternavn_u8*)

`dk.identifiers.navn.`**`normalize`**(*fornavn*, *etternavn*)
Normalize double spaces and title case.

`dk.identifiers.navn.`**`normalize2u8`**(*navn*)
Normalized := Utf-8 And Title Case.

dk.identifiers.navn.**normalize2uni**(*navn_u8*)
Normalized := Unicode And Title Case.

dk.identifiers.navn.**shorten**(*lengde*, *fornavn*, *etternavn*)
Apply algorithms described above, first to first-names, then to last-names. Return tuple of (fornavn, etternavn).

dk.identifiers.navn.**shorten_fname**(*length*, *fname*)
Try to shorten the first-name *fname* to *length* characters (raise ValueError if we fail).

Algorithm: Consider the last of the person's first names and replace it with its initial. If the resulting name is still too long, then call ourselves recursively.

dk.identifiers.navn.**shorten_fname_u8**(*length*, *fname*)

dk.identifiers.navn.**shorten_lname**(*length*, *lname*)
Try to shorten the last-name *lname* to *length* characters (raise ValueError if we fail).

Algorithm: Consider the first of the person's last names and replace it with its initial. If the resulting name is still too long, then call ourselves recursively.

dk.identifiers.navn.**shorten_lname_u8**(*length*, *lname*)

dk.identifiers.navn.**shorten_u8**(*lengde*, *fornavn*, *etternavn*)

dk.identifiers.navn.**test_module**()

```
>>> forkort_navn(25, 'Bjorn Steinar', 'Fjeld Pettersen')
u'Bjorn S Fjeld Pettersen'
>>> forkort_navn(22, 'Bjorn Steinar', 'Fjeld Pettersen')
u'Bjorn S F Pettersen'
>>> forkort_navn(18, 'Bjorn Steinar', 'Fjeld Pettersen')
Traceback (most recent call last):
...
ValueError: Bjorn S F Pettersen kan ikke forkortes til 18 bokstaver
```

```
>>> v = forkort_navn(25, u'Bjørn Øystein', u'Fjeld Pettersen')
>>> u'Bjørn Ø Fjeld Pettersen'
>>> v == u'Bjørn Ø Fjeld Pettersen'
True
>>> v = forkort_navn(21, u'Bjørn Øystein', u'Ødal Pettersen')
>>> v == u'Bjørn Ø Ø Pettersen'
True
```

```
>>> v = forkort_navn_u8(21, u'Bjørn Øystein'.encode('u8'), u'Ødal Pettersen'.
↪encode('u8'))
>>> v == u'Bjørn Ø Ø Pettersen'.encode('u8')
True
```

Normalisering av navn til utf-8 med stor forbokstav som eneste store bokstav.

```
>>> normalize2u8(u'Bjørn') == u'Bjørn'.encode('u8')
True
>>> normalize2u8(u'BJørn') == u'Bjørn'.encode('u8')
True
>>> normalize2u8(u'BJØrn'.encode('u8')) == u'Bjørn'.encode('u8')
True
>>> normalize2u8(u'geir-arne') == 'Geir-Arne'
True
```

dk.identifiers.persnr module

Norwegian 'Personnummer' module.

exception dk.identifiers.persnr.**PersnrException**
Bases: exceptions.ValueError

Base exception for persnr module.

class dk.identifiers.persnr.**TestingPersnr**
Bases: object

Class to generate personnummer.

next_persnr()

dk.identifiers.persnr.**calc_par1**(*ppnr*)
Calculate the first parity digit.

dk.identifiers.persnr.**calc_par2**(*ppnr*)
Calculate the second parity digit.

dk.identifiers.persnr.**calc_parity**(*ppnr*)
Calculate parity digits.

dk.identifiers.persnr.**calc_year**(*yr2*, *inr*)
Find the 4-digit year, following all the rules.

dk.identifiers.persnr.**check_individnr**(*inr*, *year4*)
Hopelessly inefficient way of checking the individnr.

