# Moodle String Generator Documentation

***Release 0.3.1***

**Edward Villegas-Pulgarin**

**Aug 22, 2023**

# Contents:

Because I am not teacher since long time and I am not considering in future, this project is unmainteined.

CHAPTER 1

# Moodle String Generator

Python package to generate common strings used in moodle for users (not admin level) and manage exported files from moodle and generate files for import.

- Free software: MIT license
- Documentation: https://moodlesg.readthedocs.io.

## 1.1 Description

MoodleSG (Moodle String Generator) is a python package with the purpose of help to generate common strings using for users (not admin level) when manage a course in moodle, e.g. answer formulas, grade formulas and question generation in markdown or XML.

This package is not intended to use as admin level and is developed because management of some formulas are very uncomfortable because not variable is available as user level (some universities has very strange methods for grade calculation and vary each semester or some math problems has many substitutions). So, you finish with a longer formula with pattern repetitions that you add manually and many brackets and then your math expression for grade or answer formula is very susceptible to fail.

Also, you can think in some automatic question generation that Moodle not support but you can create and export as a question database of questions (not calculated questions that support numeric or text variables only -not both in the same question-) or import from a question database backup and modify that.

## 1.2 Features

n: null support

b: basic support

p: partial support

f: full support

- [p] String generation of moodle math formulas.
  - [f] Grade calculations.
  - [p] Calculated question.
- [n] String generation of questions for text plain moodle editor (not XML nor HTML).
- [n] String manipulation of question bank.
  - [n] Export of question strings to moodle supported question import formats.
  - [n] Import of question strings from moodle question export formats.
  - [n] Management question bank locally and generate from collections of question strings.

## 1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

# Installation

## 2.1 Stable release

To install Moodle String Generator, run this command in your terminal:

```
$ pip install moodlesg
```

This is the preferred method to install Moodle String Generator, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

## 2.2 From sources

The sources for Moodle String Generator can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/cosmoscalibur/moodlesg
```

Or download the tarball:

```
$ curl  -OL https://github.com/cosmoscalibur/moodlesg/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

# Usage

To use Moodle String Generator in a project:

```
import moodlesg
```

At this moment only works the `math` module.

## 3.1 Math module

To use Moodle String Generator math utility

```
import moodlesg.math as mm
```

You can find in `mm` all [intended] public members as a common and easy interface to grade and answer calculation, with names that remember python math core.

Not all is finished but is usable (grade calculation is full supported).

CHAPTER 4

# moodlesg

## 4.1 moodlesg package

### 4.1.1 Subpackages

#### moodlesg.math package

#### Subpackages

#### moodlesg.math.functions package

#### Submodules

#### moodlesg.math.functions.common module

This module implement moodle math functions that are common between grade and answer calculation.

`moodlesg.math.functions.common.`**`abs`**`(`*expr*`)`
Absolute value.

`moodlesg.math.functions.common.`**`acos`**`(`*expr*`)`
Arc cosine – output in radians.

It is the same that `arccos` moodle math function.

`moodlesg.math.functions.common.`**`acosh`**`(`*expr*`)`
Inverse hyperbolic cosine – output in radians.

It is the same that `arccosh` moodle math function.

`moodlesg.math.functions.common.`**`asin`**`(`*expr*`)`
Arc sine – output in radians.

It is the same that `arcsin` moodle math function.

moodlesg.math.functions.common.**asinh**(*expr*)
Inverse hyperbolic sine.– output in radians.

It is the same that `arcsinh` moodle math function.

moodlesg.math.functions.common.**atan**(*expr*)
Arc tangent – output in radians.

It is the same that `arctan` moodle math function.

moodlesg.math.functions.common.**atanh**(*expr*)
Inverse hyperbolic tangent– output in radians.

