Guidelines for Unit Tests

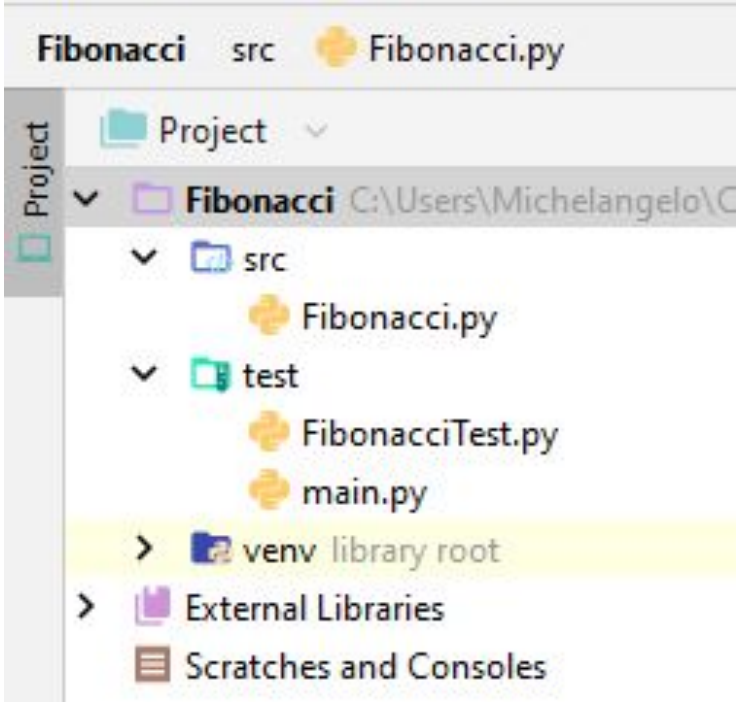
Giuseppe Scanniello

Simone Romano

Michelangelo Esposito

Separate test code from production one

Test classes and tested classes should be located in different source folders



A.A.A. pattern

You should be able to divide a test method into three subsequent sections:

- **Arrange---**in this section, you should arrange all necessary preconditions and inputs
  - You can also setup methods to arrange all necessary preconditions and inputs
- **Act---**in this section, you should exercise the unit to be tested
- **Assert---**in this section, you should make sure that the expected results have been occurred

Why A.A.A. pattern?

Once you have identified the three sections, you can easily grasp if your test method **smells bad**

```
def test_bad_push_pop_test(self):
    # Arrange
    stack = Stack()
    size = stack.size()          # Warning: action in the Arrange section!
    self.assertEqual(0, size)    # Warning: assertion in the Arrange section!

    # Act
    stack.push(1)
    size = stack.size()

    # Assert
    self.assertEqual(1, size)
    popped = stack.pop()         # Warning: action in the Assert section!
    self.assertEqual(1, popped)
    size = stack.size()          # Warning: action in the Assert section!
    self.assertEqual(0, size)
```

One assert per test

A unit test should contain just one assert

- If it contains more than one assert, then split that test into two tests having one assert each

Why one assert per test?

- If a test method has more than one assert, it is hard to tell which assertion has caused a test failure

Fast

Tests should run quickly

- If they depend on expensive resources, replace those resources with test doubles

Why fast tests?

- Developers need immediate feedback
- If tests run slow, you will not run them frequently; if you do not run them frequently, you will not find problems early enough to fix them easily

Independent

Tests should not depend on each other

Why independent tests?

When tests depend on each other, then the first one can cause the other tests to fail so making diagnosis difficult

Repeatable

Tests should be repeatable in any environment

- You can use test doubles to make tests repeatable

Why repeatable tests?