dk.identifiers.persnr.**check_parity**(*pnr*)
Check the last two digits, which are parity controls.

dk.identifiers.persnr.**check_pnr**(*pnr*, *birthday*, *sex*)

```
>>> check_pnr('02057035768', datetime.date(1970, 5, 2), 'm')
True
```

dk.identifiers.persnr.**check_pnr_structure**(*pnr*)
Raise exception if the structure of the personnummer is incorrect.

dk.identifiers.persnr.**date**(*pnr*)
Find the birth date and return as datetime.date() object.

dk.identifiers.persnr.**gender**(*pnr*)
Extract the gender as 'M' for male and 'F' for female.

dk.identifiers.persnr.**generate_pnr**(*day*, *gndr*)
Generate all persnrs for a given gender on a given day.

dk.identifiers.persnr.**is_persnr**(*pnr*, *country='NO'*)
Return True if pnr is a valid persnr.

dk.identifiers.persnr.**list_pnr**(*day=None*, *gender='M'*)
List all persnrs for a given gender on a given day.

dk.identifiers.persnr.**multiply_reduce**(*avec*, *bvec*)
Multiply each item in a with corresponding item in b, then sum the result.

dk.identifiers.persnr.**splitpnr**(*pnr*)
Split the personnummer into it's parts.

dk.identifiers.persnr.**testing_persnr**(*n=0*)
Create a persnr for use in unit tests. If different tests need separate persnr, pass a unique small integer as a parameter.

dk.identifiers.persnr.**year**(*pnr*)
Extract the year from pnr.

Module contents

dk.js package

Submodules

dk.js.js module

This module is dedicated to creating javascript snippets that can be consumed elsewhere. (uses MochiKit and seems to be imported in way too many places...)

dk.js.js.**focus**(*item*)
Output javascript to focus on item.

dk.js.js.**javascript**(*txt*, ***args*)
Compress javascript into a single line, for in-situ on___ handlers.

dk.js.js.**link**(*url*)
Same as a html.a element

dk.js.js.**set_datefield**(*name*, *year*, *month*, *day*)
The year/month/day values need to be calculated as in the SetDateButton in widgets.

dk.js.js.**setpnumber_connect**(*postnrid*)
Add onblur handler that connects the above event handler.

dk.js.js.**setpnumber_function**(*postnrid*, *poststedid*)
Event handler to load poststed from postnr, through an ajax call.

dk.js.js.**submit_form**(*formname*)
JS sniplet that will submit the named form when being the target of an event-handler.

Module contents

dk.ttcal package

Submodules

dk.ttcal.calfns module

dk.ttcal.calfns.**chop**(*it*, *n*)
Chop iterator into *n* size chuchks.

dk.ttcal.calfns.**isoweek**(*year*, *week*)
Iterate over the days in isoweek *week* of *year*.

dk.ttcal.calfns.**rangecmp**()
Compare half-open intervals [a, b) and [c, d) They compare equal if there is overlap.

dk.ttcal.calfns.**rangetuple**(*x*)

dk.ttcal.day module

Date (single day) operations.

class dk.ttcal.day.**Day**
Bases: datetime.date

A calendar date.

Month
Return a Month object representing the month *self* belongs to.

Year
Return a Year object representing the year *self* belongs to.

between_tuple()
Return a tuple of datetimes that is convenient for sql *between* queries.

code
One letter code representing the dayname.

compare(*other*)
Return how similar self is to other, i.e. the smallest factor they have in common ('day', 'month', or 'year'). Returns None if the Days are in different years.

date()
Excplicitly convert to datetime.date.

datetime(*hour=0*, *minute=0*, *second=0*)
Extend *self* to datetime.

datetuple()
Return year, month, day.

day_code = ['M', 'U', 'W', 'H', 'F', 'A', 'S']

day_name = [u'mandag', u'tirsdag', u'onsdag', u'torsdag', u'fredag', u'l\xf8rdag', u's\xf8ndag']

dayname
The semi-localized name of self.

