Lab 11: Unit Testing

What is testing?

Testing is the practice of writing methods that invoke the application code to determine if there are any errors.

It does not prove that the code is correct

If merely reports if the conditions the tester thought of are handled correctly.

More importantly:-

It ensures that any new changes made did not break the existing working code

Types of Tests

Unit Tests

- Tests that cover low-level aspects of a system
- For each module, does each operation perform as expected

Integration Tests

- Tests that check that modules work together in combination
- All sorts of hidden assumptions are surfaced when code written by different developers are used in tandem.

Source: Prof. Kenneth Anderson Fall 2015

Types of Tests

System Tests

- Tests performed to ensure that all major functionality has been implemented.
- Conducted to evaluate the entire integrated system.
- It is a form of black-box testing and should require no knowledge of the inner design of the code or logic.

Acceptance Tests (Usually done manually by the clients)

- Tests performed by the user to check that the delivered system meets their needs.
- In large, custom projects, developers will be on-site to install system and then respond to problems as they arise.

Source: Prof. Kenneth Anderson Fall 2015

Unit Testing

- Unit testing, specifically tests a single "unit" of code in isolation.
- A unit could be an entire module, a single class or function, or almost anything in between. What's important, however, is that the code is isolated from other code we're not testing.

Python Unit Testing Framework : Unittest / PyUnit

- Provides test automation and aggregation of tests into test suite

Components of Unittest

1. Test fixtures : Preparation to perform tests and the associated cleanup actions

Eg: setUp() and tearDown() functions

2. Test case : To check for specific response to a particular set of inputs

NOTE: Unittest provides a base class called “TestCase” which could be used to create new test cases.

3. Test Suite : Collection of test cases

NOTE: Uses TestSuite class

4. Test Runner: Returns result of executing the tests.

Anatomy of a Unit Test [in Python]

We will use Unittest : The unit test framework for Python.

A test case is created by subclassing unittest.TestCase

A unit test consists of one or more assertions (statements that assert that a property of the code being tested is true)

The individual tests are defined with methods whose names start with the word test. This naming convention informs the test runner about which methods represent tests.

Example

```
# Application code
def sum(a, b):
        return a + b


# Test code
def test_sum_of_two_numbers():
        expected_sum = 10
        actual_sum = sum(2, 8)
        assertEqual(expected_sum, actual_sum)
```

Method	Checks that
`assertEqual(a, b)`	`a == b`
`assertNotEqual(a, b)`	`a != b`
`assertTrue(x)`	`bool(x) is True`
`assertFalse(x)`	`bool(x) is False`
`assertIs(a, b)`	`a is b`
`assertIsNot(a, b)`	`a is not b`
`assertIsNone(x)`	`x is None`
`assertIsNotNone(x)`	`x is not None`
`assertIn(a, b)`	`a in b`
`assertNotIn(a, b)`	`a not in b`
`assertIsInstance(a, b)`	`isinstance(a, b)`
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`