Pytest: Where Bugs Come to Die, No Exceptions

Jay Smallwood

April 4, 2025

Learning Intentions

- Be able to explain when and why a piece of code needs tests.
- Be able to structure tests using the Arrange-Act-Assert framework.
- Be able to write and run simple tests using pytest & ipytest.
- Understand that writing testable code requires a slightly different method of writing code.
- Have all the necessary understanding to be able to start writing tests on your own code **today**.

Structure

- Setup local environments.
- Examples of unit testing.
- What is testing? Why do we write tests?
- What code requires tests? What code does not require tests?
- What is Arrange-Act-Assert?
- Refactoring code to make it testable.
- Public & Private functions - what to test?
- Write a test on one of your own functions.

Setting up the environment

- Clone ¡insert git repo¿
- Follow the instructions in the README.md

Some simple tests...

Refer to Example 1 in the notebook.

Why Test?

What do you think testing is? Why do we do it?

It feels like...

Credit: Hasbro

Why Test?

- Writing automated unit tests allows us to change code and ensure we do not break existing functionality.

Why Test?

- Writing automated unit tests allows us to change code and ensure we do not break existing functionality.
- Unit tests are a contract for the functionality of code. They communicate the intent of the code.

Why Test?

- Writing automated unit tests allows us to change code and ensure we do not break existing functionality.
- Unit tests are a contract for the functionality of code. They communicate the intent of the code.
- Unit tests act as documentation for code.

Benefits & Myths...

Benefits of testing:

- Promotes code reuse.
- Less bugs! A test suite that evolves as you fix bugs so they NEVER occur again.
- Documents the code.
- Forces you to write more modular code.

Benefits & Myths...

Benefits of testing:

- Promotes code reuse.
- Less bugs! A test suite that evolves as you fix bugs so they NEVER occur again.
- Documents the code.
- Forces you to write more modular code.

Myths about testing:

- Testing is hard.
- Testing takes too much time.
- Writing tests is only if you want it to be "perfect".

Does this code need tests?

- I have a piece of code that runs on a server every day at 9am.
- It's run for 15 years and has not been altered since 1992.
- Should I go and write tests for this function?

Which of these need tests?

- Simple 5 line script that calls an API & sends an email.
- Public function of a library that is in active development.
- Private function of a library that is in active development.

Examples

Refer to example 2 in the Jupyter notebook.

Arrange-Act-Assert

- **Arrange:** Create inputs to the function or class you are testing.
- **Act:** Call the function or class you are testing.
- **Assert:** Assert that you get the output you expected.

What is Testable Code?

- Deterministic.

What is Testable Code?

- Deterministic.
- In a function or class.

What is Testable Code?

- Deterministic.
- In a function or class.
- De-coupled.

Refactoring Example

See Example 3 in the Jupyter notebook.