display
Return the 'class' of self.

first
Define self == self.first for polymorphic usage with other classes.

format(*fmt=None*)
Emulate Django's date filter.

classmethod from_idtag(*tag*)
Return Day from idtag.

static get_day_name(*daynum*, *length=None*)
Return dayname for daynum.

idtag
Return the idtag for *self*: dyyyymmddmm.

in_month
True iff the day is in its month.

isoyear
Return the *isoyear* of *self*.

last
Define self == self.last for polymorphic usage with other classes.

middle
Return the day that splits the date range in half.

next()
Return Tomorrow (for use in templates).

classmethod parse(*strval*)
Parse date value from a string. Allowed syntax include

```
yyyy-mm-dd, yyyy-m-dd, yyyy-mm-d, yyyy-m-d
dd-mm-yyyy, etc.
dd/mm/yyyy, ...
dd.mm.yyyy, ...
ddmmyyyy
```

prev()
Return Yesterday (for use in templates).

range()
Return an iterator for the range of *self*.

rangetuple()

special
True if the database has an entry for this date (sets special_hours).

timetuple()
Create timetuple from datetuple. (to interact with datetime objects).

today
True if self is today.

week
Return a Week object representing the week *self* belongs to.

weekday
True if self is a weekday.

weekend
True if self is Saturday or Sunday.

weeknum
Return the isoweek of *self*.

class dk.ttcal.day.**Days**(*start*, *end*, *start_week=False*)
Bases: list

A contigous set of days.

between_tuple()
Return a tuple of datetimes that is convenient for sql *between* queries.

first
1st day

last
last day

middle
Return the day that splits the date range in half.

range()
Return an iterator for the range of *self*.

class dk.ttcal.day.**Today**
Bases: *dk.ttcal.day.Day*

Special subclass for today's date.

today = True

dk.ttcal.duration module

Extension of datetime.timedelta.

class dk.ttcal.duration.**Duration**
Bases: datetime.timedelta

A duration of time.

duration_tuple()
Return self as hours, minutes, seconds.

hrs
The number of hours in self.

mins
The number of minutes in self.

classmethod parse(*txt*)
Parse a textual representation into a Duration object. Format HHH:MM:SS.

secs
The number of seconds in self.

classmethod sum(*sequence*, *start=None*)
Return the sum of sequence. (built-in sum only works with numbers).

toint()
Convert self to integer.

dk.ttcal.month module

class dk.ttcal.month.**Month**(*year=None*, *month=None*, *date=None*)
Bases: object

A calendar month.

Month

Year
Return a Year object for the year-part of this month.

between_tuple()
Return a tuple of datetimes that is convenient for sql *between* queries.

datetuple()
First date in month.

daycount
The number of days in this month (as an int).

dayiter()

days()
Return a list of days (*class*:ttcal.Day) in this month.

first
First day in month.

format(*fmt=None*)
Format according to format string. Default format is monthname, four-digit-year.

classmethod from_date(*d*)
Create a Month from the date d.

classmethod from_idtag(*tag*)
Parse idtag into *class*:Month.

idtag()
Return a text representation that is parsable by the from_idtag function (above), and is useable as part of an url.

last
Last day in month.

mark(*d*, *value='mark'*, *method='replace'*)

marked_days()

middle
Return the day that splits the date range in half.

month = None

month_name = ['', 'Januar', 'Februar', 'Mars', 'April', 'Mai', 'Juni', 'Juli', 'August', 'September', 'Oktober', 'Novembe

next()
Next month.

numdays()
The number of days in the month.

classmethod parse(*txt*)
Parse a textual representation into a Month object. Format YYYY-MM?

prev()
Previous month.

range()
Return an iterator for the range of *self*.

rangetuple()

timetuple()
Create timetuple from datetuple. (to interact with datetime objects).

year = None

dk.ttcal.week module

class dk.ttcal.week.**Week**(*days*, *month*)
Bases: object

between_tuple()
Return a tuple of datetimes that is convenient for sql *between* queries.

current
True if today is in week.

datetuple()
First day of this week.

days = None

first
1st day of week.

classmethod from_idtag(*tag*)

idtag()

last
Last day of week.

middle
Return the day that splits the date range in half.

month = None

num = None

range()
Return an iterator for the range of *self*.

rangetuple()

until_today()

classmethod weeknum(*n=None*, *year=None*)

year = None

dk.ttcal.year module

class dk.ttcal.year.**Year**(*year=None*)
Bases: object

H1
First half of this year.

