
mathmaker Documentation

Release 0.7.3

Nicolas Hainaux <nh.techn@gmail.com>

Nov 17, 2017

Contents

1	User's guide	1
1.1	Overview	1
1.2	Quickstart	1
1.3	Contribute	3
1.4	Contributors	3
1.5	Changelog	4
1.6	Advanced features	6
1.7	Install notes	14
2	Developer's documentation	17
2.1	Guided tour	17
2.2	Start working on mathmaker	20
2.3	A deeper look in the source code	33
2.4	What can be done?	36
2.5	mathmaker package	36
3	Indices and tables	79
	Python Module Index	81

Contents:

1.1 Overview

Mathmaker creates elementary maths worksheets with detailed solutions.

The output documents can be compiled into pdf files by lualatex. Examples of available themes are: first degree equations, pythagorean theorem, fractions calculation. . .

It can run from command line, but can be controlled via http requests too.

[License](#)

[Documentation \(master release\)](#)

[Documentation \(latest development release\)](#).

1.2 Quickstart

1.2.1 Install

Required (non python) dependencies are python ≥ 3.6 , euktoeps $\geq 1.5.4$, xmllint, msgfmt, lualatex, luaotfload-tool and a bunch of LaTeX packages(1)

To install them:

- on Ubuntu, install python3.6 if it is not already installed, then:

```
$ sudo apt-get install eukleides libxml2-utils gettext texlive-latex-base
```

And for the yet missing LaTeX packages: you can either install the complete texlive distribution (takes some room on the hard disk): run `$ sudo apt-get install texlive-full`, or install only the necessary packages:

```
$ sudo apt-get install texlive-luatex texlive-latex-recommended texlive-xetex
↳ texlive-pstricks texlive-font-utils texlive-latex-extra texlive-base texlive-
↳ science texlive-pictures texlive-generic-recommended texlive-fonts-recommended
↳ texlive-fonts-extra
```

- on FreeBSD(2):

```
$ sudo pkg install python36 py36-sqlite3 gettext eukleides libxml2 texlive-full
$ rehash
```

Note: Check how to fix eukleides install in [the complete documentation](#)

Once you're done, you can proceed installing mathmaker:

```
$ pip3 install mathmaker
```

(this will automatically install some extra python3 libraries too: mathmakerlib, polib, ruamel.yaml, intspan and python-daemon).

Note: Check how to fix install if it stops with an error in python-daemon install, in [the complete documentation](#)

1.2.2 Basic use

```
$ mathmaker pythagorean-theorem-short-test > out.tex
$ lualatex out.tex
```

or directly:

```
$ mathmaker pythagorean-theorem-short-test --pdf > out.pdf
```

Get the list of all provided sheets:

```
$ mathmaker list
```

1.2.3 Some settings

Check `mathmaker --help` to see which settings can be changed as command line arguments.

Some more settings can be overridden by user defined values in `~/.config/mathmaker/user_config.yaml`. Read [the complete documentation](#) for more information.

1.2.4 Advanced use

It's possible to create your own sheets in yaml. Read [the complete documentation](#) for more information.

1.3 Contribute

You can contribute to mathmaker:

1.3.1 As a wordings contributor

Mathmaker needs contexts for problems wordings. There are already some, but the more there is, the better. Existing wordings can be found [here](#). You can submit any new idea as an enhancement proposal [there](#) (should be written in english, french or german).

Any question can be sent to nh dot techn (hosted at gmail dot com).

1.3.2 As a translator

You can help translating mathmaker to your language (or any language you like, if you have enough elementary maths vocabulary for that).

If the translation to your language isn't started yet, there are several pot files to get [here](#) (see explanations about their respective roles [there](#)). You can use an editor like [poedit](#) or any other you like better, to create po files from them and start to translate.

If you want to add missing translations, or to correct some, you can find the po files in the subdirectories [here](#).

Once you're done, you can make a pull request [here](#).

Any question can be sent to nh dot techn (hosted at gmail dot com).

1.3.3 As a developer

Before submitting a PR, please ensure you've had a look at the [writing rules](#).

More details can be found in the [documentation for developers](#).

Any question can be sent to nh dot techn (hosted at gmail dot com).

1.4 Contributors

1.4.1 Development

- Lead developer: Nicolas Hainaux
- Developers: Vaibhav Gupta
- Clever advices: Olivier Cecillon

1.4.2 Translation

- French: Nicolas Hainaux

1.4.3 Problems wordings

Nicolas Hainaux

1.4.4 Patience and chocolate cakes

Sophie Reboud

1.5 Changelog

1.5.1 New in version 0.7.2 (2017-10-18)

- Add mental calculation sheets for third level (“yellow belt”)
- The mental calculation pdf may now be “interactive” (answers can be written in text fields, they can be validated pushing a button; this validation is done by some javascript)
- The daemon now accepts an optional argument in the request (written right after sheet’s name, separated with a ‘|’. Only interactive is allowed, so far)

1.5.2 New in version 0.7.1-3 (2017-08-30)

- Patch the daemon to let it accept the new YAML sheet names.
- The output dir will always be in user’s home.
- Fix several bugs.

1.5.3 New in version 0.7.1 (2017-08-29)

- Support for python3.6 only, drop support for older python versions.
- Mental calculation sheets can now be created as slideshows. Add a default slideshows series for white belt, 1st and 2d stripes.
- Reorganization of mental calculation in belts: White belt, 1st stripe and 2d stripe have been added (including new sheets: addition/subtraction, fraction of a rectangle, complements)
- New sheet: order of precedence in operations.
- YAML files will be used to store sheets. The previous ways (XML and Python) will be dropped.
- Huge reorganization of the lib/ source code.
- Fair bunch of bug fixes.
- Issue warnings instead of exceptions when the version of a dependency could not be determined. [0.7.1dev5 (2017-05-04)]
- New sheets about trigonometry: [0.7.1dev4 (2017-05-03)]
 - vocabulary in the right triangle
 - write the correct formulae
 - calculate a length
 - calculate an angle
- New sheets: [0.7.1dev3 (2016-10-21)]
 - intercept theorem: “butterfly” configuration
 - intercept theorem: converse

- New sheets: [0.7.1dev2 (2016-10-13)]
 - expansion of simple brackets (declined in two versions)
 - clever multiplications (mental calculation)
 - intercept theorem: write the correct quotients' equalities
 - intercept theorem: solve simple exercises
- A new sheet (declined in two versions): expansion of double brackets. Defined in an xml sheet as for mental calculation sheets. [0.7.1dev1 (2016-09-14)]

1.5.4 New in version 0.7.0-6 (2016-08-19)

- Added a setting to let the user change mathmaker's path (to be used by the daemon)
- Bugfix [0.7.0-5 (2016-08-19)]
- If an IP address is passed as parameter to mathmaker's daemon, it will return a 429 http status code (too many requests) if the last request from the same address is not older than 10 seconds. [0.7.0-4 (2016-08-19)]
- Fixed the install of locale files and font listing file [0.7.0-3 (2016-07-18)]

1.5.5 New in version 0.7 (2016-07-15)

- Standardized structure (`mathmaker` becomes pip3-installable, available on PyPI and github; its documentation is hosted on readthedocs; tests are made with `py.test`)
- A daemon is added (`mathmakerd`) to provide communication with `mathmaker` through http connections.
- A bunch of mental calculation sheets
- The use of XML frameworks for the sheets (yet only for mental calculation, so far)

Footnotes:

1. Complete list of recommended LaTeX packages (list up-to-date for 0.7 release):

CTAN Package Name	Package name (Ubuntu 14.04)
fontspec	texlive-latex-recommended
polyglossia	texlive-xetex
geometry	texlive-latex-base
graphicx	texlive-pstricks
epstopdf	texlive-font-utils
tikz	texlive-latex-extra
amssymb	texlive-base
amsmath	texlive-latex-base
siunitx	texlive-science
cancel	texlive-pictures
array	texlive-latex-base
ulem	texlive-generic-recommended
textcomp	texlive-latex-base
eurosym	texlive-fonts-recommended
lxfonts	texlive-fonts-extra
multicol	texlive-latex-base

- Using `pkg`, you'll have to install `texlive-full`; if you wish to install only the relevant LaTeX packages, you'll have to browse the ports, I haven't done this yet so cannot tell you exactly which ones are necessary.

1.6 Advanced features

1.6.1 User settings

The default settings are following:

```
PATHS:
  EUKTOEPS: euktoeps
  XMLLINT: xmllint
  LUALATEX: lualatex
  LUAOTFLOAD_TOOL: luaotfload-tool
  MSGFMT: msgfmt
  # OUTPUT_DIR *must* be relative (to user's home)
  OUTPUT_DIR: .mathmaker/outfiles/

LOCALES:
  # Available values can be checked in the locale directory.
  LANGUAGE: en_US
  ENCODING: UTF-8
  # Values can be 'euro', 'sterling', 'dollar'
  CURRENCY: dollar

LATEX:
  FONT:
  ROUND_LETTERS_IN_MATH_EXPR: False

DOCUMENT:
  # Double quotes around the template strings are mandatory.
  # {n} will be replaced by the successive numbering items (1, 2, 3... or
  # a, b, c... etc.)
  QUESTION_NUMBERING_TEMPLATE: "{n}."
  # nothing; bold; italics; underlined
  QUESTION_NUMBERING_TEMPLATE_WEIGHT: bold
  QUESTION_NUMBERING_TEMPLATE_SLIDESHOWS: "{n}."
  QUESTION_NUMBERING_TEMPLATE_SLIDESHOWS_WEIGHT: regular

DAEMON:
  MATHMAKER_EXECUTABLE: mathmaker
```

Some explanations:

- The `PATHS` section is here to provide a mean to change the paths to `euktoeps`, `lualatex` and `xmllint` mainly. In case one of them is not reachable the way it is set in this section, you can change that easily.
- The `PATHS` section contains also an `OUTPUT_DIR` entry. This is the directory where `mathmaker` will store the possible picture files (`.euk` and `.eps`). Change it at your liking, but as it must be a subdirectory of user's own directory, it must be a relative path.
- The entries under `LOCALES` allow to change the language, encoding, and default currency used.
- The `LATEX` section contains an entry to set the font to use (be sure it is available on your system). The `ROUND_LETTERS_IN_MATH_EXPR` entry is disabled by default (set to `False`). If you set it to `True`, a special font will be used in math expressions, that will turn all letters (especially the 'x') into a rounded version. This

is actually the `lxfonts` LaTeX package. It doesn't fit well with any font. Using "Ubuntu" as font and setting `ROUND_LETTERS_IN_MATH_EXPR`: to `True` gives a nice result though.

- The entries under `DOCUMENT`: allow to change some values to format the output documents.

Your settings file must be `~/.config/mathmaker/user_config.yaml`.

1.6.2 Command-line options

Several command-line options correspond to settings that are defined in `~/.config/mathmaker/user_config.yaml`. Any option redefined in command-line options will override the setting from the configuration file.

Type `mathmaker --help` to get information on these command-line options.

1.6.3 http server (mathmakerd)

Once everything is installed, it's possible to run a server to communicate with mathmaker through a web browser.

Run the server:

```
$ mathmakerd start
```

Then go to your web browser and as url, you can enter:

```
http://127.0.0.1:9999/?sheetname=<sheetname>
```

and replace `<sheetname>` by an available sheet's name (from `mathmaker list`), for instance:

```
http://127.0.0.1:9999/?sheetname=pythagorean-theorem-short-test
```

`mathmaker` will create the new sheet, compile it and return the pdf result to be displayed in the web browser.

At the moment, `mathmakerd stop` doesn't work correctly, you'll have to kill it directly (`ps aux | grep mathmakerd` then kill with the appropriate pid).

It's possible to pass an IP address in an extra parameter named `ip`, like:

```
http://127.0.0.1:9999/?sheetname=pythagorean-theorem-short-test&ip=127.0.0.1
```

In this case, `mathmakerd` will check if the last request is older than 10 seconds (this is hardcoded, so far) and if not, then a http status 429 will be returned. In order to do that, `mathmakerd` uses a small database that it erases when the last request is older than one hour (also hardcoded, so far).

1.6.4 YAML sheets

As a directive to `mathmaker` it is possible to give a path to yaml file.

Creating a new yaml file that can be used as a model by `mathmaker` is more for advanced users, though it's not that difficult.

Example

Let's have a look at `mathmaker/data/frameworks/algebra/expand.yaml`, where four sheets are defined:

```
simple: !!omap
- title: "Algebra: expand simple brackets"
- exercise: !!omap
  - details_level: medium
  - text_exc: "Expand and reduce the following expressions:"
  - questions: expand simple -> intriplets_2to9 (5)

simple_detailed_solutions: !!omap
- title: "Algebra: expand simple brackets"
- exercise: !!omap
  - text_exc: "Expand and reduce the following expressions:"
  - questions: expand simple -> intriplets_2to9 (5)

double: !!omap
- title: "Algebra: expand and reduce double brackets"
- exercise: !!omap
  - details_level: medium
  - text_exc: "Expand and reduce the following expressions:"
  - questions: expand double -> intpairs_2to9;;intpairs_2to9 (5)

double_detailed_solutions: !!omap
- title: "Algebra: expand and reduce double brackets"
- exercise: !!omap
  - text_exc: "Expand and reduce the following expressions:"
  - question: expand double -> intpairs_2to9;;intpairs_2to9 (5)
```

The four top-level keys are the sheets' names. These names must not contain spaces (not supported).

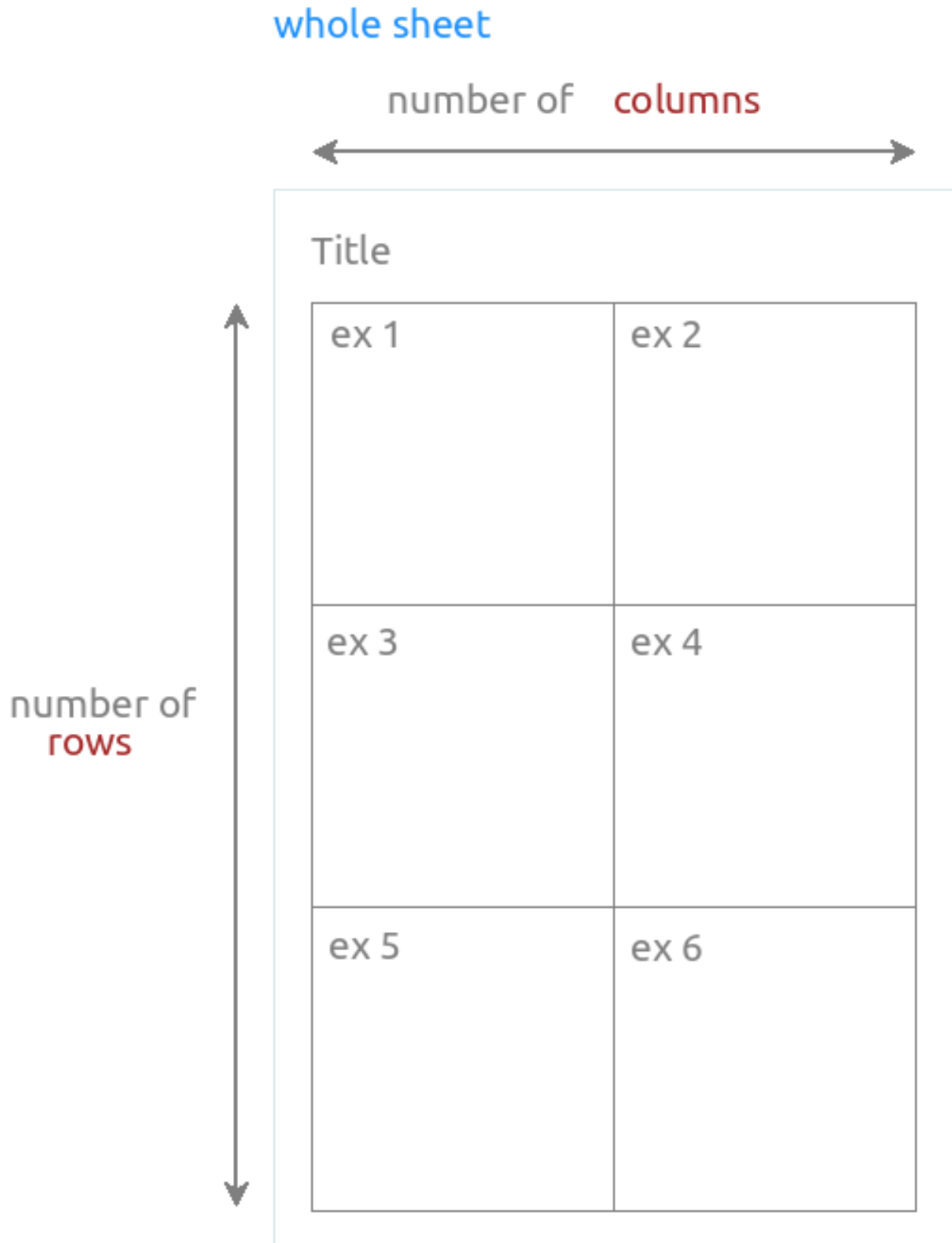
A list of keys is defined below each sheet's name. No one is mandatory. If you do not define the `title`, then the default value will be used (for titles, this is an empty string).

Sheet's keys

Possible keys for sheets, at the moment, are:

- `preset` allows to preset a number of other keys. Possible values: `default`, `mental_calculation`. Default is `default`. The `mental_calculation` value will remove the exercise's titles and the exercises layout.
- `header`, `title`, `subtitle`, `text` allow to customize the header, title, subtitle and text of the sheet. Default value is an empty string for each of them.
- `answers_title` allows to customize the title for the answers' sheet. It defaults to `Answers`.
- `layout` contains the layout description of the sheet, if necessary (see below).
- Any key starting with `exercise` will contain the list of questions of one exercise. It is not possible to use the same key several times (YAML forbids it), so if you want to define several exercises, say two, for instance, you'll have to use `exercise` and `exercise2`, for instance (if you use numbers, it will have no effect on the order of exercises in the output).

Sheet's layout



If a sheet contains no `layout` key (or if its value is left empty), then the default layout will be used (all exercises printed one after the other, unlike the 3x2 grid in the figure above).

The `layout` key can list a `unit` key, whose value will be used for columns widths (see below). `unit` defaults to `cm`.

The `layout` key can list a `font_size_offset` key, whose value is a relatively small integer allowing to change the font size (for instance, set it at +1 or +2 to enlarge all fonts, or -1 or -2 to reduce all fonts' size)

So far, no `spacing` key is available for sheets' layout, but the spacing between exercises can be set in the properties.

Finally, the `layout` key can list one `wordings` key and/or one `answers` key. They allow to define different settings for wordings and answers, but both work the same way.

The properties are defined as `key=value` pairs separated by commas (actually a comma and a space). For instance: `rowxcol=?x2, print=3 3`

- `rowxcol` can contain `none` (default: no layout) or the number of rows and columns as a multiplication of two integers: `rxc`, for instance: `3x2`. This would mean 3 rows and 2 columns, what would define 6 "cells", like in the figure above. As a convenience, you can use a `x` instead of a `x`, like this: `3x2`.
- `colwidths` is ignored if `rowxcol` contains `none`. If `rowxcol` contains a `rxc` definition, then `colwidths` defaults to `auto`: the width of all columns will be calculated automatically (all equal). Otherwise, you can set the values you like, separated by spaces, like: `4.5 4.5 9` what would make the two first columns 4.5 units wide and the last, 9 units wide (See `unit` key description above). Note that there must be as many values as the number of columns defined in the "`rxc`" definition.
- `print` is the number of exercises to print, either one after the other, or per "cell". It defaults to `auto`. If `rowxcol` contains `none`, then `print` can either be a natural number (how many exercises/questions to print), or `auto`, and then all exercises (left) will be printed, without distributing them among columns. If `rowxcol` contains a `rxc` definition, then an `auto` value would mean that each "cell" will contain one question. Otherwise, you can tell how many questions you want in each cell, row after row, as integers separated by spaces, like: `2 1 1 3 1 1` what would put (with `rowxcol=2x3`) 2 questions (or exercises) in the first cell, then 1 question in each other cell of the first row, then 3 questions in the first cell of the second row, and 1 question in each cell left. There must be as many numbers as cells. As a convenience, you can add a `.` or a `/` to separate the rows, like: `2 1 1 / 3 1 1` (These two signs will simply be ignored). Each row must contain as many numbers as defined in the `rxc` definition. If the number of rows is left undefined (?) then only the first row has to be defined (extra rows will be ignored) as a pattern for all rows (the default still being `auto`, i.e. 1 question per cell).
- `newpage` can be turned to `true` in order to insert a jump to next page.

Examples of sheet's layouts:

Say you have 4 exercises and you want to put the answers of the two first ones in 2 two columns, and then print the left ones one after another:

```
layout:
  answers: rowxcol=1x2, print=1 1,
          print=2
```

ex 1 ex 2

ex 3

ex 4

Note: YAML allows to write the same string (“scalar”) on several lines. This is practical for readability. In the example above, we could have written `rowxcol=1×2, print=1 1, print=2` all on the same line.

If you have 3 exercises and you want to print 2 answers on a first page, then jump to next page and print the answer of the third one, then your sheet’s layout may be:

```
layout:
  answers: print=2,
           newpage=true,
           print=1
```

The !!omap label

The sheets’ names keys as well as the `exercise` keys are labeled `!!omap`. This is required in order to ensure the order of the created sheets will be the same as the one defined in the sheet. Forgetting these labels won’t prevent `mathmaker` from running, but the final order may be changed (what does not mean it will be randomly reorganized at each run). In this example, this wouldn’t have any consequence as there’s only one `exercise` key in each sheet and only one `question` key in each exercise.

Exercise’s keys

Possible keys for sheets, at the moment, are:

- `preset` (same as for sheets)
- `layout_variant` can be `default`, `tabular` or `slideshow`.
- `layout_unit` defaults to `cm`.
- `shuffle` can be `true` or `false`. It defaults to `false`, except for `mental_calculation` preset. If set to `true`, then the questions will be shuffled.
- `details_level` can be `maximum` (default), `medium` or `none` (default for `mental_calculation` preset). Some types of questions can be configured to output the answer with different levels of details.
- `q_numbering` defines the numbering of the questions of the exercise. It can be `default`, `alphabet`, `alphabetical`, `numeric` or `disabled`. The three first values are synonyms.
- `start_number` defines the first number, when numbering the questions. Must be an integer greater or equal to 1.
- `spacing` defines the spacing between two consecutive exercises. It defaults to `' '` (i.e. nothing). Otherwise, you can set it at `newline`, `newline_twice`, or a value that will be inserted in a LaTeX `addvspace{}` command, for instance `spacing=40.0pt` will result in a `addvspace{40.0pt}` inserted at the end of each exercise. `spacing` can be overridden in the `layout` key (in either or both `wordings` and `answers` keys) of the exercise, in order to set different spacings for the wordings and the answers.
- `newpage` can be turned to `true` in order to insert a jump to next page.
- `q_spacing` can be used to set a default value for the spacing between two consecutive questions.
- `text_exc` and `text_ans` allow to customize the wording of the exercise and of its answer. `text_exc` defaults to nothing (empty string). `text_ans` defaults to `"Example of detailed solutions:"` with default preset, but also to an empty string with `mental_calculation` preset.
- The `question` and `mix` keys allow to define the exercise’s questions. As YAML does not allow to use the same key, if you want to define several `question` keys, nor several `mix` keys, you’ll have to use the same

trick for them as for the `exercise key`: `question1`, `question2` etc. or `mix1`, `mix2`, etc. See below the paragraphs about `question` and `mix`.

Exercise's layout

It works the same way as a Sheet's layout, with some differences:

- In `rowxcol`, the first number can be replaced by a `?`. In that case, the number of rows will be automatically calculated, depending on the number of questions and the number of columns.

Examples of exercise's layouts:

```
layout:
wordings: rowxcol=4×3
answers: rowxcol=4×3
```

will basically distribute the questions in 4 rows of 3 columns. Same for wordings and for answers.

```
layout:
wordings: rowxcol=?×3, colwidths=5 5 8, print=1 1 2
```

will distribute, only for wordings, the questions in 3 columns of widths 5 cm, 5 cm and 8 cm. There will be 1 question in the left cell of each row, 1 question in the middle cell of each row and 2 questions in the right cell of each row.

If you have 6 expressions, say A, B, C, D, E and F to distribute:

```
layout:
wordings: rowxcol=?×2
```

will distribute the questions in 2 columns of 3 rows, 1 question per row, i.e.:

A = B =

C = D =

E = F =

whereas:

```
layout:
wordings: rowxcol=?×2, print=3 / 3
```

will distribute the questions in 2 columns of 1 row, 3 questions per row, i.e.:

A = D =

B = E =

C = F =

The question key

Example of a simple question:

```
question: expand simple -> inttriplets_2to9 (5)
```


This question says: “I want 5 questions about expand a simple braces expression, the numbers being integers between 2 and 9”.

It is actually divided in two parts, separated by this arrow `->`. The first part concerns the kind of question and possibly its specific attributes, the second part concerns the numbers’ source to be used to create the question.

The question’s `id` and attributes

The left part of the scalar (string) matching a `question` key **must start** with two parts (words or `several_words`) separated by a space. This is the `id` of the question. Possible extra attributes can follow it, separated by commas (actually a comma and a space). Each extra attribute will be written as a pair `key=value`.

For instance:

```
question: calculation order_of_operations, subvariant=only_positive, spacing=15.0pt ->
↳ singleint_5to20;;intpairs_2to9, variant=5 (1)
```

In this example, a question of kind “calculation order_of_operations” will be created, with only positive numbers, spaced of 15.0pt.

