

Unit Testing with Python

Introduction to Unit Testing in Python

- **Unit testing** involves testing individual units of code to ensure they function as expected.
- Helps catch bugs early, simplifies debugging, and improves code quality.
- *unittest* is Python's built-in framework for writing and running tests ([Tutorial](#))

Basics of the *unittest* Framework

- **TestCase Class:** Inherit from *unittest.TestCase* to create test cases.
- **Assertions:** Use various assert methods (e.g., *assertEqual*, *assertTrue*, ...) to check code behavior.
- **Example:**

```
def test_addition(self):  
    self.assertEqual(1 + 1, 2)
```

- **Test Organization:** Group related tests in a single class.
- *setUp* and *tearDown*: Methods to prepare a consistent environment before/after each test.

```
def setUp(self):  
    self.obj = MyClass()  
  
def tearDown(self):  
    del self.obj
```

- **Write Clear Test Names:** Use descriptive names for your test methods.
- **Test Small Units:** Focus on small, independent units.
- **Regularly Run Tests:** Integrate tests into your development workflow.

1. **Basic Test Case:** Create a function that adds two numbers. Write a unit test to verify the result.
2. **Test Fixtures:** Implement *setUp* and *tearDown* methods to initialize and clean up resources (e.g., open/close files).
3. **Test Assertions:** Write a function that reverses a string, then create unit tests using different ``assert`` methods like *assertEqual*, *assertTrue*, and *assertIn*.

4. **Handling Exceptions:** Write a function that raises a *ValueError* for invalid inputs, and write a test case that checks if the exception is raised correctly.
5. **Skipping Tests:** Write tests for a function that multiplies numbers, but skip one of the tests under certain conditions (e.g., skip if a number is negative).
6. **Test Discovery:** Create multiple test files and use Python's test discovery feature to run them together.

Role-playing: Form a group with another student. One group member assumes the role of the Python programmer, while the other becomes the tester, who will develop unit tests using the *unittest* framework.

Once the task is completed and validated, switch roles with your group member (i.e., the programmer becomes the tester and vice versa).

