Unit Testing 3 cheatsheet

Pytest

Installation

```
$ pip install pytest
```

1. File names should begin or end with `test`, as in `test_example.py` or `example_test.py`.
2. Function names should begin with `test_`. So for instance: `test_example`.
3. If tests are defined as methods on a class, the class should start with `Test`, as in `TestExample`.
4. You can run `pytest --collect-only` to see which tests `pytest` will discover, without running them.

```
# test_additions.py
def add_two_numbers(a, b):
    return a + b

def test_add_two_numbers():
    expected = 5
    actual = add_two_numbers(4, 1)
    assert expected == actual
```

Copy the code to a Python file, run `python -m pytest` and watch the output. Hopefully you should see some information about 1 test passing.

Mocking

`Mock()` creates a Mock object `return_value` Specifies the return value when the mock is called (*stub*) `side_effect` Specifies some other function when the mock is called. `call_count` Returns the amount of times the mock has been called `called_with` Returns the parameters passed into the mock when called `called` Returns a `bool` indicating if the mock has been called or not

```
from random import randint
from unittest.mock import Mock

def add_two_numbers(a, randint):
    return a + randint(1, 10)

def test_add_two_numbers():
    # Creates a new mock instance
    mock_get_random_number = Mock()
    mock_get_random_number.return_value = 5

    expected = 10
    actual = add_two_numbers(5, mock_get_random_number)
    assert expected == actual
    assert mock_get_random_number.call_count = 1
    assert mock_get_random_number.called
```

Mocking Assertions

`assert_called()` Fails if mock is not called `assert_not_called()` Fails if mock is called `assert_called_with(*args)` Fails if the mock is not called with the specified params `reset_mock()` Resets mock back to the initial state. Useful if testing one mock under multiple scenarios

```
mock_function = Mock()
mock_function.return_value = True
mock_function() # True
mock_function.call_count # 1
mock_function() # True
mock_function.reset_mock()
mock_function.assert_called() # Fails
```

Patching

`@patch("path.to.module.method")` creates a Mock object which needs to be passed in as function parameter

```
from unittest.mock import patch

def hello_world(): # No DI
    print("Hello World!") # Dependency

@patch("builtins.print")
def test_prints_hello_world(mock_print):
    hello_world() # Act
    mock_print.assert_called_with("Hello World!") # Passes
```

Unit Testing Terms and Definitions

- `Mock` : A piece of *fake* code standing in to replace some *real* code.
- `Stub` : Dummy data serving to replace real data usually returned from an external source.