The question’s `nb` and its attributes

The right part (after `->`) starts with the name of the numbers’ source (`intpairs_*to*`, `singleint_*to*` etc. see the already existing questions to know what to use, so far there’s no doc about them. Some questions require multiple sources, like the one in the example above, they’re written in row, joined by `;;`). It may be followed by attributes, just like the left part, and **must end** with an integer between braces, what is the number of questions to create with this numbers’ source.

The example above will create 1 question, variant number 5, and use the sources `singleint_5to20` and `intpairs_2to9`.

Note that you can put several different numbers’ sources inside one `question`. For instance:

```
questions: calculation order_of_operations, subvariant=only_positive, spacing=15.0pt -
↳> singleint_5to20;;intpairs_2to9, variant=5 (1)
-
↳> singleint_5to20;;intpairs_2to9, variant=7 (1)
```

or:

```
questions: expand simple -> inttriplets_2to9 (3)
-> inttriplets_5to15 (3)
```

This means there will be six questions, all being of “expand simple” kind, but the three first ones will use integers between 2 and 9; and the three last ones will use integers between 5 and 15.

The `mix` key

“Mixes” are primarily meant to allow to distribute numbers’ sources randomly on several questions types. This will only work if all numbers’ sources match all the questions of the same `mix`.

They can also be used to control the randomness of questions in an exercise. For instance, you want that the 3 first questions of an exercise are in random order, and then the 3 next ones too, but not all the 6 questions in random order. Then you can set two `mix` keys, say `mix1` and `mix2` and put 3 questions inside each `mix`.

The questions and numbers' sources inside `mix` are not displayed as in simple `question`, but under different keys, the ones starting by `question`, the others by `nb`.

```
- mix0:
  - question: calculation order_of_operations, subvariant=only_positive, pick=4, nb_
↪variant=decimal1, spacing=15.0pt
  - nb: singleint_5to20;;intpairs_2to9, variant=0, required=true (1)
      singleint_5to20;;intpairs_2to9, variant=1,3,5,7, required=true (1)
      singleint_5to20;;intpairs_2to9, variant=2,3,6,7, required=true (1)
      singleint_5to20;;intpairs_2to9, variant=0-7, required=true (1)
- mix1:
  - question: calculation order_of_operations, subvariant=only_positive, pick=6, nb_
↪variant=decimal1, spacing=15.0pt
  - nb: singleint_3to12;;intpairs_2to9, variant=8-23, required=true (2)
      singleint_3to12;;intpairs_2to9, variant=116-155, required=true (1)
      singleint_3to12;;intpairs_2to9, variant=156-187, required=true (1)
      singleint_3to12;;intpairs_2to9, variant=8-23,100-187 (2)
```

Note: Inside a `mix`, the `<question>`'s `pick` attribute tells how many times to create such a question. If unspecified, default value is 1. This attribute has no effect outside `mix` keys.

The rules to follow in a `mix` list are:

- Any numbers' source must be assignable to any of the questions of the section.
- Put at least as many numbers' sources as there are questions.

If you put more number's sources as there are questions, the extraneous ones will be ignored. This is useful when there are a lot of possibilities to pick from and you want to define special features to each of them, if chosen (like different number sources depending on variant or subvariant).

If among the sources you want to ensure there will be at least one of a certain type, you can set the `required` attribute of `nb` to `true`.

Also, note that the question's variant can be redefined as `nb`'s attribute (it overrides the one defined in `question`, if any).

Conclusion

Now the question is: how to know about the questions kinds and subkinds, and the possible contexts, variants or whatever other attributes? Well it is planned to add an easy way to know that (like a special directive) but there's nothing yet. The better, so far, may be to look at the provided sheets in `mathmaker/data/frameworks/` and see what's in there.

1.7 Install notes

1.7.1 eukleides install fix for FreeBSD

`eukleides` currently does not work out of the box. The `pkg`-installed version has a functional `euktoeps` script, it is required, so keep it somewhere. Then do `pkg remove eukleides` and re-install it from source:

- get the 1.5.4 source from <http://www.eukleides.org/>, for instance `wget http://www.eukleides.org/files/eukleides-1.5.4.tar.bz2`
- then `tar xvzf eukleides-1.5.4.tar.bz2`

- then possibly modify the prefix for install in the `Config` file, at your liking
- remove the making of documentation and manpages from the `install` target in the `Makefile` (they cause errors)
- install the required dependencies to compile `eukleides`: `pkg install bison flex gmake gcc`
- do `gmake` and then `gmake install`. This will provide functional binaries.
- replace the `euktoeps` script by the one you did get from the `pkg` installed version.
- if necessary (if `lualatex` complains about not finding `eukleides.sty`), reinstall `eukleides.sty` and `eukleides.tex` correctly:

```
# mkdir /usr/local/share/texmf-dist/tex/latex/eukleides
# cp /usr/local/share/texmf/tex/latex/eukleides/eukleides.* /usr/local/
↪share/texmf-dist/tex/latex/eukleides/
# mktexlsr
```

1.7.2 python-daemon error at install

You might get an error before the end of `mathmaker`'s installation:

```
error: The 'python-daemon>=2.1.1' distribution was not found and is required by_
↪mathmaker
```

or:

```
File "/home/nico/dev/mathmaker/venv/test071_bis/lib/python3.6/site-packages/
↪setuptools/command/easy_install.py", line 250, in finalize_options
    'dist_version': self.distribution.get_version(),
File "/tmp/easy_install-myl7eaei/python-daemon-2.1.2/version.py", line 656, in get_
↪version
File "/tmp/easy_install-myl7eaei/python-daemon-2.1.2/version.py", line 651, in get_
↪version_info
File "/tmp/easy_install-myl7eaei/python-daemon-2.1.2/version.py", line 552, in get_
↪changelog_path
File "/usr/lib/python3.6/posixpath.py", line 154, in dirname
    p = os.fspath(p)
TypeError: expected str, bytes or os.PathLike object, not NoneType
```

Fix it this way:

```
# pip3 install python-daemon --upgrade
```

And finish the install:

```
# pip3 install mathmaker
```


2.1 Guided tour

2.1.1 Foreword

This code has been developed chunk by chunk over more than 10 years now, starting with python2.3 or 4. Now it is a python3.6 software and my python skills have fortunately improved over the years. Problem is that several old parts, even after big efforts to correct the worst pieces, are not very idiomatic, especially the core.

Luckily there are unit tests, and whatever one might think about them, it's not difficult to admit that they're extremely useful to check nothing got broken when the core parts are written anew or debugged.

The documentation has been originally written using `doxygen`. Despite the fact it is an excellent documentation software, I have decided to move to `Sphinx` because it corresponds closer to python best practices. So, all doxygen-style comments will be turned into docstrings so mathmaker can use Sphinx to build the documentation. At the moment this work is just started, so the auto-generated Sphinx documentation is quite uncomplete now.

So, a part of the work to do is surely to bring new features, but another part, more annoying, is to turn ugly old parts into the right pythonic idioms. That's why at places you'll see that this or this other module is deprecated and should be "reduced", or rewritten. A list of such things to do is available on [sourceforge](#).

2.1.2 The issue

It is utmost important to understand that mathmaker is not a software intended to *compute* mathematical stuff, but to *display* it. For instance, resolving a first-degree equation is not in itself a goal of mathmaker, because other softwares do that already (and we don't even need any software to do it). Instead, mathmaker will determine and display the steps of this resolution. Whenever possible, mathmaker solutions will try to mimic the pupils' way of doing things.

For instance, it won't automatically simplify a fraction to make it irreducible in one step, but will try to reproduce the steps that pupils usually need to simplify the fraction. So the GCD is only used to check when the fraction is irreducible and for the cases where there's no other choice, but not as the mean to simplify a fraction directly (not before pupils learn how to use it, at least).

Another example is the need of mathmaker to control the displaying of decimal and integer numbers perfectly. Of course, most of the time, it doesn't matter if a computer tells that $5.2 \times 5.2 = 27.040000000000003$ or $3.9 \times 3.9 = 15.209999999999999$ because everyone knows that the correct results are 27.04 and 15.21 and because the difference is not so important, so in many situations, this precision will be sufficient. But, mathmaker can't display to pupils that the result of 5.2×5.2 is 27.040000000000003.

Also, the human rules we use to write maths are full of exceptions and odd details we don't notice usually because we're familiar to them. We would never write

$$+2x^2 + 1x - 1(+5 - 1x)$$

but instead

$$2x^2 + x - (5 - x)$$

There are many conventions in the human way to write maths and many exceptions.

These are the reasons why the core is quite complex: re-create these writing rules and habits on a computer and let the result be readable by pupils is not an easy thing.

2.1.3 Workflow

Mathmaker creates Sheets of maths Exercises.

Each Exercise contains Questions.

Each Question uses objects from the core, that embed enough information to compute and write the text of the Question and also the answer.

The main executable (`entry_point()` in `mathmaker/cli.py`) performs following steps:

- Load the default settings from configuration files.
- Setup the main logger.
- Check that the correct dependencies are installed.
- Parse the command-line arguments, updates the settings accordingly.
- Install the language and setup shared objects, like the database connection.
- If the main directive is `list`, it just write the directives list to stdout
- Otherwise, it checks that the directive matches a known sheet (either a yaml or xml file or a sheet's name that mathmaker provides) and writes the result to the output (`stdout` or a file) (xml will be dropped in 0.7.2)

2.1.4 The directories

Directories that are relevant to git, at the root:

```
.
├── docs
├── mathmaker
├── tests
└── toolbox
```

- The usual `docs/` and `tests/` directories
- `mathmaker/` contains the actual python source code
- `toolbox/` contains several standalone scripts that are useful for developers only (not users)

- Several usual files (`.flake8` etc.)
- `outfiles/` (not listed here, because it is not relevant to git) is where the garbage is put (figures created when testing, etc.). Sometimes it is useful to remove all garbage files it contains.

mathmaker/'s content:

```
$ tree -d -L 1 mathmaker -I __pycache__
mathmaker
├── data
├── lib
├── locale
└── settings
```

- `data/` is where the database is stored, but also yaml files containing additional wordings, translations etc.
- `lib/` contains all useful classes and submodules (see below).
- `locale/` contains all translation files.
- `settings/` contains the functions dedicated to setup the settings and also the default settings files themselves.

lib/'s content:

```
$ tree -d -L 3 mathmaker/lib -I __pycache__
mathmaker/lib
├── constants
├── core
├── document
│   ├── content
│   │   ├── algebra
│   │   ├── calculation
│   │   ├── geometry
│   │   └── ... (maybe some others in the future)
│   └── frames
├── machine
├── old_style_sheet
│   ├── exercise
│   └── question
└── tools
```

- `constants/` contains several constants (but `pythagorean.py` must be replaced by requests to the database)
- `core/` contains all mathematical objects, numeric or geometric
- `document/` contains the frames for sheets, exercises in questions, under `document/frames/`, and the questions' content, under `document/content/`.
- `machine/` contains the "typewriter"
- `old_style_sheet/` contains all old style sheets, exercises and questions. All of this is obsolete (will be replaced by generic objects that take their data from yaml files and created by the objects defined in `document/frames/`)
- `tools/` contains collections of useful functions
 - `__init__.py` contains various functions
 - `database.py` contains all functions required to interact with mathmaker's database
 - `frameworks.py` contains a collection of useful functions to handle the collection of yaml sheet files
 - `ignition.py` contains several functions called at startup

- `maths.py` contains some extra mathematical functions
- `wording.py` contains a collection of useful functions to handle wordings
- `xml.py` contains a collection of useful functions to handle the xml files (obsolete, will disappear in 0.7.2)
- `shared.py` contains objects and variables that need to be shared (except settings), like the database connection

2.1.5 Overview of the main classes

A Machine is like a typewriter: it turns all printable objects (Sheets, and everything they contain) into LaTeX. It knows how to turn a mathematical expression in LaTeX format. It knows how to draw figures from the geometrical objects (using `eukleides`).

The Sheet objects given to a Machine contain guidelines for the Machine: the layout of the Sheet and what Exercises it contains.

The Exercise objects contain Questions and also layout informations that might be specific to the exercise (for instance, display the equations' resolutions in two columns).

The Question objects contain the mathematical objects from the core and uses them to compute texts and answers. The real content is in `lib/document/content/*/*.py`. The appropriate module is used by the Question object (defined in `lib/document/frames/question.py`) to create the question's mathematical objects, wording and answer.

The objects from the core are all different kinds of mathematical objects, like Sums, Products, Equations or Triangles, Tables... For instance, a Question about Pythagora's theorem would embed a `RightTriangle` (which itself embeds information on its sides, vertices, angles; and enough methods to create a picture of it) but also fields telling if the figure should be drawn in the Question's text or if only a description of the figure should be given; if the hypotenuse should be calculated or another side; if the result should be a rounded decimal and how precise it should be etc.

When a new Sheet is created, all objects it contains are created randomly, following some rules, though, to avoid completely random uninteresting results.

More details about the core objects a little bit below, in the paragraph about *The core*.

2.2 Start working on mathmaker

2.2.1 Short version

Warning: The work is currently (0.7.1) done with python 3.6.

Install dependencies:

- Ubuntu:

```
$ sudo apt-get install eukleides libxml2-utils gettext texlive-full
```

- FreeBSD:

```
$ sudo pkg install python36 py36-sqlite3 gettext eukleides libxml2 texlive-full
$ rehash
```


And FreeBSD users should check the *eukleides install fix for FreeBSD*

To install mathmaker in dev mode in a venv, get to the directory where you want to work, and (assuming git and python3.6 are installed):

- Ubuntu:

```
$ python3 -m venv dev0
$ source dev0/bin/activate
(dev0) $ pip3 install pytest tox flake8 pydocstyle sphinx sphinx-autodoc-
↪annotation sphinx-rtd-theme
(dev0) $ mkdir mathmaker
(dev0) $ cd mathmaker/
(dev0) $ git clone https://github.com/nicolashainaux/mathmaker.git
(dev0) $ python3 setup.py develop
```

- FreeBSD:

```
$ python3 -m venv dev0
$ source dev0/bin/activate.csh
[dev0] $ sudo pip3 install pytest tox flake8 pydocstyle sphinx sphinx-autodoc-
↪annotation sphinx-rtd-theme
[dev0] $ mkdir mathmaker
[dev0] $ cd mathmaker/
[dev0] $ git clone https://github.com/nicolashainaux/mathmaker.git
[dev0] $ python3 setup.py develop
```

Usage: get to an empty directory and:

```
(dev0) $ mathmaker test_11_2 > out.tex
(dev0) $ lualatex out.tex
```

You can check out .pdf with the pdf viewer you like.

Run the tools:

```
(dev0) $ cd path/to/mathmaker/tools/
(dev0) $ ./build_db.py
(dev0) $ ./update_pot_files
```

Most of the tests are stored under `tests/`. Some others are doctests. Any new test or doctest will be added automatically to the tests run by `py.test` or `tox`.

Run the tests:

```
(dev0) $ py.test
(dev0) $ tox
```

Tox will ignore missing python interpreters.

Edit the settings:

```
(dev0) $ cd path/to/mathmaker/settings/
(dev0) $ mkdir dev/
(dev0) $ cp default/*.yaml dev/
```

In `dev/logging.yaml` you can set the `__main__` logger to INFO (take care to define log rotation for `/var/log/mathmaker`). Set the `dbg` logger to DEBUG.

Each debugging logger can be enabled/disabled individually in `debug_conf.yaml` (by setting it to DEBUG or INFO).

See *Loggers: main, daemon, debugging, output watcher* for more details on how to setup new loggers (and debugging loggers).

You can override settings in `dev/user_config.yaml` to your liking.

Before starting, you should read at least the *Auxiliary tools* and *Writing rules* sections. It is certainly worth also to have a look at *Advanced features*.

Hope you'll enjoy working on mathmaker!

2.2.2 Detailed version

Dev environment

Note: python3.6 is mandatory for mathmaker development

Install external dependencies

You'll need to install the same dependencies as users do (see *Install*). In addition, `xgettext` is required to extract the `gettext` messages from `py` files. In Ubuntu 14.04 it's in the `gettext` package.

Get mathmaker's source code from github repo

In the folder of your choice:

```
$ git clone https://github.com/nicolashainaux/mathmaker.git
```

Setup a python virtual environment

It is strongly advised to install mathmaker in develop mode inside of a python virtual environment. This allows to install the required libraries without conflicting with other projects or python software on the same computer. Just get to the directory of your choice, and to create a virtual environment named `dev0`, you type:

```
$ python3 -m venv dev0
```

From there, you can activate it:

on Ubuntu:

```
$ source dev0/bin/activate
```

on FreeBSD:

```
$ source dev0/bin/activate.csh
```

Install mathmaker

Once your virtual environment is activated, go to mathmaker's root:

```
(dev0) $ cd path/to/mathmaker/
```

You should see something like:

```
(dev0) $ ls
CHANGELOG.rst docs LICENSE MANIFEST.in mathmaker README.md README.rst
↪requirements.txt setup.py tests tools tox.ini
```

There you can install mathmaker in developer mode:

```
(dev0) $ python3 setup.py develop
```

It's possible to clean the project's main directory:

```
(dev0) $ python3 setup.py clean
```

Run mathmaker and tools

From now on, it is possible to run `mathmaker` from your virtual environment. As `mathmaker` is installed in developer mode, any change in the source files will be effective when running `mathmaker`. Go to a directory where you can leave temporary files (each sheet requiring pictures will produce picture files, by default), and test it:

```
(dev0) $ cd path/to/garbage/directory/
(dev0) $ mathmaker test_11_2 > out.tex
(dev0) $ lualatex out.tex
```

You can check out `.pdf` with the pdf viewer you like.

You can also run the tools:

```
(dev0) $ cd path/to/mathmaker/
(dev0) $ cd toolbox/
(dev0) $ ./build_db.py
(dev0) $ ./update_pot_files
```

Somewhat below, more informations about the *Auxiliary tools*.

Once you're done working with `mathmaker`, you can deactivate the virtual environment:

```
(dev0) $ deactivate
$
```

Note that it is possible to run `mathmaker` outside the virtual environment this way:

```
$ cd path/to/mathmaker/
$ python3 -m mathmaker.cli
```

But it requires to have installed the python dependencies yourself on the host system (e.g. the computer) and maybe also to have set `$PYTHONPATH` correctly (and exported it).

Other dependencies

Linters

It is recommended to install linters for PEP 8 and PEP 257 (see *Writing rules*):

```
(dev0) $ pip3 install flake8
(dev0) $ pip3 install pydocstyle
```

Test dependencies

In addition you should install at least `py.test`, and also `tox` if you intend to run `tox` tests:

```
(dev0) $ pip3 install pytest
(dev0) $ pip3 install tox
```

Below is more information about *testing*.

Documentation dependencies

You'll need to install these dependencies in the virtual environment:

```
(dev0) $ pip3 install sphinx sphinx-rtd-theme
```

`sphinx-rtd-theme` is the theme used for mathmaker's documentation. It's the `readthedocs` theme.

Note: `sphinx-autodoc-annotation` makes writing docstrings lighter when using python3 annotations. Problem is, this package currently has a bug that prevents to build the doc on `readthedocs`.

Below is more information about *documentation*.

Dev settings

You can make a copy of the default configuration files:

```
(dev0) $ cd path/to/mathmaker/
(dev0) $ cd settings/
(dev0) $ mkdir dev/
(dev0) $ cp default/*.yaml dev/
```

Then you can edit the files in `mathmaker/settings/dev/` to your liking. Any value redefined there will override all other settings (except the options from the command line).

In `logging.yaml` the `loggers` part is interesting. I usually set the `__main__` logger to `INFO` (this way, informations about starting and stopping mathmaker are recorded to `/var/log/mathmaker`, take care to define the log rotation if you do so) and the `dbg` logger to `DEBUG`. This second setting is important because it will allow to enable debugging loggers in `debug_conf.yaml`.

`debug_conf.yaml` allows to trigger each debugging logger individually by setting it to `DEBUG` instead of `INFO`.

And in `user_config.yaml` it is especially nice to define an output directory where all garbage files will be stored, but also to set the language, the font etc.

For instance, my `settings/dev/user_config.yaml` contains this:

```
# SOFTWARE'S CONFIGURATION FILE

PATHS:
  OUTPUT_DIR: /home/nico/dev/mathmaker/poubelle/
```

```

LOCALES:
    LANGUAGE: fr_FR
    CURRENCY: euro

LATEX:
    FONT: Ubuntu
    ROUND_LETTERS_IN_MATH_EXPR: True

DOCUMENT:
    QUESTION_NUMBERING_TEMPLATE_SLIDESHOWS: "n° {n}"

```

See [Settings](#) to learn more about the way settings are handled by mathmaker.

Testing

Run the tests

The testing suite is run by `py.test` this way:

```
(dev0) $ py.test
```

or this way:

```
(dev0) $ python3 setup.py test
```

Where do they live?

Most of the tests belong to `tests/`. Any function whose name starts with `test_` written in any python file whose name also starts with `test_` (and stored somewhere under `tests/`) and will be automatically added to the tests run by `py.test`.

Some more tests are written as [doctests](#) (see also [pytest documentation about doctests](#)) in the docstrings of the functions. It's possible to add doctests, especially for simple functions (sometimes it is redundant with the tests from `tests/`, but this is not a serious problem). The configuration for tests is so that any new doctest will be automatically added to the tests run by `py.test`.

Tox

To test mathmaker against different versions of python, you can run `tox` this way:

```
(dev0) $ tox
```

or this way:

```
(dev0) $ python3 setup.py tox
```

Be sure you have different versions of python installed correctly on your computer before starting this. The missing versions will be skipped anyway. Note that it is not a purpose of mathmaker to run under a lot of python versions (several python3 versions are OK, but no support for python2 is planned, unless someone really wants to do that).

Loggers: main, daemon, debugging, output watcher

See *Dev settings* to know how to use the settings files and enable or disable logging and debugging.

Main logger

`__main__` is intended to be used for messages relating to `mathmaker` general working. In particular, it should be used to log any error that forces `mathmaker` to stop, before it stops.

In order to use this `__main__` logger, you can write this at the start of any function (assuming you have imported settings at the top of the file):

```
log = settings.mainlogger
```

And then inside this function:

```
log.error("message")
```

(or `log.warning("message")` or `log.critical("message")` depending on the severity level).

If an Exception led to stop `mathmaker`, then the message should include its Traceback (if you notice this is not the case somewhere, you can modify this and make a pull request). For instance in `cli.py`:

```
try:
    shared.machine.write_out(str(sh))
except Exception:
    log.error("An exception occurred during the creation of the sheet.",
            exc_info=True)
    shared.db.close()
    sys.exit(1)
```

Daemon logger

This logger is intended to be used by the daemon script. Works the same way as the main logger.

Debugging logger

`dbg` is the logger dedicated to debugging and ready to use. No need to write `sys.stderr.write(msg)` anywhere.

If there's no logger object in the function you want to print debugging messages, you can create one this way:

- Add the matching entry in `debug_conf.yaml` (both the `settings/default/` and `settings/dev/` versions, but set to `INFO` in the `settings/default/` version). For short modules, you can add only one level, and for modules containing lots of functions or classes, two levels should be added, like the example of the extract below:

```
dbg:
  db: INFO
  wording:
    merge_nb_unit_pairs: INFO
    setup_wording_format_of: INFO
    insert_nonbreaking_spaces: INFO
  class_or_module_name:
    fct: DEBUG
```

- Import the settings at the top of the file, if it's not done yet:

```
from mathmaker import settings
```

- Create the logger at the start of the function (i.e. locally):

```
def fct():
    log = settings.dbg_logger.getChild('class_or_module_name.fct')
```

- Then where you need it, inside `fct`, write messages this way:

```
log.debug("the message you like")
```

Later when you need to disable this logger, you just set it to INFO instead of DEBUG in `settings/dev/debug_conf.yaml`. See *Dev settings* for information on these files.

A summary of the conventions used to represent the different core objects (i.e. what their `__repr__()` returns):

```
Value: .|
Item: {±| }
Item (strided): s{±| }

Product: <| × ... ×| > ^|
Sum: [| + ... +| ] ^|
Quotient: Q# ± (|/|) ^| #Q
Fraction: F# ± (|/|) ^| #F
Monomial: <<| × X ^| >> ^|
Expandable: <X:| ×| :X> ^|
BinomialIdentity: <B:| ×| :B> ^|
Polynomial: [| | + ... +| ] ^|
```

Output Watcher logger

This is another debugging logger. It can be used to check whether output is as expected, in order to detect bugs that do not crash mathmaker. Works the same way as the main logger. The log messages are sent to another facility (local4), in order to be recorded independently.

System log configuration

Systems using rsyslog

The communication with `rsyslog` goes through a local Unix socket (no need to load `rsyslog` TCP or UDP modules).

Note: The default socket is `/dev/log` for Linux systems, and `/var/run/log` for FreeBSD. These values are defined in the `logging*.yaml` settings files.

`rsyslog` may be already enabled and running by default (Ubuntu) or you can install, enable and start it (in Manjaro, `# systemctl enable rsyslog` and `# systemctl start rsyslog`).

Ensure `/etc/rsyslog.conf` contains these lines:

```
$ModLoad imuxsock
$IncludeConfig /etc/rsyslog.d/*.conf
```

Then create (if not created yet) a 'local' configuration file, like: `/etc/rsyslog.d/40-local.conf` and put (or add) in it:

```
# Local user rules for rsyslog.
#
#
local4.*                /var/log/mathmaker_output.log
local5.*                /var/log/mathmaker.log
local6.*                /var/log/mathmakerd.log
```

Then save it and restart:

- in Ubuntu: `# service rsyslog restart`
- in Manjaro: `# systemctl restart rsyslog`

Warning: Do not create `/var/log/mathmaker.log` yourself with the wrong rights, otherwise nothing will be logged.