It is the same that `arctanh` moodle math function.

moodlesg.math.functions.common.**average**(*expr1*, **expr2*)
Returns the average of the values in a list of arguments.

moodlesg.math.functions.common.**ceil**(*expr*)
Maps a real number to the smallest following integer.

moodlesg.math.functions.common.**cos**(*expr*)
Cosine – in radians!!! Convert your degree measurement to radians before you take the cos of it.

moodlesg.math.functions.common.**cosh**(*expr*)
Hyperbolic cosine – in radians!!! Convert your degree measurement to radians before you take the cosh of it.

moodlesg.math.functions.common.**exp**(*expr*)
Calculates the exponent of e.

moodlesg.math.functions.common.**floor**(*expr*)
Maps a real number to the largest previous integer.

moodlesg.math.functions.common.**max**(*expr1*, **expr2*)
Returns the maximum value in a list of arguments.

moodlesg.math.functions.common.**min**(*expr1*, **expr2*)
Returns the minimum value in a list of arguments.

moodlesg.math.functions.common.**round**(*expr*, *count*)
Rounds number to count decimal digits.

moodlesg.math.functions.common.**sin**(*expr*)
Sine – in radians!!! Convert your degree measurement to radians before you take the sin of it.

moodlesg.math.functions.common.**sinh**(*expr*)
Hyperbolic sine – in radians!!! Convert your degree measurement to radians before you take the sinh of it.

moodlesg.math.functions.common.**sqrt**(*expr*)
Square root.

moodlesg.math.functions.common.**sum**(*expr1*, **expr2*)
Returns the sum of all arguments.

moodlesg.math.functions.common.**tan**(*expr*)
Tangent – in radians!!! Convert your degree measurement to radians before you take the tan of it.

moodlesg.math.functions.common.**tanh**(*expr*)
Hyperbolic tangent – in radians!!! Convert your degree measurement to radians before you take the tanh of it.

### moodlesg.math.functions.compatibility module

This module create a common interface to call math functions of grade and answer calculation. Some functions in moodle version has different names and other cases has no equivalent (grade calculation support less functions).

### Examples

In Moodle you can evaluate a natural logarithm in both, grade and answer calculation, but the function has different name. If you need to evaluate in grade calculation `ln` is the function but in answer calculation is `log`. With MoodleSG both cases are covered with *`log()`* (common convention in programming languages).

```
>>> mm.mmParams.grade()
>>> a = mm.moodle_var('a')
>>> mm.log(a)
Expression('ln([[a]])')
```

```
>>> mm.mmParams.answer()
>>> b = mm.moodle_var('b')
>>> mm.log(b)
Expression('log({b})')
```

moodlesg.math.functions.compatibility.**log**(*expr*)
Return natural logarithm of a expression.

moodlesg.math.functions.compatibility.**log10**(*expr*)
Return base-10 logarithm of a expression.

moodlesg.math.functions.compatibility.**mod**(*expr1*, *expr2*)
Calculates the remainder of a division.

Also supported in the builtin `%` (modulus) operator.

moodlesg.math.functions.compatibility.**pow**(*expr1*, *expr2*)
Exponential expression. Raises a number to the power of another.

### moodlesg.math.functions.extended module

moodlesg.math.functions.extended.**bool**(*expr*)
Implement truth value testing.

Expression is considered true if its result is nonzero.

moodlesg.math.functions.extended.**trunc**(*expr*)

### Module contents

Module `functions` contains submodules of functions that grouping supported functions for moodle (native and extended).

### Submodules

### moodlesg.math.base module

Module `base` contains definitions required for building moodle mathematical expressions.

**class** moodlesg.math.base.**Expression**(*expression*)
Implement a generic mathematical expression in moodle format.

**Parameters** **expression** (*str*) – Math moodle expression.

**Returns** Moodle math object.

**Return type** *Expression*

#### Examples

```
>>> str(mm.Expression('a'))
'a'
```

```
>>> a = mm.Expression('a')
>>> b = mm.Expression('b')
```

```
>>> str(a + b)
'(a + b)'
>>> str(a + 1)
'(a + 1)'
```

**class** moodlesg.math.base.**Settings**
Representation of moodle math parameters.

If you need change parameters, access to them throught *mmParams*.

**answer**()

**comma_semicolon**()

**decimal_sep = '.'**

**grade**()

**list_sep = ';'**

**math_type = 'grade'**

**period_comma**()

**period_semicolon**()

**setup**(**kwargs*)

moodlesg.math.base.**answer_var**(*answer_variable*)
Implement a moodle variable for using in correct answer calculation.

**Parameters** **answer_variable** (*str*) – Variable name string.

**Returns** Moodle variable ready for calculations.

**Return type** *Expression*

### Examples

We can create a moodle answer variable for calculation as follow:

```
>>> a = mm.answer_var('a')
>>> str(a)
'{a}'
```

**See also:**

*Expression grade_var()*.

moodlesg.math.base.**grade_var**(*id_activity*)
Implement a moodle variable for using in grade calculation.

