Part 1: Mahler Symphony No. 9

Edward Linscott
EPFL

https://www.youtube.com/watch?v=tkChdHBuoiQ&t=3237s

Part 2: Sticky Notes and Strong Songs

Edward Linscott
EPFL

Part 2: Sticky Notes and Strong Songs

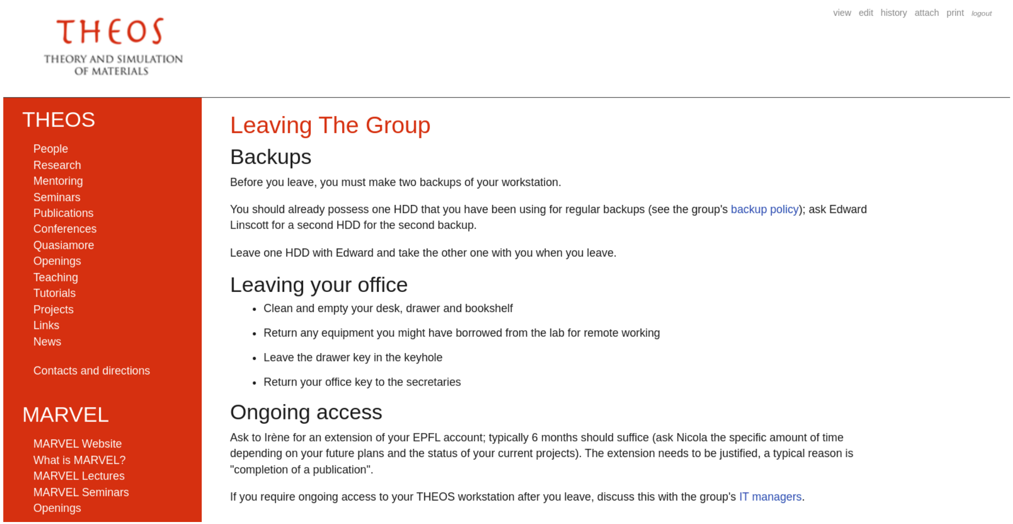
Created by the previous director of l’OCL!

Part 2: Sticky Notes and Strong Songs

Created by the previous director of l'OCL!

Leaving the group

Edward Linscott
EPFL



view edit history attach print *logout*

THEOS
THEORY AND SIMULATION OF MATERIALS

THEOS

- People
- Research
- Mentoring
- Seminars
- Publications
- Conferences
- Quasiamore
- Openings
- Teaching
- Tutorials
- Projects
- Links
- News

Contacts and directions

MARVEL

- MARVEL Website
- What is MARVEL?
- MARVEL Lectures
- MARVEL Seminars
- Openings

Leaving The Group

Backups

Before you leave, you must make two backups of your workstation.

You should already possess one HDD that you have been using for regular backups (see the group's backup policy); ask Edward Linscott for a second HDD for the second backup.

Leave one HDD with Edward and take the other one with you when you leave.

Leaving your office

- Clean and empty your desk, drawer and bookshelf
- Return any equipment you might have borrowed from the lab for remote working
- Leave the drawer key in the keyhole
- Return your office key to the secretaries

Ongoing access

Ask to Irène for an extension of your EPFL account; typically 6 months should suffice (ask Nicola the specific amount of time depending on your future plans and the status of your current projects). The extension needs to be justified, a typical reason is "completion of a publication".

If you require ongoing access to your THEOS workstation after you leave, discuss this with the group's IT managers.

How to do programming and stuff (part 2)

Edward Linscott

THEOS group meeting, 27 July 2022

Previously on HTDPAS...

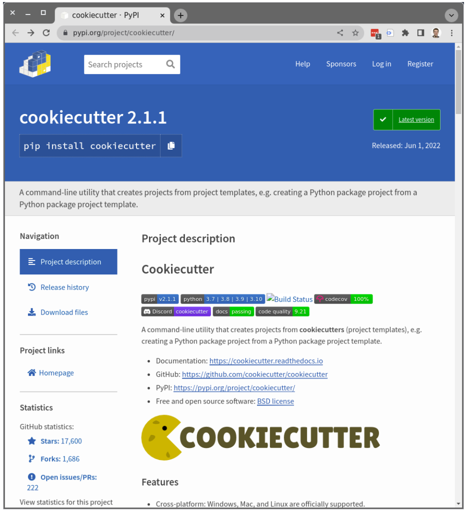
Edward Linscott
EPFL

- vscode
- environments
- languages
- programming paradigms
- linting
- testing
- typing
- pre-commit
- ...

I’m beginning a new project, where do I start?

Cookiecutters

`cookiecutter`



cookiecutter 2.1.1

```
pip install cookiecutter
```

Released: Jun 1, 2022

A command-line utility that creates projects from project templates, e.g. creating a Python package project from a Python package project template.