To format the messages in a nicer way, it's possible to add this in `/etc/rsyslog.conf`:

```
$template MathmakerTpl,"%$now% %timegenerated:12:23:date-rfc3339% %syslogtag%%msg%\n"
```

and then, modify `/etc/rsyslog.d/40-local.conf` like:

```
local4.*                /var/log/mathmaker_output.log;MathmakerTpl
local5.*                /var/log/mathmaker.log;MathmakerTpl
local6.*                /var/log/mathmakerd.log;MathmakerTpl
```

Tools to check everything's fine: after having restarted rsyslog, enable some more informations output:

```
# export RSYSLOG_DEBUGLOG="/var/log/myrsyslogd.log"
# export RSYSLOG_DEBUG="Debug"
```

and running the configuration validation:

```
# rsyslogd -N2 | grep "mathmaker"
```

should show something like (errorless):

```
rsyslogd: version 7.4.4, config validation run (level 2), master config /etc/rsyslog.
↪conf
2564.153590773:7f559632b780: ACTION 0x2123160 [builtin:omfile:/var/log/mathmaker.
↪log;MathmakerTpl]
2564.154126386:7f559632b780: ACTION 0x2123990 [builtin:omfile:/var/log/mathmakerd.
↪log;MathmakerTpl]
2564.158461309:7f559632b780: ACTION 0x2123160 [builtin:omfile:/var/log/mathmaker.
↪log;MathmakerTpl]
2564.158729012:7f559632b780: ACTION 0x2123990 [builtin:omfile:/var/log/mathmakerd.
↪log;MathmakerTpl]
rsyslogd: End of config validation run. Bye.
```


Once you've checked this works as expected, do not forget to configure your log rotation.

Note: mathmaker does not support systemd journalisation (the default one in Manjaro). You may have to setup systemd too (enable `ForwardToSyslog` in its conf file) in order to get `rsyslog` recording messages. Also you may need to add `$ModLoad imjournal` in `/etc/rsyslog.conf` and to create the file `/var/spool/rsyslog`. For a better setup, see <https://www.freedesktop.org/wiki/Software/systemd/syslog/>. A workaround to prevent duplicate messages could be to discard the unwanted ones, like described here: <http://www.rsyslog.com/discarding-unwanted-messages/>.

Documentation

Current state

As stated in the *Foreword*, the documentation is being turned from doxygen to Sphinx, so there are missing parts .

Any new function or module has to be documented as described in [PEP 257](#).

The doxygen documentation for version 0.6 is [here](#). The core parts are still correct, so far.

Format

This documentation is written in [ReStructured Text](#) format.

There are no newlines inside paragraphs. Set your editor to wrap lines automatically to your liking.

Make html

To produce the html documentation:

```
(dev0) mathmaker [dev] $ $ cd docs/  
(dev0) mathmaker/docs [dev] $ $ make html
```

If modules have changed (new ones, deleted ones), it is necessary to rebuild the autogenerated index:

```
(dev0) mathmaker/docs [dev] $ sphinx-apidoc -f -o . ../mathmaker
```

Auxiliary tools

Several standalone scripts live in the `toolbox/` directory under root. They can be useful for several tasks that automate the handling of data.

The two most useful ones are both meant to be run from the `toolbox/` directory. They are:

- `build_db.py`, used to update the database when there are new entries to add in it. If new words of 4 letters are added to any po file, `build_db.py` should be run, it will add them to the database. If new wordings are entered in `mathmaker/data/wordings/*.xml` (obsolete: xml files will be replaced by yaml files up from 0.7.2), then it should be run too. See details in the docstring. And if a new table is required, it should be added in this script. For instance, the pythagorean triples should live in the database and will be added to this list soon or later.

- `update_pot_files`, a shell script making use of `xgettext` and of the scripts `merge_py_updates_to_main_pot_file`, `merge_yaml_updates_to_pot_file` and `merge_xml_updates_to_pot_file` (this last one will be removed in 0.7.2). Run `update_pot_files` to update `locale/mathmaker.pot` when new strings to translate have been added to python code (i.e. inside a call to `_()`) or new entries have been added to any yaml or xml (xml files will be turned to yaml files in 0.7.2) file from `mathmaker/data` (only entries matching a number of identifiers are taken into account, see `DEFAULT_KEYWORDS` in the source code to know which ones exactly).
- `build_index.py` will build the index of available sheets. Run it when you need to test a new sheet.

`import_msgstr` and `retrieve_po_entries` are useful on some rare occasions. See their docstrings for more explanations. They both have a `--help` option.

`pythagorean_triples_generator` shouldn't be of any use any more (later on maybe a part of its code will be incorporated to `build_db.py`, that's why it's still around here)

Writing rules

It is necessary to write the cleanest code possible. It has not been the case in the past, but the old code is updated chunk by chunk and **any new code portion must follow python's best practices**, to avoid adding to the mess, and so, must:

- Use idioms (to learn some, it is recommended to read Jeff Knupp's [Writing Idiomatic Python](#))
- Conform to the [PEP 8 – Style Guide for Python](#)
- Conform to the [PEP 257 – Docstring Conventions](#)

And of course, all the code is written in english.

As to PEP 8, mathmaker 's code being free from errors, the best is to use a linter, like `flake8`. They also exist as plugins to various text editors or IDE (see [Atom packages](#) for instance). Three error codes are ignored (see `.flake8`):

- E129 because it is triggered anytime a comment is used to separate a multiline conditional of an `if` statement from its nested suite. A choice has been made to wrap multiline conditions in `()` and realize the separation with next indented block using a `# __` comment (or any other comment if it's necessary) and this complies with PEP 8 (second option here):

Acceptable options in this situation include, but are not limited to:

```
# No extra indentation.
if (this_is_one_thing and
    that_is_another_thing):
    do_something()

# Add a comment, which will provide some distinction in editors
# supporting syntax highlighting.
if (this_is_one_thing and
    that_is_another_thing):
    # Since both conditions are true, we can frobnicate.
    do_something()
```

- W503 because PEP 8 does not compel to break before binary operators (the choice of breaking *after* binary operators has been done).
- E704 because on some occasions it is OK to put several *short* statements on one line in the case of `def`. It is the case in several test files using lines like `def v0(): return Value(4)`

Other choices are:

- A maximum line length of 79
- Declare `_` as builtin, otherwise all calls to `_()` (i.e. the translation function installed by `gettext`) would trigger flake8's error F821 (undefined name).
- No complexity check. This might change in the future, but the algorithms in the core are complex. It's not easy to make them more simple (if anyone wants to try, (s)he's welcome).
- Name modules, functions, instances, and other variables in lower case, whenever possible using a single word but if necessary, using `several_words_separated_with_underscores`.
- Name classes in CapitalizedWords, like: `SuchAWonderfullClass` (don't use `mixedCase`, like `wrongCapitalizedClass`).
- All `import` statements must be at the top of any module. It must be avoided to add `from ... import ...` at the top of some functions, but sometimes it's necessary. A solution to avoid this is always preferred.
- All text files (including program code) are encoded in UTF-8.

As to PEP 257, this is also a good idea to use a linter, but lots of documentation being written as doxygen comments, the linter will detect a lot of missing docstrings. Just be sure the part you intend to push does not introduce new PEP 257 errors (their number must decrease with time, never increase).

The text of any docstring is marked up with `reStructuredText`.

The module `mathmaker.lib.tools.wording` can be considered as a reference on how to write correct docstrings. As an example, the code of two functions is reproduced here.

Note: The use of python3's annotations and `sphinx-autodoc-annotation` would automatically add the types (including return type) to the generated documentation. If `sphinx-autodoc-annotation`'s bug is corrected, the `:type ...: ...` and `:rtype: ...` lines will be removed.

```
def cut_off_hint_from(sentence: str) -> tuple:
    """
    Return the sentence and the possible hint separated.

    Only one hint will be taken into account.

    :param sentence: the sentence to inspect
    :type sentence: str
    :rtype: tuple

    :Examples:

    >>> cut_off_hint_from("This sentence has no hint.")
    ('This sentence has no hint.', '')
    >>> cut_off_hint_from("This sentence has a hint: |hint:length_unit|")
    ('This sentence has a hint:', 'length_unit')
    >>> cut_off_hint_from("Malformed hint:|hint:length_unit|")
    ('Malformed hint:|hint:length_unit|', '')
    >>> cut_off_hint_from("Malformed hint: |hint0:length_unit|")
    ('Malformed hint: |hint0:length_unit|', '')
    >>> cut_off_hint_from("Two hints: |hint:unit| |hint:something_else|")
    ('Two hints: |hint:unit|', 'something_else')
    """
    last_word = sentence.split()[-1][0]
    hint_block = ""
    if (is_wrapped(last_word, braces='|'|')
        and last_word[1:-1].startswith('hint:')):
```

```
# __
hint_block = last_word
if len(hint_block):
    new_s = " ".join(w for w in sentence.split() if w != hint_block)
    hint = hint_block[1:-1].split(sep=':')[1]
    return (new_s, hint)
else:
    return (sentence, "")

def merge_nb_unit_pairs(arg: object):
    r"""
    Merge all occurrences of {nbN} {\*_unit} in arg.wording into {nbN\*_unit}.

    In the same time, the matching attribute arg.nbN\*_unit is set with
    Value(nbN, unit=Unit(arg.\*_unit)).into_str(display_SI_unit=True)
    (the possible exponent is taken into account too).

    :param arg: the object whose attribute wording will be processed. It must
        have a wording attribute as well as nbN and \*_unit attributes.
    :type arg: object
    :rtype: None

    :Example:

    >>> class Object(object): pass
    ...
    >>> arg = Object()
    >>> arg.wording = 'I have {nb1} {capacity_unit} of water.'
    >>> arg.nb1 = 2
    >>> arg.capacity_unit = 'L'
    >>> merge_nb_unit_pairs(arg)
    >>> arg.wording
    'I have {nb1_capacity_unit} of water.'
    >>> arg.nb1_capacity_unit
    '\\SI{2}{L}'
    """
```

Atom packages

This paragraph lists useful packages for atom users (visit the links to have full install and setup informations):

- flake8 linter provider: [linter-flake8](#) (Note: you should let the settings as is, except for the “Project config file” entry where you can write “.flake8” to use mathmaker project’s settings.)
- pydocstyle linter provider: [linter-pydocstyle](#)
- python3’s highlighter: [MagicPython](#) (MagicPython is able to highlight correctly python3’s annotations. You’ll have to disable the language-python core package.)
- To edit rst documentation: [language-restructuredtext](#) and [rst-preview-pandoc](#)

2.3 A deeper look in the source code

2.3.1 Settings

Everything happens in `mathmaker/settings/__init__.py` (it would be better to have everything happening rather in something like `mathmaker/settings/settings.py`, so this will most certainly change).

This module is imported by the main script at start, that run its `init()` function. After that, any subsequent `from mathmaker import settings` statement will make `settings.*` available.

The values shared as `settings.*` are: the paths to different subdirectories of the project, the loggers and the values read from *configuration* files. (Plus at the moment, two default values that should move to some other place).

None of these values is meant to be changed after it has been set by the main script, what calls `settings.init()` and then corrects some of them depending on the command-line options. Once this is done, these values can be considered actually as constants (they are not really constants as they are setup and corrected, so no UPPERCASE naming).

`tests/conftest.py` `` uses the ```settings` module the same way `mathmaker/cli.py` does.

2.3.2 Configuration

`load_config()` handles this and is defined in `mathmaker/lib/tools/__init__.py`. It works the same way for any of the `*.yaml` files. It first loads the default values from `mathmaker/settings/default/filename.yaml`. Then it updates any value found redefined in any of these files: `/etc/mathmaker/filename.yaml`, `~/.config/mathmaker/filename.yaml` and `mathmaker/settings/dev/filename.yaml`. Any missing file is skipped (except the first one: the default settings are part of the code, are shipped with it and must be present).

An extended dict class is used to deal easier with dicts created from yaml files. See in `mathmaker/lib/tools/__init__.py`.

2.3.3 The daemon

It's a daemonized web server that allows to communicate with mathmaker through http requests. See *http server (mathmakerd)*.

2.3.4 Shared

Three resources are shared: *the database*, the LaTeX machine and the sources.

`mathmaker/lib/shared.py` works a similar way as the `settings` module. It is initialized once in the main script and then its resources are used.

2.3.5 The database

The aim of the database is to avoid having to create a lot of randomly values and store them in lists or dicts everytime we need something.

It is considered as a source among others.

2.3.6 The sources

They concern as well numbers as words or letters or anything one can think of.

Note: Old style sheets don't use sources.

When random numbers are required, most of the time, we don't need complete random. For instance if we want a pair of integers for the multiplication tables between 2 and 9, we don't want to ask the same question twice in a row.

The sources manage this randomness. Anytime we need to use a source, we can use its `next()` method to get the next random data, without worrying in the same time whether it's the same as the previous one or not.

So we have sources for names, for words having a limited number of letters, for different kinds of numbers but also for mini-problems wordings.

So far, there are two kinds of sources: the ones that are provided by the database, and the ones that are provided by the function `generate_values()`. They both reside in `mathmaker/lib/tools/database.py`.

All sources are initialized in `mathmaker/lib/shared.py`. There you can see which one has its values provided by the database, which are the other ones.

The database provides an easy way to ensure the next value will be different from the last one: we simply “stamp” each drawn value and the next time we draw a value among the yet unstamped ones. When they're all stamped, we reset all stamps and redraw. There's a tiny possibility to draw two times in a row the same value, so far, but it's so tiny we can safely ignore it. (This could be improved later). The values drawn from `generate_values()` are so different the ones from the others that it's very unlikely to draw the same ones two times in a row.

2.3.7 The real and the fake translation files

`mathmaker/locale/mathmaker.pot` is a real translation file.

The other `mathmaker/locale/*.pot` files are “fake” ones. They are used to get random words in a specific language, but the words do not need to be the exact translation from a source language.

For instance, `w4l.pot` contains words of four different letters. It wouldn't make sense to translate the english word “BEAN” into a word of the same meaning AND having exactly four different letters, in another language. This wouldn't work for french, for instance. In general this would only work for rare exceptions (like “HALF” can be translated to “DEMI” in french).

The same applies to `feminine_names.pot` and `masculine_names.pot`. These files are used to get random names, but we don't need to *translate* them.

So, the entries in these “fake” translation files are only labeled entries, with nothing to translate.

A translator only needs to provide a number of entries (at least 10) in each of these files. No matter how many, no matter which `msgid` do they match. So: in `masculine_names.po` are several masculine names required, in `feminine_names.po` are several feminine names required and in `w4l.po` are several words of four unique letters required. Each time, at least 10, and then, the more the better.

2.3.8 The sheets, exercises and questions

There is still a bunch of “old-style” written sheets, that were not generated from yaml documents. I won't describe them thoroughly. They will disappear in the future, when they're replaced by their yaml counterparts. They are kept in `lib/old_style_sheet/`, so far. They use the classes `S_Structure` and `S_Generic`. `S_Structure` handles the layout of the sheet depending on the `SHEET_LAYOUT` dict you can find at the top of any sheet module.

Another bunch have been written in XML. They will disappear. So far, mathmaker can read the sheets data from a xml or a yaml document, but the xml format will be dropped in 0.7.2, so don't bother with it now.

So, all new sheets are stored in yaml files (under data/frameworks/theme/subtheme.yaml, for instance data/frameworks/algebra/expand.yaml).

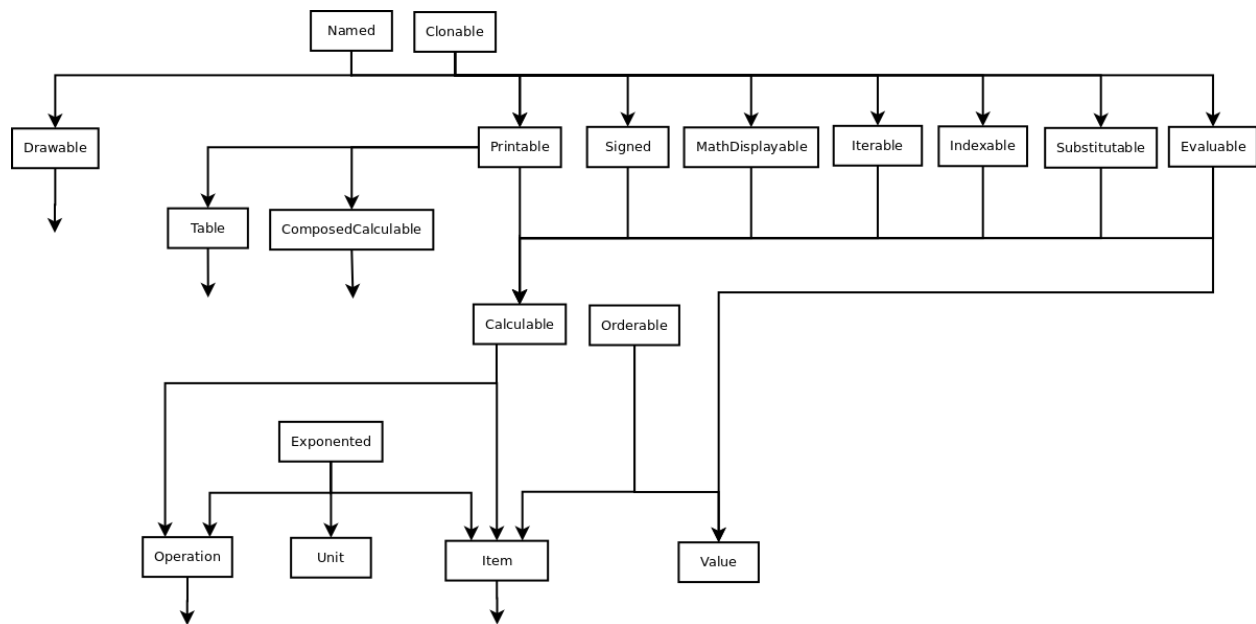
They are handled by sheet.py, exercise.py and question.py in lib/document/frames/.

2.3.9 The core

Diagram

You can check the 0.6 version (i.e. from doxygen) of the [top of the core diagram](#), though it will be somewhat changed later, it still can be used as reference for some time.

Unfinished draft of future plans:



Core objects' summary

Objects at left; associated `__repr()` at right:

Value : \pm nbjstr	Value : \square .
Item : \pm value ^{EXP}	Item : $(\pm \square^{\square})$
Product : $(\pm \text{Exponented} \times \dots \times \text{Exponented})^{\text{EXP}}$	Product : $\langle \square \times \dots \times \square \rangle^{\square}$
Sum : $(\pm \text{Exponented} + \dots + \text{Exponented})^{\text{EXP}}$	Sum : $[\square + \dots + \square]^{\square}$
Quotient : $\pm \left(\frac{\text{Exponented}}{\text{Exponented}} \right)^{\text{EXP}}$	Quotient : $Q \# \pm (\square / \square)^{\square} \# Q$
Fraction : $\pm \left(\frac{\text{Product}}{\text{Product}} \right)^{\text{EXP}}$	Fraction : $F \# \pm (\square / \square)^{\square} \# F$
Monomial : $(\text{Item} \times X^{\text{value}})^{\text{EXP}}$	Monomial : $\langle \langle \square \times X^{\square} \rangle \rangle^{\square}$
Expandable : $(\text{Sum} \times \text{Sum})^{\text{EXP}}$	Expandable : $\langle X : \square \times \square : X \rangle^{\square}$
BinomialIdentity : $(\text{Sum} \times \text{Sum})^{\text{EXP}}$	BinomialIdentity : $\langle B : \square \times \square : B \rangle^{\square}$
Polynomial : $(\pm \text{Monomial} + \dots + \text{Monomial})^{\text{EXP}}$	Polynomial : $[(\square + \dots + \square)]^{\square}$

Core objects' details

The “old” doc for 0.6 version is available [here](#) and mainly still correct for 0.7 version. When things will have settled down to something more stable, an updated documentation will be published chunk by chunk.

2.4 What can be done?

See the [tickets on sourceforge](#) and especially the ones for the 1.0 version.

2.5 mathmaker package

2.5.1 Subpackages

mathmaker.lib package

Subpackages

mathmaker.lib.constants package

Submodules

mathmaker.lib.constants.content module

mathmaker.lib.constants.latex module

mathmaker.lib.constants.numeration module

mathmaker.lib.constants.pythagorean module

`mathmaker.lib.constants.pythagorean.get_legs_matching_given_hypotenuse` (*side_length*)

`mathmaker.lib.constants.pythagorean.get_legs_matching_given_leg` (*side_length*)

mathmaker.lib.constants.units module

Module contents

mathmaker.lib.core package

Submodules

mathmaker.lib.core.base module

```
class mathmaker.lib.core.base.Clonable
    Bases: object
```



```

    clone ()
class mathmaker.lib.core.base.Drawable
    Bases: mathmaker.lib.core.base.NamedObject
    eps_filename
    euk_filename
    into_euk ()
    into_pic (create_pic_file=True)
    name
class mathmaker.lib.core.base.NamedObject
    Bases: mathmaker.lib.core.base.Clonable
    name
class mathmaker.lib.core.base.Printable
    Bases: mathmaker.lib.core.base.NamedObject
    into_str (**options)
    jsprinted
        Shortcut for self.into_str(force_expression_begins=True, js_repr=True)
        This returns the string of the Printable object, assuming it starts the expression.
    printed
        Shortcut for self.into_str(force_expression_begins=True)
        This returns the string of the Printable object, assuming it starts the expression.

```

mathmaker.lib.core.base_calculus module

```

class mathmaker.lib.core.base_calculus.AngleItem (raw_value=None, copy_this=None,
                                                    from_this_angle=None)
    Bases: mathmaker.lib.core.base_calculus.Item
    Represent Angles' names, like  $\widehat{ABC}$  (handled as Items).
    into_str (**options)
        Return the printed version of the AngleItem.
        If the AngleItem is literal, it will be displayed like a literal Item yet wrapped in a 'wide hat'. If it is not
        numeric, the unit will be automatically discarded.
class mathmaker.lib.core.base_calculus.BinomialIdentity (arg, **options)
    Bases: mathmaker.lib.core.base_calculus.Expandable
    a
        Gets the 'a' term of the BinomialIdentity
    b
        Gets the 'b' term of the BinomialIdentity
    expand ()
    expand_and_reduce_next_step (**options)
    get_a ()
    get_b ()

```

```
    get_kind()
    into_str (**options)
    kind
        kind of BinomialIdentity it, e.g. 'sum_square'|'difference_square'|'squares_difference'
class mathmaker.lib.core.base_calculus.CommutativeOperation
    Bases: mathmaker.lib.core.base_calculus.Operation
    append (elt)
    compact_display
        compact_display field of a CommutativeOperation
    contains_a_rounded_number ()
    contains_exactly (object)
    evaluate (**options)
    get_compact_display ()
    get_first_letter ()
    get_info ()
    get_sign ()
    info
        info field of a CommutativeOperation
    is_displ_as_a_single_int ()
    is_displ_as_a_single_neutral (neutral_elt)
    is_displ_as_a_single_numeric_Item ()
    remove (elt)
    set_compact_display (arg)
    set_info (arg)
    set_sign (arg)
    throw_away_the_neutrals ()
class mathmaker.lib.core.base_calculus.Division (arg, ignore_1_denominator=False,
                                                **options)
    Bases: mathmaker.lib.core.base_calculus.Quotient
    Same as Quotient, but using  $\div$  sign as default.
class mathmaker.lib.core.base_calculus.Expandable (arg, **options)
    Bases: mathmaker.lib.core.base_calculus.Product
    calculate_next_step (**options)
    expand (**options)
    expand_and_reduce_ ()
    expand_and_reduce_next_step (**options)
    is_expandable ()
```

```
class mathmaker.lib.core.base_calculus.Fraction(arg, ignore_1_denominator=False,
**options)
```

Bases: *mathmaker.lib.core.base_calculus.Quotient*

```
calculate_next_step(**options)
completely_reduced()
evaluate(**options)
expand_and_reduce_next_step(**options)
get_first_letter()
get_same_deno_reduction_in_progress()
get_simplification_in_progress()
get_status()
is_a_decimal_number()
is_reducible()
minimally_reduced(ignore_1_denominator=False)
    Return the simplest reduction step possible
reduced_by(n, ignore_1_denominator=False)
    Divide numerator and denominator by n.
replace_striked_out()
same_deno_reduction_in_progress
    Fraction's same_deno_reduction_in_progress field
set_down_numerator_s_minus_sign()
set_same_deno_reduction_in_progress(arg)
set_status(arg)
simplification_in_progress
    Fraction's simplification_in_progress status
simplification_line()
simplified()
status
    Fraction's status
```

```
class mathmaker.lib.core.base_calculus.Function(copy_this=None, name='f', var=
    {+x (unit=") ^1.}, fct=<function
    Function.<lambda>>, num_val=.1.,
    display_mode='literal', inv_fct=None,
    unlocked=False)
```

Bases: *mathmaker.lib.core.base_calculus.Item*

Represent the image of a number under a function like f(x) or cos(ABC).

argument

```
calculate_next_step(**options)
    Will only swap to numeric argument, no automatic evaluation.
```

```
contains_a_rounded_number()
    f(...) neither its argument never have any reason to be rounded.
```

contains_exactly (*object*)

True if the looked for object is self.

evaluate (***options*)

Return the value of f(number).

expand_and_reduce_next_step (***options*)

Same as calculate_nnext_step(), for Functions.

fct

The lambda function to use for evaluation.

get_first_letter ()

Return the first letter of Function's name.

get_iteration_list ()

Return [variable, exponent].

get_minus_signs_nb ()

1 if Function object has a negative sign and no even exponent, else 0.

image_notation

into_str (***options*)

Creates the str version of the Function.

inv_fct

The lambda function to use for evaluation.

is_displ_as_a_single_0 ()

f(x) is never a single 0.

is_displ_as_a_single_1 ()

f(x) is never a single 1.

is_displ_as_a_single_int ()

f(x) is never a single int.

is_displ_as_a_single_minus_1 ()

f(x) is never a single -1.

is_displ_as_a_single_neutral (*elt*)

f(x) is never a single neutral element.

is_displ_as_a_single_numeric_Item ()

f(x) is never a single numeric Item (like any single number).

is_expandable ()

f(x) is not expandable.

is_literal (*displ_as=False*) → bool

Return True if Function is to be considered literal.

