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

Виконав:
ст. гр. ІТ-32
Шоха А.А.
Прийняв:
Щербак С.С.

Львів-2023

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

Завдання на лабораторну роботу

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.

Хід роботи

Код програми:

import unittest

```
import sys

from constants import home_path
sys.path.append(home_path + 'source/lab2')

from calculator import Calculator

from constants import home_path
sys.path.append(home_path + 'source/lab1')

from calculator_funcs import *


class TestCalculator(unittest.TestCase):
    def setUp(self):
        self.calculator = Calculator()
```

```
        self.calculator.num1 = 5
        self.calculator.num2 = 3

    def test_addition(self):

self.assertEqual(self.calculator.operations['+'](self.calculat
or.num1, self.calculator.num2), 8)

self.assertEqual(self.calculator.operations['+'](self.calculat
or.num1, -self.calculator.num2), 2)

    def test_subtraction(self):

self.assertEqual(self.calculator.operations['-'](self.calculat
or.num1, self.calculator.num2), 2)

self.assertEqual(self.calculator.operations['-'](-self.calcula
tor.num1, -self.calculator.num2), -2)

    def test_multiplication(self):

self.assertEqual(self.calculator.operations['*'](self.calculat
or.num1, self.calculator.num2), 15)

self.assertEqual(self.calculator.operations['*'](self.calculat
or.num1, -self.calculator.num2), -15)

self.assertEqual(self.calculator.operations['*'](self.calculat
or.num1, 0), 0)

    def test_division(self):
        with self.assertRaises(ZeroDivisionError):

self.calculator.operations['/'](self.calculator.num1, 0)

self.assertAlmostEqual(self.calculator.operations['/'](self.ca
lculator.num1, self.calculator.num2), 1.6666666666666667)
          self.assertEqual(self.calculator.operations['/'](10,
-5), -2)

    def test_invalid_operator(self):
          self.calculator.operations = {'$': add}   # Invalid
operator
        with self.assertRaises(KeyError):

self.calculator.operations['+'](self.calculator.num1,
self.calculator.num2)

if __name__ == '__main__':
    unittest.main()
```

Висновок: під час виконання лабораторної роботи було створено юніт-тести для додатка-калькулятора на основі класів.