H2
Last half of this year.

Month
For orthogonality in the api.

Q1
1st quarter.

Q2
2nd quarter.

Q3
3rd quarter.

Q4
4th quarter.

Year

april

august

between_tuple()
Return a tuple of datetimes that is convenient for sql *between* queries.

datetuple()
January 1.

dayiter()

december

february

first
First day of first month.

format(*fmt=None*)
Format according to format string. Default format is monthname, four-digit-year.

classmethod from_idtag(*tag*)
Year tags have the lower-case letter y + the four digit year, eg. y2008.

halves()
Both halves of the year.

idtag()
Year tags have the lower-case letter y + the four digit year, eg. y2008.

january

july

june

last
Last day of last month.

march

mark(*d*, *value='mark'*)

mark_period(*p*, *value='mark'*)

marked_days()

may

middle
Return the day that splits the date range in half.

next()
Next year.

november

october

prev()
Previous year.

quarters()
Every quarter in this year.

range()
Return an iterator for the range of *self*.

rangetuple()

rows()

rows4()

september

timetuple()

Module contents

Date classes (originally from TikTok).

dk.ttcal.**from_idtag**(*idtag*)
Return a class from idtag.

Submodules

dk.age module

Age (date subtraction) routines.

class dk.age.**age**(*dob*, *today=None*)
Bases: object

The number of years, months, and days since date of birth.

dk.age.**birthday_this_year**(*birthday*, *today=None*)
Return the date of the birthday in the current year.

dk.age.**days_ago**(*n*, *dato=None*)
The date that is *n* days before *dato* (or today).

dk.age.**next_birthday**(*birthday*, *today=None*)
Return the date of the next birthday for someone born on date *birthday*.

dk.age.**previous_birthday**(*birthday*, *today=None*)
Return the previous birthday relative to `today`.

dk.age.**weeks_ago**(*n*, *today=None*)
The date that is *n* weeks before *today*.

dk.age.**years_ago**(*n*, *today=None*)
The date that is *n* years before *today*.

dk.asciify module

Convert unicode strings to visually similar ascii representations.

dk.asciify.**ascii_name**(*name*)
Convert name from unicode to a ascii representation that (while surely a grave bastardization) can be used as a filename without (ever!) causing problems.

dk.asciify.**asciify**(*s*, *spaces=None*, *legal=None*, *replacement=''*)
Convert unicode string *s* to a similarly looking ascii string.

If *spaces* is specified, runs of space characters are replaced with exactly one *spaces*.

If *legal* is specified (as a string), only characters from *legal* will be in the result, otherwise all characters from ascii 32 to ascii 127 are allowed.

If *replacement* is passed, then any characters that are elided will be replaced by *replacement*.

dk.asciify.**slug**(*txt*)
Same as above, but intended for URIs.

dk.dkimport module

Convenience function for importing a fqdn from a package. (to hide the baroque nature of __import__).

dk.dkimport.**defined_symbols**(*module*, *attrfilter=None*, *itemfilter=None*)
Return symbols that are defined in module.

dk.dkimport.**dkimport**(*name*)
Import and return the item specified by name:

Usage:

```
>>> item = dkimport('dk.core.dkimport.dkimport')
>>> item.__name__
'dkimport'
```

dk.dkimport.**dkimport_functions**(*modname*, ***kw*)
Return all functions from all direct sub-modules of *modname*.

