Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра ІСМ



**Звіт**

до лабораторної роботи № 6

з дисципліни

​*Спеціалізовані мови програмування*

на тему:

“ **Розробка та Unit тестування Python додатку**”

Виконав студент РІ-31

**Гвоздь Северин**

Прийняв: Щербак С.С.

Львів – 2024

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

**Завдання лабораторної 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.

Файл запуску:

import unittest, os, sys

sys.path.append(os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../classes')))

from calc import BasicCalculator, ScientificCalculator

class TestBasicCalculator(unittest.TestCase):

    def setUp(self):

        self.calc = BasicCalculator()

    def test\_addition(self):

        self.assertEqual(self.calc.calculate(10, 5, '+'), 15)

        self.assertEqual(self.calc.calculate(-3, -7, '+'), -10)

    def test\_subtraction(self):

        self.assertEqual(self.calc.calculate(10, 5, '-'), 5)

        self.assertEqual(self.calc.calculate(-3, -7, '-'), 4)

    def test\_multiplication(self):

        self.assertEqual(self.calc.calculate(10, 5, '\*'), 50)

        self.assertEqual(self.calc.calculate(0, 5, '\*'), 0)

        self.assertEqual(self.calc.calculate(-3, 5, '\*'), -15)

    def test\_division(self):

        self.assertEqual(self.calc.calculate(10, 5, '/'), 2)

        self.assertEqual(self.calc.calculate(-10, 5, '/'), -2)

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

        with self.assertRaises(ZeroDivisionError):

            self.calc.calculate(10, 0, '/')

class TestScientificCalculator(unittest.TestCase):

    def setUp(self):

        self.calc = ScientificCalculator()

    def test\_power(self):

        self.assertEqual(self.calc.calculate(2, 3, '^'), 8)

    def test\_square\_root(self):

        self.assertEqual(self.calc.calculate(9, None, '√'), 3)

    def test\_remainder(self):

        self.assertEqual(self.calc.calculate(10, 3, '%'), 1)

    def test\_invalid\_operator(self):

        with self.assertRaises(ValueError):

            self.calc.calculate(5, 3, '#')

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

    unittest.main()

**Висновок**

Виконавши ці завдання, у мене є набір юніт-тестів, які перевіряють правильність основних арифметичних операцій у мому додатку-калькуляторі. Ці тести допомогають виявити та виправити будь-які проблеми або помилки, які можуть виникнути під час розробки чи обслуговування мого додатку, забезпечуючи його надійність і точність