- If your tests are not repeatable in any environment, then you will always have an excuse for why they fail
- It is a matter of trust in your tests

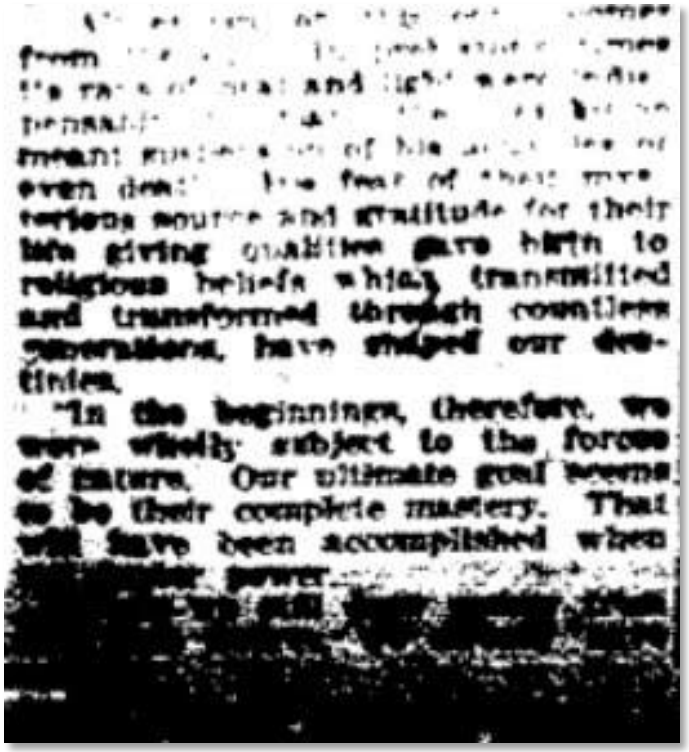
Self-validating

The tests should have a Boolean outcome, namely either they pass or fail

Why self-validating tests?

You would like to have completely automated tests so that no manual work is needed to understand whether a test has passed or not

Further guidelines for unit tests



"In the beginnings, therefore, we were wholly subject to the forces of nature. Our ultimate goal seems to be their complete mastery. That will have been accomplished when

A test should be readable and meaningful

Self-documenting tests

A test should not be overprotective

- Remove redundant assertions

Tests should be maintainable

- Avoid duplication

Yes, tests should be maintainable

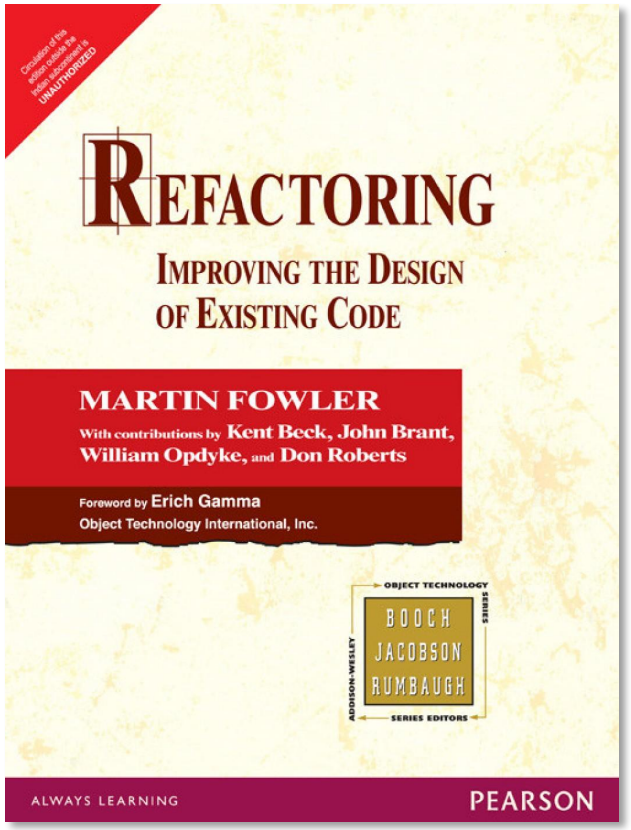
Do not use conditionals in your tests

Never failing tests do not make sense

Do not write tests without assertions

Tests should span the best of both worlds

Test happy paths - Test sad paths



Refactor test code

Test code IS code