dk.dkimport.**dkimport_star**(*modname*, ***kw*)
Import all names from module *modname*, similar to:

```
from modname import *
```

Available `**kw` arguments:

filefilter A function that receives a filename (with extension but without path) that should return True if the filename should be included.

useful for modularly implementing functionality, yet making it dynamically available from the top namespace:

```
...path/cmds/a.py
    def a(): ...
...path/cmds/b.py
    def b(): ...
...path/cmds/c.py
    def c(): ...

...path/cmds/__init__.py
    from dk.dkimport import dkimport_star as _dki

    for _item in _dki('..path.cmds'):
```

```
        if hasattr(_item, '__name__'):
            globals()[_item.__name__] = _item
```

dk.dkimport.**load_files_from**(*module_path*, *module_name*, *filefilter=None*)
Load all .py files in module.

dk.dklogger module

Convenience function for installing a module level logger:

```
from dk import dklogger
logger = dklogger.dklogger(__name__, debug=1, info=1)
logger.setLevel(dklogger.DEBUG)
```

to prevent logging to stdout, pass *stream=None*

dk.dklogger.**dklogger**(*name*, *debug=False*, *info=False*, *fname=None*, *stream=<open file '<stdout>'*, *mode 'w'>*, *format='%(asctime)s %(levelname)-7s\n %(pathname)s@%(funcName)s:%(lineno)d %(message)s'*, *datefmt='%Y-%m-%d %H:%M:%S'*)

dk.findapps module

Module for finding all apps folders.

dk.findapps.**appfolder**(*appname*)
Return app folder for appname.

dk.findapps.**appfolders**()
Find all django app folders, yield absolute paths to the folders.

dk.findapps.**appname**(*folder*)
Return the app name for the (app)folder).

dk.findapps.**appnames**()
Find the name of all the apps.

dk.findapps.**is_appfolder**(*path*)
Is the path an app folder?

dk.fstr module

class dk.fstr.**fstr**
Bases: str

String sub-class with a split() method that splits a given indexes.

Usage:

```
>>> r = fstr('D2008022002')
>>> print r.split(1, 5, 7, 9)
['D', '2008', '02', '20', '02']
>>> _, year, _ = r.split(1,5)
>>> year
'2008'
```

split(**ndxs*)

dk.grid module

A 2D grid with slicing.

Usage:

```
>>> t = grid(emptyval=0)
>>> t.apply[:5,:5] = lambda v, (y,x):y*x
>>> t
 0 0 0  0  0
 0 1 2  3  4
 0 2 4  6  8
 0 3 6  9 12
 0 4 8 12 16
>>> t.apply[:5,:5] = lambda v, p:v*2
>>> t
 0 0  0  0  0
 0 2  4  6  8
 0 4  8 12 16
 0 6 12 18 24
 0 8 16 24 32
>>> t2 = grid.copy(t, lambda orig, (y,x):orig[y,x] / 2)
>>> t2
 0 0 0  0  0
 0 1 2  3  4
 0 2 4  6  8
 0 3 6  9 12
 0 4 8 12 16
```

class dk.grid.**Empty**(*emptyval=None*)
Bases: *dk.proxy.proxy*

setval(*v*)

class dk.grid.**grid**(*rows=0*, *cols=0*, *emptyval=None*)
Bases: object

This is a tabular object of two dimensions that supports slice notation.

Deleted = <->

apply

apply_cell(*y*, *x*, *fn*)

apply_iterator()
You can implement the game of life style actions with this iterator:

```
def average((y,x)):
    return sum(t[y-1:y+1, x-1:x+1]) / 9.0
t.apply[:2, :2] = lambda value, key: average(key)
```

columns

classmethod copy(*tgrid*, *fn=None*)

copy_area()

del_cell(*y*, *x*)

empty_col(*x*)

empty_row(*y*)

get_cell(*y*, *x*)

get_col(*x*)

get_row(*y*)

height

insert_col(*xpos=None*, *count=1*)

insert_row(*ypos=None*, *count=1*)

isempty(*y1*, *x1*, *y2*, *x2*)

key_iterator()

keyiter(*ykey*, *xkey*, *debug=False*)

keys

lastcol

lastrow

move_area()

next_nonempty_down(*ykey*, *xkey*)

next_nonempty_right(*ykey*, *xkey*)

notempty(*y1*, *x1*, *y2*, *x2*)

print_row(*y*)

purge()
Remove rows and columns that are empty.

