МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

**Лабораторна робота №6**

**з дисципліни**

**СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ**

**на тему**

**«Розробка та Unit тестування Python додатку»**

Виконав:

ст. гр. ІТ-21сп

Одноріг Д.І.

Прийняв:

Щербак С.С.

Львів-2023

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

**Хід роботи**

**Завдання 1:** Тестування Додавання

Напишіть юніт-тест, щоб перевірити, що операція додавання в вашому додатку-калькуляторі працює правильно. Надайте тестові випадки як для позитивних, так і для негативних чисел.

**Завдання 2:** Тестування Віднімання

Створіть юніт-тести для переконання, що операція віднімання працює правильно. Тестуйте різні сценарії, включаючи випадки з від'ємними результатами.

**Завдання 3:** Тестування Множення

Напишіть юніт-тести, щоб перевірити правильність операції множення в вашому калькуляторі. Включіть випадки з нулем, позитивними та від'ємними числами.

**Завдання 4:** Тестування Ділення

Розробіть юніт-тести для підтвердження точності операції ділення. Тести повинні охоплювати ситуації, пов'язані з діленням на нуль та різними числовими значеннями.

**Завдання 5:** Тестування Обробки Помилок

Створіть юніт-тести, щоб перевірити, як ваш додаток-калькулятор обробляє помилки. Включіть тести для ділення на нуль та інших потенційних сценаріїв помилок. Переконайтеся, що додаток відображає відповідні повідомлення про помилки.

Код програми:

import unittest

from Lab2.calc import Calculator

class TestCalculator(unittest.TestCase):

def test\_add(self):

calc = Calculator()

result = calc.add(150, 50)

self.assertEqual(result, 200)

def test\_subtr(self):

calc = Calculator()

result = calc.subtract(17, 5)

self.assertEqual(result, 12)

#self.assertEqual(result, 11) #Провокуємо помилку в тестуванні

def test\_multiply(self):

calc = Calculator()

result = calc.multiply(10, 10)

self.assertEqual(result, 100)

def test\_divide(self):

calc = Calculator()

result = calc.divide(333, 3)

self.assertEqual(result, 111)

def test\_divide\_zero(self):

calc = Calculator()

with self.assertRaises(ValueError):

calc.divide(100, 0)

def test\_add\_negative(self):

calc = Calculator()

result = calc.add(-150, 50)

self.assertEqual(result, -100)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

На рис. 1 зображено демонстрацію успішного тестування.

*Зображення, що містить текст, знімок екрана, Мультимедійне програмне забезпечення, програмне забезпечення

Автоматично згенерований опис*

*Рис. 1. Результат успішного тестування.*

На рис. 2 зображено демонстрацію провального тестування через невірно заданий параметр результату result.

*Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис*

*Рис. 2. Результат провального тестування.*

**Висновок:** виконуючи дану лабораторну роботу, я створив набір юніт-тестів, які перевіряють правильність основних арифметичних операцій у моєму додатку-калькуляторі.