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

Борисов П.Р.

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**

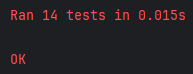
Розробіть юніт-тести для підтвердження точності операції ділення. Тести повинні охоплювати ситуації, пов'язані з діленням на нуль та різними числовими значеннями.

**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  
import math  
from BLL.lab2.CalculatorCore import CalculatorCore  
from unittest.mock import patch  
  
  
class TestCalculatorCore(unittest.TestCase):  
  
 def setUp(self):  
 self.calculator = CalculatorCore()  
  
 def test\_add(self):  
 result = self.calculator.calculate\_number(5, 3, '+')  
 self.assertEqual(result, 8)  
  
 def test\_subtract(self):  
 result = self.calculator.calculate\_number(5, 3, '-')  
 self.assertEqual(result, 2)  
  
 def test\_multiply(self):  
 result = self.calculator.calculate\_number(5, 3, '\*')  
 self.assertEqual(result, 15)  
  
 def test\_divide(self):  
 result = self.calculator.calculate\_number(6, 3, '/')  
 self.assertEqual(result, 2)  
  
 def test\_divide\_by\_zero(self):  
 with self.assertRaises(ValueError):  
 self.calculator.calculate\_number(5, 0, '/')  
  
 def test\_square\_root(self):  
 result = self.calculator.calculate\_number(9, 0, '√')  
 self.assertEqual(result, 3)  
  
 def test\_power(self):  
 result = self.calculator.calculate\_number(2, 3, '^')  
 self.assertEqual(result, 8)  
  
 def test\_modulo(self):  
 result = self.calculator.calculate\_number(10, 3, '%')  
 self.assertEqual(result, 1)  
  
 def test\_memory\_store\_and\_retrieve(self):  
 self.calculator.store\_in\_memory(10)  
 self.assertEqual(self.calculator.retrieve\_from\_memory(), 10)  
  
 def test\_add\_to\_history(self):  
 self.calculator.add\_to\_history('5 + 3', 8)  
 self.assertIn('5 + 3 - 8', self.calculator.history)  
  
 def test\_view\_history\_empty(self):  
 self.assertEqual(self.calculator.view\_history(), 'Empty history')  
  
 def test\_view\_history\_non\_empty(self):  
 self.calculator.add\_to\_history('5 + 3', 8)  
 self.assertEqual(self.calculator.view\_history(), '5 + 3 - 8')  
  
 def test\_change\_decimal\_places(self):  
 with unittest.mock.patch('builtins.input', return\_value='4'):  
 self.calculator.change\_decimal\_places()  
 self.assertEqual(self.calculator.settings['decimal\_places'], 4)  
  
 def test\_toggle\_memory\_function(self):  
 current\_status = self.calculator.settings['use\_memory']  
 self.calculator.toggle\_memory\_function()  
 self.assertNotEqual(self.calculator.settings['use\_memory'], current\_status)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

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.