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Statistics

GitHub statistics:

- Stars: 17,600
- Forks: 1,686
- Open issues/PRs: 222

View statistics for this project

Project description

Cookiecutter

pypi v2.1.1 | python 3.7 | 3.8 | 3.9 | 3.10 | Build Status | codecov 100% | Discord cookiecutter | docs passing | code quality 9.21

A command-line utility that creates projects from **cookiecutters** (project templates), e.g. creating a Python package project from a Python package project template.

- Documentation: https://cookiecutter.readthedocs.io
- GitHub: https://github.com/cookiecutter/cookiecutter
- PyPI: https://pypi.org/project/cookiecutter/
- Free and open source software: BSD license

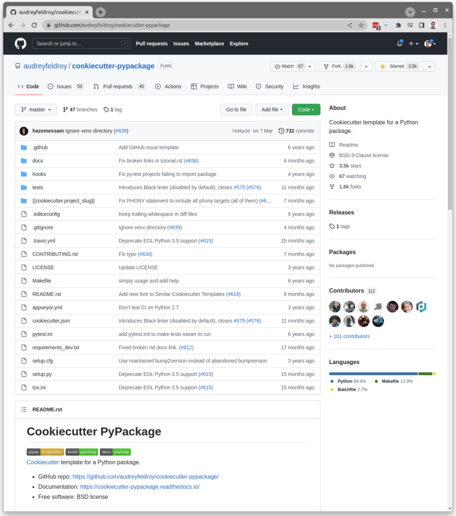
COOKIECUTTER

Features

- Cross-platform: Windows, Mac, and Linux are officially supported.

+

a template



audreyfeldroy / cookiecutter-pypackage

hazemessam Ignore venv directory (#639) 7696a30 on 7 Mar 732 commits

.github	Add GitHub issue template	6 years ago
docs	Fix broken links in tutorial.rst (#636)	6 months ago
hooks	Fix py.test projects failing to import package	4 years ago
tests	Introduces Black linter (disabled by default), closes #575 (#576)	11 months ago
{{cookiecutter.project_slug}}	Fix PHONY statement to include all phony targets (all of them) (#6…	7 months ago
.editorconfig	Keep trailing whitespace in diff files	5 years ago
.gitignore	Ignore venv directory (#639)	4 months ago
.travis.yml	Deprecate EOL Python 3.5 support (#615)	15 months ago
CONTRIBUTING.rst	Fix typo (#634)	7 months ago
LICENSE	Update LICENSE	3 years ago
Makefile	simply usage and add help	6 years ago
README.rst	Add new fork to Similar Cookiecutter Templates (#618)	9 months ago
appveyor.yml	Don't test CI on Python 2.7	3 years ago
cookiecutter.json	Introduces Black linter (disabled by default), closes #575 (#576)	11 months ago
pytest.ini	add pytest.init to make tests easier to run	6 years ago
requirements_dev.txt	Fixed broken rtd docs link. (#612)	17 months ago
setup.cfg	Use maintained bump2version instead of abandoned bumpversion	3 years ago
setup.py	Deprecate EOL Python 3.5 support (#615)	15 months ago
tox.ini	Deprecate EOL Python 3.5 support (#615)	15 months ago

About

Cookiecutter template for a Python package.

- Readme
- BSD-3-Clause license
- 3.5k stars
- 67 watching
- 1.6k forks

Releases

1 tags

Packages

No packages published

Contributors 112

+ 101 contributors

Languages

● Python 84.4% ● Makefile 12.9% ● Batchfile 2.7%

README.rst

Cookiecutter PyPackage

pyup 4 updates | build passing | docs passing

Cookiecutter template for a Python package.

- GitHub repo: https://github.com/audreyfeldroy/cookiecutter-pypackage/
- Documentation: https://cookiecutter-pypackage.readthedocs.io/
- Free software: BSD license

Cookiecutters

Example: `cookiecutter https://github.com/audreyfeldroy/cookiecutter-pypackage`

Cookiecutters

Edward Linscott
EPFL

Lots of available options!
See e.g. `https://github.com/audreyfeldroy/cookiecutter-pypackage#similar-cookiecutter-templates`

