

Unit Testing

JUNE 30, 2021

What is Unit Testing?

Unit testing is the testing of individual components.

https://en.wikipedia.org/wiki/Unit_testing

Python Unit Tests

Almost all languages have unit testing frameworks.

Python unit testing framework documentation: <https://docs.python.org/3/library/unittest.html>

Asserts are where the test logic is - here are some common ones:

`assertEqual(a, b)`

`assertNotEqual(a, b)`

`assertTrue(a)`

`assertFalse(a)`

`assertRaises(a)`

`assertAlmostEqual(a, b)`

`assertNotAlmostEqual(a, b)`

Test File Template


```
import unittest

from yourCodeFileName import function1, function2

class TestFileName(unittest.TestCase):
    def test_function1(self):
        self.assertEqual(function1(1), 0)

    def test_function2(self):
        self.assertEqual(function2(2,1), 3)
        self.assertEqual(function2(2.1, 1.2), 3.3)

if __name__ == '__main__':
    unittest.main()
```



Be sure to update
at least the parts
in *italics* for your
program

Running Tests

You can run the testing script on the command line as follows:

```
python3 -m unittest testFileName.py
```

. means pass, so if you have 2 tests that pass you'll see ..

.F means first test passed, second test failed

Today's Activity

1. Create unit tests for your program that loads API data into a database
 2. Setup a Github Action to automatically run your tests on every commit
- You will go into your TA's breakout room and work on the activity.
 - Feel free to work with others in your room and ask questions of the group.
 - The TA is there to assist you. If you want to ask me a question, come back to the main zoom room.

Setup Github Action

You can do the following from either the terminal in Codio or on the GitHub website:

1. Create a directory called `.github` (don't forget the leading dot!)
2. Inside of the `.github` directory, create a directory called `workflows`.
3. Inside of the `workflows` directory, create a `yml` file with the code on the next slide. This file will be used to automatically run your tests. Name it accordingly.

YAML code to set up a GitHub Action

You may have to add tool installs. For my example, I had to install pandas, requests, sqlalchemy, and datetime. See the example below for how to add a pandas install. Be sure to update the name for each tool you install.

- name: Install pandas

```
run: python -m pip install pandas
```

**Make sure to replace the file name
with the name of your unit tests file**



```
name: Tests
on: push
jobs:
  unit-tests:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Setup python
        uses: actions/setup-python@v2
        with:
          python-version: 3.6
      - name: Install tools
        run: python -m pip install --upgrade pip pytest
      - name: Test with unittest
        run: |
          python3 -m unittest test.py
```


Additional Testing Terminology

- A **stub** can replace an object that isn't built yet. It's a dummy piece of code that lets the test run, but you don't care what happens to it. A stub could be replaced when the functionality is added.
- A **mock** is a fake object that mimics an actual object. It's also a dummy piece of code, but you verify that it is called correctly as part of the test.
- A **stub** will never fail a unit test, but a **mock** can.
- A **fake** can refer to either a mock or a stub - any piece of code that is pretending to be fully implemented, production code.
- A spy is an observation point to check if code calls a component. **Spies** are a way to check if a function was called or to provide a custom return value. We can use **spies** to test components that depend on a service to avoid actually calling the service's methods to get a value. This helps keep our unit tests focused on testing the internals of the component itself instead of its dependencies.
- A **dummy** is an object that is passed around but never used

Sources:

<https://stackoverflow.com/questions/3459287/whats-the-difference-between-a-mock-stub>

<https://www.digitalocean.com/community/tutorials/angular-testing-with-spies>

Comparing Fakes

Parameters	Stub	Mock
Data Source	The data source of stubs is hardcoded. It is usually tightly coupled to the test suite.	Data on mocks is set up by the tests.
Created by	Stubs are usually handwritten, and some are generated by tools.	Mocks are usually created by using the third-party library such as Mockito, JMock, and WireMock.
Usage	Stubs are mainly used for simple test suites.	Mocks are mainly used for large test suites.

Parameters	Mock	Spy
Usage	Mocks are used to create fully mock or dummy objects. It is mainly used in large test suites.	Spies are used for creating partial or half mock objects. Like mock, spies are also used in large test suites.
Default behavior	When using mock objects, the default behavior of methods (when not stubbed) is do nothing (performs nothing.)	When using spy objects, the default behavior of the methods (when not stubbed) is the real method behavior.



Questions?
