

**Batch: B1 Roll No.: 16010421 Experiment No.:2 Aim:** To perform unit testing using JUnit/PyTest.

\_

**Resources needed:** Java, JUnit / Python, PyTest

# Theory:

Unit Testing in Python is done to identify bugs early in the development stage of the application when bugs are less recurrent and less expensive to fix. A unit test is a scripted code level test designed in Python to verify a small "unit" of functionality. Unit test is an object oriented framework based around test fixtures.

JUnit is a [unit testing](https://en.wikipedia.org/wiki/Unit_testing) [framework](https://en.wikipedia.org/wiki/Software_framework) for the [Java programming language](https://en.wikipedia.org/wiki/Java_(programming_language)). JUnit has been important in the development of [test-driven development](https://en.wikipedia.org/wiki/Test-driven_development), and is one of a family of [unit](https://en.wikipedia.org/wiki/Unit_testing) [testing](https://en.wikipedia.org/wiki/Unit_testing) frameworks.

Unit testing is used to verify a small chunk of code by creating a path, function or a method. The term "unit" exists earlier than the object-oriented era. It is basically a natural abstraction of an object oriented system i.e. a Java class or object (its instantiated form).

# Unit testing and its importance

Unit testing is used to identify defects early in software development cycle.

Unit Testing will compel to read our own code. i.e. a developer starts spending more time in reading than writing.

Defects in the design of code affect the development system. A successful code breeds the confidence of developer.

# Procedure:

1. Explore any one tool for unit testing.

Perform unit testing.

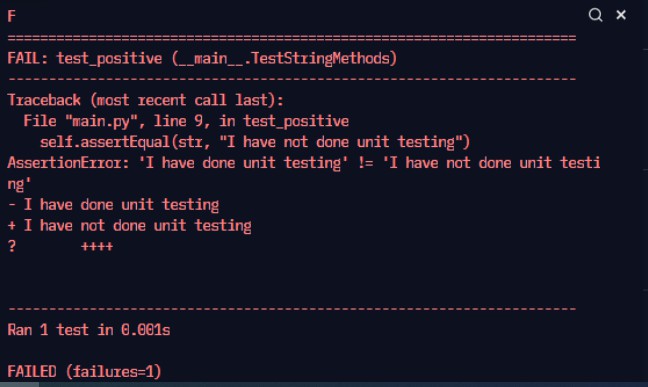
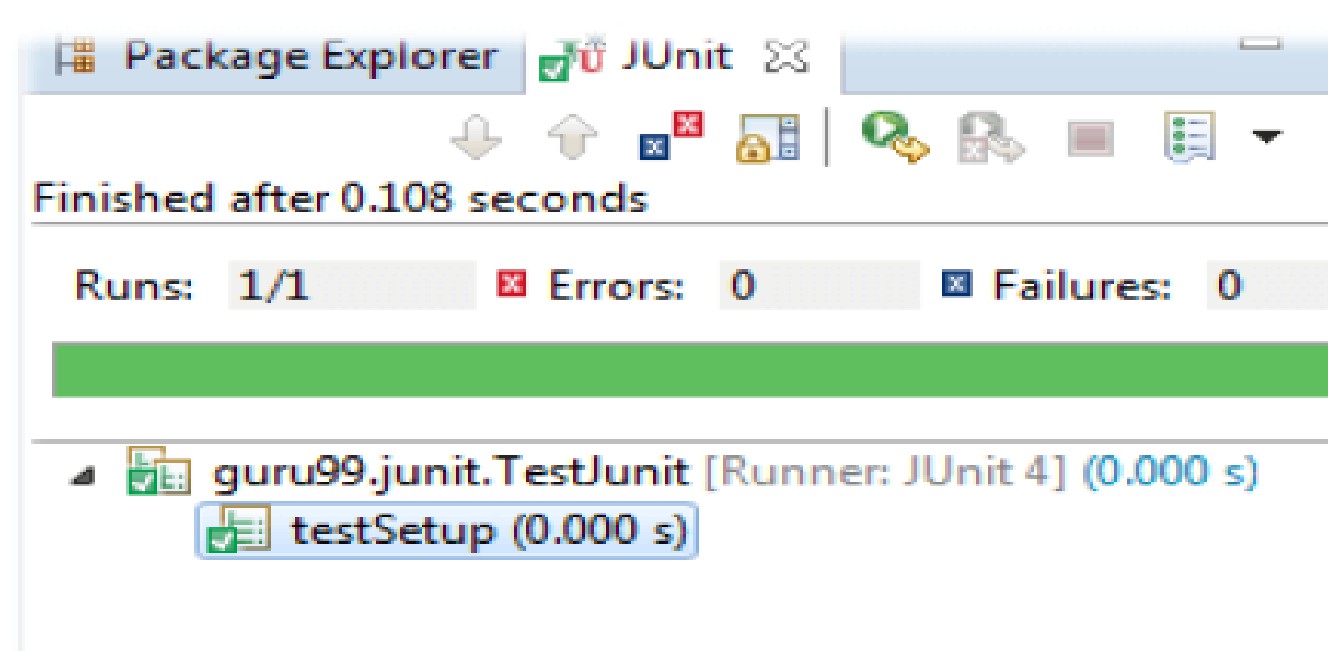


2.

