МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра інформаційних систем та мереж

Лабораторна робота №6
з дисципліни: «Спеціалізовані мови програмування»
на тему: «Розробка та Unit тестування Python додатку»

Виконала:
студентка групи ІТ-32
Моляща Ю.А.
Прийняв:
Щербак С.С.

Львів - 2023

Лабораторна робота №6

«Розробка та Unit тестування Python додатку»

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:

main.py

```
# Include the parent directory in the system's import path
import sys
import os

current_dir = os.path.dirname(os.path.abspath(__file__))
parent_dir = os.path.abspath(os.path.join(current_dir, '..'))
sys.path.append(parent_dir)

import unittest
```

```
from lab1 import calculations


class AdditionTestCase(unittest.TestCase):
    """
    A test case class for testing addition, subtraction, multiplication
    and division functions in the calculations module.
    """
    def setUp(self):
        self.calculator = calculations
        self.first_number = 80
        self.second_number = 30


    def test_addition(self):
                    self.assertEqual(self.calculator.addition(self.first_number,
self.second_number), 110)
                   self.assertEqual(self.calculator.addition(-self.first_number,
-self.second_number), -110)


        with self.assertRaises(TypeError):
                                self.calculator.addition(f"{self.first_number}",
self.second_number)
            self.calculator.addition(f"{self.first_number}", "kachow")
            self.calculator.addition(self.first_number, "kachow")


    def test_subtraction(self):
                self.assertEqual(self.calculator.subtraction(self.first_number,
self.second_number), 50)
                self.assertEqual(self.calculator.subtraction(self.first_number,
-self.second_number), 110)


        with self.assertRaises(TypeError):
                            self.calculator.subtraction(f"{self.first_number}",
self.second_number)
            self.calculator.subtraction(f"{self.first_number}", "kachow")
            self.calculator.subtraction(self.first_number, "kachow")
```

```
    def test_multiplication(self):
                self.assertEqual(self.calculator.multiplication(self.first_number,
self.second_number), 2400)
                self.assertEqual(self.calculator.multiplication(self.first_number,
-self.second_number), -2400)
            self.assertEqual(self.calculator.multiplication(0, self.second_number),
0)

        with self.assertRaises(TypeError):
                            self.calculator.multiplication(f"{self.first_number}",
self.second_number)
            self.calculator.multiplication(f"{self.first_number}", "kachow")
            self.calculator.multiplication(self.first_number, "kachow")

    def test_division(self):
                    self.assertEqual(self.calculator.division(self.first_number,
self.second_number), 2.6666666666666665)
                    self.assertEqual(self.calculator.division(self.first_number,
-self.second_number), -2.6666666666666665)
        self.assertEqual(self.calculator.division(0, self.second_number), 0)

        with self.assertRaises(TypeError):
                                self.calculator.division(f"{self.first_number}",
self.second_number)
            self.calculator.division(f"{self.first_number}", "kachow")
            self.calculator.division(self.first_number, "kachow")
        with self.assertRaises(ZeroDivisionError):
            self.calculator.division(self.first_number, 0)

def main():
    unittest.main()

if __name__ == "__main__":
    main()
```

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