

Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Звіт про виконання лабораторної роботи №6
з дисципліни «Спеціалізовані мови програмування»
на тему «Розробка та Unit тестування Python додатку»

Виконав:
студент групи РІ-32
Гусак Віктор
Прийняв:
Щербак С.С.

Мета роботи: Створення 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 Помилки

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.

Хід роботи

calculator_test.py:

```
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(
__file__))))

import unittest
from functions.calculator_functions import perform_calculation

class TestCalculatorFunctions(unittest.TestCase):

    def testAdd(self):
        self.assertEqual(perform_calculation(1, '+', 2), 3)

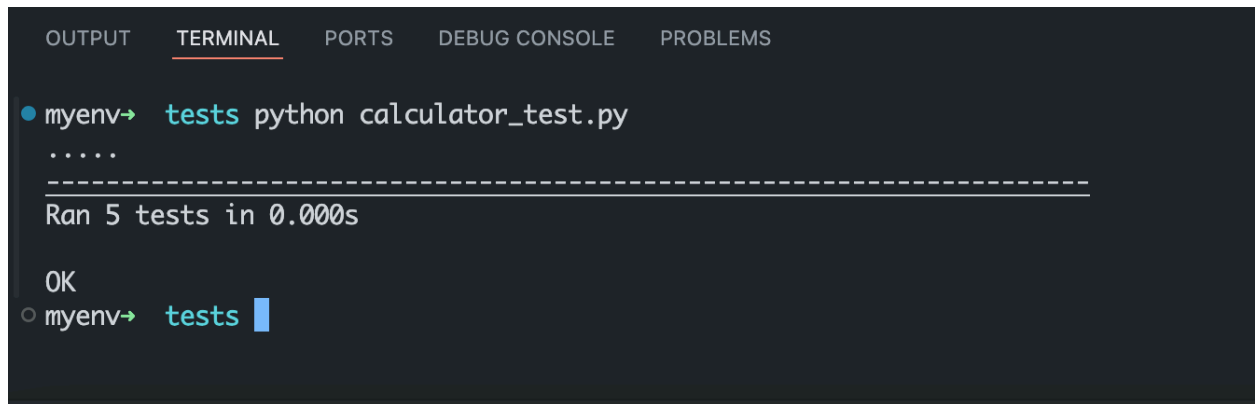
    def testSubtract(self):
        self.assertEqual(perform_calculation(5, '-', 3), 2)

    def testMultiplication(self):
        self.assertEqual(perform_calculation(4, '*', 2), 8)

    def testDivision(self):
        self.assertEqual(perform_calculation(10, '/', 2), 5)

    def testExceptions(self):
        self.assertEqual(perform_calculation(10, '/', 0),
        "Помилка! Ділення на нуль.")

    def run():
        unittest.main()
```



The image shows a terminal window with a dark background. At the top, there are tabs: OUTPUT, TERMINAL (which is selected and underlined), PORTS, DEBUG CONSOLE, and PROBLEMS. Below the tabs, the terminal shows a command prompt 'myenv→' followed by 'tests python calculator_test.py'. The output consists of five dots '.....' followed by a dashed line and the text 'Ran 5 tests in 0.000s'. Below this, there is an 'OK' status and another command prompt 'myenv→' followed by 'tests' and a blue cursor block.

```
OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE  PROBLEMS
● myenv→ tests python calculator_test.py
.....
-----
Ran 5 tests in 0.000s

OK
○ myenv→ tests █
```

Рис. 1. Приклад роботи програми

Висновки: Виконавши ці завдання, було отримано набір юніт-тестів, які перевіряють правильність основних арифметичних операцій у вашому додатку-калькуляторі.