oemof-tools

Release 0.4.3

oemof-developer-group

Oct 12, 2023

CONTENTS

1 Overview **1**
- 1.1 Installation . . . 1
- 1.2 Documentation . . . 1
- 1.3 Development . . . 2

2 Installation **3**

3 Usage **5**
- 3.1 Economics . . . 5
- 3.2 Helpers . . . 5
- 3.3 Logger . . . 5
- 3.4 Debugging . . . 6

4 Reference **7**
- 4.1 oemof.tools package . . . 7

5 Contributing **11**
- 5.1 Bug reports . . . 11
- 5.2 Documentation improvements . . . 11
- 5.3 Feature requests and feedback . . . 11
- 5.4 Development . . . 12

6 Authors **13**

7 Changelog **15**
- 7.1 0.4.2 (2022-06-15) . . . 15
- 7.2 0.4.1 (2021-02-21) . . . 15
- 7.3 0.4.0 (2020-05-11) . . . 15

8 Indices and tables **17**

Python Module Index **19**

Index **21**

CHAPTER

ONE

OVERVIEW

Documentation

Tests

Package

Tiny tools of the oemof project.

- Free software: MIT license

1.1 Installation

```
pip install oemof.tools
```

You can also install the in-development version with:

```
pip install https://github.com/oemof/oemof-tools/archive/dev.zip
```

1.2 Documentation

https://oemof-tools.readthedocs.io/

1.3 Development

To run the all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER

TWO

INSTALLATION

At the command line:

```
pip install oemof.tools
```

CHAPTER

THREE

USAGE

The oemof tools package contains little helpers to create your own application. You can use a configuration file in the ini-format to define computer specific parameters such as paths, addresses etc.. Furthermore a logging module helps you creating log files for your application.

List of oemof tools

- *Economics*
- *Helpers*
- *Logger*
- *Debugging*

3.1 Economics

Calculate the annuity. See the API-doc of *annuity()* for all details.

3.2 Helpers

Excess oemof's default path. See the API-doc of `helpers` for all details.

3.3 Logger

The main purpose of this function is to provide a logger with well set default values but with the opportunity to change the most important parameters if you know what you want after a while. This is what most new users (or users who do not want to care about loggers) need. If you are an advanced user with your own ideas it might be easier to copy the whole function to your application and adapt it to your own wishes.

```
define_logging(logpath=None, logfile='oemof.log', file_format=None,
               screen_format=None, file_datefmt=None, screen_datefmt=None,
               screen_level=logging.INFO, file_level=logging.DEBUG,
               log_path=True, timed_rotating=None):
```

By default down to INFO all messages are written on the screen and down to DEBUG all messages are written in the file. The file is placed in $HOME/.oemof/log_files as oemof.log. But you can easily pass your own path and your

own filename. You can also change the logging level (screen/file) by changing the screen_level or the file_level to logging.DEBUG, logging.INFO, logging.WARNING.... . You can stop the logger from logging the oemof version or commit with *log_version=False* and the path of the file with *log_path=False*. Furthermore, you can change the format on the screen and in the file according to the python logging documentation. You can also change the used time format according to this documentation.

```
file_format = "%(asctime)s - %(levelname)s - %(module)s - %(message)s"
file_datefmt = "%x - %X"
screen_format = "%(asctime)s-%(levelname)s-%(message)s"
screen_datefmt = "%H:%M:%S"
```

You can also change the behaviour of the file handling (TimedRotatingFileHandler) by passing a dictionary with your own options (timed_rotating).

See the API-doc of *define_logging()* for all details.

3.4 Debugging

3.4.1 SuspiciousUsageWarning

The SuspiciousUsageWarning can help to find untypical usage of oemof's libraries. However, if you know what you are doing such warnings might be annoying. Therefore, it is possible to control the appearance of this warning.

```
# switch on SuspiciousUsageWarning
warnings.filterwarnings("always", category=SuspiciousUsageWarning)

# raise an error instead of a warning
warnings.filterwarnings("error", category=SuspiciousUsageWarning)

# switch off SuspiciousUsageWarning
warnings.filterwarnings("ignore", category=SuspiciousUsageWarning)
```

For more information about the handling of warnings see the warnings section in the python documentation.

