МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Лабораторна робота №6

з дисципліни

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

на тему

РОЗРОБКА ТА UNIT ТЕСТУВАННЯ PYTHON ДОДАТКУ

Виконала:

ст. гр. ІТ-21сп

Кобленц Д.А.

Прийняв:

Щербак С.С.

Львів-2023

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

**План 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 BLL.Lab2.Calculator import Calculator

class UnitTestsCalculator(unittest.TestCase):

def setUp(self):

# This method is called before each test. It sets up the Calculator object for testing.

self.calculator = Calculator()

def test\_add(self):

# This test checks if the addition operation works correctly.

first\_input = float(5)

second\_input = float(3)

operator = '+'

self.assertEqual(self.calculator.calculate(first\_input, operator, second\_input), float(8))

print("Added successful")

def test\_subtract(self):

# This test checks if the subtraction operation works correctly.

first\_input = float(5)

second\_input = float(3)

operator = '-'

self.assertEqual(self.calculator.calculate(first\_input, operator, second\_input), float(2))

print("Subtracted successful")

def test\_multiply(self):

# This test checks if the multiplication operation works correctly.

first\_input = float(5)

second\_input = float(3)

operator = '\*'

self.assertEqual(self.calculator.calculate(first\_input, operator, second\_input), float(15))

print("Multiplied successful")

def test\_divide(self):

# This test checks if the division operation works correctly.

first\_input = float(6)

second\_input = float(3)

operator = '/'

self.assertEqual(self.calculator.calculate(first\_input, operator, second\_input), float(2))

print("Divided successful")

def test\_square\_root(self):

# This test checks if the square root operation works correctly.

first\_input = float(9)

second\_input = float(0)

operator = '√'

self.assertEqual(self.calculator.calculate(first\_input, operator, second\_input), float(3))

print("Square root get successful")

def test\_power(self):

# This test checks if the power operation works correctly.

first\_input = float(-2)

second\_input = float(2)

operator = '^'

self.assertEqual(self.calculator.calculate(first\_input, operator, second\_input), float(4))

print("Power successful")

def test\_leftover(self):

# This test checks if the modulo operation works correctly.

first\_input = float(10)

second\_input = float(3)

operator = '%'

self.assertEqual(self.calculator.calculate(first\_input, operator, second\_input), float(1))

print("Leftover successful")

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

# This test checks if the Calculator correctly handles division by zero.

first\_input = float(5)

second\_input = float(0)

operator = '/'

self.assertIsNone(self.calculator.calculate(first\_input, operator, second\_input))

def test\_square\_root\_of\_negative(self):

# This test checks if the Calculator correctly handles square root of a negative number.

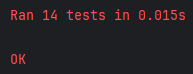
first\_input = float(-9)

second\_input = float(0)

operator = '√'

self.assertIsNone(self.calculator.calculate(first\_input, operator, second\_input))

На рисунку 1 зображено результат програми.



**Рис. 1** Результат виконання коду

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