Introduction to pytest

Exercises

Bruhin Software
https://bruhin.software/

Europython 2021
July 27th 2021

1 Setup

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

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

Activate the environment:

- `.venv\Scripts\activate` (Windows)
- `source .venv/bin/activate` (Unix)

Then install pytest within the activated environment:

```
pip install pytest
```

Now let's see if it works:

```
pytest -h
```

Exercise code: `https://t.cmpl.cc/ep2021.zip`

2 Basics

2.1 Getting started

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

2.2 Persisting command line options

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

2.3 Asserting expected exceptions

`basic/test_raises.py`

- Use `pytest.raises` in a new test function

3 Marks

3.1 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

3.2 Parametrizing

`basic/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

4 Fixtures

4.1 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)

4.2 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`

4.3 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

4.4 Caching fixture results

`fixtures/test_fixture_scope.py`

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

4.5 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

4.6 conftest.py fixtures

Shift one of your fixture functions to a new `conftest.py` file in the same directory

4.7 Autouse fixtures

- 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