Parameters displ_as – if displ_as is True, it's about knowing whether the object should be considered literal for display, otherwise, it's about knowing whether it can be numerically evaluated (directly, without replacing its variable by a Value).

is_null ()

True if self evaluates to 0.

is_numeric (*displ_as=False*)

Return True if current display mode is numeric.

Parameters **displ_as** – if `displ_as` is True, it's about knowing whether the object should be considered numeric for display, otherwise, it's about knowing whether it can be numerically evaluated (directly, without replacing its variable by a Value).

multiply_symbol_is_required (*object, position*)

True if \times is required between self and next object in a Product.

num_val

requires_brackets (*position*)

True if Function requires brackets inside a Product.

requires_inner_brackets ()

Return True for cases like $(-f(x))^2$

set_literal_mode ()

Set the display mode to 'literal'.

set_numeric_mode ()

Set the display mode to 'numeric'.

substitute (*subst_dict*)

Substitute the argument by its value, if available in `subst_dict`.

unlocked

var

class `mathmaker.lib.core.base_calculus.Item` (*arg, **options*)

Bases: `mathmaker.lib.core.root_calculus.Exponented`

calculate_next_step (***options*)

contains_a_rounded_number ()

contains_exactly (*object*)

convert_to (*unit*)

digits_number ()

evaluate (***options*)

expand_and_reduce_next_step (***options*)

force_display_sign_once

Item's `force_display_sign_once` field

get_first_letter ()

get_force_display_sign_once ()

get_is_out_striked ()

get_iteration_list ()

get_minus_signs_nb ()

get_raw_value ()

get_unit ()

get_value_inside ()

into_str (***options*)

is_displ_as_a_single_0 ()

is_displ_as_a_single_1 ()

is_displ_as_a_single_int ()

is_displ_as_a_single_minus_1 ()

is_displ_as_a_single_neutral (*elt*)

is_displ_as_a_single_numeric_Item ()

is_expandable ()

is_literal (*displ_as=False*) → bool

Return True if Item is to be considered literal.

Parameters **displ_as** – not applicable to Items

is_null ()

is_numeric (*displ_as=False*)

is_out_striked

Item's `is_out_striked` field

multiply_symbol_is_required (*object, position*)

needs_to_get_rounded (*precision*)

raw_value

Item's raw value

requires_brackets (*position*)

requires_inner_brackets ()

rounded (*precision*)

set_force_display_sign_once (*arg*)

set_is_out_striked (*arg*)

set_unit (*arg*)

set_value_inside (*arg*)

turn_into_fraction ()

unit

Unit of the Item

value_inside

Item's Value

class `mathmaker.lib.core.base_calculus.Monomial` (*arg, **options*)

Bases: `mathmaker.lib.core.base_calculus.Product`

coeff

Monomial's coefficient

degree

Monomial's degree

get_coeff ()

get_degree ()

get_first_letter ()

get_raw_value ()

```

get_sign()
get_value_inside()
is_negative()
is_null()
is_numeric(displ_as=False)
is_positive()
letter
    Monomial's letter
raw_value
    0-degree-Monomial's value
set_coeff(arg)
set_degree(arg)
set_letter(letter)
sign
    Monomial's sign
value_inside
    0-degree Monomial's Value inside

```

```

class mathmaker.lib.core.base_calculus.Operation
    Bases: mathmaker.lib.core.root_calculus.Exponented
element
    element field of Operation
get_element()
get_iteration_list()
get_neutral()
get_symbol()
is_expandable()
is_literal(displ_as=False) → bool
    Return True if Operation is to be considered literal.
    Parameters displ_as – not applicable to Operations
is_numeric(displ_as=False)
neutral
    neutral field of Operation
operator(arg1, arg2)
reset_element()
set_element(n, arg)
set_symbol(arg)
symbol
    symbol field of Operation

```

```
class mathmaker.lib.core.base_calculus.Polynomial (arg)
    Bases: mathmaker.lib.core.base_calculus.Sum

    degree
        Real degree of the Polynomial

    get_degree ()

    get_max_degree ()

class mathmaker.lib.core.base_calculus.Product (arg, compact_display=True)
    Bases: mathmaker.lib.core.base_calculus.CommutativeOperation

    calculate_next_step (**options)

    expand_and_reduce_next_step (**options)

    factor
        To access the factors of the Product.

    get_factors_list (given_kind)

    get_factors_list_except (object)

    get_first_factor ()

    get_minus_signs_nb ()

    into_str (**options)

    is_displ_as_a_single_0 ()

    is_displ_as_a_single_1 ()

    is_displ_as_a_single_minus_1 ()

    is_null ()

    is_reducible ()

    multiply_symbol_is_required (object, position)

    operator (arg1, arg2)

    order ()

    reduce_ ()

    requires_brackets (position)

    requires_inner_brackets ()

    set_factor (n, arg)

class mathmaker.lib.core.base_calculus.Quotient (arg, ignore_1_denominator=False,
                                                **options)
    Bases: mathmaker.lib.core.base_calculus.Operation

    calculate_next_step (**options)

    contains_a_rounded_number ()

    contains_exactly (object)

    denominator
        denominator field of Quotient

    evaluate (**options)
```

```

expand_and_reduce_next_step (**options)
    If possible, expands Quotient's numerator and/or denominator.

get_denominator ()

get_iteration_list ()

get_minus_signs_nb ()

get_numerator ()

get_sign ()

into_str (**options)

invert ()

is_displ_as_a_single_0 ()

is_displ_as_a_single_1 ()

is_displ_as_a_single_int ()

is_displ_as_a_single_minus_1 ()

is_displ_as_a_single_neutral (elt)

is_displ_as_a_single_numeric_Item ()

is_null ()

multiply_symbol_is_required (object, position)

numerator
    numerator field of Quotient

operator (arg1, arg2)

requires_brackets (position)

requires_inner_brackets ()

set_denominator (arg)

set_numerator (arg)

class mathmaker.lib.core.base_calculus.SquareRoot (arg, **options)
    Bases: mathmaker.lib.core.base_calculus.Function

calculate_next_step (**options)

contains_a_rounded_number ()

contains_exactly (object)

expand_and_reduce_next_step (**options)

force_display_sign_once
    Item's force_display_sign_once field

get_force_display_sign_once ()

get_iteration_list ()

get_minus_signs_nb ()

into_str (**options)

is_displ_as_a_single_0 ()

```

`is_displ_as_a_single_1()`
`is_displ_as_a_single_int()`
`is_displ_as_a_single_minus_1()`
`is_displ_as_a_single_neutral(elt)`
`is_displ_as_a_single_numeric_Item()`
`is_expandable()`
`is_literal(displ_as=False)`
Return True if SquareRoot is to be considered literal.

Parameters `displ_as` – not applicable to SquareRoots

`is_null()`
`is_numeric(displ_as=False)`
`multiply_symbol_is_required(object, position)`
`requires_brackets(position)`
`requires_inner_brackets()`
`set_force_display_sign_once(arg)`
`turn_into_fraction()`

class `mathmaker.lib.core.base_calculus.Sum(arg)`
Bases: `mathmaker.lib.core.base_calculus.CommutativeOperation`
`calculate_next_step(**options)`
`expand_and_reduce_next_step(**options)`
`force_inner_brackets_display`
force_inner_brackets_display field of a Sum
`get_force_inner_brackets_display()`
`get_literal_terms()`
`get_minus_signs_nb()`
`get_numeric_terms()`
`get_terms_lexicon()`
`intermediate_reduction_line()`
`into_str(**options)`
`is_displ_as_a_single_0()`
`is_displ_as_a_single_1()`
`is_displ_as_a_single_minus_1()`
`is_null()`
`is_reducible()`
`multiply_symbol_is_required(object, position)`
`next_displayable_term_nb(position)`
`numeric_terms_require_to_be_reduced()`

operator (*arg1, arg2*)
reduce_ ()
requires_brackets (*position*)
requires_inner_brackets ()
set_force_inner_brackets_display (*arg*)
set_term (*n, arg*)
term
 To access the terms of the Sum.

mathmaker.lib.core.base_geometry module

class mathmaker.lib.core.base_geometry.**Angle** (*arg, **options*)
 Bases: *mathmaker.lib.core.base.Drawable, mathmaker.lib.core.base.Printable*
into_euk ()
into_str (***options*)
label
label_display_angle
mark
measure
points
vertex

class mathmaker.lib.core.base_geometry.**Point** (*name=None, x=None, y=None*)
 Bases: *mathmaker.lib.core.base.Drawable*
into_euk ()
name
rotate (*center, angle, **options*)
x
x_exact
xy
y
y_exact

class mathmaker.lib.core.base_geometry.**Ray** (*arg, **options*)
 Bases: *mathmaker.lib.core.base.Drawable*
into_euk ()

class mathmaker.lib.core.base_geometry.**Segment** (*arg, **options*)
 Bases: *mathmaker.lib.core.base.Drawable*
dividing_points (*n=1, prefix='a'*)
 Create the list of Points that divide the Segment in n parts.

Parameters `n` (*int*) – the number of parts (so it will create $n - 1$ points) `n` must be greater or equal to 1

into_euk ()

invert_length_name ()

Swap points' names in the length name. E.g. AB becomes BA.

label

Label of the Segment (the displayed information).

label_into_euk ()

Return the label correctly positionned along the Segment.

length

Fake length of the Segment (the one used in a problem).

length_has_been_set

Whether the (fake) length has been set or not.

length_name

Length's name of the Segment, like AB.

mark

points

real_length

Real length (build length) of the Segment.

revert ()

setup_label (*flag*)

class `mathmaker.lib.core.base_geometry.Vector` (*arg*, ***options*)

Bases: `mathmaker.lib.core.base_geometry.Point`

bisector_vector (*arg*)

Return a vector colinear to the bisector of self and another vector.

Parameters `arg` (`Vector`) – the other vector

into_euk ()

norm

Return the norm of self.

orthogonal_unit_vector (*clockwise=True*)

Return a unit vector that's (default clockwise) orthogonal to self.

If clockwise is set to False, then the anti-clockwise orthogonal vector is returned.

slope

Return the slope of self.

unit_vector ()

Return the unit vector built from self

mathmaker.lib.core.calculus module

class `mathmaker.lib.core.calculus.ComposedCalculable`

Bases: `mathmaker.lib.core.base.Printable`

```

class mathmaker.lib.core.calculus.CrossProductEquation (arg)
    Bases: mathmaker.lib.core.calculus.Equation

    get_variable_obj ()

    get_variable_position ()

    solve_next_step (**options)

    variable_obj
        Variable object of the Equation

    variable_position
        Variable position in the Equation

class mathmaker.lib.core.calculus.Equality (objects, subst_dict=None, **options)
    Bases: mathmaker.lib.core.calculus.ComposedCalculable, mathmaker.lib.core.root_calculus.Substitutable

    content
        The content to be substituted (list containing literal objects).

    elements
        Elements of the object

    equal_signs
        Equal signs of the object

    get_elements ()

    get_equal_signs ()

    into_str (**options)

class mathmaker.lib.core.calculus.Equation (arg, **options)
    Bases: mathmaker.lib.core.calculus.ComposedCalculable

    auto_resolution (**options)

    get_left_hand_side ()

    get_number ()

    get_right_hand_side ()

    get_variable_letter ()

    into_str (**options)

    left_hand_side
        Left hand side of the Equation

    name

    number
        Number of the Equation

    right_hand_side
        Right hand side of the Equation

    set_hand_side (left_or_right, arg)

    set_number (arg)

    solve_next_step (**options)

```

variable_letter

Variable letter of the Equation

class mathmaker.lib.core.calculus.**Expression** (*integer_or_letter, object*)

Bases: *mathmaker.lib.core.calculus.ComposedCalculable*

auto_expansion_and_reduction (***options*)

get_right_hand_side ()

into_str (***options*)

right_hand_side

Right hand side of the object

set_right_hand_side (*arg*)

class mathmaker.lib.core.calculus.**QuotientsEquality** (*arg, displ_as_qe=True, ignore_1_denos=True, subst_dict=None*)

Bases: *mathmaker.lib.core.calculus.Table*

A shortcut to create Tables as quotients equalities.

class mathmaker.lib.core.calculus.**Table** (*arg, displ_as_qe=False, ignore_1_denos=None, subst_dict=None*)

Bases: *mathmaker.lib.core.base.Printable, mathmaker.lib.core.root_calculus.Substitutable*

class **SubstitutableList** (**args, subst_dict=None*)

Bases: *list, mathmaker.lib.core.root_calculus.Substitutable*

A list that can call substitute() on its elements.

content

The content to be substituted (list containing literal objects).

auto_resolution (*col0=0, col1=1, subst_dict=None, **options*)

cell

t.cell is the complete Table t.cell[i][j] is a cell

content

cross_product (*col, x_position, remove_1_deno=True, **options*)

displ_as_qe

get_cell ()

ignore_1_denos

into_crossproduct_equation (*col0=0, col1=1*) → *mathmaker.lib.core.calculus.CrossProductEquation*
Create a CrossProductEquation from two columns.

Ensure there is only one literal among the four cells before using it.

Parameters

- **col0** – the number of the first column to use
- **col1** – the number of the second column to use

into_str (*as_a_quotients_equality=None, ignore_1_denos=None, **options*)

is_numeric (*displ_as=False*)

class mathmaker.lib.core.calculus.**Table_UP** (*coeff, first_line, info, displ_as_qe=False*)

Bases: *mathmaker.lib.core.calculus.Table*

coeff

the coefficient of the Table_UP

crossproducts_info

infos about the cross products

get_coeff()

get_crossproducts_info()

into_crossproduct_equation (*arg*)

mathmaker.lib.core.geometry module

class mathmaker.lib.core.geometry.**InterceptTheoremConfiguration** (*points_names=None, butterfly=False, sketch=True, build_ratio=None, build_dimensions=None, rotate_around_isobarycenter='no'*)

Bases: *mathmaker.lib.core.geometry.Triangle*

butterfly

chunk

enlargement_ratio

into_euk (***options*)

Create the euk file content, as a str

point

ratios_equalities () → mathmaker.lib.core.calculus.Table

Return a Table matching the ratios equalities.

ratios_equalities_substituted () → mathmaker.lib.core.calculus.Table_UP

Return the ratios equalities containing known numbers.

It is returned as a Table_UP object.

ratios_for_converse () → mathmaker.lib.core.calculus.Table

Return a Table matching the ratios equality for converse.

set_lengths (*lengths_list, enlargement_ratio*)

Set all (“fake”) lengths of the figure.

The given lengths’ list matches the three small sides. The ratio will be used to compute all other segments’ sides. As these lengths are the “fake” ones (not the ones used to draw the figure, but the ones that will show up on the figure), this ratio is the “fake” one (not the same as self.ratio).

Parameters

- **lengths_list** (*a list (of Values)*) – the list of the lengths for small0, small1, small2
- **enlargement_ratio** (*any Evaluable*) – the enlargement ratio of the exercise.

small

u

v

class mathmaker.lib.core.geometry.**Polygon** (*arg*, *shapecolor=""*, ***options*)

Bases: *mathmaker.lib.core.base.Drawable*

angle

filename

into_euk ()

Build the euk file content.

Return type str

lengths_have_been_set

name

nature

perimeter

rename (*n*)

rotation_angle

set_lengths (*lengths_list*)

setup_labels (*flags_list*, *segments_list=None*)

Tells what to display along each segment of the list.

If no segments' list is provided, it defaults to the Polygon's sides' list. It is expected that both the flags' and segments' lists have the same length. Meaning of the flags' list: - a '?' will be displayed for each Segment flagged as None or '?' - its length will be displayed if it's flagged as anything else

evaluating to True

- nothing will be displayed if it's flagged as anything else evaluating to False

Parameters

- **flags_list** (*list*) – the list of the flags
- **segments_list** (*list (of Segments)*) – the list of the Segments to flag

side

vertex

class mathmaker.lib.core.geometry.**Rectangle** (*arg*, ***options*)

Bases: *mathmaker.lib.core.geometry.Polygon*

area

length

set_lengths (*lengths_list*)

width


```
class mathmaker.lib.core.geometry.RectangleGrid(arg, layout='2x2', fill='0x0', aut-
ofit=False, fillcolor='lightgray',
startvertex=None)
```

Bases: *mathmaker.lib.core.geometry.Rectangle*

fill (fill='0x0', fillcolor=None, startvertex=None)

```
class mathmaker.lib.core.geometry.RightTriangle(arg, **options)
```

Bases: *mathmaker.lib.core.geometry.Triangle*

hypotenuse

leg

pythagorean_equality (****options**)

pythagorean_substequality (****options**)

right_angle

```
setup_for_trigonometry(angle_nb=None, trigo_fct=None, angle_val=None,
up_length_val=None, down_length_val=None, length_unit=None,
only_mark_unknown_angle=False, mark_angle='simple')
```

Setup labels, determine subst_dict and stores configuration details.

Exactly one parameter among the three *_val ones must be left to None. According to the chosen trigo_fct and this parameter, this method will create the correct subst_dict.

Parameters

- **angle_nb** (*int*) – must be either 0 or 2 (index of an acute angle)
- **trigo_fct** (*str*) – must belong to ['cos', 'sin', 'tan']
- **angle_val** (*Value (or leave it to None to use it as the unknown value to calculate)*) – the angle's Value
- **up_length_val** (*Value (or leave it to None to use it as the unknown value to calculate)*) – the length's Value of the side that's at the numerator of the trigonometric formula
- **down_length_val** (*Value (or leave it to None to use it as the unknown value to calculate)*) – the length's Value of the side that's at the denominator of the trigonometric formula
- **length_unit** (*anything that can be used as argument for Units*) – the length's unit to use for lengths

```
side_adjacent_to(angle=None)
```

Return the side adjacent to given angle.

Parameters **angle** (*must be self.angle[0] or self.angle[2]*) – one of the acute angles

```
side_opposite_to(angle=None)
```

Return the side opposite to given angle.

Parameters **angle** (*must be self.angle[0] or self.angle[2]*) – one of the acute angles

```
trigonometric_equality(angle=None, trigo_fct=None, subst_dict=None, autosetup=False)
```

Return the required trigonometric equality.

Parameters

- **angle** (*Angle*) – the acute Angle to use

- **trigo_fct** (*str*) – either ‘cos’, ‘sin’ or ‘tan’
- **subst_dict** (*dict*) – a correct substitution dictionary
- **autosetup** (*bool*) – if enabled, will take the angle, **trigo_fct** and **subst_dict** from pre-configured values (requires to have called **setup_for_trigonometry()** previously).

trigonometric_ratios ()

Definitions of the three standard trigonometric ratios.

class mathmaker.lib.core.geometry.**Square** (*arg*, ***options*)

Bases: *mathmaker.lib.core.geometry.Polygon*

area

set_lengths (*lengths_list*)

set_marks (*arg*)

set_side_length (*side_length*)

side_length

class mathmaker.lib.core.geometry.**Triangle** (*arg*, ***options*)

Bases: *mathmaker.lib.core.geometry.Polygon*

mathmaker.lib.core.root_calculus module

class mathmaker.lib.core.root_calculus.**Calculable**

Bases: *mathmaker.lib.core.root_calculus.Evaluable*

calculate_next_step (***options*)

expand_and_reduce_next_step (***options*)

get_iteration_list ()

is_displ_as_a_single_0 ()

is_displ_as_a_single_1 ()

is_displ_as_a_single_int ()

is_displ_as_a_single_minus_1 ()

is_displ_as_a_single_neutral (*elt*)

is_displ_as_a_single_numeric_Item ()

multiply_symbol_is_required (*object*, *position*)

requires_brackets (*position*)

requires_inner_brackets ()

substitute (*subst_dict*)

class mathmaker.lib.core.root_calculus.**Evaluable**

Bases: *mathmaker.lib.core.base.Printable*

alphabetical_order_cmp (*other_object*)

contains_a_rounded_number ()

contains_exactly (*object*)

evaluate (***options*)

```

get_first_letter()
is_literal()
is_null()
is_numeric(displ_as=False)

```

class `mathmaker.lib.core.root_calculus.Exponented`
 Bases: `mathmaker.lib.core.root_calculus.Signed`

```

exponent
    Exponent of the Function
exponent_must_be_displayed()
get_exponent()
set_exponent(arg)

```

class `mathmaker.lib.core.root_calculus.Signed`
 Bases: `mathmaker.lib.core.root_calculus.Calculable`

```

get_minus_signs_nb()
get_sign()
is_negative()
is_positive()
set_opposite_sign()
set_sign(arg)
sign
    Sign of the object

```

class `mathmaker.lib.core.root_calculus.Substitutable`(*subst_dict=None*)
 Bases: `object`

Any object whose (literal) value(s) can be substituted by numeric ones.

Any Substitutable must define a `content` property, should include an optional `subst_dict` argument in its `__init__()` method and must ensure that a `_subst_dict` is defined (an easy way to do this is calling `Substitutable.__init__(self, subst_dict=subst_dict)`). The `substitute()` method is redefined by some Substitutable objects.

```

content
    The content to be substituted (list containing literal objects).
subst_dict
    Get the default dictionary to use for substitution.
substitute(subst_dict=None)
    If a subst_dict has been defined, it is used for literals substitution.

```

class `mathmaker.lib.core.root_calculus.Unit`(*arg, **options*)
 Bases: `mathmaker.lib.core.root_calculus.Exponented`

```

exponent
into_str(textwrap=True, js_repr=False, **options)
name

```

```
class mathmaker.lib.core.root_calculus.Value (arg, text_in_maths=True, **options)
```

```
    Bases: mathmaker.lib.core.root_calculus.Signed
```

```
    abs_value
```

```
    calculate_next_step (**options)
```

```
    contains_a_rounded_number ()
```

```
    contains_exactly (objct)
```

```
    digits_number ()
```

```
    evaluate (**options)
```

```
    get_first_letter ()
```

```
    get_has_been_rounded ()
```

```
    get_iteration_list ()
```

```
    get_sign ()
```

```
    get_unit ()
```

```
    has_been_rounded
```

```
        'has been rounded' state of the Value
```

```
    into_str (textwrap=True, js_repr=False, **options)
```

```
    is_a_perfect_square ()
```

```
    is_an_integer ()
```

```
    is_displ_as_a_single_0 ()
```

```
    is_displ_as_a_single_1 ()
```

```
    is_displ_as_a_single_int ()
```

```
    is_displ_as_a_single_minus_1 ()
```

```
    is_displ_as_a_single_numeric_Item ()
```

```
    is_literal (displ_as=False) → bool
```

```
        Return True if Value is to be considered literal.
```

```
        Parameters displ_as – not applicable to Values
```

```
    is_null ()
```

```
    is_numeric (displ_as=False)
```

```
    needs_to_get_rounded (precision)
```

```
    raw_value
```

```
    rounded (precision)
```

```
    set_has_been_rounded (arg)
```

```
    set_opposite_sign ()
```

```
    set_sign (arg)
```

```
    set_unit (arg)
```

```
    sign
```

```
        Sign of the Value
```

sqrt ()
substitute (*subst_dict*)
unit
 Unit of the Value

mathmaker.lib.core.utils module

mathmaker.lib.core.utils.**check_lexicon_for_substitution** (*objets*, *subst_dict*,
how_many)

mathmaker.lib.core.utils.**gather_literals** (*xpr*)
 Return all literal Values\AngleItems of an expression.

Parameters *xpr* (*Calculable*) – the expression to iter over

mathmaker.lib.core.utils.**put_term_in_lexicon** (*provided_key*, *associated_coeff*, *lexi*)

mathmaker.lib.core.utils.**reduce_literal_items_product** (*provided_list*)

Module contents

mathmaker.lib.machine package

Submodules

mathmaker.lib.machine.LaTeX module

class mathmaker.lib.machine.LaTeX.**LaTeX** (*language*, *create_pic_files=True*, ***options*)

Bases: *mathmaker.lib.machine.Structure.Structure*

addvspace (*height='30.0pt'*, ***options*)
 Add a vertical space.

create_table (*size*, *content*, ***options*)

insert_dashed_hline (***options*)

insert_nonbreaking_space (***options*)

insert_picture (*drawable_arg*, ***options*)

insert_vspace (***options*)

reset_exercises_counter ()

set_font_size_offset (*arg*)

set_redirect_output_to_str (*arg*)

translate_font_size (*arg*)

type_string (*objct*, ***options*)

write (*given_string*, ***options*)

write_document_begins (*variant='default'*)

write_document_ends ()

write_document_header (*variant='default'*)

`write_exercise_number()`

`write_frame(content, uncovered=False, only=False, duration=None, numbering="")`

Write a slideshow's frame to the output

Parameters

- **content** (*str*) – the frame's content
- **uncovered** (*bool*) – whether to split the content in several slides that will show one after the other. Mostly useful for title. The content's parts must be delimited by '|'. Difference with **only** is the text will be replaced, not only made invisible.
- **only** (*bool*) – whether to split the content in several slides that will show one after the other. Mostly useful for answers. The content's parts must be delimited by '|'. Difference with **uncovered** is the text will be replaced, not only made invisible.
- **duration** (*number (int or float)*) – the duration of the frame. If it's None, then no duration will be set.