CHAPTER

FOUR

REFERENCE

4.1 oemof.tools package

4.1.1 Submodules

4.1.2 oemof.tools.debugging module

Module contains tools facilitating debugging

This file is part of project oemof (github.com/oemof/oemof). It's copyrighted by the contributors recorded in the version control history of the file, available from its original location oemof/oemof/tools/economics.py

SPDX-License-Identifier: MIT

exception oemof.tools.debugging.**ExperimentalFeatureWarning**

Bases: UserWarning

Warn the user about use of experimental features.

New modules first go to "experimental" ti highlight their unmature state. Sometimes, functionality is added to existing code. We use this warning to warn users in these cases.

exception oemof.tools.debugging.**SuspiciousUsageWarning**

Bases: UserWarning

Warn the user about potentially dangerous usage.

Some ways of using *oemof* are not necessarily wrong but could lead to hard to find bugs if done accidentally instead of intentionally. We use these warnings, and you can do too ;), in your code to warn users about these cases. If you know what you are doing and these warnings point you to things you are doing intentionally, you can easily switch them off.

Note: TODO: Fix ref! See *SuspiciousUsageWarning* for more information.

Examples

```
>>> import warnings
>>> warnings.filterwarnings("ignore", category=SuspiciousUsageWarning)
```

4.1.3 oemof.tools.economics module

Module to collect useful functions for economic calculation.

This file is part of project oemof (github.com/oemof/oemof). It's copyrighted by the contributors recorded in the version control history of the file, available from its original location oemof/oemof/tools/economics.py

SPDX-License-Identifier: MIT

`oemof.tools.economics.`**annuity**(*capex*, *n*, *wacc*, *u=None*, *cost_decrease=0*)

Calculates the annuity of an initial investment 'capex', considering the cost of capital 'wacc' during a project horizon 'n'

In case of a single initial investment, the employed formula reads:

$$\text{annuity} = \text{capex} \cdot \frac{(\text{wacc} \cdot (1+\text{wacc})^n)}{((1+\text{wacc})^n - 1)}$$

In case of repeated investments (due to replacements) at fixed intervals 'u', the formula yields:

$$\text{annuity} = \text{capex} \cdot \frac{(\text{wacc} \cdot (1+\text{wacc})^n)}{((1+\text{wacc})^n - 1)} \cdot \left(\frac{1 - \left(\frac{(1-\text{cost\_decrease})}{(1+\text{wacc})} \right)^n}{1 - \left(\frac{(1-\text{cost\_decrease})}{(1+\text{wacc})} \right)^u} \right)$$

Parameters

- **capex** (*float*) – Capital expenditure for first investment. Net Present Value (NPV) or Net Present Cost (NPC) of investment
- **n** (*int*) – Horizon of the analysis, or number of years the annuity wants to be obtained for (n>=1)
- **wacc** (*float*) – Weighted average cost of capital (0<wacc<1)
- **u** (*int*) – Lifetime of the investigated investment. Might be smaller than the analysis horizon, 'n', meaning it will have to be replaced. Takes value 'n' if not specified otherwise (u>=1)
- **cost_decrease** (*float*) – Annual rate of cost decrease (due to, e.g., price experience curve). This only influences the result for investments corresponding to replacements, whenever u<n. Takes value 0, if not specified otherwise (0<cost_decrease<1)

Returns

float – annuity

4.1.4 oemof.tools.logger module

Helpers to log your modeling process with oemof.

This file is part of project oemof (github.com/oemof/oemof). It's copyrighted by the contributors recorded in the version control history of the file, available from its original location oemof/oemof/tools/logger.py

SPDX-License-Identifier: MIT

`oemof.tools.logger.`**define_logging**(*logpath=None*, *logfile='oemof.log'*, *file_format=None*, *screen_format=None*, *file_datefmt=None*, *screen_datefmt=None*, *screen_level=20*, *file_level=30*, *log_path=True*, *timed_rotating=None*)

Initialise customisable logger.