An example repository: `autoibrav`

See the notebook

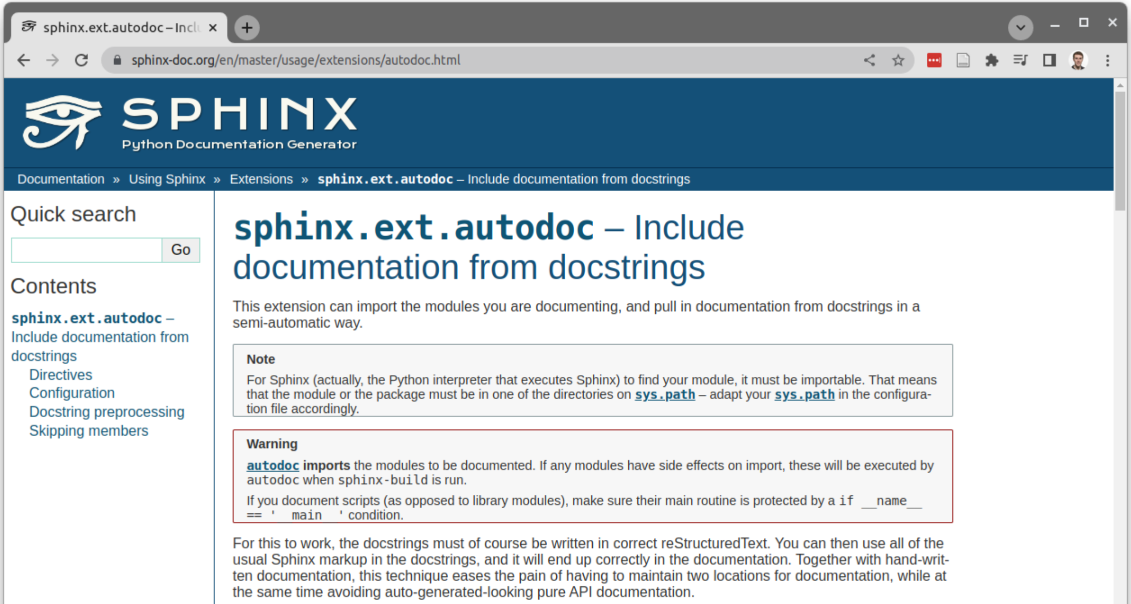
Two super-easy things we can do:

- documentation
- tests

Note: the following is specific to python

Documentation

Edward Linscott
EPFL


sphinx.ext.autodoc – Incl
sphinx-doc.org/en/master/usage/extensions/autodoc.html
SPHINX
Python Documentation Generator

Documentation » Using Sphinx » Extensions » **sphinx.ext.autodoc** – Include documentation from docstrings

Quick search

Go

Contents

`sphinx.ext.autodoc` – Include documentation from docstrings

This extension can import the modules you are documenting, and pull in documentation from docstrings in a semi-automatic way.

Note

For Sphinx (actually, the Python interpreter that executes Sphinx) to find your module, it must be importable. That means that the module or the package must be in one of the directories on **`sys.path`** – adapt your **`sys.path`** in the configuration file accordingly.

Warning

`autodoc` **imports** the modules to be documented. If any modules have side effects on import, these will be executed by `autodoc` when `sphinx-build` is run.

If you document scripts (as opposed to library modules), make sure their main routine is protected by a `if __name__ == '__main__'` condition.

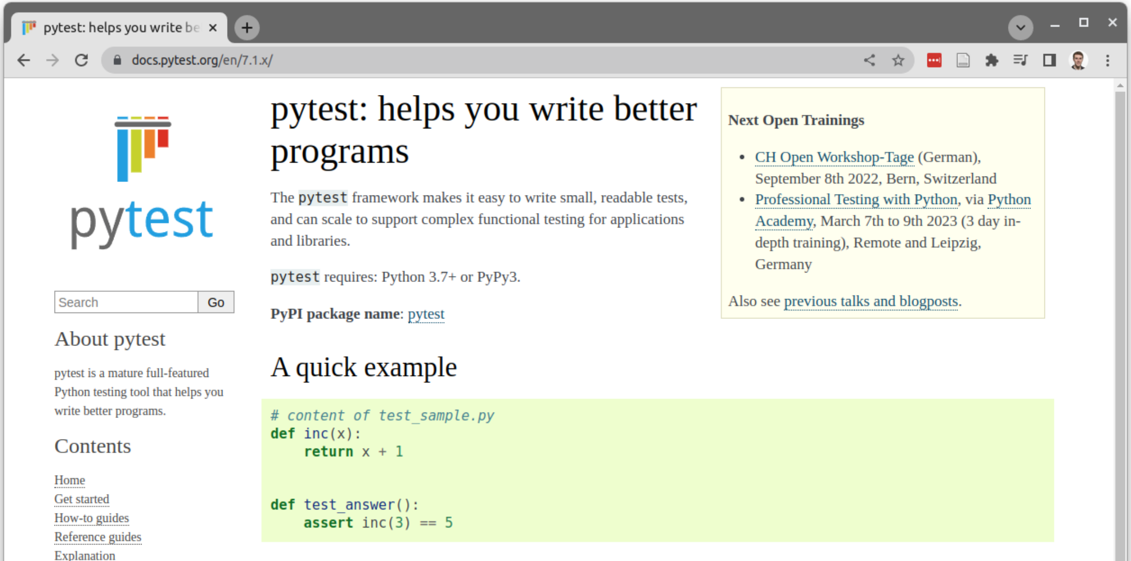
For this to work, the docstrings must of course be written in correct reStructuredText. You can then use all of the usual Sphinx markup in the docstrings, and it will end up correctly in the documentation. Together with hand-written documentation, this technique eases the pain of having to maintain two locations for documentation, while at the same time avoiding auto-generated-looking pure API documentation.