Return type *str*

`write_jump_to_next_page()`

`write_layout(size, col_widths, content, **options)`

`write_math_style1(given_string)`

`write_math_style2(given_string, **kwargs)`

`write_new_line(check="", check2="", check3="", check4="")`

`write_new_line_twice(**options)`

`write_out(latex_document: str, pdf_output=False)`

Writes the given document to the output.

If `pdf_output` is set to `True` then the document will be compiled into a pdf and the pdf content will be written to output.

Parameters

- **latex_document** – contains the entire LaTeX document
- **pdf_output** – if `True`, output will be written in pdf format

`write_set_font_size_to(arg)`

mathmaker.lib.machine.Structure module

`class mathmaker.lib.machine.Structure.Structure(language)`

Bases: `object`

Abstract mother class of machine objects.

`clone(language)`

`create_table(size, content, **options)`

`insert_dashed_hline(**options)`

`insert_nonbreaking_space(**options)`

`insert_picture(drawable_arg, **options)`

`insert_vspace(**options)`

```

reset_exercises_counter ()
set_font_size_offset (arg)
set_redirect_output_to_str (arg)
translate_font_size (arg)
type_string (object, **options)
write (given_string, **options)
write_document_begins ()
write_document_ends ()
write_document_header ()
write_exercise_number ()
write_frame (content, uncovered=False, only=False, duration=None, numbering="")
    Write frame to the output
write_jump_to_next_page ()
write_layout (size, col_widths, content, **options)
write_math_style1 (given_string)
write_math_style2 (given_string)
write_new_line (**options)
write_new_line_twice (**options)
write_out (given_string, **options)
write_set_font_size_to (arg)

```

Module contents

mathmaker.lib.old_style_sheet package

Subpackages

mathmaker.lib.old_style_sheet.exercise package

Subpackages

mathmaker.lib.old_style_sheet.exercise.question package

Submodules

mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpressionExpansion module

class mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpressionExpansion.Q_Algebra

Bases: `mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure`

`answer_to_str()`

`text_to_str()`

mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpressionReduction module

class mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpressionReduction.Q_AlgebraExpressionReduction

Bases: `mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure`

`answer_to_str()`

`text_to_str()`

mathmaker.lib.old_style_sheet.exercise.question.Q_Calculation module

class mathmaker.lib.old_style_sheet.exercise.question.Q_Calculation.Q_Calculation(*q_kind='default_nothing'*, ***options*)

Bases: `mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure`

`answer_to_str()`

`text_to_str()`

mathmaker.lib.old_style_sheet.exercise.question.Q_Equation module

class mathmaker.lib.old_style_sheet.exercise.question.Q_Equation.Q_Equation(*q_kind='default_nothing'*, ***options*)

Bases: `mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure`

`answer_to_str()`

`text_to_str()`

mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization module

class mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization.Q_Factorization(*q_kind='default_nothing'*, ***options*)

Bases: `mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure`

`answer_to_str()`

`text_to_str()`

mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization.level_01(*q_subkind='default_nothing'*, ***options*)

`mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization.level_02` (*q_subkind*,
***options*)

`mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization.level_03` (*q_subkind*,
***options*)

`mathmaker.lib.old_style_sheet.exercise.question.Q_Model` module

`mathmaker.lib.old_style_sheet.exercise.question.Q_RightTriangle` module

class `mathmaker.lib.old_style_sheet.exercise.question.Q_RightTriangle.Q_RightTriangle` (*q_kind*,
***options*)

Bases: `mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure`

`answer_to_str()`

`text_to_str()`

`mathmaker.lib.old_style_sheet.exercise.question.Q_Structure` module

class `mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure` (*q_kind*,
AVAILABLE_Q_KIND_V,
***options*)

Bases: `object`

`answer_to_str(**options)`

`hint_to_str(**options)`

`text_to_str(**options)`

`to_str(ex_or_answers)`

Module contents

Submodules

`mathmaker.lib.old_style_sheet.exercise.X_AlgebraExpressionExpansion` module

class `mathmaker.lib.old_style_sheet.exercise.X_AlgebraExpressionExpansion.X_AlgebraExpress`

Bases: `mathmaker.lib.old_style_sheet.exercise.X_Structure.X_Structure`

mathmaker.lib.old_style_sheet.exercise.X_AlgebraExpressionReduction module

class mathmaker.lib.old_style_sheet.exercise.X_AlgebraExpressionReduction.X_AlgebraExpress

Bases: *mathmaker.lib.old_style_sheet.exercise.X_Structure.X_Structure*

mathmaker.lib.old_style_sheet.exercise.X_Calculation module

class mathmaker.lib.old_style_sheet.exercise.X_Calculation.X_Calculation (*x_kind='default_nothing'*,
***op-*
tions)

Bases: *mathmaker.lib.old_style_sheet.exercise.X_Structure.X_Structure*

mathmaker.lib.old_style_sheet.exercise.X_Equation module

class mathmaker.lib.old_style_sheet.exercise.X_Equation.X_Equation (*x_kind='default_nothing'*,
***op-*
tions)

Bases: *mathmaker.lib.old_style_sheet.exercise.X_Structure.X_Structure*

mathmaker.lib.old_style_sheet.exercise.X_Factorization module

class mathmaker.lib.old_style_sheet.exercise.X_Factorization.X_Factorization (*x_kind='default_no'*
***op-*
tions)

Bases: *mathmaker.lib.old_style_sheet.exercise.X_Structure.X_Structure*

mathmaker.lib.old_style_sheet.exercise.X_Model module

mathmaker.lib.old_style_sheet.exercise.X_RightTriangle module

class mathmaker.lib.old_style_sheet.exercise.X_RightTriangle.X_RightTriangle (*x_kind='default_no'*
***op-*
tions)

Bases: *mathmaker.lib.old_style_sheet.exercise.X_Structure.X_Structure*

mathmaker.lib.old_style_sheet.exercise.X_Structure module

class mathmaker.lib.old_style_sheet.exercise.X_Structure.X_Structure (*x_kind*,
AVAILABLE_X_KIND_VALUES,
X_LAYOUTS,
X_LAYOUT_UNIT,
num-
ber_of_questions=6,
***op-*
tions)

Bases: object

to_str (*ex_or_answers*)

Module contents

Submodules

mathmaker.lib.old_style_sheet.AlgebraBalance_01 module

```
class mathmaker.lib.old_style_sheet.AlgebraBalance_01.AlgebraBalance_01 (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.AlgebraBinomialIdentityExpansion module

```
class mathmaker.lib.old_style_sheet.AlgebraBinomialIdentityExpansion.AlgebraBinomialIdentityExpansion (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure

    write_answers ()
```

mathmaker.lib.old_style_sheet.AlgebraExpressionExpansion module

```
class mathmaker.lib.old_style_sheet.AlgebraExpressionExpansion.AlgebraExpressionExpansion (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.AlgebraExpressionReduction module

```
class mathmaker.lib.old_style_sheet.AlgebraExpressionReduction.AlgebraExpressionReduction (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.AlgebraFactorization_01 module

```
class mathmaker.lib.old_style_sheet.AlgebraFactorization_01.AlgebraFactorization_01 (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.AlgebraFactorization_02 module

```
class mathmaker.lib.old_style_sheet.AlgebraFactorization_02.AlgebraFactorization_02 (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.AlgebraFactorization_03 module

```
class mathmaker.lib.old_style_sheet.AlgebraFactorization_03.AlgebraFactorization_03 (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.AlgebraMiniTest0 module

```
class mathmaker.lib.old_style_sheet.AlgebraMiniTest0.AlgebraMiniTest0 (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.AlgebraMiniTest1 module

class mathmaker.lib.old_style_sheet.AlgebraMiniTest1.**AlgebraMiniTest1** (**options)
Bases: *mathmaker.lib.old_style_sheet.S_Structure.S_Structure*

mathmaker.lib.old_style_sheet.AlgebraShortTest module

class mathmaker.lib.old_style_sheet.AlgebraShortTest.**AlgebraShortTest** (**options)
Bases: *mathmaker.lib.old_style_sheet.S_Structure.S_Structure*

mathmaker.lib.old_style_sheet.AlgebraTest module

class mathmaker.lib.old_style_sheet.AlgebraTest.**AlgebraTest** (**options)
Bases: *mathmaker.lib.old_style_sheet.S_Structure.S_Structure*

mathmaker.lib.old_style_sheet.AlgebraTest2 module

class mathmaker.lib.old_style_sheet.AlgebraTest2.**AlgebraTest2** (**options)
Bases: *mathmaker.lib.old_style_sheet.S_Structure.S_Structure*

mathmaker.lib.old_style_sheet.ConverseAndContrapositiveOfPythagoreanTheoremShortTest module

class mathmaker.lib.old_style_sheet.ConverseAndContrapositiveOfPythagoreanTheoremShortTest
Bases: *mathmaker.lib.old_style_sheet.S_Structure.S_Structure*

mathmaker.lib.old_style_sheet.EquationsBasic module

class mathmaker.lib.old_style_sheet.EquationsBasic.**EquationsBasic** (**options)
Bases: *mathmaker.lib.old_style_sheet.S_Structure.S_Structure*

mathmaker.lib.old_style_sheet.EquationsClassic module

class mathmaker.lib.old_style_sheet.EquationsClassic.**EquationsClassic** (**options)
Bases: *mathmaker.lib.old_style_sheet.S_Structure.S_Structure*

mathmaker.lib.old_style_sheet.EquationsHarder module

class mathmaker.lib.old_style_sheet.EquationsHarder.**EquationsHarder** (**options)
Bases: *mathmaker.lib.old_style_sheet.S_Structure.S_Structure*

mathmaker.lib.old_style_sheet.EquationsShortTest module

class mathmaker.lib.old_style_sheet.EquationsShortTest.**EquationsShortTest** (**options)
Bases: *mathmaker.lib.old_style_sheet.S_Structure.S_Structure*

mathmaker.lib.old_style_sheet.EquationsTest module

```
class mathmaker.lib.old_style_sheet.EquationsTest.EquationsTest (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.FractionSimplification module

```
class mathmaker.lib.old_style_sheet.FractionSimplification.FractionSimplification (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.FractionsProductAndQuotient module

```
class mathmaker.lib.old_style_sheet.FractionsProductAndQuotient.FractionsProductAndQuotient
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.FractionsSum module

```
class mathmaker.lib.old_style_sheet.FractionsSum.FractionsSum (**options)
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.PythagoreanTheoremShortTest module

```
class mathmaker.lib.old_style_sheet.PythagoreanTheoremShortTest.PythagoreanTheoremShortTest
    Bases: mathmaker.lib.old_style_sheet.S_Structure.S_Structure
```

mathmaker.lib.old_style_sheet.S_Model module**mathmaker.lib.old_style_sheet.S_Structure module**

```
class mathmaker.lib.old_style_sheet.S_Structure.S_Structure (font_size_offset,
                                                         sheet_layout_unit,
                                                         sheet_layout, layout_type,
                                                         **options)
```

Bases: object

answers_title_to_str()

sheet_header_to_str()

sheet_text_to_str()

sheet_title_to_str()

texts_to_str (*ex_or_answers, n_of_first_ex*)

Module contents

mathmaker.lib.tools package

Submodules

mathmaker.lib.tools.database module

mathmaker.lib.tools.database.**classify_tag** (*tag*)

mathmaker.lib.tools.database.**db_table** (*tag*)

Table's name possibly associated to tag.

mathmaker.lib.tools.database.**generate_random_decimal_nb** (*position=None*,
width='random', *generation_type=None*,
pos_matches_invisible_zero=False,
unique_figures=True,
grow_left=False, *numberof=False*, *digits_positions=None*,
***unused*)

mathmaker.lib.tools.database.**generate_values** (*source_id*)

class mathmaker.lib.tools.database.**mc_source**

Bases: object

next (*source_id, qkw=None, **kwargs*)

mathmaker.lib.tools.database.**parse_sql_creation_query** (*qr*)

Retrieve table's name and columns' names from sql query.

mathmaker.lib.tools.database.**postprocess_decimalfractionssums_query** (*qr*,
qkw=None,
***kwargs*)

Create two decimal fractions from the drawn decimal number.

Parameters **qr** (*tuple*) – the result of the query (containing the decimal number)

Return type tuple

mathmaker.lib.tools.database.**postprocess_percentage_query** (*qr*, *source_id*,
*qkw=None, **kwargs*)

Create the two numbers from the query result, depending on *source_id*.

Parameters

- **qr** (*tuple*) – the result of the query (containing the number(s))
- **source_id** (*str*) – the original source id
- **qkw** (*dict*) – the question's keywords (attributes)

Return type tuple

mathmaker.lib.tools.database.**preprocess_decimalfractions_pairs_tag** (*qkw=None*,
***kwargs*)

Create the SQL query according to possible *qkw*'s overlap value.

Parameters **qkw** (*dict*) – keywords provided by the question

Return type dict

`mathmaker.lib.tools.database.preprocess_decimals_query` (*qkw=None*)

Create the SQL query according to possible *qkw*.

`mathmaker.lib.tools.database.preprocess_extdecimals_query` (*qkw=None*)

Create the SQL query according to possible *qkw*.

`mathmaker.lib.tools.database.preprocess_int_pairs_tag` (*tag, qkw=None*)

`mathmaker.lib.tools.database.preprocess_percentage_tag` (*tag, qkw=None*)

Deal with quarters, halves... numbers' sources.

As the initial tag (*source_id*) may be modified, it is returned along the tag to use, in first position, so all return statements are of the form return tag,...

`mathmaker.lib.tools.database.preprocess_qkw` (*table_name, qkw=None*)

Add relevant questions keywords to build the query.

`mathmaker.lib.tools.database.preprocess_single_nb_tag` (*tag*)

From single..._mintomax, get and return min and max in a dictionary.

`mathmaker.lib.tools.database.preprocess_units_pairs_tag` (*tag, last_draw=None, qkw=None*)

Create the SQL query according to *last_draw* content and possible *qkw*.

class `mathmaker.lib.tools.database.source` (*table_name, cols, **kwargs*)

Bases: object

next (***kwargs*)

class `mathmaker.lib.tools.database.sub_source` (*source_id, **kwargs*)

Bases: object

next (*qkw=None, **kwargs*)

mathmaker.lib.tools.frameworks module

`mathmaker.lib.tools.frameworks.build_exercises_list` (*data*)

Return the list of exercises from a sheet.

Parameters *data* (*dict*) – the sheet's data, as read from YAML file

Return type list

`mathmaker.lib.tools.frameworks.build_index` ()

Create the index of all (YAML) sheets available.

`mathmaker.lib.tools.frameworks.build_questions_list` (*data*)

Return the list of questions from an exercise.

Parameters *data* (*dict*) – the exercise's data, as read from YAML file (extract from the complete sheet's data)

Return type list

`mathmaker.lib.tools.frameworks.get_attributes` (*filename, tag*)

Gathers the "attributes" of all *filename*'s keys matching *tag*.

Parameters

- **filename** (*str*) – The YAML file name.
- **tag** (*str*) – The tag we're looking for.

Return type list

`mathmaker.lib.tools.frameworks.get_q_modifier(q_type, nb_source)`

`mathmaker.lib.tools.frameworks.list_all_sheets()`

Creates the list of all available mathmaker's sheets.

The list is displayed as a tabular.

Returns The list as str

`mathmaker.lib.tools.frameworks.load_sheet(theme, subtheme, sheet_name)`

Retrieve sheet data from yaml file.

Parameters

- **theme** (*str*) – the theme where to find the sheet
- **subtheme** (*str*) – the subtheme where to find the sheet
- **sheet_name** (*str*) – the name of the sheet

Return type OrderedDict

`mathmaker.lib.tools.frameworks.parse_qid(qid)`

Return question's kind and subkind from question's attribute "id".

`mathmaker.lib.tools.frameworks.read_index()`

Read the index of all (YAML) sheets available.

`mathmaker.lib.tools.frameworks.read_layout(data)`

Create the layout dictionary from the raw data.

Parameters **data** (*dict or list*) – the dictionary loaded from YAML file (might be list of dict)

Return type dict

mathmaker.lib.tools.ignition module

This module gathers functions that should be run at startup.

These functions check dependencies, settings consistency and setup the language for gettext translations.

`mathmaker.lib.tools.ignition.check_dependencies(euktoeps='euktoeps', xmlint='xmllint', lualatex='lualatex', luaotfload_tool='luaotfload-tool') → bool`

Will check all mathmaker's dependencies.

`mathmaker.lib.tools.ignition.check_dependency(name: str, goal: str, path_to: str, required_version_nb: str) → bool`

Will check if a dependency is installed plus its version number.

The version number is supposed to be displayed at the end of the line containing 'version' when calling `executable -version` (or the equivalent).

Parameters

- **name** (*str*) – the dependency's name.
- **goal** (*str*) – tells shortly why mathmaker needs it for
- **path_to** (*str*) – the path to the executable to test

- **required_version_nb** (*str*) – well, the required version number

Return type bool

`mathmaker.lib.tools.ignition.check_font()` → bool

Will check if settings.font belongs to data/fonts_list.txt.

It will first check if the exact name is in the list, then if one line of the list starts with the exact name.

`mathmaker.lib.tools.ignition.check_settings_consistency` (*language=None*,
od=None)

Will check the consistency of several settings values.

The checked values are: whether the language is supported as a LaTeX package that mathmaker uses, the output directory (is it an existing directory?) and whether the chosen font is usable by lualatex.

`mathmaker.lib.tools.ignition.install_gettext_translations` (**kwargs)

Will install output's language (gettext functions).

`mathmaker.lib.tools.ignition.warning_msg` (*name: str*, *path_to: str*, *c_out: str*, *c_err: str*,
gkw: str, *g_out: str*, *g_err: str*)

Return the formatted warning message.

Parameters

- **name** – name of the software
- **path_to** – the path to the software
- **c_out** – output of the call to *software -version*
- **c_err** – error output of the call to *software -version*
- **gkw** – keyword used to grep the version from output
- **g_out** – output of the call to *grep*...
- **g_err** – error output of the call to *grep*...

mathmaker.lib.tools.maths module

`mathmaker.lib.tools.maths.barycenter` (*points_list*, *barycenter_name*, *weights=None*)

`mathmaker.lib.tools.maths.coprime_generator` (*n*)

`mathmaker.lib.tools.maths.coprimes_to` (*n*, *span*)

List numbers coprime to n inside provided span.

Parameters

- **n** (*int*) – integer number
- **span** (*list*) – a list of integer numbers

Return type list

`mathmaker.lib.tools.maths.gcd` (*a*, *b*)

`mathmaker.lib.tools.maths.generate_decimal` (*width*, *places_scale*, *start_place*)

`mathmaker.lib.tools.maths.is_even` (*object*)

`mathmaker.lib.tools.maths.is_uneven` (*object*)

`mathmaker.lib.tools.maths.lcm` (*a*, *b*)

`mathmaker.lib.tools.maths.lcm_of_the_list` (*l*)

`mathmaker.lib.tools.maths.mean` (*numberList*, *weights=None*)

`mathmaker.lib.tools.maths.not_coprimes_to` (*n*, *span*, *exclude=None*)
List numbers NOT coprime to *n* inside provided span.

Parameters

- **n** (*int*) – integer number
- **span** (*list*) – a list of integer numbers
- **exclude** (*list*) – a list of number to always exclude from the results

Return type list

`mathmaker.lib.tools.maths.prime_factors` (*n*)
Return all the prime factors of a positive integer

Taken from <https://stackoverflow.com/a/412942/3926735>.

`mathmaker.lib.tools.maths.pupil_gcd` (*a*, *b*)

`mathmaker.lib.tools.maths.sign_of_product` (*signed_objctlist*)

`mathmaker.lib.tools.maths.ten_power_gcd` (*a*, *b*)

mathmaker.lib.tools.wording module

Use these functions to process sentences or objects containing a wording.

`mathmaker.lib.tools.wording.cut_off_hint_from` (*sentence: str*) → tuple
Return the sentence and the possible hint separated.

Only one hint will be taken into account.

Parameters **sentence** (*str*) – the sentence to inspect

Return type tuple

Examples

```
>>> cut_off_hint_from("This sentence has no hint.")
('This sentence has no hint.', '')
>>> cut_off_hint_from("This sentence has a hint: |hint:length_unit|")
('This sentence has a hint:', 'length_unit')
>>> cut_off_hint_from("Malformed hint:|hint:length_unit|")
('Malformed hint:|hint:length_unit|', '')
>>> cut_off_hint_from("Malformed hint: |hint0:length_unit|")
('Malformed hint: |hint0:length_unit|', '')
>>> cut_off_hint_from("Two hints: |hint:unit| |hint:something_else|")
('Two hints: |hint:unit|', 'something_else')
```

`mathmaker.lib.tools.wording.extract_formatting_tags_from` (*s: str*)
Return all tags found wrapped in {}. Punctuation has no effect.

Parameters **s** – the sentence where to look for {tags}.

`mathmaker.lib.tools.wording.handle_valueless_names_tags` (*arg: object*, *sentence: str*)
Each {name} tag found triggers an `arg.name` attribute to be randomly set.

All concerned tags are: {name}, {nameN}, {masculine_name}, {masculine_nameN}, {feminine_name}, {feminine_nameN}.

If the tag embeds a value, like in `{name=John}`, then it's ignored by this function. If `arg` already has an attribute matching the tag, then it's also ignored by this function.

Now, say `arg` has no attributes like `name`, `name1`, etc. then, if sentence contains:

- “`{name}`” then `arg.name` will receive a random name.
- “`{name1}`”, then `arg.name1` will receive a random name.
- “`{name=Michael}`”, then this function ignores it.
- “`{feminine_name}`”, then `arg.feminine_name` will get a random feminine name.

Parameters

- **arg** – the object that attributes must be checked and possibly set
- **sentence** – the sentence where to look for “name” tags.

`mathmaker.lib.tools.wording.handle_valueless_unit_tags` (*arg: object, sentence: str*)

Each `{*_unit}` tag triggers an `arg.*_unit` attribute to be randomly set.

For instance, if `{length_unit}` is found, then `arg.length_unit` will get a random length unit. Moreover, if `{area_unit}` or `{volume_unit}` are found, `arg.length_unit` is set accordingly too. If `arg.length_unit` does already exist, then `arg.area_unit` will be set accordingly (and not randomly any more).

`{*_unitN}`, `<*_unit>` and `<*_unitN>` tags will be handled the same way by this function.

If the tag embeds a value, like in `{capacity_unit=dL}`, then it's ignored by this function. If `arg` already has an attribute matching the tag, then it's also ignored by this function.

Parameters

- **arg** (*object*) – the object that attributes must be checked and possibly set
- **sentence** (*str*) – the sentence where to look for “unit” tags.

Return type None

`mathmaker.lib.tools.wording.insert_nonbreaking_spaces` (*sentence: str*)

Replace spaces by nonbreaking ones between a number and the next word.

Parameters **sentence** – the sentence to process

`mathmaker.lib.tools.wording.is_unit` (*word: str*) → bool

Return True if word is a “unit” tag (e.g. ends with `_unit`).

Punctuation has no effect.

Parameters **word** – the word to inspect

`mathmaker.lib.tools.wording.is_unitN` (*word*)

Return True if word is a “unitN” tag (e.g. ends with `_unitN`).

Punctuation has no effect.

Parameters **word** – the word to inspect

`mathmaker.lib.tools.wording.is_wrapped` (*word: str, braces='{}', extra_braces=""*) → bool

Return True if word is wrapped between braces.

Parameters

- **word** – the word to inspect
- **braces** – to change the default `{}` braces to something else,

like [] or <> :param extra_braces: to add extra braces around the usual ones. Like in ({{tag}}) or [{{tag}}] :Examples:

```
>>> is_wrapped('{word}')
True
>>> is_wrapped('{word},')
False
>>> is_wrapped('<word>')
False
>>> is_wrapped('<word>', braces='<>')
True
>>> is_wrapped('{{word}}')
False
>>> is_wrapped('{{word}}', extra_braces='()')
True
>>> is_wrapped('[{word}]', extra_braces='()')
False
```

mathmaker.lib.tools.wording.**is_wrapped_P**(word: str, braces='{}', extra_braces="") → bool
Return True if word is wrapped between braces & followed by a punctuation.

Parameters

- **word** – the word to inspect
- **braces** – to change the default {} braces to something else,

like [] or <> :param extra_braces: to add extra braces around the usual ones. Like in ({{tag}}) or [{{tag}}]

Examples

```
>>> is_wrapped_P('{word}')
False
>>> is_wrapped_P('{word},')
True
>>> is_wrapped_P('<word>')
False
>>> is_wrapped_P('<word>', braces='<>')
False
>>> is_wrapped_P('<word>:', braces='<>')
True
>>> is_wrapped_P('{{word}}', extra_braces='()')
False
>>> is_wrapped_P('{{word}}.', extra_braces='()')
True
>>> is_wrapped_P('[{word}]?', extra_braces='[]')
True
```

mathmaker.lib.tools.wording.**is_wrapped_p**(word: str, braces='{}', extra_braces="") → bool
Return True if word is wrapped between braces. Punctuation has no effect.

Parameters

- **word** – the word to inspect
- **braces** – to change the default {} braces to something else,

like [] or <> :param extra_braces: to add extra braces around the usual ones. Like in ({{tag}}) or [{{tag}}]

Examples