**Parameters** **id_activity** (*str*) – Moodle ID of the activity.

**Returns** Moodle ID ready for grade calculation.

**Return type** *Expression*

### Examples

We can create a moodle id activity for calculation as follow:

```
>>> a = mm.grade_var('a')
>>> str(a)
'[[a]]'
>>> str(a + 1)
'([[a]] + 1)'
```

**See also:**

*Expression answer_var()*

moodlesg.math.base.**mmParams = <moodlesg.math.base.Settings instance>**
Object that represents moodle math parameters and manages them.

**Type** (*Settings*)

moodlesg.math.base.**moodle_var**(*name_variable*)
Create variable type according to mmParams.math_type.

**Parameters** **name_variable** (*str*) – Identifier of the moodle variable.

**Returns** Moodle variable of correct type.

**Return type** *Expression*

### Examples

```
>>> mm.mmParams.grade()
>>> mm.moodle_var('a')
Expression('[[a]]')
```

```
>>> mm.mmParams.answer()
>>> mm.moodle_var('a')
Expression('{a}')
```

**See_Also:** *Expression grade_var() answer_var()*

### moodlesg.math.constants module

Module `constants` define common math moodle constants between grade and answer calculations.

In Moodle, actually there is no predefined constant that is allowed other than pi() as a function without parameter. With this module, is posible use $\pi$ as a constant (not call to a function) and define others constants.

`moodlesg.math.constants.`**`PI = Expression('pi()')`**
Define $\pi$.

### moodlesg.math.settings module

### Module contents

Module `moodlesg.math` implements a python interface to math manipulation requiered in *grade* and *answer* moodle formulas.

You can use variable assigment, relation operators, mixing types (moodle expressions with python data types) and commons names of functions in both type of formulas, grade and answers.

### References

## 4.1.2 Module contents

Top-level package for Moodle String Generator.

CHAPTER 5

# Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

## 5.1 Types of Contributions

### 5.1.1 Report Bugs

Report bugs at https://github.com/cosmoscalibur/moodlesg/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

### 5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

### 5.1.4 Write Documentation

Moodle String Generator could always use more documentation, whether as part of the official Moodle String Generator docs, in docstrings, or even on the web in blog posts, articles, and such.

### 5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/cosmoscalibur/moodlesg/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## 5.2 Get Started!

Ready to contribute? Here's how to set up *moodlesg* for local development.

1. Fork the *moodlesg* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/moodlesg.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv moodlesg
$ cd moodlesg/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 moodlesg tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

## 5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/cosmoscalibur/moodlesg/pull_requests and make sure that the tests pass for all supported Python versions.

## 5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_moodlesg
```

## 5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

# Credits

## 6.1 Development Lead

- Edward Villegas-Pulgarin <cosmoscalibur@gmail.com>

## 6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

# History

## 7.1 0.3.1 (2018-05-17)

- Define changes according to mmParams.
- Remove sdist distribution from travis deployment.
- Update classifiers.
- Support moodle math setting: decimal and list separator, math type.

## 7.2 0.3.0 (2018-05-13)

- Fix installation wheels: Removed wheels for python2.7.
- Caps convention for constant: pi → PI.
- Better documentation.
- Full support of common functions between grade and answer calculations.

## 7.3 0.2.2 (2018-05-11)

- Fix docstrings.
- Added new math functions and docstrings (common and extended module).

## 7.4 0.2.0 (2018-05-06)

- Full implementation of moodle math expressions.

- Added pi constant.
- Some math functions common between grade and answer calculation.

## 7.5 0.1.2 (2018-03-14)

- Use super method for init of derived classes.
- Removed support of python 3.3.
- Added basic definition of moodle mathematical variable classes (since 0.1.1).

## 7.6 0.1.0 (2018-03-08)

- First release on PyPI.

CHAPTER 8

# Indices and tables

- genindex
- modindex
- search

# Bibliography

[grade] Grade calculations. Moodle 3.4 Docs.

[answer] Calculated question type Moodle 3.4 Docs.

# Python Module Index

## m

# Index

## P



## R



## S



## T