МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра ІСМ

******

Звіт

до лабораторної роботи №6

На тему “Розробка та Unit тестування Python додатку”

З дисципліни “Спеціалізовані мови програмування”

*Виконав:*

*ст. гр. ІТ-31*

*Іштван Комоній*

*Прийняв:*

*Щербак С. С.*

*Львів - 2023*

**Мета роботи:** Створення юніт-тестів для додатка-калькулятора на основі класів.

**Хід роботи**

**Програмний код:**

/calculator/tests/tests

import unittest

from calulator.operations.operations import Addition, Division, Multiplication, Subtraction

from calulator.ui.error\_handler.error\_handler import handle\_error

class CalculatorTest(unittest.TestCase):

# addition

def test\_addition\_positive\_numbers(self):

addition\_operation = Addition(5, 3)

self.assertEqual(addition\_operation.execute(), 8)

def test\_addition\_negative\_numbers(self):

addition\_operation = Addition(-2, -4)

self.assertEqual(addition\_operation.execute(), -6)

def test\_addition\_mixed\_numbers(self):

addition\_operation = Addition(4, -1)

self.assertEqual(addition\_operation.execute(), 3)

# substraction

def test\_subtraction\_positive\_numbers(self):

subtraction\_operation = Subtraction(5, 3)

self.assertEqual(subtraction\_operation.execute(), 2)

def test\_subtraction\_negative\_numbers(self):

subtraction\_operation = Subtraction(-2, -4)

self.assertEqual(subtraction\_operation.execute(), 2)

def test\_subtraction\_mixed\_numbers(self):

subtraction\_operation = Subtraction(4, -1)

self.assertEqual(subtraction\_operation.execute(), 5)

def test\_subtraction\_zero(self):

subtraction\_operation = Subtraction(5, 5)

self.assertEqual(subtraction\_operation.execute(), 0)

def test\_subtraction\_negative\_result(self):

subtraction\_operation = Subtraction(10, 15)

self.assertEqual(subtraction\_operation.execute(), -5)

# multiplication

def test\_multiplication\_positive\_numbers(self):

multiplication\_operation = Multiplication(5, 3)

self.assertEqual(multiplication\_operation.execute(), 15)

def test\_multiplication\_negative\_numbers(self):

multiplication\_operation = Multiplication(-2, -4)

self.assertEqual(multiplication\_operation.execute(), 8)

def test\_multiplication\_mixed\_numbers(self):

multiplication\_operation = Multiplication(4, -1)

self.assertEqual(multiplication\_operation.execute(), -4)

def test\_multiplication\_zero(self):

multiplication\_operation = Multiplication(5, 0)

self.assertEqual(multiplication\_operation.execute(), 0)

def test\_multiplication\_by\_zero(self):

multiplication\_operation = Multiplication(0, 5)

self.assertEqual(multiplication\_operation.execute(), 0)

# division

def test\_division\_positive\_decimal\_numbers(self):

division\_operation = Division(10.5, 2.5)

self.assertEqual(division\_operation.execute(), 4.2)

def test\_division\_negative\_decimal\_numbers(self):

division\_operation = Division(-10.5, -2.5)

self.assertEqual(division\_operation.execute(), 4.2)

def test\_division\_mixed\_decimal\_numbers(self):

division\_operation = Division(4.5, -2)

self.assertEqual(division\_operation.execute(), -2.25)

def test\_division\_by\_zero(self):

with self.assertRaises(ValueError):

Division(10.5, 0).execute()

# error handling

def test\_division\_by\_zero\_error(self):

with self.assertRaises(ValueError):

handle\_error(Division(10, 0).execute())

def test\_invalid\_operation\_error(self):

with self.assertRaises(TypeError):

handle\_error(Division('abc', 0).execute())

def test\_invalid\_number\_error(self):

with self.assertRaises(TypeError):

handle\_error(Division(10, 'abc').execute())

def test\_calculator(self):

self.test\_addition\_positive\_numbers()

self.test\_addition\_negative\_numbers()

self.test\_addition\_mixed\_numbers()

self.test\_subtraction\_positive\_numbers()

self.test\_subtraction\_negative\_numbers()

self.test\_subtraction\_mixed\_numbers()

self.test\_subtraction\_zero()

self.test\_subtraction\_negative\_result()

self.test\_multiplication\_positive\_numbers()

self.test\_multiplication\_negative\_numbers()

self.test\_multiplication\_mixed\_numbers()

self.test\_multiplication\_zero()

self.test\_multiplication\_by\_zero()

self.test\_division\_positive\_decimal\_numbers()

self.test\_division\_negative\_decimal\_numbers()

self.test\_division\_mixed\_decimal\_numbers()

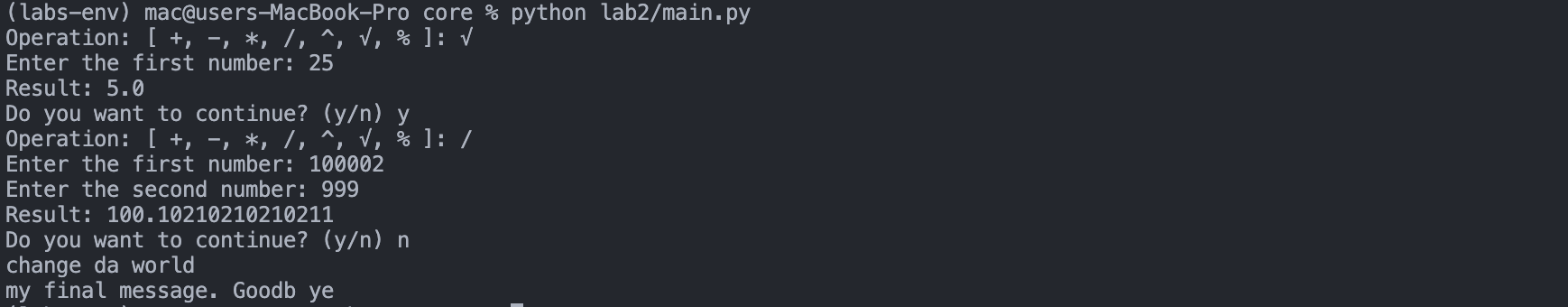
self.test\_division\_by\_zero()

self.test\_division\_by\_zero\_error()

self.test\_invalid\_operation\_error()

self.test\_invalid\_number\_error()

Результат виконання програми:



**Висновок:** Під час виконання даної лабораторної роботи було створено юніт-тести для додатка-калькулятора на основі класів.