```

>>> is_wrapped_p('{word}')
True
>>> is_wrapped_p('{word},')
True
>>> is_wrapped_p('<word>')
False
>>> is_wrapped_p('<word>', braces='<>')
True
>>> is_wrapped_p('<word>:', braces='<>')
True
>>> is_wrapped_p('{{word}}.')
False
>>> is_wrapped_p('{{word}}.', extra_braces='()')
True
>>> is_wrapped_p('[{word}]?', extra_braces='[]')
True

```

`mathmaker.lib.tools.wording.merge_nb_unit_pairs` (*arg: object, w_prefix=""*)

Merge all occurrences of {nbN} {*_unit} in arg.wording into {nbN*_unit}.

In the same time, the matching attribute `arg.nbN*_unit` is set with `Value(nbN, unit=Unit(arg.*_unit)).into_str(display_SI_unit=True)` (the possible exponent is taken into account too).

Parameters `arg` (*object*) – the object whose attribute wording will be processed. It must have a wording attribute as well as `nbN` and `*_unit` attributes.

Return type None

Example

```

>>> class Object(object): pass
...
>>> arg = Object()
>>> arg.wording = 'I have {nb1} {capacity_unit} of water.'
>>> arg.nb1 = 2
>>> arg.capacity_unit = 'L'
>>> merge_nb_unit_pairs(arg)
>>> arg.wording
'I have {nb1_capacity_unit} of water.'
>>> arg.nb1_capacity_unit
'\\SI{2}{L}'

```

`mathmaker.lib.tools.wording.post_process` (*sentence: str*)

Apply all desired post processes to a sentence without {tags}.

So far, this is only the replacement of spaces following a number and preceding a word by nonbreaking spaces.

Parameters `sentence` – the sentence to post process

`mathmaker.lib.tools.wording.process_attr_values` (*sentence: str*) → tuple

Build a dict with all {key=val} occurrences in sentence. Update such tags.

All {key=val} occurrences will be replaced by {key}.

Parameters `sentence` – the sentence to process

Returns this couple: (transformed_sentence, {key:val, ... })

`mathmaker.lib.tools.wording.setup_wording_format_of` (*w_object: object, w_prefix=""*)

Set `w_object`'s attributes according to the tags found in `w_object.wording`.

This is the complete process of the wording. `w_object.wording` will also be modified in the process.

For instance, if `w_object.wording` is: “Here are one {name}, {nb1} {length_unit1} of roads, and a cube of {nb2} {volume_unit=cm}. What is the side’s length of the cube? |hint:length_unit|”

Then `w_object.wording` becomes: “Here are one {name}, {nb1_length_unit1} of roads, and a cube of {nb2_volume_unit}. What is the side’s length of the cube?”

`w_object.name` will be set with a random name,

`w_object.nb1_length_unit1` will be set with: `‘\SI{<value of nb1>}{<random length unit>}’`

`w_object.length_unit` will be set to centimeters

`w_object.nb2_volume_unit` will be set with: `‘\SI{<value of nb2>}{cm^{3}}’`

`w_object.hint` will be set with: `‘cm’`

If `w_prefix` is set, the “wording” processed attributes will be `w_object.<prefix>wording` and `w_object.<prefix>wording_format`. This allows to process several different wordings.

Parameters

- **w_object** – The object having a ‘wording’ attribute to process.
- **w_prefix** – The possible prefix of the “wording” attributes to

process.

`mathmaker.lib.tools.wording.unwrapped(word: str) → str`

Remove first and last char plus possible punctuation of word.

Examples

```
>>> unwrapped('{word}')
'word'
>>> unwrapped('{word},')
'word'
>>> unwrapped('{word}:')
'word'
```

`mathmaker.lib.tools.wording.wrap(word: str, braces='{}', o_str=None, e_str=None) → str`

Return the word wrapped between the two given strings.

Using `o_str` and `e_str` (e.g. opening str and ending str) will override braces content. It’s interesting when one want to wrap the word with something longer than a char.

Parameters

- **word** (`str`) – the chunk of text to be wrapped
- **braces** (`str`) – the pair of braces that will wrap the word
- **o_str** (`str`) – prefix the word.
- **e_str** (`str`) – suffix the word

Return type `str`

Examples

```
>>> wrap('wonderful')
'{wonderful}'
>>> wrap('wonderful', braces='<>')
'<wonderful>'
>>> wrap('wonderful', o_str='<<', e_str='>>')
'<<wonderful>>'
```

```
>>> wrap('wonderful', e_str='}*')
'{wonderful}*'
```

`mathmaker.lib.tools.wording.wrap_latex_keywords` (*s: str*) → str
 Replace some {kw} by {{kw}}, to prevent format() from using them as keys.

mathmaker.lib.tools.xml module

`mathmaker.lib.tools.xml.check_q_consistency` (*q_attrib, sources*)
 (Unfinished) Check the consistency of question's kind, subkind and source.

`mathmaker.lib.tools.xml.get_exercises_list` (*file_name*)
 Retrieves the exercises' list from *file_name*.

Parameters *file_name* (*str*) – The XML file name.

Return type list

`mathmaker.lib.tools.xml.get_sheet_config` (*file_name*)
 Retrieves the sheet configuration values from *file_name*.

Parameters *file_name* (*str*) – The XML file name.

Return type tuple

`mathmaker.lib.tools.xml.get_xml_schema_path` ()

`mathmaker.lib.tools.xml.get_xml_sheets_paths` ()

Returns all paths to default xml frameworks.

They are returned as a dictionary like: {id: path_to_matching_file.xml, ...} the id being the filename without its extension.

Return type dict

Module contents

Various auxiliary functions.

`mathmaker.lib.tools.check_unique_letters_words` (*words_list, n*)
 Check if each word of the list contains exactly n letters, all unique.

`mathmaker.lib.tools.difference_of_orders_of_magnitude` (*unit1, unit2*)
 Return the required power of 10 to multiply unit1 by, to get unit2.

Parameters

- **unit1** (*str*) – the first unit
- **unit2** (*str*) – the second unit (to get from unit1)

Return type decimal.Decimal

class `mathmaker.lib.tools.ext_dict`

Bases: dict

A dict with more methods.

flat (*sep='.'*)

Return a recursively flattened dict.

If the dictionary contains nested dictionaries, this function will return a one-level (“flat”) dictionary.

Example

```
>>> d = ext_dict({'a': {'a1': 3, 'a2': {'z': 5}}, 'b': 'data'})
>>> d.flat() == {'a.a1': 3, 'a.a2.z': 5, 'b': 'data'}
True
```

recursive_update (*d2*)

Update self with d2 key/values, recursively update nested dicts.

Example

