Python Unit Testing

In this section, you'll learn about unit testing in Python by using the unittest modules to make your code more robust.

What you'll learn:

- Write effective test cases using the `unittest` module
- Run unit tests fast
- Skip tests unconditionally and conditionally
- Using test doubles including mocks, stubs, and fakes
- Parameterize tests
- Generate test coverage reports

Section 1. Introduction to unit testing in Python

This section introduces you to the unit testing and the unittest module. After completing this section, you'll know how to define and execute unit tests effectively.

▸

- What is unit testing (https://www.pythontutorial.net/python-unit-testing/python-unittest/) – introduce you to the unittest testing and how to use the unittest module to perform unit tests.
- Test fixtures (https://www.pythontutorial.net/python-unit-testing/python-test-fixtures/) – learn how to use test fixtures including setUp() and tearDown() to carry out the steps before and after test methods.
- Skipping tests (https://www.pythontutorial.net/python-unit-testing/python-unittest-skip-test/) – guide you on how to skip a test method or test class.
- Running unittest (https://www.pythontutorial.net/python-unit-testing/python-run-unittest/) – show you various commands to run unit tests.

Section 2. assert methods

This section covers the assert methods so that you know how to each of them more effectively.

- assert methods (https://www.pythontutorial.net/python-unit-testing/python-unittest-assert/) – introduce to you a brief overview of the assert methods of the `TestCase` class.
- assertEqual() (https://www.pythontutorial.net/python-unit-testing/python-assertequal/) – test if two values are equal.
- assertAlmostEqual() (https://www.pythontutorial.net/python-unit-testing/python-assertalmostequal/) – test if two values are approximately equal.
- assertIs() (https://www.pythontutorial.net/python-unit-testing/python-assertis/) – test if two objects are the same.
- assertIsInstance() (https://www.pythontutorial.net/python-unit-testing/python-assertisinstance/) – test if an object is an instance of a class or a tuple of classes.
- assertIsNone() (https://www.pythontutorial.net/python-unit-testing/python-assertisnone/) – test if an expression is None.
- assertTrue() (https://www.pythontutorial.net/python-unit-testing/python-asserttrue/) – test if an expression is True.

- assertIn() (https://www.pythontutorial.net/python-unit-testing/python-assertin/) – test if a member is in a container.

Section 3. Test doubles

This section introduces to you the test doubles to decouple the system under test code from the rest of the system so that code can be tested in isolation.

- Mock (https://www.pythontutorial.net/python-unit-testing/python-unittest-mock/) – learn how to use the Mock class to mimic behaviors of another function or class.
- patch() (https://www.pythontutorial.net/python-unit-testing/python-patch/) – show you how to use the patch() to temporarily replace an object with another object for testing.
- Stubs (https://www.pythontutorial.net/python-unit-testing/python-stubs/) – show you how to use the MagicMock class & patch() to create stubs.
- Mocking requests module (https://www.pythontutorial.net/python-unit-testing/python-mock-requests/) – learn how to mock the requests module to test an API call using the unittest module.

Section 4. Test coverage & Parameterized tests

This section introduces you to the test coverage and how to define parameterized tests using the `subTest()` context manager.

- Generating test coverage reports (https://www.pythontutorial.net/python-unit-testing/python-unittest-coverage/) – learn about test coverage and how to generate the test coverage report using the coverage module.
- Defining parameterized tests using subTest() (https://www.pythontutorial.net/python-unit-testing/python-unittest-subtest/) – show you how to define parameterized tests using the unittest's subTest() context manager.