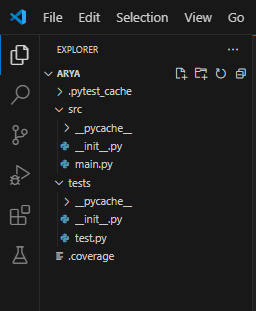
3.

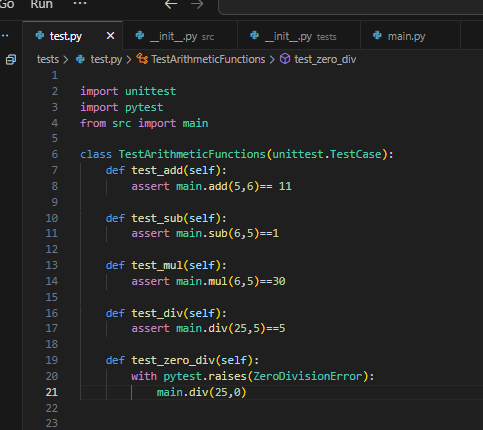
Install JUnit for Java or PyTest for Python

|  |  |
| --- | --- |
| **JUnit** | **PyTest** |
| **Step 1)** Create a java class named TestJUnit.java and provide a simple assert statement. | **Step 1)** Create a python class named Tests and provide a simple assert statement. |
| **Step 2)** Create a Test Runner class to execute above test. | **Step 2)** Run above code. Here is the output or result for your test. |
| **Step 3)** To execute the test, follow below steps:   1. Right click on TestRunner.java and click on "Run As" as shown below 2. Another window will be open once you click on "Run As", click on " JUnit Test" as shown below: | **Step 3)** Create a python class named Tests and provide a simple assert statement. |
| **Step 4)** Here is the output or result for your test. If it is successfully executed, it will show a green check mark in front of it. | **Step 4)** Here is the output or result for your test. If it is not successfully executed |

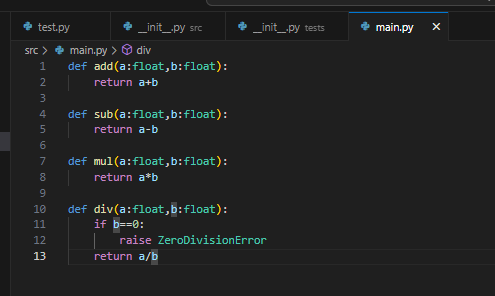


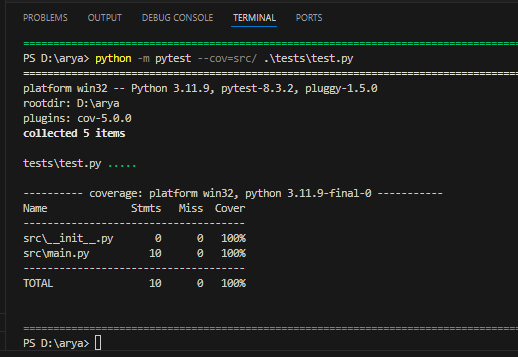
# Results: (Document printout as per the format)

Folder structure



Test.py file which contains all the unit tests

main.py file which contains the application code



Written test cases with 100% coverage.

# Questions:

1. Explain unit testing framework.

A unit testing framework is a tool for testing individual units of code, like functions or methods, to ensure they work correctly. It provides a structured way to write tests, check results with assertions, automate test execution, and generate reports on test outcomes. Key features often include setup and teardown functions, isolation of tests, and support for mocking dependencies. Examples include JUnit for Java, pytest for Python, and Jest for JavaScript.

1. Explain Test-Driven Development TDD.

Test-Driven Development (TDD) is a method where you write tests before coding. The process involves:

Write a Test: Create a test for a new feature. Run the Test: Ensure it fails (as expected).

Write Code: Implement the minimum code to pass the test. Run the Test Again: Check that it now passes.

Refactor: Improve the code while keeping tests green. This cycle helps ensure code correctness and quality.

1. What is Assert in unit testing?

In unit testing, an assert is a function or method used to check if a condition is true during the test execution. If the condition is true, the test passes; if it's false, the test fails.

For example:

assertEqual(a, b) checks if a is equal to b. assertTrue(condition) checks if condition is true. assertFalse(condition) checks if condition is false.

Assertions help verify that the code behaves as expected and are central to identifying and diagnosing issues in tests.

**Outcomes:** CO2: Demonstrating designing and execution of test cases using testing techniques.

# Conclusion: (Conclusion to be based on outcomes)

Understood and implemented pytest with 100% coverage.

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**

**References:**

**Books/ Journals/ Websites:**

1. https://junit.org/junit4/
2. [https://www.guru99.com/junit-tutorial.html](http://www.guru99.com/junit-tutorial.html)
3. <https://www.guru99.com/download-installation-junit.html>
4. https://docs.pytest.org/en/6.2.x/getting-started.html
5. <https://www.guru99.com/python-unit-testing-guide.html>
6. [https://www.tutorialspoint.com/pytest/pytest\_environment\_setup.html](http://www.tutorialspoint.com/pytest/pytest_environment_setup.html)