```
>>> d = ext_dict({'a': 1, 'b': {'a': 7, 'c': 10}})
>>> d.recursive_update({'a': 24, 'd': 13, 'b': {'c': 100}})
>>> print(d == {'a': 24, 'd': 13, 'b': {'a': 7, 'c': 100}})
True
```

`mathmaker.lib.tools.fix_math_style2_fontsize` (*text*)

Turn all frac to dfrac.

Return type str

`mathmaker.lib.tools.generate_header_comment` (*document_format*, *comment_symbol*='%')

Return the header comment for output text files.

`mathmaker.lib.tools.load_config` (*file_tag*, *settingsdir*)

Will load the values from the yaml config file, named *file_tag*.yaml.

The default configuration values are loaded from `mathmaker/settings/default/.yaml`, then `load_config` will update with values found successively in `/etc/mathmaker/.yaml`, then in `~/config/mathmaker/.yaml`, finally in `mathmaker/settings/dev/.yaml`.

`mathmaker.lib.tools.parse_layout_descriptor` (*d*, *sep*=None, *special_row_chars*=None, *min_row*=0, *min_col*=0)

Parse a “layout” string, e.g. ‘3×4’. Return number of rows, number of cols.

Parameters

- **d** (*str*) – the “layout” string
- **sep** (*None or str or a list of str*) – the separator’s list. Default to ‘×’
- **special_row_chars** (*None or list*) – a list of special characters allowed instead of natural numbers. Defaults to []
- **min_row** (*positive int*) – a minimal value that the number of rows must respect. It is not checked is nrow is a special char
- **min_col** (*positive int*) – a minimal value that the number of columns must respect

Return type tuple

`mathmaker.lib.tools.physical_quantity` (*unit*)

Return the name of the physical quantity the unit belongs to.

Parameters *unit* (*str*) – the unit

Return type str

`mathmaker.lib.tools.po_file_get_list_of` (*what*, *language*, *arg*)

`mathmaker.lib.tools.retrieve_fonts` (*fonts_list_file*='mathmaker/data/fonts_list.txt', *datadir*='mathmaker/data', *force*=False) → tuple

Store in a file the list of the fonts available for lualatex.

`mathmaker.lib.tools.rotate(l, n)`
Rotate list `l` of `n` places, to the right if `n > 0`; else to the left.

Submodules

mathmaker.lib.shared module

`mathmaker.lib.shared.init()`

Module contents

mathmaker.settings package

Module contents

class `mathmaker.settings.ContextFilter` (*name=""*)

Bases: `logging.Filter`

Removes the 'dbg.' at the beginning of logged messages.

filter (*record*)

`mathmaker.settings.config_dbglogger(sd)`

Configures `dbg_logger`, using to the configuration file values.

class `mathmaker.settings.default_object`

Bases: `object`

`mathmaker.settings.init()`

class `mathmaker.settings.path_object` (***dirs*)

Bases: `object`

2.5.2 Submodules

2.5.3 mathmaker.cli module

`mathmaker.cli.entry_point()`

2.5.4 mathmaker.daemon module

class `mathmaker.daemon.MathmakerHTTPRequestHandler` (*request, client_address, server*)

Bases: `http.server.BaseHTTPRequestHandler`

do_GET ()

`mathmaker.daemon.run()`

2.5.5 Module contents

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

m

mathmaker.lib, 77

mathmaker.lib.constants, 36

mathmaker.lib.constants.content, 36

mathmaker.lib.constants.latex, 36

mathmaker.lib.constants.numeration, 36

mathmaker.lib.constants.pythagorean, 36

mathmaker.lib.constants.units, 36

mathmaker.lib.core, 57

mathmaker.lib.core.base, 36

mathmaker.lib.core.base_calculus, 37

mathmaker.lib.core.base_geometry, 47

mathmaker.lib.core.calculus, 48

mathmaker.lib.core.geometry, 51

mathmaker.lib.core.root_calculus, 54

mathmaker.lib.core.utils, 57

mathmaker.lib.machine, 59

mathmaker.lib.machine.LaTeX, 57

mathmaker.lib.machine.Structure, 58

mathmaker.lib.old_style_sheet, 66

mathmaker.lib.old_style_sheet.AlgebraBalance_01, 63

mathmaker.lib.old_style_sheet.AlgebraBinomialIdentityExpansion, 63

mathmaker.lib.old_style_sheet.AlgebraExpressionExpansion, 63

mathmaker.lib.old_style_sheet.AlgebraExpressionReduction, 63

mathmaker.lib.old_style_sheet.AlgebraFactorization_01, 63

mathmaker.lib.old_style_sheet.AlgebraFactorization_02, 63

mathmaker.lib.old_style_sheet.AlgebraFactorization_03, 63

mathmaker.lib.old_style_sheet.AlgebraMiniTest0, 63

mathmaker.lib.old_style_sheet.AlgebraMiniTest1, 64

mathmaker.lib.old_style_sheet.AlgebraShortTest, 64

mathmaker.lib.old_style_sheet.AlgebraTest, 64

mathmaker.lib.old_style_sheet.AlgebraTest2, 64

mathmaker.lib.old_style_sheet.ConverseAndContrapos, 64

mathmaker.lib.old_style_sheet.EquationsBasic, 64

mathmaker.lib.old_style_sheet.EquationsClassic, 64

mathmaker.lib.old_style_sheet.EquationsHarder, 64

mathmaker.lib.old_style_sheet.EquationsShortTest, 64

mathmaker.lib.old_style_sheet.EquationsTest, 65

mathmaker.lib.old_style_sheet.exercise, 63

mathmaker.lib.old_style_sheet.exercise.question, 63

mathmaker.lib.old_style_sheet.exercise.question.Q_A, 59

mathmaker.lib.old_style_sheet.exercise.question.Q_B, 60

mathmaker.lib.old_style_sheet.exercise.question.Q_C, 60

mathmaker.lib.old_style_sheet.exercise.question.Q_D, 60

mathmaker.lib.old_style_sheet.exercise.question.Q_E, 60

mathmaker.lib.old_style_sheet.exercise.question.Q_F, 61

mathmaker.lib.old_style_sheet.exercise.X_AlgebraExp, 61

mathmaker.lib.old_style_sheet.exercise.X_AlgebraExp, 61

mathmaker.lib.old_style_sheet.exercise.X_Calculati, 62

62
mathmaker.lib.old_style_sheet.exercise.X_Equation,
62
mathmaker.lib.old_style_sheet.exercise.X_Factorization,
62
mathmaker.lib.old_style_sheet.exercise.X_RightTriangle,
62
mathmaker.lib.old_style_sheet.exercise.X_Structure,
62
mathmaker.lib.old_style_sheet.FractionSimplification,
65
mathmaker.lib.old_style_sheet.FractionsProductAndQuotient,
65
mathmaker.lib.old_style_sheet.FractionsSum,
65
mathmaker.lib.old_style_sheet.PythagoreanTheoremShortTest,
65
mathmaker.lib.old_style_sheet.S_Structure,
65
mathmaker.lib.shared, 77
mathmaker.lib.tools, 75
mathmaker.lib.tools.database, 66
mathmaker.lib.tools.frameworks, 67
mathmaker.lib.tools.ignition, 68
mathmaker.lib.tools.maths, 69
mathmaker.lib.tools.wording, 70
mathmaker.lib.tools.xml, 75
mathmaker.settings, 77

A

- a (mathmaker.lib.core.base_calculus.BinomialIdentity attribute), 37
- abs_value (mathmaker.lib.core.root_calculus.Value attribute), 56
- addvspace() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- AlgebraBalance_01 (class in mathmaker.lib.old_style_sheet.AlgebraBalance_01), 63
- AlgebraBinomialIdentityExpansion (class in mathmaker.lib.old_style_sheet.AlgebraBinomialIdentityExpansion), 63
- AlgebraExpressionExpansion (class in mathmaker.lib.old_style_sheet.AlgebraExpressionExpansion), 63
- AlgebraExpressionReduction (class in mathmaker.lib.old_style_sheet.AlgebraExpressionReduction), 63
- AlgebraFactorization_01 (class in mathmaker.lib.old_style_sheet.AlgebraFactorization_01), 63
- AlgebraFactorization_02 (class in mathmaker.lib.old_style_sheet.AlgebraFactorization_02), 63
- AlgebraFactorization_03 (class in mathmaker.lib.old_style_sheet.AlgebraFactorization_03), 63
- AlgebraMiniTest0 (class in mathmaker.lib.old_style_sheet.AlgebraMiniTest0), 63
- AlgebraMiniTest1 (class in mathmaker.lib.old_style_sheet.AlgebraMiniTest1), 64
- AlgebraShortTest (class in mathmaker.lib.old_style_sheet.AlgebraShortTest), 64
- AlgebraTest (class in mathmaker.lib.old_style_sheet.AlgebraTest), 64
- AlgebraTest2 (class in mathmaker.lib.old_style_sheet.AlgebraTest2), 64
- alphabetical_order_cmp() (mathmaker.lib.core.root_calculus.Evaluable method), 54
- Angle (class in mathmaker.lib.core.base_geometry), 47
- angle (mathmaker.lib.core.geometry.Polygon attribute), 52
- AngleItem (class in mathmaker.lib.core.base_calculus), 37
- answer_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Algebra method), 59
- answer_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Algebra method), 60
- answer_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Calculus method), 60
- answer_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Equation method), 60
- answer_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization method), 60
- answer_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_RightTriangle method), 61
- answer_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Structure method), 61
- answers_title_to_str() (mathmaker.lib.old_style_sheet.S_Structure.S_Structure method), 65
- append() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
- area (mathmaker.lib.core.geometry.Rectangle attribute), 52
- area (mathmaker.lib.core.geometry.Square attribute), 54
- argument (mathmaker.lib.core.base_calculus.Function attribute), 39
- auto_expansion_and_reduction() (mathmaker.lib.core.calculus.Expression method), 50
- auto_resolution() (mathmaker.lib.core.calculus.Equation method), 49

- auto_resolution() (mathmaker.lib.core.calculus.Table method), 50
- ## B
- b (mathmaker.lib.core.base_calculus.BinomialIdentity attribute), 37
- barycenter() (in module mathmaker.lib.tools.maths), 69
- BinomialIdentity (class in mathmaker.lib.core.base_calculus), 37
- bisector_vector() (mathmaker.lib.core.base_geometry.Vector method), 48
- build_exercises_list() (in module mathmaker.lib.tools.frameworks), 67
- build_index() (in module mathmaker.lib.tools.frameworks), 67
- build_questions_list() (in module mathmaker.lib.tools.frameworks), 67
- butterfly (mathmaker.lib.core.geometry.InterceptTheoremConfiguration attribute), 51
- ## C
- Calculable (class in mathmaker.lib.core.root_calculus), 54
- calculate_next_step() (mathmaker.lib.core.base_calculus.Expandable method), 38
- calculate_next_step() (mathmaker.lib.core.base_calculus.Fraction method), 39
- calculate_next_step() (mathmaker.lib.core.base_calculus.Function method), 39
- calculate_next_step() (mathmaker.lib.core.base_calculus.Item method), 41
- calculate_next_step() (mathmaker.lib.core.base_calculus.Product method), 44
- calculate_next_step() (mathmaker.lib.core.base_calculus.Quotient method), 44
- calculate_next_step() (mathmaker.lib.core.base_calculus.SquareRoot method), 45
- calculate_next_step() (mathmaker.lib.core.base_calculus.Sum method), 46
- calculate_next_step() (mathmaker.lib.core.root_calculus.Calculable method), 54
- calculate_next_step() (mathmaker.lib.core.root_calculus.Value method), 56
- cell (mathmaker.lib.core.calculus.Table attribute), 50
- check_dependencies() (in module mathmaker.lib.tools.ignition), 68
- check_dependency() (in module mathmaker.lib.tools.ignition), 68
- check_font() (in module mathmaker.lib.tools.ignition), 69
- check_lexicon_for_substitution() (in module mathmaker.lib.core.utils), 57
- check_q_consistency() (in module mathmaker.lib.tools.xml), 75
- check_settings_consistency() (in module mathmaker.lib.tools.ignition), 69
- check_unique_letters_words() (in module mathmaker.lib.tools), 75
- chunk (mathmaker.lib.core.geometry.InterceptTheoremConfiguration attribute), 51
- classify_tag() (in module mathmaker.lib.tools.database), 66
- Clonable (class in mathmaker.lib.core.base), 36
- clone() (mathmaker.lib.core.base.Clonable method), 36
- clone() (mathmaker.lib.machine.Structure.Structure method), 58
- coeff (mathmaker.lib.core.base_calculus.Monomial attribute), 42
- coeff (mathmaker.lib.core.calculus.Table_UP attribute), 51
- CommutativeOperation (class in mathmaker.lib.core.base_calculus), 38
- compact_display (mathmaker.lib.core.base_calculus.CommutativeOperation attribute), 38
- completely_reduced() (mathmaker.lib.core.base_calculus.Fraction method), 39
- ComposedCalculable (class in mathmaker.lib.core.calculus), 48
- config_dbglogger() (in module mathmaker.settings), 77
- contains_a_rounded_number() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
- contains_a_rounded_number() (mathmaker.lib.core.base_calculus.Function method), 39
- contains_a_rounded_number() (mathmaker.lib.core.base_calculus.Item method), 41
- contains_a_rounded_number() (mathmaker.lib.core.base_calculus.Quotient method), 44
- contains_a_rounded_number() (mathmaker.lib.core.base_calculus.SquareRoot method), 45
- contains_a_rounded_number() (mathmaker.lib.core.root_calculus.Evaluable method), 56

- method), 54
- contains_a_rounded_number() (mathmaker.lib.core.root_calculus.Value method), 56
- contains_exactly() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
- contains_exactly() (mathmaker.lib.core.base_calculus.Function method), 39
- contains_exactly() (mathmaker.lib.core.base_calculus.Item method), 41
- contains_exactly() (mathmaker.lib.core.base_calculus.Quotient method), 44
- contains_exactly() (mathmaker.lib.core.base_calculus.SquareRoot method), 45
- contains_exactly() (mathmaker.lib.core.root_calculus.Evaluable method), 54
- contains_exactly() (mathmaker.lib.core.root_calculus.Value method), 56
- content (mathmaker.lib.core.calculus.Equality attribute), 49
- content (mathmaker.lib.core.calculus.Table attribute), 50
- content (mathmaker.lib.core.calculus.Table.SubstitutableList attribute), 50
- content (mathmaker.lib.core.root_calculus.Substitutable attribute), 55
- ContextFilter (class in mathmaker.settings), 77
- ConverseAndContrapositiveOfPythagoreanTheoremShortTest (class in mathmaker.lib.old_style_sheet.ConverseAndContrapositiveOfPythagoreanTheoremShortTest), 64
- convert_to() (mathmaker.lib.core.base_calculus.Item method), 41
- coprime_generator() (in module mathmaker.lib.tools.maths), 69
- coprimes_to() (in module mathmaker.lib.tools.maths), 69
- create_table() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- create_table() (mathmaker.lib.machine.Structure.Structure method), 58
- cross_product() (mathmaker.lib.core.calculus.Table method), 50
- CrossProductEquation (class in mathmaker.lib.core.calculus), 48
- crossproducts_info (mathmaker.lib.core.calculus.Table_UP attribute), 51
- cut_off_hint_from() (in module mathmaker.lib.tools.wording), 70
- ## D
- db_table() (in module mathmaker.lib.tools.database), 66
- default_object (class in mathmaker.settings), 77
- degree (mathmaker.lib.core.base_calculus.Monomial attribute), 42
- degree (mathmaker.lib.core.base_calculus.Polynomial attribute), 44
- denominator (mathmaker.lib.core.base_calculus.Quotient attribute), 44
- difference_of_orders_of_magnitude() (in module mathmaker.lib.tools), 75
- digits_number() (mathmaker.lib.core.base_calculus.Item method), 41
- digits_number() (mathmaker.lib.core.root_calculus.Value method), 56
- displ_as_qe (mathmaker.lib.core.calculus.Table attribute), 50
- dividing_points() (mathmaker.lib.core.base_geometry.Segment method), 47
- Division (class in mathmaker.lib.core.base_calculus), 38
- do_GET() (mathmaker.daemon.MathmakerHTTPRequestHandler method), 77
- Drawable (class in mathmaker.lib.core.base), 37
- ## E
- element (mathmaker.lib.core.base_calculus.Operation attribute), 43
- elements (mathmaker.lib.core.calculus.Equality attribute), 49
- enlargement_ratio (mathmaker.lib.core.geometry.InterceptTheoremConfiguration attribute), 51
- entry_point() (in module mathmaker.cli), 77
- eps_filename (mathmaker.lib.core.base.Drawable attribute), 37
- equal_signs (mathmaker.lib.core.calculus.Equality attribute), 49
- Equality (class in mathmaker.lib.core.calculus), 49
- Equation (class in mathmaker.lib.core.calculus), 49
- EquationsBasic (class in mathmaker.lib.old_style_sheet.EquationsBasic), 64
- EquationsClassic (class in mathmaker.lib.old_style_sheet.EquationsClassic), 64
- EquationsHarder (class in mathmaker.lib.old_style_sheet.EquationsHarder), 64
- EquationsShortTest (class in mathmaker.lib.old_style_sheet.EquationsShortTest), 64

- EquationsTest (class in mathmaker.lib.old_style_sheet.EquationsTest), 65
- euk_filename (mathmaker.lib.core.base.Drawable attribute), 37
- Evaluable (class in mathmaker.lib.core.root_calculus), 54
- evaluate() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
- evaluate() (mathmaker.lib.core.base_calculus.Fraction method), 39
- evaluate() (mathmaker.lib.core.base_calculus.Function method), 40
- evaluate() (mathmaker.lib.core.base_calculus.Item method), 41
- evaluate() (mathmaker.lib.core.base_calculus.Quotient method), 44
- evaluate() (mathmaker.lib.core.root_calculus.Evaluable method), 54
- evaluate() (mathmaker.lib.core.root_calculus.Value method), 56
- expand() (mathmaker.lib.core.base_calculus.BinomialIdentity method), 37
- expand() (mathmaker.lib.core.base_calculus.Expandable method), 38
- expand_and_reduce_() (mathmaker.lib.core.base_calculus.Expandable method), 38
- expand_and_reduce_next_step() (mathmaker.lib.core.base_calculus.BinomialIdentity method), 37
- expand_and_reduce_next_step() (mathmaker.lib.core.base_calculus.Expandable method), 38
- expand_and_reduce_next_step() (mathmaker.lib.core.base_calculus.Fraction method), 39
- expand_and_reduce_next_step() (mathmaker.lib.core.base_calculus.Function method), 40
- expand_and_reduce_next_step() (mathmaker.lib.core.base_calculus.Item method), 41
- expand_and_reduce_next_step() (mathmaker.lib.core.base_calculus.Product method), 44
- expand_and_reduce_next_step() (mathmaker.lib.core.base_calculus.Quotient method), 44
- expand_and_reduce_next_step() (mathmaker.lib.core.base_calculus.SquareRoot method), 45
- expand_and_reduce_next_step() (mathmaker.lib.core.base_calculus.Sum method), 46
- expand_and_reduce_next_step() (mathmaker.lib.core.root_calculus.Calculable method), 54
- Expandable (class in mathmaker.lib.core.base_calculus), 38
- exponent (mathmaker.lib.core.root_calculus.ExponentedOperation attribute), 55
- exponent (mathmaker.lib.core.root_calculus.Unit attribute), 55
- exponent_must_be_displayed() (mathmaker.lib.core.root_calculus.Exponented method), 55
- Exponented (class in mathmaker.lib.core.root_calculus), 55
- Expression (class in mathmaker.lib.core.calculus), 50
- ext_dict (class in mathmaker.lib.tools), 75
- extract_formatting_tags_from() (in module mathmaker.lib.tools.wording), 70
- ## F
- factor (mathmaker.lib.core.base_calculus.Product attribute), 44
- fct (mathmaker.lib.core.base_calculus.Function attribute), 40
- filename (mathmaker.lib.core.geometry.Polygon attribute), 52
- fill() (mathmaker.lib.core.geometry.RectangleGrid method), 53
- filter() (mathmaker.settings.ContextFilter method), 77
- fix_math_style2_fontsize() (in module mathmaker.lib.tools), 76
- flat() (mathmaker.lib.tools.ext_dict method), 75
- force_display_sign_once (mathmaker.lib.core.base_calculus.Item attribute), 41
- force_display_sign_once (mathmaker.lib.core.base_calculus.SquareRoot attribute), 45
- force_inner_brackets_display (mathmaker.lib.core.base_calculus.Sum attribute), 46
- Fraction (class in mathmaker.lib.core.base_calculus), 38
- FractionSimplification (class in mathmaker.lib.old_style_sheet.FractionSimplification), 65
- FractionsProductAndQuotient (class in mathmaker.lib.old_style_sheet.FractionsProductAndQuotient), 65
- FractionsSum (class in mathmaker.lib.old_style_sheet.FractionsSum), 65
- Function (class in mathmaker.lib.core.base_calculus), 39

G

- gather_literals() (in module mathmaker.lib.core.utils), 57
- gcd() (in module mathmaker.lib.tools.maths), 69
- generate_decimal() (in module mathmaker.lib.tools.maths), 69
- generate_header_comment() (in module mathmaker.lib.tools), 76
- generate_random_decimal_nb() (in module mathmaker.lib.tools.database), 66
- generate_values() (in module mathmaker.lib.tools.database), 66
- get_a() (mathmaker.lib.core.base_calculus.BinomialIdentity method), 37
- get_attributes() (in module mathmaker.lib.tools.frameworks), 67
- get_b() (mathmaker.lib.core.base_calculus.BinomialIdentity method), 37
- get_cell() (mathmaker.lib.core.calculus.Table method), 50
- get_coeff() (mathmaker.lib.core.base_calculus.Monomial method), 42
- get_coeff() (mathmaker.lib.core.calculus.Table_UP method), 51
- get_compact_display() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
- get_crossproducts_info() (mathmaker.lib.core.calculus.Table_UP method), 51
- get_degree() (mathmaker.lib.core.base_calculus.Monomial method), 42
- get_degree() (mathmaker.lib.core.base_calculus.Polynomial method), 44
- get_denominator() (mathmaker.lib.core.base_calculus.Quotient method), 45
- get_element() (mathmaker.lib.core.base_calculus.Operation method), 43
- get_elements() (mathmaker.lib.core.calculus.Equality method), 49
- get_equal_signs() (mathmaker.lib.core.calculus.Equality method), 49
- get_exercises_list() (in module mathmaker.lib.tools.xml), 75
- get_exponent() (mathmaker.lib.core.root_calculus.Exponented method), 55
- get_factors_list() (mathmaker.lib.core.base_calculus.Product method), 44
- get_factors_list_except() (mathmaker.lib.core.base_calculus.Product method), 44
- get_first_factor() (mathmaker.lib.core.base_calculus.Product method), 44
- get_first_letter() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
- get_first_letter() (mathmaker.lib.core.base_calculus.Fraction method), 39
- get_first_letter() (mathmaker.lib.core.base_calculus.Function method), 40
- get_first_letter() (mathmaker.lib.core.base_calculus.Item method), 41
- get_first_letter() (mathmaker.lib.core.base_calculus.Monomial method), 42
- get_first_letter() (mathmaker.lib.core.root_calculus.Evaluable method), 54
- get_first_letter() (mathmaker.lib.core.root_calculus.Value method), 56
- get_force_display_sign_once() (mathmaker.lib.core.base_calculus.Item method), 41
- get_force_display_sign_once() (mathmaker.lib.core.base_calculus.SquareRoot method), 45
- get_force_inner_brackets_display() (mathmaker.lib.core.base_calculus.Sum method), 46
- get_has_been_rounded() (mathmaker.lib.core.root_calculus.Value method), 56
- get_info() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
- get_is_out_striked() (mathmaker.lib.core.base_calculus.Item method), 41
- get_iteration_list() (mathmaker.lib.core.base_calculus.Function method), 40
- get_iteration_list() (mathmaker.lib.core.base_calculus.Item method), 41
- get_iteration_list() (mathmaker.lib.core.base_calculus.Operation method), 43
- get_iteration_list() (mathmaker.lib.core.base_calculus.Quotient method), 45
- get_iteration_list() (mathmaker.lib.core.base_calculus.SquareRoot method), 45
- get_iteration_list() (mathmaker.lib.core.root_calculus.Calculable method), 54

get_iteration_list()	(mathmaker.lib.core.root_calculus.Value method), 56	maker.lib.core.base_calculus.Monomial method), 42
get_kind()	(mathmaker.lib.core.base_calculus.BinomialIdentity method), 37	get_right_hand_side() (mathmaker.lib.core.calculus.Equation method), 49
get_left_hand_side()	(mathmaker.lib.core.calculus.Equation method), 49	get_right_hand_side() (mathmaker.lib.core.calculus.Expression method), 50
get_legs_matching_given_hypotenuse()	(in module mathmaker.lib.constants.pythagorean), 36	get_same_deno_reduction_in_progress() (mathmaker.lib.core.base_calculus.Fraction method), 39
get_legs_matching_given_leg()	(in module mathmaker.lib.constants.pythagorean), 36	get_sheet_config() (in module mathmaker.lib.tools.xml), 75
get_literal_terms()	(mathmaker.lib.core.base_calculus.Sum method), 46	get_sign() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
get_max_degree()	(mathmaker.lib.core.base_calculus.Polynomial method), 44	get_sign() (mathmaker.lib.core.base_calculus.Monomial method), 42
get_minus_signs_nb()	(mathmaker.lib.core.base_calculus.Function method), 40	get_sign() (mathmaker.lib.core.base_calculus.Quotient method), 45
get_minus_signs_nb()	(mathmaker.lib.core.base_calculus.Item method), 41	get_sign() (mathmaker.lib.core.root_calculus.Signed method), 55
get_minus_signs_nb()	(mathmaker.lib.core.base_calculus.Product method), 44	get_sign() (mathmaker.lib.core.root_calculus.Value method), 56
get_minus_signs_nb()	(mathmaker.lib.core.base_calculus.Quotient method), 45	get_simplification_in_progress() (mathmaker.lib.core.base_calculus.Fraction method), 39
get_minus_signs_nb()	(mathmaker.lib.core.base_calculus.SquareRoot method), 45	get_status() (mathmaker.lib.core.base_calculus.Fraction method), 39
get_minus_signs_nb()	(mathmaker.lib.core.base_calculus.Sum method), 46	get_symbol() (mathmaker.lib.core.base_calculus.Operation method), 43
get_minus_signs_nb()	(mathmaker.lib.core.base_calculus.Sum method), 46	get_terms_lexicon() (mathmaker.lib.core.base_calculus.Sum method), 46
get_minus_signs_nb()	(mathmaker.lib.core.root_calculus.Signed method), 55	get_unit() (mathmaker.lib.core.base_calculus.Item method), 41
get_neutral()	(mathmaker.lib.core.base_calculus.Operation method), 43	get_unit() (mathmaker.lib.core.root_calculus.Value method), 56
get_number()	(mathmaker.lib.core.calculus.Equation method), 49	get_value_inside() (mathmaker.lib.core.base_calculus.Item method), 41
get_numerator()	(mathmaker.lib.core.base_calculus.Quotient method), 45	get_value_inside() (mathmaker.lib.core.base_calculus.Monomial method), 43
get_numeric_terms()	(mathmaker.lib.core.base_calculus.Sum method), 46	get_variable_letter() (mathmaker.lib.core.calculus.Equation method), 49
get_q_modifier()	(in module mathmaker.lib.tools.frameworks), 68	get_variable_obj() (mathmaker.lib.core.calculus.CrossProductEquation method), 49
get_raw_value()	(mathmaker.lib.core.base_calculus.Item method), 41	get_variable_position() (mathmaker.lib.core.calculus.CrossProductEquation method), 49
get_raw_value()	(math-	get_xml_schema_path() (in module mathmaker.lib.tools.xml), 75

- get_xml_sheets_paths() (in module mathmaker.lib.tools.xml), 75
- ## H
- handle_valueless_names_tags() (in module mathmaker.lib.tools.wording), 70
- handle_valueless_unit_tags() (in module mathmaker.lib.tools.wording), 71
- has_been_rounded (mathmaker.lib.core.root_calculus.Value attribute), 56
- hint_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure method), 61
- hypotenuse (mathmaker.lib.core.geometry.RightTriangle attribute), 53
- ## I
- ignore_1_denos (mathmaker.lib.core.calculus.Table attribute), 50
- image_notation (mathmaker.lib.core.base_calculus.Function attribute), 40
- info (mathmaker.lib.core.base_calculus.CommutativeOperation attribute), 38
- init() (in module mathmaker.lib.shared), 77
- init() (in module mathmaker.settings), 77
- insert_dashed_hline() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- insert_dashed_hline() (mathmaker.lib.machine.Structure.Structure method), 58
- insert_nonbreaking_space() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- insert_nonbreaking_space() (mathmaker.lib.machine.Structure.Structure method), 58
- insert_nonbreaking_spaces() (in module mathmaker.lib.tools.wording), 71
- insert_picture() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- insert_picture() (mathmaker.lib.machine.Structure.Structure method), 58
- insert_vspace() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- insert_vspace() (mathmaker.lib.machine.Structure.Structure method), 58
- install_gettext_translations() (in module mathmaker.lib.tools.ignition), 69
- InterceptTheoremConfiguration (class in mathmaker.lib.core.geometry), 51
- intermediate_reduction_line() (mathmaker.lib.core.base_calculus.Sum method), 46
- into_crossproduct_equation() (mathmaker.lib.core.calculus.Table method), 50
- into_crossproduct_equation() (mathmaker.lib.core.calculus.Table_UP method), 51
- into_euk() (mathmaker.lib.core.base.Drawable method), 37
- into_euk() (mathmaker.lib.core.base_geometry.Angle method), 47
- into_euk() (mathmaker.lib.core.base_geometry.Point method), 47
- into_euk() (mathmaker.lib.core.base_geometry.Ray method), 47
- into_euk() (mathmaker.lib.core.base_geometry.Segment method), 48
- into_euk() (mathmaker.lib.core.base_geometry.Vector method), 48
- into_euk() (mathmaker.lib.core.geometry.InterceptTheoremConfiguration method), 51
- into_euk() (mathmaker.lib.core.geometry.Polygon method), 52
- into_pic() (mathmaker.lib.core.base.Drawable method), 37
- into_str() (mathmaker.lib.core.base.Printable method), 37
- into_str() (mathmaker.lib.core.base_calculus.AngleItem method), 37
- into_str() (mathmaker.lib.core.base_calculus.BinomialIdentity method), 38
- into_str() (mathmaker.lib.core.base_calculus.Function method), 40
- into_str() (mathmaker.lib.core.base_calculus.Item method), 41
- into_str() (mathmaker.lib.core.base_calculus.Product method), 44
- into_str() (mathmaker.lib.core.base_calculus.Quotient method), 45
- into_str() (mathmaker.lib.core.base_calculus.SquareRoot method), 45
- into_str() (mathmaker.lib.core.base_calculus.Sum method), 46
- into_str() (mathmaker.lib.core.base_geometry.Angle method), 47
- into_str() (mathmaker.lib.core.calculus.Equality method), 49
- into_str() (mathmaker.lib.core.calculus.Equation method), 49
- into_str() (mathmaker.lib.core.calculus.Expression method), 50
- into_str() (mathmaker.lib.core.calculus.Table method), 50
- into_str() (mathmaker.lib.core.root_calculus.Unit method), 55
- into_str() (mathmaker.lib.core.root_calculus.Value method), 56
- inv_fct (mathmaker.lib.core.base_calculus.Function at-

tribute), 40		maker.lib.core.base_calculus.Sum method), 46
invert() (mathmaker.lib.core.base_calculus.Quotient method), 45		is_displ_as_a_single_1() (mathmaker.lib.core.root_calculus.Calculable method), 54
invert_length_name() (mathmaker.lib.core.base_geometry.Segment method), 48		is_displ_as_a_single_1() (mathmaker.lib.core.root_calculus.Value method), 56
is_a_decimal_number() (mathmaker.lib.core.base_calculus.Fraction method), 39		is_displ_as_a_single_int() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
is_a_perfect_square() (mathmaker.lib.core.root_calculus.Value method), 56		is_displ_as_a_single_int() (mathmaker.lib.core.base_calculus.Function method), 40
is_an_integer() (mathmaker.lib.core.root_calculus.Value method), 56		is_displ_as_a_single_int() (mathmaker.lib.core.base_calculus.Item method), 42
is_displ_as_a_single_0() (mathmaker.lib.core.base_calculus.Function method), 40		is_displ_as_a_single_int() (mathmaker.lib.core.base_calculus.Quotient method), 45
is_displ_as_a_single_0() (mathmaker.lib.core.base_calculus.Item method), 41		is_displ_as_a_single_int() (mathmaker.lib.core.base_calculus.SquareRoot method), 46
is_displ_as_a_single_0() (mathmaker.lib.core.base_calculus.Product method), 44		is_displ_as_a_single_int() (mathmaker.lib.core.root_calculus.Calculable method), 54
is_displ_as_a_single_0() (mathmaker.lib.core.base_calculus.Quotient method), 45		is_displ_as_a_single_int() (mathmaker.lib.core.root_calculus.Value method), 56
is_displ_as_a_single_0() (mathmaker.lib.core.base_calculus.SquareRoot method), 45		is_displ_as_a_single_minus_1() (mathmaker.lib.core.base_calculus.Function method), 40
is_displ_as_a_single_0() (mathmaker.lib.core.base_calculus.Sum method), 46		is_displ_as_a_single_minus_1() (mathmaker.lib.core.base_calculus.Item method), 42
is_displ_as_a_single_0() (mathmaker.lib.core.root_calculus.Calculable method), 54		is_displ_as_a_single_minus_1() (mathmaker.lib.core.base_calculus.Product method), 44
is_displ_as_a_single_0() (mathmaker.lib.core.root_calculus.Value method), 56		is_displ_as_a_single_minus_1() (mathmaker.lib.core.base_calculus.Quotient method), 45
is_displ_as_a_single_1() (mathmaker.lib.core.base_calculus.Function method), 40		is_displ_as_a_single_minus_1() (mathmaker.lib.core.base_calculus.SquareRoot method), 46
is_displ_as_a_single_1() (mathmaker.lib.core.base_calculus.Item method), 41		is_displ_as_a_single_minus_1() (mathmaker.lib.core.base_calculus.Sum method), 46
is_displ_as_a_single_1() (mathmaker.lib.core.base_calculus.Product method), 44		is_displ_as_a_single_minus_1() (mathmaker.lib.core.root_calculus.Calculable method), 54
is_displ_as_a_single_1() (mathmaker.lib.core.base_calculus.Quotient method), 45		is_displ_as_a_single_minus_1() (mathmaker.lib.core.root_calculus.Value method), 56
is_displ_as_a_single_1() (mathmaker.lib.core.base_calculus.SquareRoot method), 45		is_displ_as_a_single_neutral() (math-
is_displ_as_a_single_1() (math-		

`maker.lib.core.base_calculus.CommutativeOperation` method), 43
`method`), 38
`is_displ_as_a_single_neutral()` (`math-`
`maker.lib.core.base_calculus.Function`
`method`), 40
`is_displ_as_a_single_neutral()` (`math-`
`maker.lib.core.base_calculus.Item` `method`),
42
`is_displ_as_a_single_neutral()` (`math-`
`maker.lib.core.base_calculus.Quotient`
