Testing in Python

Evgeniy Slobodkin

ITMO Univeristy

Fall 2023

Disclaimer

- Testing is challenging

Disclaimer

- Testing is challenging
- Mainly we will focus on unit tests

Disclaimer

- Testing is challenging
- Mainly we will focus on unit tests
- Many facts in this lecture are language-agnostic

Why do we need tests?

- Verify that new code works as expected

Why do we need tests?

- Verify that new code works as expected
- Verify that old code works as expected (*Regression testing*)

Why do we need tests?

- Verify that new code works as expected
- Verify that old code works as expected (*Regression testing*)
- Document your code

Why do we need tests?

- Verify that new code works as expected
- Verify that old code works as expected (*Regression testing*)
- Document your code
- Help newcomers with diving into the project

Why do we need tests?

- Verify that new code works as expected
- Verify that old code works as expected (*Regression testing*)
- Document your code
- Help newcomers with diving into the project
- Discipline us (!)

DOCTEST MODULE[1]

Possible way to run your doctests: `python -m doctest main.py`

[1] https://docs.python.org/3/library/doctest.html

When to use doctest?

- Test your documentation
- Do not test logic by `doctest`

UNITTEST MODULE

- Testing Framework in Python standard library
- Inspired by JUnit
- TODO

PYTEST

- Initially was originated from the PyPy project
- Many powerful features
- One **assert** for any case
- There is only reason to use `unittest` instead of `pytest`: no third-party libraries are allowed in your project

Test parametrization[1]

[1] https://docs.pytest.org/en/latest/example/parametrize.html

Test parametrization[1]

Do not do this!

[1] https://docs.pytest.org/en/latest/example/parametrize.html

Test parametrization[1]

[1] https://docs.pytest.org/en/latest/example/parametrize.html

Fixtures[1]

From pytest doc: *In testing, a fixture provides a defined, reliable and consistent context for the tests.*

[1]https://docs.pytest.org/en/latest/explanation/fixtures.html

Fixtures[1]

[1] https://docs.pytest.org/en/latest/explanation/fixtures.html

Factory as Fixture Pattern

Factory as Fixture Pattern

Libraries for Fake Data Generation

The most common are Mimesis[1] and Faker[2]

- Provide generators for different entities
- Provide localization
- Do not forget about the seed![3]

[1] https://mimesis.name/en/master/

[2] https://faker.readthedocs.io/en/master/

[3] https://mimesis.name/en/master/random_and_seed.html

Test organization

One of the examples of project organization (my preferred):

Mocking

- How to test code that calls other services or sends messages (for instance, Telegram Bot)?
- How to test code that calls some still unimplemented functions (but with known API)?

Mock it!

Different mocks[1]

- Mock
- MagicMock
- NonCallableMock
- NonCallableMagicMock
- PropertyMock
- AsyncMock

[1] `https://docs.python.org/3/library/unittest.mock.html`

Useful pytest plugins

- pytest-randomly[1]
- pytest-cov[2]
- pytest-asyncio[3]
- pytest-xdist[4]
- pytest-mock[5]
- pytest-timeout[6]

[1] https://pypi.org/project/pytest-randomly/
[2] https://pypi.org/project/pytest-cov/
[3] https://pypi.org/project/pytest-asyncio/
[4] https://pypi.org/project/pytest-xdist/
[5] https://pypi.org/project/pytest-mock/
[6] https://pypi.org/project/pytest-timeout/

Property-based testing

Hypothesis

Example