Project Setup and Packaging in Python

SPP 2363 - Tutorial

Marvin Friede

Mulliken Center for Theoretical Chemistry

January 10, 2023

Introduction

Motivation

- easily available for others

  $\longrightarrow$ `pip install .`
- maintainable, common setup for developers

Introduction

Motivation

- easily available for others

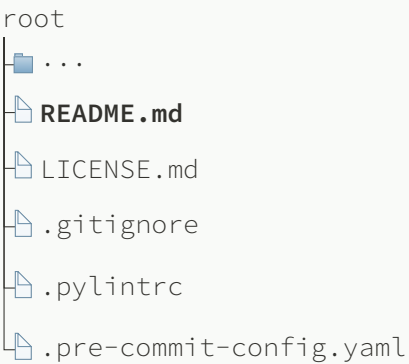
  $\longrightarrow$ `pip install .`
- maintainable, common setup for developers

Disclaimer

- there is no single best way
- only recommendations and personal preferences

GitHub: https://github.com/marvinfriede/template-python-project

Project Setup: Configuration Files



Project Setup: Configuration Files

LICENSE.md

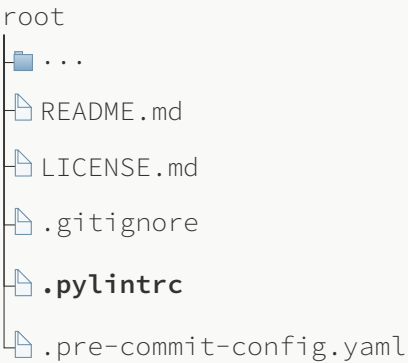
- https://choosealicense.com/

Project Setup: Configuration Files

.gitignore

- exclude files from version control (artifacts)
- https://www.gitignore.io

Project Setup: Configuration Files



.pylintrc

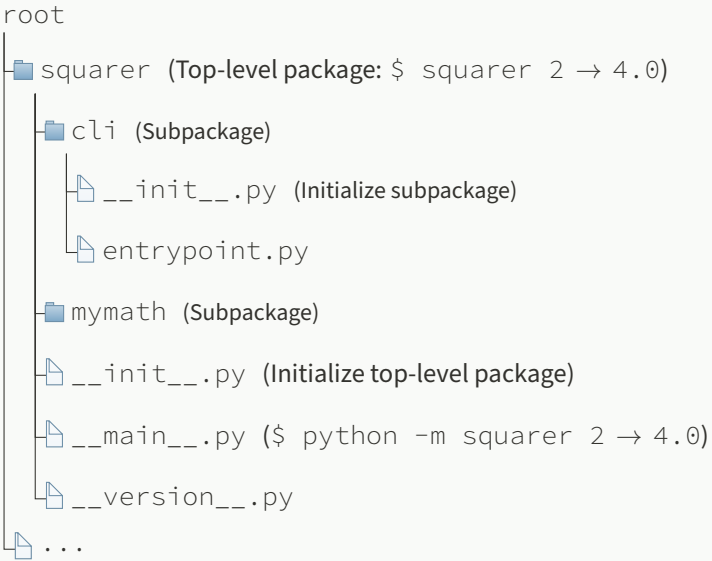
- uphold best-practices and styles
- https://google.github.io/styleguide/pyguide.html

Project Setup: Configuration Files

.pre-commit-config.yaml

- apply rules before committing
- formatting, trailing commas, type checking, ...

Project Setup: Source Code



Project Setup: Source Code

Content of `__init__.py`

squarer/cli/__init__.py

```
"""
Command line interface
======================

This module contains functionality for the CLI.
"""

from .entrypoint import console_entry_point
```

squarer/cli/entrypoint.py

```
def console_entry_point():
    ...
```

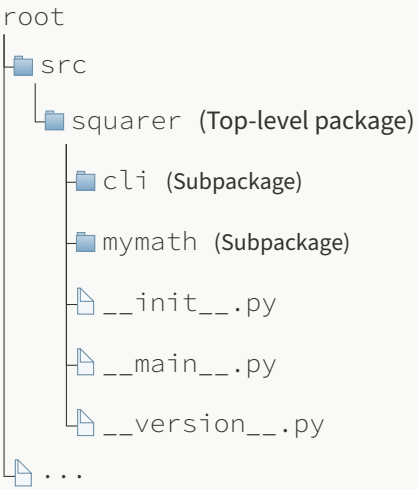
Project Setup: Source Code

Content of `__init__.py`

`squarer/__init__.py` (top-level package)

```
"""
Squarer
=======

Dummy command line tool to square a number.
"""

from .cli import console_entry_point
# from .cli.entrypoint import console_entry_point

from .__version__ import __version__
from .mymath import square_a_number as square
```