raise_indexerror(*y*, *x*)

range_check(*y*, *x*, *throw=True*)

remove_col(*xpos*, *count=1*)

remove_row(*ypos*, *count=1*)

resize(*yndx*, *xndx*, *pr=False*)
Resize so that self[yndx,xndx] is valid.

reverse_key_iterator()

reversed

rows

set_cell(*y*, *x*, *v*)

size

transpose()

value_iterator()

values

width

dk.grid.**indexiter**(*length*, *ndx*)

class dk.grid.**point**
Bases: tuple

x

y

dk.grid.**point_xiter**(*start*, *end*)

dk.grid.**point_yiter**(*start*, *end*)

class dk.grid.**rect**(*x*, *y*, *w*, *h*)
Bases: object

NE

NW

SE

SW

corners

isomorphic(*other*)
Same shape?

opposite(*corner*)

x

x2

y

y2

class dk.grid.**table_iterator**(*iterfn*)
Bases: object

class dk.grid.**value_iterator**(*gridobj*, *ykey*, *xkey*)
Bases: object

indices(*direction='RD'*)

iter(*direction='RD'*)

ndx_base(*direction='RD'*)

rect()

dk.iplist module

Collections of distinct ip-addresses.

class dk.iplist.**IPList**(*iterable=()*)
Bases: object

List (well, actually more of a set) of ip-addresses (well, actually using subnets...).

add(*ipaddy*)
Add ipaddy to self.

network()
Return the list of networks that cover our list of ip-addys.

pack()
Convert to compressed, but db friendly, notation. This fits ~124 ip addys into 250 bytes, if they are sufficiently contigous.

unpack(*b64val*)
Reverse steps in pack().

dk.proxy module

Proxy class that forwards __special__ methods too.

class dk.proxy.**proxy**(*obj*)
Bases: object

Proxy class that forwards __special__ methods too.

dk.text module

dk.text.**u**(*obj*)
Return obj as a unicode string. If obj is a (non-)unicode string, then first try to decode it as utf-8, then as iso-8859-1.

dk.text.**u8**(*obj*)
Return a utf-8 representation of obj.

dk.text.**unicode_repr**(*obj*)
Return obj as a unicode string. If obj is a (non-)unicode string, then first try to decode it as utf-8, then as iso-8859-1.

dk.text.**utf8**(*obj*)
Return a utf-8 representation of obj.

dk.utidy module

Micro tidy.

Usage:

```
>>> print utidy('''
... <form name="FirmaForm" id="FirmaForm" method="POST" autocomplete="off"
... action="." class="fForm"><input type="hidden" name="__cmd"
... value="FirmaForm"></form>hello
... ''')
...
<form action="." autocomplete="off" class="fForm" id="FirmaForm" method="POST" name=
↪"FirmaForm">
    <input name="__cmd" type="hidden" value="FirmaForm">
</form>>
hello
```

class dk.utidy.**HtmlTag**(*txt*)
Bases: object

attre = <_sre.SRE_Pattern object>

normalize_attrs(*attrs*)

normalize_class(*val*)

normalize_style(*val*)

dk.utidy.**simplify_simple_tags**(*html*)
Put tags without any nested children on one line, i.e. turn:

```
<h1>
    foo
</h1>
```

into:

```
<h1>foo</h1>
```

dk.utidy.**tokenize_html**(*html*)

dk.utidy.**utidy**(*html*, *level=0*, *indent=' '*, *simplify=False*)
micro-tidy

Normalizes the html.

dk.utils module

FIXME: many of these really should go in their own modules...