Parameters

- **logfile** (*str*) – Name of the log file, default: oemof.log
- **logpath** (*str*) – The path for log files. By default a ".oemof" folder is created in your home directory with subfolder called 'log_files'.
- **file_format** (*str*) – Format of the file output. Default: "%(asctime)s - %(levelname)s - %(module)s - %(message)s"
- **screen_format** (*str*) – Format of the screen output. Default: "%(asctime)s-%(levelname)s-%(message)s"
- **file_datefmt** (*str*) – Format of the datetime in the file output. Default: None
- **screen_datefmt** (*str*) – Format of the datetime in the screen output. Default: "%H:%M:%S"
- **screen_level** (*int*) – Level of logging to stdout. Default: 20 (logging.INFO)
- **file_level** (*int*) – Level of logging to file. Default: 30 (logging.WARNING)
- **log_path** (*boolean*) – If True the used file path is logged while initialising the logger.
- **timed_rotating** (*dict*) – Option to pass parameters to the TimedRotatingFileHandler.

Returns

str (*Place where the log file is stored.*)

Notes

By default the INFO level is printed on the screen and the DEBUG level in a file, but you can easily configure the logger. Every module that wants to create logging messages has to import the logging module. The oemof logger module has to be imported once to initialise it.

Examples

To define the default logger you have to import the python logging library and this function. The first logging message should be the path where the log file is saved to.

```
>>> import logging
>>> from oemof.tools import logger
>>> mypath = logger.define_logging(
...     log_path=True, timed_rotating={'backupCount': 4},
...     screen_level=logging.ERROR, screen_datefmt = "no_date")
>>> mypath[-9:]
```

(continues on next page)

(continued from previous page)

```
'oemof.log'
>>> logging.debug("Hallo")
```

oemof.tools.logger.**extend_basic_path**(*subfolder*)

Returns a path based on the basic oemof path and creates it if necessary. The subfolder is the name of the path extension.

oemof.tools.logger.**get_basic_path**()

Returns the basic oemof path and creates it if necessary. The basic path is the '.oemof' folder in the $HOME directory.

CHAPTER

FIVE

CONTRIBUTING

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

5.1 Bug reports

When 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.

5.2 Documentation improvements

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

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/oemof/oemof-tools/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 code contributions are welcome :)

5.4 Development

To set up *oemof-tools* for local development:

1. Fork oemof-tools (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:oemof/oemof-tools.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

5. 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
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER

SIX

AUTHORS

- Caroline Möller
- Guido Plessmann
- Hendrik Hyskens
- Simon Hilpert
- Stephan Günther
- Uwe Krien

CHAPTER

SEVEN

CHANGELOG

7.1 0.4.2 (2022-06-15)

- Add Python support for Python 3.10 and drop support for 3.7
- Move CI from Appveyor and Travis to Github Actions
- Use Black Code Style for oemof.tools
- Change default logging level for file logging from DEBUG to WARNING

7.2 0.4.1 (2021-02-21)

- Fix compatibility problem (by naming submodules in init script)

7.3 0.4.0 (2020-05-11)

- Move the code of the oemof.tools repository to a stand-alone repository.
- Use cookiecutter to create a wider testing structure and a more standardised package structure.
- Switch from nose to pytest
- Remove the version or commit logger
- Add test to increase test coverage

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

oemof.tools.debugging, 7
oemof.tools.economics, 8
oemof.tools.logger, 9

INDEX

A

annuity() (*in module oemof.tools.economics*), 8

D

define_logging() (*in module oemof.tools.logger*), 9

E

ExperimentalFeatureWarning, 7
extend_basic_path() (*in module oemof.tools.logger*), 10

G

get_basic_path() (*in module oemof.tools.logger*), 10

M

module
    oemof.tools.debugging, 7
    oemof.tools.economics, 8
    oemof.tools.logger, 9

O

oemof.tools.debugging
    module, 7
oemof.tools.economics
    module, 8
oemof.tools.logger
    module, 9

S

SuspiciousUsageWarning, 7