Project Setup: Source Code

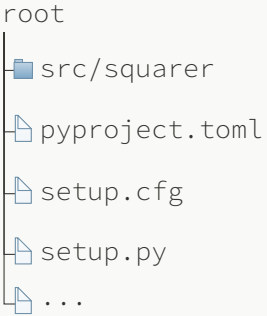


The `src/` directory

Avoid possible packaging errors with the `src/` directory layout!

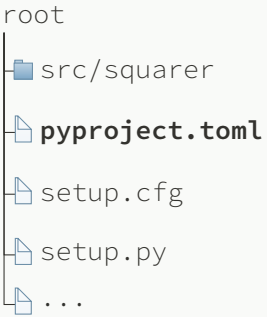
Packaging

...with `setuptools`



Packaging

...with setuptools



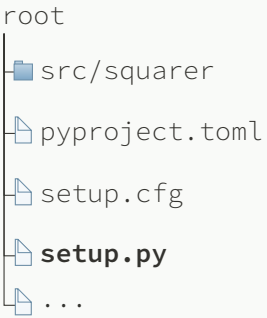

```toml
[build-system]
requires = ["setuptools"]
build-backend = "setuptools.build_meta"

[tool.pytest.ini_options]
...

[tool.mypy]
...
```

pyproject.toml

- minimal build specification to use with setuptools
- configuration of other tools (black, pytest, mypy, ...)

Packaging

...with `setuptools`




```python
from setuptools import setup

if __name__ == "__main__":
    setup()
```

setup.py

- only call `setuptools.setup()`
- configuration goes in `setup.cfg` (declarative style)

Testing

...with `pytest`

- root
  - src/squarer
  - test(s)
    - test_cli
    - test_mymath
      - __init__.py
      - test_square.py
    - __init__.py
    - conftest.py (configuration for all tests)
  - ...

pytest

- command line tool
- automatic test discovery
  - ⟶ `test_*`, `*_test`, `Test*`
- test utility
  - ⟶ parametrization
  - ⟶ error/exception testing
  - ⟶ ...

Testing

...with pytest

Generic test (test/test_math/test_square.py)

```python
import pytest
from squarer.mymath import square_a_number

def test_squarer_2() -> None:
    value = 2
    expected = value * value
    actual = square_a_number(value)

    assert pytest.approx(expected) == actual
```

Testing

...with pytest

Test with parametrization (test/test_mymath/test_square.py)

```python
import pytest
from squarer.mymath import square_a_number

def test_squarer_2() -> None:
    value = 2
    expected = value * value
    actual = square_a_number(value)
    assert pytest.approx(expected) == actual

@pytest.mark.parametrize("value", [1.0, 2, -3.0])
def test_squarer(value: int | float) -> None:
    expected = value * value
    actual = square_a_number(value)
    assert pytest.approx(expected) == actual
```

Testing

...with `pytest`

Exception test (`test/test_mymath/test_square.py`)

```python
import pytest
from squarer.mymath import square_a_number

def test_squarer_fail() -> None:
    with pytest.raises(TypeError):
        square_a_number("2")  # type: ignore
```

Implementation of squaring function (`src/squarer/mymath/ calc.py`)

```python
def square_a_number(a: float | int) -> float | int:
    if not isinstance(a, (float, int)):
        raise TypeError("Float or int expected.")

    return a * a
```

Summary

Project Layout

- source code: `src/<package_name>`
- tests: `test` or `tests`
- root: configuration files for packaging and development, README, ...

Packaging

- `pyproject.toml` (minimal), `setup.cfg` (main, declarative), `setup.py`
- install with `pip install -e .`

Tests

- `pytest`: automatic test discovery, useful utility functions
- `tox`: testing multiple environments

GitHub: https://github.com/marvinfriede/template-python-project
⟶ also contains *GitHub Actions*!

Thanks for Your Attention!