dk.utils.**HourMinute**(*v*)
Format 7.10 as 7t 06m.

class dk.utils.**Ordered**
Bases: dict

Mapping that maintains insertion order. (Should be removed and the adt versions should be used).

items()

keys()

values()

dk.utils.**dkpath**(*pth=None*)
Usage

```
dkpath() => (w:)/xxxxxxx/
         => (/home)/xxxxxxxx/

dkpath('app/') => ../xxxxxxxx/app/
```

dk.utils.**hm_to_float**(*h*, *m*)
Convert 7, 30 to 7.5 hours.

dk.utils.**hour_minute**(*v*)
Convert 7.5 (hours) to (7, 30) i.e. 7 hours and 30 minutes.

dk.utils.**html2u8**(*s*)
Convert charrefs for Norwegian vowels to their utf-8 counterparts.

dk.utils.**identity**(*x*)
Return the argument unchanged. This function is often called *identity* in programming language and type theory (the type is t -> t, which turns out to be a difficult type for most classical static type systems).

dk.utils.**kr_ore**(*n*)
Convert the øre-value n to a proper NOK string value.

dk.utils.**kronestring**(*kr*)
Return a string version of the integer value kr, with space as the thousand-separator.

dk.utils.**latin1**(*obj*)
Return a latin-1 representation of obj.

dk.utils.**lower_case**(*s*, *encoding='u8'*)
Return a lower cased (byte-)string version of s encoded in encoding.

dk.utils.**mk_post**(*model*)
Encode model (a dict-like object) into a dict where:

- all values are strings
- None values are removed
- date values are expanded into year/month/day parts

Note:: this function is superceeded by maint.client._encode_date which does this transparently for unit tests.

dk.utils.**nlat**(*v*)
Normalize and recover from utf-8 stored in varchar columns.

dk.utils.**normalize**(*v*)
Return a string version of v such that

normalize(u) == normalize(v) iff **not** (u != v)

e.g.:

normalize(None) == normalize('') == normalize(u'')

dk.utils.**orestring**(*n*)
Return a string version of the integer øre value. Either a two-digit string or a dash (as in 5,-).

dk.utils.**root**()
Return the root of the source tree.

dk.utils.**single_line**(*txt*)
Remove multiple spaces and newlines.

dk.utils.**srcpath**(*base*, *pth*)
Return an absolute path based on the relative path *pth*. Useful in tests, where we don't know what the absolute path is, and we can't use relative paths since we don't know which folder the tests are run from.

In a test file xxxxxxx/foo/test/test_foo.py:

```
path = 'foo/test'
fp = open(srcpath(path, 'data/testdata.txt'))
```

dk.utils.**title_case**(*s*, *encoding='u8'*)
Return a title cased (byte-)string version of s encoded in encoding.

dk.utils.**title_case_lastname**(*s*, *encoding='u8'*)
Return a title cased version of s encoded in encoding. If it looks like s is already title cased, then leave it alone (in case of manual override and complex capitalization rules for last names).

dk.utils.**u**(*obj*)
Return obj as a unicode string. If obj is a (non-)unicode string, then first try to decode it as utf-8, then as iso-8859-1.

dk.utils.**u8**(*obj*)
Return a utf-8 representation of obj.

dk.utils.**ulower_case**(*val*)
Call val.lower(). Return '' if val is None.

dk.utils.**unhtml**(*s*, *toencoding=None*)
Convert charrefs for Norwegian vowels to their unicode counterparts.

dk.utils.**unicode_repr**(*obj*)
Return obj as a unicode string. If obj is a (non-)unicode string, then first try to decode it as utf-8, then as iso-8859-1.

dk.utils.**utf8**(*obj*)
Return a utf-8 representation of obj.

dk.utils.**utitle_case**(*val*)
(safer) val.title() implementation.

dk.utils.**utitle_case_lastname**(*s*)
Return a title cased version of s. If s contains a recognized special case, then return it unchanged.

Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y