Documentation

See the examples in `autoibrav/docs/modules.rst`

Testing with pytest

Who here believes I implemented `classify` correctly?



pytest: helps you write better programs

The `pytest` framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.

`pytest` requires: Python 3.7+ or PyPy3.

PyPI package name: pytest

Next Open Trainings

- CH Open Workshop-Tage (German), September 8th 2022, Bern, Switzerland
- Professional Testing with Python, via Python Academy, March 7th to 9th 2023 (3 day in-depth training), Remote and Leipzig, Germany

Also see previous talks and blogposts.

About pytest

pytest is a mature full-featured Python testing tool that helps you write better programs.

Contents

A quick example

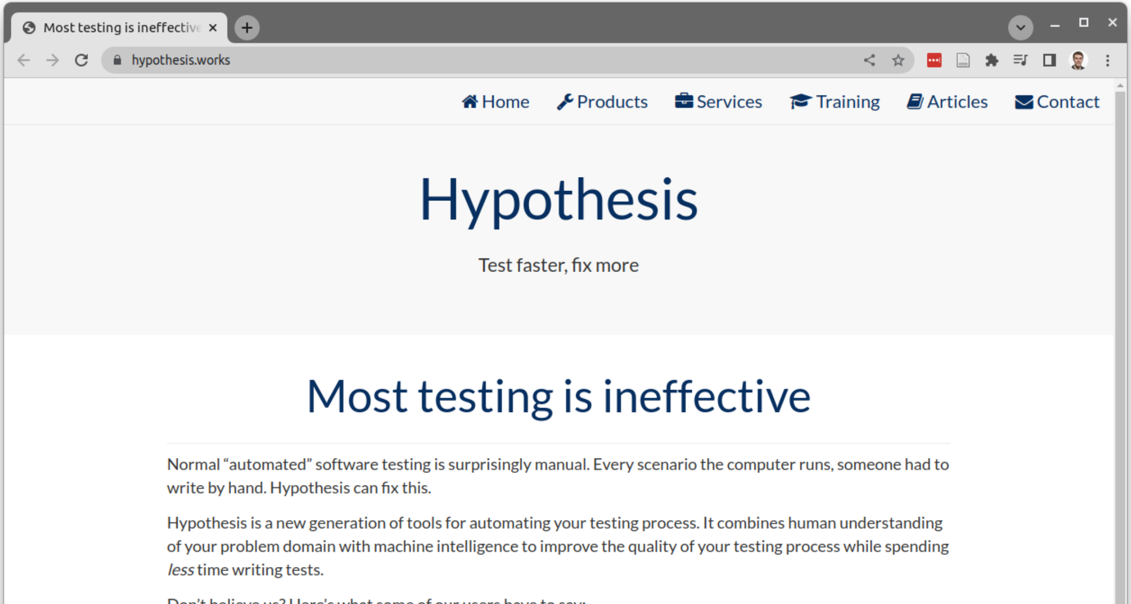
```python
# content of test_sample.py
def inc(x):
    return x + 1


def test_answer():
    assert inc(3) == 5
```

Testing with pytest

See the test examples in `autoibrav`

Edward Linscott
EPFL



Hypothesis

Test faster, fix more

Most testing is ineffective

Normal "automated" software testing is surprisingly manual. Every scenario the computer runs, someone had to write by hand. Hypothesis can fix this.

Hypothesis is a new generation of tools for automating your testing process. It combines human understanding of your problem domain with machine intelligence to improve the quality of your testing process while spending *less* time writing tests.

Don't believe us? Here's what some of our users have to say:

`hypothesis`

Edward Linscott
EPFL

See `hypothesis_example`
See `autoibrav`
See `hypothesis write hypothesis_example.hypothesis_example.decode`

Testing with pytest

A few notes:

- unit vs integration tests (e.g. QE)
- pure functions and functional programming
- there are a few things about `pytest` that are a bit “smelly” (magic fixture injection, test function discovery, parameterization syntax, ...)

Summary

- Don't know where to start? Use a cookiecutter
- For python[1] projects...
    - docstrings + `autodocs` for nice documentation
    - use `pytest` to test the code

[1] There is autodocs and pytest support for other languages, but I haven't tried them...