Unit Test

References:
Unit Testing Framework - Python Docs

What is unittest?

unittest is an object oriented testing framework It has four main components:

- test fixture : A *test fixture* represents the preparation and cleanup for testing.
- test case: A *test case* is the individual unit of testing.
- test suite: A *test suite* is a collection of test cases, test suites, or both. It is used to aggregate related tests and suites.
- test runner: A *test runner* is a component which orchestrates the execution of tests.

Simple Example:

```
import unittest                                 # Import framework

class TestStringMethods(unittest.TestCase):     # Define test suite

    def test_upper(self):                       # Define a test case
        self.assertEqual('foo'.upper(), 'FOO')  # Call runner

    def test_isupper(self):                     # Define another test case
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):                       # And another
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        with self.assertRaises(TypeError):
            s.split(2)

if __name__ == '__main__':                    # If in main, return result of test
    unittest.main()
```

setUp() and tearDown()

`setUp()` and `tearDown()` allow for the automation of the preparation and clean-up for any number of test cases in a *test suite*.

- `setUp()` : Use to automate setup for various test cases in a suite. This code will be run before each test case.
- `tearDown()` : Will clean after the test has been run, whether or not the test has been successful.

Where should testing code go?

Testing code can go in the same module that it is intended to test. But there are several benefits to including it in its own module:

- The test module can be run standalone from the command line.
- The test code can more easily be separated from shipped code.
- There is less temptation to change the test, rather than change the code.
- Test code should be modified less frequently than the code it tests.
- Tested can be refactored more easily.
- If the testing strategy changes then there is no reasons to change the source.