`method`), 45
`is_displ_as_a_single_neutral()` (`math-`
`maker.lib.core.base_calculus.SquareRoot`
`method`), 46
`is_displ_as_a_single_neutral()` (`math-`
`maker.lib.core.root_calculus.Calculable`
`method`), 54
`is_displ_as_a_single_numeric_Item()` (`math-`
`maker.lib.core.base_calculus.CommutativeOperation`
`method`), 38
`is_displ_as_a_single_numeric_Item()` (`math-`
`maker.lib.core.base_calculus.Function`
`method`), 40
`is_displ_as_a_single_numeric_Item()` (`math-`
`maker.lib.core.base_calculus.Item` `method`),
42
`is_displ_as_a_single_numeric_Item()` (`math-`
`maker.lib.core.base_calculus.Quotient`
`method`), 45
`is_displ_as_a_single_numeric_Item()` (`math-`
`maker.lib.core.base_calculus.SquareRoot`
`method`), 46
`is_displ_as_a_single_numeric_Item()` (`math-`
`maker.lib.core.root_calculus.Calculable`
`method`), 54
`is_displ_as_a_single_numeric_Item()` (`math-`
`maker.lib.core.root_calculus.Value` `method`),
56
`is_even()` (in module `mathmaker.lib.tools.maths`), 69
`is_expandable()` (`mathmaker.lib.core.base_calculus.Expandable`
`method`), 38
`is_expandable()` (`mathmaker.lib.core.base_calculus.Function`
`method`), 40
`is_expandable()` (`mathmaker.lib.core.base_calculus.Item`
`method`), 42
`is_expandable()` (`mathmaker.lib.core.base_calculus.Operation`
`method`), 43
`is_expandable()` (`mathmaker.lib.core.base_calculus.SquareRoot`
`method`), 46
`is_literal()` (`mathmaker.lib.core.base_calculus.Function`
`method`), 40
`is_literal()` (`mathmaker.lib.core.base_calculus.Item`
`method`), 42
`is_literal()` (`mathmaker.lib.core.base_calculus.Operation`
`method`), 43
`is_literal()` (`mathmaker.lib.core.base_calculus.SquareRoot`
`method`), 46
`is_literal()` (`mathmaker.lib.core.root_calculus.Evaluable`
`method`), 55
`is_literal()` (`mathmaker.lib.core.root_calculus.Value`
`method`), 56
`is_negative()` (`mathmaker.lib.core.base_calculus.Monomial`
`method`), 43
`is_negative()` (`mathmaker.lib.core.root_calculus.Signed`
`method`), 55
`is_null()` (`mathmaker.lib.core.base_calculus.Function`
`method`), 40
`is_null()` (`mathmaker.lib.core.base_calculus.Item`
`method`), 42
`is_null()` (`mathmaker.lib.core.base_calculus.Monomial`
`method`), 43
`is_null()` (`mathmaker.lib.core.base_calculus.Product`
`method`), 44
`is_null()` (`mathmaker.lib.core.base_calculus.Quotient`
`method`), 45
`is_null()` (`mathmaker.lib.core.base_calculus.SquareRoot`
`method`), 46
`is_null()` (`mathmaker.lib.core.base_calculus.Sum`
`method`), 46
`is_null()` (`mathmaker.lib.core.root_calculus.Evaluable`
`method`), 55
`is_null()` (`mathmaker.lib.core.root_calculus.Value`
`method`), 56
`is_numeric()` (`mathmaker.lib.core.base_calculus.Function`
`method`), 40
`is_numeric()` (`mathmaker.lib.core.base_calculus.Item`
`method`), 42
`is_numeric()` (`mathmaker.lib.core.base_calculus.Monomial`
`method`), 43
`is_numeric()` (`mathmaker.lib.core.base_calculus.Operation`
`method`), 43
`is_numeric()` (`mathmaker.lib.core.base_calculus.SquareRoot`
`method`), 46
`is_numeric()` (`mathmaker.lib.core.calculus.Table`
`method`), 50
`is_numeric()` (`mathmaker.lib.core.root_calculus.Evaluable`
`method`), 55
`is_numeric()` (`mathmaker.lib.core.root_calculus.Value`
`method`), 56
`is_out_striked()` (`mathmaker.lib.core.base_calculus.Item` `at-`
`tribute`), 42
`is_positive()` (`mathmaker.lib.core.base_calculus.Monomial`
`method`), 43
`is_positive()` (`mathmaker.lib.core.root_calculus.Signed`
`method`), 55
`is_reducible()` (`mathmaker.lib.core.base_calculus.Fraction`
`method`), 39
`is_reducible()` (`mathmaker.lib.core.base_calculus.Product`

method), 44
 is_reducible() (mathmaker.lib.core.base_calculus.Sum method), 46
 is_uneven() (in module mathmaker.lib.tools.maths), 69
 is_unit() (in module mathmaker.lib.tools.wording), 71
 is_unitN() (in module mathmaker.lib.tools.wording), 71
 is_wrapped() (in module mathmaker.lib.tools.wording), 71
 is_wrapped_P() (in module mathmaker.lib.tools.wording), 72
 is_wrapped_p() (in module mathmaker.lib.tools.wording), 72
 Item (class in mathmaker.lib.core.base_calculus), 41

J

jsprinted (mathmaker.lib.core.base.Printable attribute), 37

K

kind (mathmaker.lib.core.base_calculus.BinomialIdentity attribute), 38

L

label (mathmaker.lib.core.base_geometry.Angle attribute), 47
 label (mathmaker.lib.core.base_geometry.Segment attribute), 48
 label_display_angle (mathmaker.lib.core.base_geometry.Angle attribute), 47
 label_into_euk() (mathmaker.lib.core.base_geometry.Segment method), 48
 LaTeX (class in mathmaker.lib.machine.LaTeX), 57
 lcm() (in module mathmaker.lib.tools.maths), 69
 lcm_of_the_list() (in module mathmaker.lib.tools.maths), 69
 left_hand_side (mathmaker.lib.core.calculus.Equation attribute), 49
 leg (mathmaker.lib.core.geometry.RightTriangle attribute), 53
 length (mathmaker.lib.core.base_geometry.Segment attribute), 48
 length (mathmaker.lib.core.geometry.Rectangle attribute), 52
 length_has_been_set (mathmaker.lib.core.base_geometry.Segment attribute), 48
 length_name (mathmaker.lib.core.base_geometry.Segment attribute), 48
 lengths_have_been_set (mathmaker.lib.core.geometry.Polygon attribute), 52
 letter (mathmaker.lib.core.base_calculus.Monomial attribute), 43

level_01() (in module mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization), 60
 level_02() (in module mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization), 60
 level_03() (in module mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization), 61
 list_all_sheets() (in module mathmaker.lib.tools.frameworks), 68
 load_config() (in module mathmaker.lib.tools), 76
 load_sheet() (in module mathmaker.lib.tools.frameworks), 68

M

mark (mathmaker.lib.core.base_geometry.Angle attribute), 47
 mark (mathmaker.lib.core.base_geometry.Segment attribute), 48
 mathmaker (module), 77
 mathmaker.cli (module), 77
 mathmaker.daemon (module), 77
 mathmaker.lib (module), 77
 mathmaker.lib.constants (module), 36
 mathmaker.lib.constants.content (module), 36
 mathmaker.lib.constants.latex (module), 36
 mathmaker.lib.constants.numeration (module), 36
 mathmaker.lib.constants.pythagorean (module), 36
 mathmaker.lib.constants.units (module), 36
 mathmaker.lib.core (module), 57
 mathmaker.lib.core.base (module), 36
 mathmaker.lib.core.base_calculus (module), 37
 mathmaker.lib.core.base_geometry (module), 47
 mathmaker.lib.core.calculus (module), 48
 mathmaker.lib.core.geometry (module), 51
 mathmaker.lib.core.root_calculus (module), 54
 mathmaker.lib.core.utils (module), 57
 mathmaker.lib.machine (module), 59
 mathmaker.lib.machine.LaTeX (module), 57
 mathmaker.lib.machine.Structure (module), 58
 mathmaker.lib.old_style_sheet (module), 66
 mathmaker.lib.old_style_sheet.AlgebraBalance_01 (module), 63
 mathmaker.lib.old_style_sheet.AlgebraBinomialIdentityExpansion (module), 63
 mathmaker.lib.old_style_sheet.AlgebraExpressionExpansion (module), 63
 mathmaker.lib.old_style_sheet.AlgebraExpressionReduction (module), 63
 mathmaker.lib.old_style_sheet.AlgebraFactorization_01 (module), 63
 mathmaker.lib.old_style_sheet.AlgebraFactorization_02 (module), 63

- mathmaker.lib.old_style_sheet.AlgebraFactorization_03 (module), 63
- mathmaker.lib.old_style_sheet.AlgebraMiniTest0 (module), 63
- mathmaker.lib.old_style_sheet.AlgebraMiniTest1 (module), 64
- mathmaker.lib.old_style_sheet.AlgebraShortTest (module), 64
- mathmaker.lib.old_style_sheet.AlgebraTest (module), 64
- mathmaker.lib.old_style_sheet.AlgebraTest2 (module), 64
- mathmaker.lib.old_style_sheet.ConverseAndContrapositiveOfPythagoreanTheoremShortTest (module), 64
- mathmaker.lib.old_style_sheet.EquationsBasic (module), 64
- mathmaker.lib.old_style_sheet.EquationsClassic (module), 64
- mathmaker.lib.old_style_sheet.EquationsHarder (module), 64
- mathmaker.lib.old_style_sheet.EquationsShortTest (module), 64
- mathmaker.lib.old_style_sheet.EquationsTest (module), 65
- mathmaker.lib.old_style_sheet.exercise (module), 63
- mathmaker.lib.old_style_sheet.exercise.question (module), 61
- mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpansion (module), 59
- mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpressionReduction (module), 60
- mathmaker.lib.old_style_sheet.exercise.question.Q_Calculation (module), 60
- mathmaker.lib.old_style_sheet.exercise.question.Q_Equation (module), 60
- mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization (module), 60
- mathmaker.lib.old_style_sheet.exercise.question.Q_RightTriangle (module), 61
- mathmaker.lib.old_style_sheet.exercise.question.Q_Structure (module), 61
- mathmaker.lib.old_style_sheet.exercise.X_AlgebraExpressionMultiplication (module), 61
- mathmaker.lib.old_style_sheet.exercise.X_AlgebraExpressionReduction (module), 62
- mathmaker.lib.old_style_sheet.exercise.X_Calculation (module), 62
- mathmaker.lib.old_style_sheet.exercise.X_Equation (module), 62
- mathmaker.lib.old_style_sheet.exercise.X_Factorization (module), 62
- mathmaker.lib.old_style_sheet.exercise.X_RightTriangle (module), 62
- mathmaker.lib.old_style_sheet.exercise.X_Structure (module), 62
- mathmaker.lib.old_style_sheet.FractionSimplification (module), 65
- mathmaker.lib.old_style_sheet.FractionsProductAndQuotient (module), 65
- mathmaker.lib.old_style_sheet.FractionsSum (module), 65
- mathmaker.lib.old_style_sheet.PythagoreanTheoremShortTest (module), 65
- mathmaker.lib.old_style_sheet.S_Structure (module), 65
- mathmaker.lib.shared (module), 77
- mathmaker.lib.tools (module), 75
- mathmaker.lib.tools.shortcuts (module), 66
- mathmaker.lib.tools.frameworks (module), 67
- mathmaker.lib.tools.ignition (module), 68
- mathmaker.lib.tools.maths (module), 69
- mathmaker.lib.tools.wording (module), 70
- mathmaker.lib.tools.xml (module), 75
- mathmaker.settings (module), 77
- MathmakerHTTPRequestHandler (class in mathmaker.daemon), 77
- mc_source (class in mathmaker.lib.tools.database), 66
- mean() (in module mathmaker.lib.tools.maths), 69
- measure (mathmaker.lib.core.base_geometry.Angle attribute), 47
- merge_nb_unit_pairs() (in module mathmaker.lib.tools.wording), 73
- Expansion (mathmaker.lib.core.base_calculus.Fraction method), 42
- Reduction (mathmaker.lib.core.base_calculus.Monomial (class in mathmaker.lib.core.base_calculus), 42
- multiply_symbol_is_required() (mathmaker.lib.core.base_calculus.Function method), 41
- multiply_symbol_is_required() (mathmaker.lib.core.base_calculus.Item method), 44
- multiply_symbol_is_required() (mathmaker.lib.core.base_calculus.Product method), 44
- multiply_symbol_is_required() (mathmaker.lib.core.base_calculus.Quotient method), 45
- multiply_symbol_is_required() (mathmaker.lib.core.base_calculus.SquareRoot method), 46
- multiply_symbol_is_required() (mathmaker.lib.core.base_calculus.Sum method), 46
- multiply_symbol_is_required() (mathmaker.lib.core.root_calculus.Calculable method), 54

N

name (mathmaker.lib.core.base.Drawable attribute), 37

name (mathmaker.lib.core.base.NamedObject attribute), 37

name (mathmaker.lib.core.base_geometry.Point attribute), 47

name (mathmaker.lib.core.calculus.Equation attribute), 49

name (mathmaker.lib.core.geometry.Polygon attribute), 52

name (mathmaker.lib.core.root_calculus.Unit attribute), 55

NamedObject (class in mathmaker.lib.core.base), 37

nature (mathmaker.lib.core.geometry.Polygon attribute), 52

needs_to_get_rounded() (mathmaker.lib.core.base_calculus.Item method), 42

needs_to_get_rounded() (mathmaker.lib.core.root_calculus.Value method), 56

neutral (mathmaker.lib.core.base_calculus.Operation attribute), 43

next() (mathmaker.lib.tools.database.mc_source method), 66

next() (mathmaker.lib.tools.database.source method), 67

next() (mathmaker.lib.tools.database.sub_source method), 67

next_displayable_term_nb() (mathmaker.lib.core.base_calculus.Sum method), 46

norm (mathmaker.lib.core.base_geometry.Vector attribute), 48

not_coprimes_to() (in module mathmaker.lib.tools.maths), 70

num_val (mathmaker.lib.core.base_calculus.Function attribute), 41

number (mathmaker.lib.core.calculus.Equation attribute), 49

numerator (mathmaker.lib.core.base_calculus.Quotient attribute), 45

numeric_terms_require_to_be_reduced() (mathmaker.lib.core.base_calculus.Sum method), 46

O

Operation (class in mathmaker.lib.core.base_calculus), 43

operator() (mathmaker.lib.core.base_calculus.Operation method), 43

operator() (mathmaker.lib.core.base_calculus.Product method), 44

operator() (mathmaker.lib.core.base_calculus.Quotient method), 45

operator() (mathmaker.lib.core.base_calculus.Sum method), 46

order() (mathmaker.lib.core.base_calculus.Product method), 44

orthogonal_unit_vector() (mathmaker.lib.core.base_geometry.Vector method), 48

P

parse_layout_descriptor() (in module mathmaker.lib.tools), 76

parse_qid() (in module mathmaker.lib.tools.frameworks), 68

parse_sql_creation_query() (in module mathmaker.lib.tools.database), 66

path_object (class in mathmaker.settings), 77

perimeter (mathmaker.lib.core.geometry.Polygon attribute), 52

physical_quantity() (in module mathmaker.lib.tools), 76

po_file_get_list_of() (in module mathmaker.lib.tools), 76

Point (class in mathmaker.lib.core.base_geometry), 47

point (mathmaker.lib.core.geometry.InterceptTheoremConfiguration attribute), 51

points (mathmaker.lib.core.base_geometry.Angle attribute), 47

points (mathmaker.lib.core.base_geometry.Segment attribute), 48

Polygon (class in mathmaker.lib.core.geometry), 52

Polynomial (class in mathmaker.lib.core.base_calculus), 43

post_process() (in module mathmaker.lib.tools.wording), 73

postprocess_decimalfractionssums_query() (in module mathmaker.lib.tools.database), 66

postprocess_percentage_query() (in module mathmaker.lib.tools.database), 66

preprocess_decimalfractions_pairs_tag() (in module mathmaker.lib.tools.database), 66

preprocess_decimals_query() (in module mathmaker.lib.tools.database), 67

preprocess_extdecimals_query() (in module mathmaker.lib.tools.database), 67

preprocess_int_pairs_tag() (in module mathmaker.lib.tools.database), 67

preprocess_percentage_tag() (in module mathmaker.lib.tools.database), 67

preprocess_qkw() (in module mathmaker.lib.tools.database), 67

preprocess_single_nb_tag() (in module mathmaker.lib.tools.database), 67

preprocess_units_pairs_tag() (in module mathmaker.lib.tools.database), 67

prime_factors() (in module mathmaker.lib.tools.maths), 70

- Printable (class in mathmaker.lib.core.base), 37
- printed (mathmaker.lib.core.base.Printable attribute), 37
- process_attr_values() (in module mathmaker.lib.tools.wording), 73
- Product (class in mathmaker.lib.core.base_calculus), 44
- pupil_gcd() (in module mathmaker.lib.tools.maths), 70
- put_term_in_lexicon() (in module mathmaker.lib.core.utils), 57
- pythagorean_equality() (mathmaker.lib.core.geometry.RightTriangle method), 53
- pythagorean_substequality() (mathmaker.lib.core.geometry.RightTriangle method), 53
- PythagoreanTheoremShortTest (class in mathmaker.lib.old_style_sheet.PythagoreanTheoremShortTest), 65
- ## Q
- Q_AlgebraExpressionExpansion (class in mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpressionExpansion), 59
- Q_AlgebraExpressionReduction (class in mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpressionReduction), 60
- Q_Calculation (class in mathmaker.lib.old_style_sheet.exercise.question.Q_Calculation), 60
- Q_Equation (class in mathmaker.lib.old_style_sheet.exercise.question.Q_Equation), 60
- Q_Factorization (class in mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization), 60
- Q_RightTriangle (class in mathmaker.lib.old_style_sheet.exercise.question.Q_RightTriangle), 61
- Q_Structure (class in mathmaker.lib.old_style_sheet.exercise.question.Q_Structure), 61
- Quotient (class in mathmaker.lib.core.base_calculus), 44
- QuotientsEquality (class in mathmaker.lib.core.calculus), 50
- ## R
- ratios_equalities() (mathmaker.lib.core.geometry.InterceptTheoremConfiguration method), 51
- ratios_equalities_substituted() (mathmaker.lib.core.geometry.InterceptTheoremConfiguration method), 51
- ratios_for_converse() (mathmaker.lib.core.geometry.InterceptTheoremConfiguration method), 51
- raw_value (mathmaker.lib.core.base_calculus.Item attribute), 42
- raw_value (mathmaker.lib.core.base_calculus.Monomial attribute), 43
- raw_value (mathmaker.lib.core.root_calculus.Value attribute), 56
- Ray (class in mathmaker.lib.core.base_geometry), 47
- read_index() (in module mathmaker.lib.tools.frameworks), 68
- read_layout() (in module mathmaker.lib.tools.frameworks), 68
- real_length (mathmaker.lib.core.base_geometry.Segment attribute), 48
- Rectangle (class in mathmaker.lib.core.geometry), 52
- RectangleGrid (class in mathmaker.lib.core.geometry), 52
- recursive_update() (mathmaker.lib.tools.ext_dict method), 76
- reduce_() (mathmaker.lib.core.base_calculus.Product method), 44
- reduce_() (mathmaker.lib.core.base_calculus.Sum method), 47
- reduce_literal_items_product() (in module mathmaker.lib.core.utils), 57
- reduce_literals() (mathmaker.lib.core.base_calculus.Fraction method), 39
- remove() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
- rename() (mathmaker.lib.core.geometry.Polygon method), 52
- replace_striked_out() (mathmaker.lib.core.base_calculus.Fraction method), 39
- requires_brackets() (mathmaker.lib.core.base_calculus.Function method), 41
- requires_brackets() (mathmaker.lib.core.base_calculus.Item method), 42
- requires_brackets() (mathmaker.lib.core.base_calculus.Product method), 44
- requires_brackets() (mathmaker.lib.core.base_calculus.Quotient method), 45
- requires_brackets() (mathmaker.lib.core.base_calculus.SquareRoot method), 46
- requires_brackets() (mathmaker.lib.core.base_calculus.Sum method), 47
- requires_brackets() (mathmaker.lib.core.root_calculus.Calculable method), 54
- requires_inner_brackets() (math-

- maker.lib.core.base_calculus.Function method), 41
 requires_inner_brackets() (mathmaker.lib.core.base_calculus.Item method), 42
 requires_inner_brackets() (mathmaker.lib.core.base_calculus.Product method), 44
 requires_inner_brackets() (mathmaker.lib.core.base_calculus.Quotient method), 45
 requires_inner_brackets() (mathmaker.lib.core.base_calculus.SquareRoot method), 46
 requires_inner_brackets() (mathmaker.lib.core.base_calculus.Sum method), 47
 requires_inner_brackets() (mathmaker.lib.core.root_calculus.Calculable method), 54
 reset_element() (mathmaker.lib.core.base_calculus.Operation method), 43
 reset_exercises_counter() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
 reset_exercises_counter() (mathmaker.lib.machine.Structure.Structure method), 58
 retrieve_fonts() (in module mathmaker.lib.tools), 76
 revert() (mathmaker.lib.core.base_geometry.Segment method), 48
 right_angle (mathmaker.lib.core.geometry.RightTriangle attribute), 53
 right_hand_side (mathmaker.lib.core.calculus.Equation attribute), 49
 right_hand_side (mathmaker.lib.core.calculus.Expression attribute), 50
 RightTriangle (class in mathmaker.lib.core.geometry), 53
 rotate() (in module mathmaker.lib.tools), 76
 rotate() (mathmaker.lib.core.base_geometry.Point method), 47
 rotation_angle (mathmaker.lib.core.geometry.Polygon attribute), 52
 rounded() (mathmaker.lib.core.base_calculus.Item method), 42
 rounded() (mathmaker.lib.core.root_calculus.Value method), 56
 run() (in module mathmaker.daemon), 77
- ## S
- S_Structure (class in mathmaker.lib.old_style_sheet.S_Structure), 65
 same_deno_reduction_in_progress (mathmaker.lib.core.base_calculus.Fraction attribute), 39
 Segment (class in mathmaker.lib.core.base_geometry), 47
 set_coeff() (mathmaker.lib.core.base_calculus.Monomial method), 43
 set_compact_display() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
 set_degree() (mathmaker.lib.core.base_calculus.Monomial method), 43
 set_denominator() (mathmaker.lib.core.base_calculus.Quotient method), 45
 set_down_numerator_s_minus_sign() (mathmaker.lib.core.base_calculus.Fraction method), 39
 set_element() (mathmaker.lib.core.base_calculus.Operation method), 43
 set_exponent() (mathmaker.lib.core.root_calculus.Exponented method), 55
 set_factor() (mathmaker.lib.core.base_calculus.Product method), 44
 set_font_size_offset() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
 set_font_size_offset() (mathmaker.lib.machine.Structure.Structure method), 59
 set_force_display_sign_once() (mathmaker.lib.core.base_calculus.Item method), 42
 set_force_display_sign_once() (mathmaker.lib.core.base_calculus.SquareRoot method), 46
 set_force_inner_brackets_display() (mathmaker.lib.core.base_calculus.Sum method), 47
 set_hand_side() (mathmaker.lib.core.calculus.Equation method), 49
 set_has_been_rounded() (mathmaker.lib.core.root_calculus.Value method), 56
 set_info() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
 set_is_out_striked() (mathmaker.lib.core.base_calculus.Item method), 42
 set_lengths() (mathmaker.lib.core.geometry.InterceptTheoremConfiguration method), 51
 set_lengths() (mathmaker.lib.core.geometry.Polygon method), 52
 set_lengths() (mathmaker.lib.core.geometry.Rectangle method), 52
 set_lengths() (mathmaker.lib.core.geometry.Square method), 54

- `set_letter()` (mathmaker.lib.core.base_calculus.Monomial method), 43
`set_literal_mode()` (mathmaker.lib.core.base_calculus.Function method), 41
`set_marks()` (mathmaker.lib.core.geometry.Square method), 54
`set_number()` (mathmaker.lib.core.calculus.Equation method), 49
`set_numerator()` (mathmaker.lib.core.base_calculus.Quotient method), 45
`set_numeric_mode()` (mathmaker.lib.core.base_calculus.Function method), 41
`set_opposite_sign()` (mathmaker.lib.core.root_calculus.Signed method), 55
`set_opposite_sign()` (mathmaker.lib.core.root_calculus.Value method), 56
`set_redirect_output_to_str()` (mathmaker.lib.machine.LaTeX.LaTeX method), 57
`set_redirect_output_to_str()` (mathmaker.lib.machine.Structure.Structure method), 59
`set_right_hand_side()` (mathmaker.lib.core.calculus.Expression method), 50
`set_same_deno_reduction_in_progress()` (mathmaker.lib.core.base_calculus.Fraction method), 39
`set_side_length()` (mathmaker.lib.core.geometry.Square method), 54
`set_sign()` (mathmaker.lib.core.base_calculus.CommutativeGroup class in mathmaker.lib.core.root_calculus), 55
`set_sign()` (mathmaker.lib.core.root_calculus.Signed method), 55
`set_sign()` (mathmaker.lib.core.root_calculus.Value method), 56
`set_status()` (mathmaker.lib.core.base_calculus.Fraction method), 39
`set_symbol()` (mathmaker.lib.core.base_calculus.Operation method), 43
`set_term()` (mathmaker.lib.core.base_calculus.Sum method), 47
`set_unit()` (mathmaker.lib.core.base_calculus.Item method), 42
`set_unit()` (mathmaker.lib.core.root_calculus.Value method), 56
`set_value_inside()` (mathmaker.lib.core.base_calculus.Item method), 42
`setup_for_trigonometry()` (mathmaker.lib.core.geometry.RightTriangle method), 53
`setup_label()` (mathmaker.lib.core.base_geometry.Segment method), 48
`setup_labels()` (mathmaker.lib.core.geometry.Polygon method), 52
`setup_wording_format_of()` (in module mathmaker.lib.tools.wording), 73
`sheet_header_to_str()` (mathmaker.lib.old_style_sheet.S_Structure.S_Structure method), 65
`sheet_text_to_str()` (mathmaker.lib.old_style_sheet.S_Structure.S_Structure method), 65
`sheet_title_to_str()` (mathmaker.lib.old_style_sheet.S_Structure.S_Structure method), 65
`side` (mathmaker.lib.core.geometry.Polygon attribute), 52
`side_adjacent_to()` (mathmaker.lib.core.geometry.RightTriangle method), 53
`side_length` (mathmaker.lib.core.geometry.Square attribute), 54
`side_opposite_to()` (mathmaker.lib.core.geometry.RightTriangle method), 53
`sign` (mathmaker.lib.core.base_calculus.Monomial attribute), 43
`sign` (mathmaker.lib.core.root_calculus.Signed attribute), 55
`sign` (mathmaker.lib.core.root_calculus.Value attribute), 56
`sign_of_product()` (in module mathmaker.lib.tools.maths), 70
`Signator` (class in mathmaker.lib.core.root_calculus), 55
`simplification_in_progress` (mathmaker.lib.core.base_calculus.Fraction attribute), 39
`simplification_line()` (mathmaker.lib.core.base_calculus.Fraction method), 39
`simplified()` (mathmaker.lib.core.base_calculus.Fraction method), 39
`slope` (mathmaker.lib.core.base_geometry.Vector attribute), 48
`small` (mathmaker.lib.core.geometry.InterceptTheoremConfiguration attribute), 51
`solve_next_step()` (mathmaker.lib.core.calculus.CrossProductEquation method), 49
`solve_next_step()` (mathmaker.lib.core.calculus.Equation method), 49
`source` (class in mathmaker.lib.tools.database), 67

- sqrt() (mathmaker.lib.core.root_calculus.Value method), 56
- Square (class in mathmaker.lib.core.geometry), 54
- SquareRoot (class in mathmaker.lib.core.base_calculus), 45
- status (mathmaker.lib.core.base_calculus.Fraction attribute), 39
- Structure (class in mathmaker.lib.machine.Structure), 58
- sub_source (class in mathmaker.lib.tools.database), 67
- subst_dict (mathmaker.lib.core.root_calculus.Substitutable attribute), 55
- Substitutable (class in mathmaker.lib.core.root_calculus), 55
- substitute() (mathmaker.lib.core.base_calculus.Function method), 41
- substitute() (mathmaker.lib.core.root_calculus.Calculable method), 54
- substitute() (mathmaker.lib.core.root_calculus.Substitutable method), 55
- substitute() (mathmaker.lib.core.root_calculus.Value method), 57
- Sum (class in mathmaker.lib.core.base_calculus), 46
- symbol (mathmaker.lib.core.base_calculus.Operation attribute), 43
- ## T
- Table (class in mathmaker.lib.core.calculus), 50
- Table.SubstitutableList (class in mathmaker.lib.core.calculus), 50
- Table_UP (class in mathmaker.lib.core.calculus), 50
- ten_power_gcd() (in module mathmaker.lib.tools.maths), 70
- term (mathmaker.lib.core.base_calculus.Sum attribute), 47
- text_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpressionExpansion.Q_AlgebraExpressionExpansion method), 60
- text_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_AlgebraExpressionReduction.Q_AlgebraExpressionReduction method), 60
- text_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Calculation.Q_Calculation method), 60
- text_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Equation.Q_Equation method), 60
- text_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Factorization.Q_Factorization method), 60
- text_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_RightTriangle.Q_RightTriangle method), 61
- text_to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure method), 61
- texts_to_str() (mathmaker.lib.old_style_sheet.S_Structure.S_Structure method), 65
- throw_away_the_neutrals() (mathmaker.lib.core.base_calculus.CommutativeOperation method), 38
- to_str() (mathmaker.lib.old_style_sheet.exercise.question.Q_Structure.Q_Structure method), 61
- to_str() (mathmaker.lib.old_style_sheet.exercise.X_Structure.X_Structure method), 62
- translate_font_size() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- translate_font_size() (mathmaker.lib.machine.Structure.Structure method), 59
- Triangle (class in mathmaker.lib.core.geometry), 54
- trigonometric_equality() (mathmaker.lib.core.geometry.RightTriangle method), 53
- trigonometric_ratios() (mathmaker.lib.core.geometry.RightTriangle method), 54
- turn_into_fraction() (mathmaker.lib.core.base_calculus.Item method), 42
- turn_into_fraction() (mathmaker.lib.core.base_calculus.SquareRoot method), 46
- type_string() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- type_string() (mathmaker.lib.machine.Structure.Structure method), 59
- ## U
- u (mathmaker.lib.core.geometry.InterceptTheoremConfiguration attribute), 52
- Unit (class in mathmaker.lib.core.root_calculus), 55
- unit (mathmaker.lib.core.base_calculus.Item attribute), 42
- unit (mathmaker.lib.core.root_calculus.Value attribute), 47
- unit_vector() (mathmaker.lib.core.base_geometry.Vector method), 48
- unlocked (mathmaker.lib.core.base_calculus.Function attribute), 41
- unwrapped() (in module mathmaker.lib.tools.wording), 71
- ## V
- v (mathmaker.lib.core.geometry.InterceptTheoremConfiguration attribute), 52
- Value (class in mathmaker.lib.core.root_calculus), 55
- value (mathmaker.lib.core.base_calculus.Item attribute), 42
- value_side (mathmaker.lib.core.base_calculus.Monomial attribute), 43
- var (mathmaker.lib.core.base_calculus.Function attribute), 41
- variable_letter (mathmaker.lib.core.calculus.Equation attribute), 49

- variable_obj (mathmaker.lib.core.calculus.CrossProductEquation attribute), 49
- variable_position (mathmaker.lib.core.calculus.CrossProductEquation attribute), 49
- Vector (class in mathmaker.lib.core.base_geometry), 48
- vertex (mathmaker.lib.core.base_geometry.Angle attribute), 47
- vertex (mathmaker.lib.core.geometry.Polygon attribute), 52
- ## W
- warning_msg() (in module mathmaker.lib.tools.ignition), 69
- width (mathmaker.lib.core.geometry.Rectangle attribute), 52
- wrap() (in module mathmaker.lib.tools.wording), 74
- wrap_latex_keywords() (in module mathmaker.lib.tools.wording), 75
- write() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- write() (mathmaker.lib.machine.Structure.Structure method), 59
- write_answers() (mathmaker.lib.old_style_sheet.AlgebraBinomialIdentityExpansion method), 63
- write_document_begins() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- write_document_begins() (mathmaker.lib.machine.Structure.Structure method), 59
- write_document_ends() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- write_document_ends() (mathmaker.lib.machine.Structure.Structure method), 59
- write_document_header() (mathmaker.lib.machine.LaTeX.LaTeX method), 57
- write_document_header() (mathmaker.lib.machine.Structure.Structure method), 59
- write_exercise_number() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_exercise_number() (mathmaker.lib.machine.Structure.Structure method), 59
- write_frame() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_frame() (mathmaker.lib.machine.Structure.Structure method), 59
- write_jump_to_next_page() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_jump_to_next_page() (mathmaker.lib.machine.Structure.Structure method), 59
- write_layout() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_layout() (mathmaker.lib.machine.Structure.Structure method), 59
- write_math_style1() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_math_style1() (mathmaker.lib.machine.Structure.Structure method), 59
- write_math_style2() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_math_style2() (mathmaker.lib.machine.Structure.Structure method), 59
- write_new_line() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_new_line() (mathmaker.lib.machine.Structure.Structure method), 59
- write_new_line_twice() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_new_line_twice() (mathmaker.lib.machine.Structure.Structure method), 59
- write_out() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_out() (mathmaker.lib.machine.Structure.Structure method), 59
- write_set_font_size_to() (mathmaker.lib.machine.LaTeX.LaTeX method), 58
- write_set_font_size_to() (mathmaker.lib.machine.Structure.Structure method), 59
- ## X
- x (mathmaker.lib.core.base_geometry.Point attribute), 47
- X_AlgebraExpressionExpansion (class in mathmaker.lib.old_style_sheet.exercise.X_AlgebraExpressionExpansion), 61
- X_AlgebraExpressionReduction (class in mathmaker.lib.old_style_sheet.exercise.X_AlgebraExpressionReduction), 62
- X_Calculation (class in mathmaker.lib.old_style_sheet.exercise.X_Calculation), 62

62
X_Equation (class in math-
maker.lib.old_style_sheet.exercise.X_Equation),
62
x_exact (mathmaker.lib.core.base_geometry.Point at-
tribute), 47
X_Factorization (class in math-
maker.lib.old_style_sheet.exercise.X_Factorization),
62
X_RightTriangle (class in math-
maker.lib.old_style_sheet.exercise.X_RightTriangle),
62
X_Structure (class in math-
maker.lib.old_style_sheet.exercise.X_Structure),
62
xy (mathmaker.lib.core.base_geometry.Point attribute),
47

Y

y (mathmaker.lib.core.base_geometry.Point attribute), 47
y_exact (mathmaker.lib.core.base_geometry.Point at-
tribute), 47