Simple, rapid and fun testing with Python

Handout / Exercises

Bruhin Software
https://bruhin.software/

April 11th, 2022

1 Setup

- We'll use Python 3.7 or newer, with pytest 7.1.
  Use `python3 --version` or `py -3 --version` (Windows) to check your version.
- You can use whatever editor/IDE you'd like – if you don't use one yet, PyCharm (Community Edition) or VS Code are good choices.
- However, we'll first start exploring pytest on the command line, in order to see how it works "under the hood" and explore various commandline arguments.
- **Download example code for exercises:** https://t.cmpl.cc/pyconde.zip

1.1 Windows: Adding `pytest` to `PATH`

On Windows, installed Python tools usually aren't available via the command-line prompt. To fix this, you will need to add the `Scripts` directory to the `Path` environment variable.

Alternatively, you can use something like `py -m pytest` etc. instead of running `pytest` directly.

Open start menu, type `env`, select *Edit environment variables for your account*.

Add something like this to the `Path` variable:

```
;C:\Python310\Scripts
```

For a per-user install, you'll probably need to use something like:

```
%LocalAppData%\Programs\Python\3.10\Scripts
```

To find out the proper folder on your system, you can run:

```
py -c "import sys; print(sys.executable)"
```

2 Virtual environments: Isolation of package installs

Virtual environments:

- Provide isolated environments for Python package installs
- Isolate different app/package-install configurations
- Are built into Python since 3.4 (but a separate `virtualenv` tool also exists)

It's recommended to set up a virtual environment for the training, so you can also experiment with pytest plugins in an isolated install.

With a virtual environment, we can avoid running `sudo pip install` ... which can mess up your system (on Linux/macOS).

2.1 Creating a virtual environment

Create a local environment (once, you can re-use `.venv`):

`python3 -m venv .venv` (alternatively: `virtualenv .venv`)

Activate the environment:

- `.venv\Scripts\activate.bat` (Windows CMD) / `Activate.ps1` (Powershell)
- `source .venv/bin/activate` (Unix)

Then install pytest and other dependencies within the activated environment:

`pip install -r code/requirements.txt` (or just `pip install pytest`)

Now let's see if it works:

```
pytest -h
```

3 Basics

3.1 Getting started

- Write a test function, play with options, help your neighbour
- Insert a `print(...)` call in a passing/failing test.

3.2 src-layout

Ionel Cristian Mărieș (`ionelmc.ro`):
“Packaging a python library”
(also for applications!)

Hynek Schlawack (`hynek.me`):
“Testing & Packaging”

3.3 Persisting command line options

- Add some options to the `addopts` variable
- See other `pytest.ini` options at end of the `pytest -h` output

3.4 Asserting expected exceptions

`basic/test_raises.py`

- Use `pytest.raises` in a new test functionn
- Try its `match=r"..."` argument to check the exception message

4 Marks

4.1 Skip and xfail

- See `pytest --markers` for reference
- Write a declaratively skipped test (using a marker)
- Write a xfail-marked test
- Use `pytest.skip()` and `pytest.xfail()` from a test function
- Run with `-v` to see skip/xfail reasons

4.2 Parametrizing

`marking/test_parametrization.py`

- Find a function to test which uses arguments (e.g. `divide`)
- Write a test for it with a single value
- Parametrize the test to test multiple inputs and expected outputs

5 Fixtures

5.1 Fixture basics

`fixtures/test_fixture.py`

- Run `pytest --setup-show test_fixture.py`, observe how fixtures are created, used and cleaned up
  (ignore the `TEARDOWN` part for now)
- Write and use another fixture function in the same test
- Add `pytest.skip("skipped")` to fixture function (note: imperative variant, not mark)

5.2 tmp_path

`fixtures/test_tmp_path.py`

- Use the `tmp_path` fixture from another test function, read the text from the file
- Returns a `pathlib.Path` object: `docs.python.org/3/library/pathlib.html`

5.3 monkeypatch

`fixtures/test_monkeypatch.py`

- Write a function reading a password from the terminal using `getpass.getpass()`
- Use `monkeypatch.setattr(module, 'attrname', lambda: 'returnvalue')` in a test of that function

5.4 Caching fixture results

`fixtures/test_fixture_scope.py`

Use `scope="module"`, observe runtime (try `--durations=5` for additional info)

5.5 Doing cleanup with yield

`fixtures/test_yield_fixture.py`

- Write `Client` class with `connect`/`disconnect` methods (could e.g. print some text)
- Add a couple of tests using `connected_client`
- Observe teardown behaviour using `-s` and/or `--setup-show`
- Modify fixture scope, check how the behavior changes

5.6 Autouse fixtures

- Write a `Database` class with prints in `__init__`, `begin` and `finish` methods.
- Write a session-scoped `database` fixture in a `conftest.py`
- Write an autoused `transaction` fixture which uses `database` and performs `database.begin()` (i.e. start transaction) and `database.finish()` (i.e. rollback changes) around each test function/method
- Write two tests with print calls in each
- Run with `--setup-show` and/or `-s` to check behaviour

6 Book

- Brian Okken: Python Testing with pytest, Second Edition (The Pragmatic Bookshelf)
- ISBN 978-1680508604
- `https://pragprog.com/titles/bopytest2/`
- Discount code: **PyConDE**
  30% off DRM-free eBook until April 18th
  (≈ $17 instead of $25; .pdf/.epub/.mobi)
- Full disclosure: I'm technical reviewer (but don't earn any money from it)

7 In-depth trainings

- **March 7th to 9th, 2023:**
  Python Academy (`python-academy.com`):
  Professional Testing with Python
  Leipzig (Germany) and remote
- **Custom training / coaching:**
  - Python
  - pytest
  - GUI programming with Qt
  - Best Practices (packaging, linting, etc.)
  - Git
  - . . .

  Remote or on-site

8 Feedback and questions

Florian Bruhin
`florian@bruhin.software`
`https://bruhin.software/`
`@the_compiler` on Twitter

Copyright 2015 – 2022 Florian Bruhin, all rights reserved.
Reuse or redistribution is prohibited without prior written permission.

Originally based on materials copyright 2013 – 2015 by Holger Krekel, used with permission.