**Lab Exercise 5- : Unit testing in python**

**Lab Exercise: Unit Testing in Python (Using unittest)**

**Objective:**

To understand how to write and run unit tests for Python code using the unittest module in a Jupyter Notebook or Google Colab environment.

**Step 1: Writing a Simple Function**

First, let's define a simple Python class with functions that we'll write unit tests for.

# Create a simple class with methods to test

class Calculator:

def add(self, a, b):

return a + b

def subtract(self, a, b):

return a - b

def multiply(self, a, b):

return a \* b

def divide(self, a, b):

if b == 0:

raise ValueError("Cannot divide by zero")

return a / b

**Step 2: Writing Unit Tests**

Now, let’s write a series of unit tests for the Calculator class using Python's built-in unittest framework.

python

Copy code

import unittest

# Create a test case for the Calculator class

class TestCalculator(unittest.TestCase):

def setUp(self):

# This method is run before each test

self.calc = Calculator()

def test\_add(self):

self.assertEqual(self.calc.add(1, 2), 3)

self.assertEqual(self.calc.add(-1, 1), 0)

self.assertEqual(self.calc.add(-1, -1), -2)

def test\_subtract(self):

self.assertEqual(self.calc.subtract(10, 5), 5)

self.assertEqual(self.calc.subtract(-1, 1), -2)

def test\_multiply(self):

self.assertEqual(self.calc.multiply(3, 3), 9)

self.assertEqual(self.calc.multiply(0, 1), 0)

def test\_divide(self):

self.assertEqual(self.calc.divide(10, 2), 5)

self.assertEqual(self.calc.divide(9, 3), 3)

# Run the tests

unittest.main(argv=[''], verbosity=2, exit=False)

**Explanation:**

1. **TestCalculator Class**: This class inherits from unittest.TestCase and contains multiple test methods.
   * setUp(): This method is executed before every individual test case, initializing the Calculator object.
   * test\_add(), test\_subtract(), test\_multiply(), test\_divide(): Each of these methods tests a specific method in the Calculator class.
   * assertEqual(): Verifies that the result of a function call is equal to the expected result.
   * assertRaises(): Tests that a specific exception (in this case, ValueError) is raised under certain conditions.
2. **Running Tests**: We use unittest.main() to run the test cases, passing argv=[''] to prevent Jupyter from taking command-line arguments, and exit=False to prevent it from shutting down the kernel after running the tests.