**Lab Exercise 3 – Unit Testing in Python**

Unit testing is a software testing technique used to verify that individual units or components of a software application work as expected. In Python, you can perform unit testing using the built-in unittest framework or other third-party testing libraries like pytest. Here's a basic overview of how to perform unit testing in Python using unittest:

**Calculator Module (calculator.py):**

Create a Python module named calculator.py with the following simple calculator functions:

def add(a, b):

return a + b

def subtract(a, b):

return a - b

def multiply(a, b):

return a \* b

def divide(a, b):

if b == 0:

raise ValueError("Cannot divide by zero")

return a / b

**Unit Testing Module (test\_calculator.py):**

Now, let's create a unit testing module to test the functions in the calculator.py module. Create a file named test\_calculator.py:

import unittest

from calculator import add, subtract, multiply, divide

class TestCalculator(unittest.TestCase):

def test\_add(self):

self.assertEqual(add(2, 3), 5)

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

self.assertEqual(add(0, 0), 0)

def test\_subtract(self):

self.assertEqual(subtract(5, 3), 2)

self.assertEqual(subtract(2, 5), -3)

self.assertEqual(subtract(0, 0), 0)

def test\_multiply(self):

self.assertEqual(multiply(2, 3), 6)

self.assertEqual(multiply(-2, 3), -6)

self.assertEqual(multiply(0, 5), 0)

def test\_divide(self):

self.assertEqual(divide(6, 3), 2)

self.assertEqual(divide(5, 2), 2.5)

self.assertEqual(divide(1, 0), ValueError)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

In this unit testing module, we have test cases for the add, subtract, multiply, and divide functions, checking their expected outcomes for various input values. The last test case checks if attempting to divide by zero raises a ValueError.

**Running the Tests:**

To run the tests, use the following command:

python -m unittest test\_calculator

You should see the test results displayed in the console. If everything is implemented correctly, you'll get an output indicating that all tests passed. If a test fails, you'll see details about which test failed and why.

This exercise is a basic introduction to unit testing in Python. You can expand on it by adding more test cases and testing edge cases to thoroughly validate the calculator functions. Unit testing is an essential practice for ensuring the correctness and reliability of your code.