МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ « ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Звіт

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

Виконала:
Студентка гр. РІ-21
Зузяк Л. Р.
Прийняв:
Щербак С.С.

Львів 2024

Мета: 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 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.

Виконавши ці завдання, у вас буде набір юніт-тестів, які перевіряють правильність основних арифметичних операцій у вашому додатку-калькуляторі. Ці тести допоможуть виявити та виправити будь-які проблеми або помилки, які можуть виникнути під час розробки чи обслуговування вашого додатку, забезпечуючи його надійність і точність

Виконання:

```
import unittest
from calculator.main import calculate
```

```python
from calculator.memory import Memory
from calculator.history import History

class TestMain(unittest.TestCase):
    def setUp(self):
        self.memory = Memory()
        self.history = History()

    def test_basic_calculations(self):
        self.assertEqual(calculate(4, '+', 3), 7)
        self.assertEqual(calculate(10, '-', 3), 7)
        self.assertEqual(calculate(3, '*', 4), 12)
        self.assertEqual(calculate(8, '/', 2), 4)

    def test_invalid_operation(self):
        result = calculate(10, '&', 5)
        self.assertIsNone(result)

    def test_divide_by_zero(self):
        result = calculate(10, '/', 0)
        self.assertIsNone(result)

    def test_sqrt_of_negative(self):
        result = calculate(-1, '√', None)
        self.assertIsNone(result)

    def test_exponentiate(self):
        self.assertEqual(calculate(2, '^', 3), 8)

    def test_modulo(self):
        self.assertEqual(calculate(10, '%', 3), 1)

    def test_memory_integration(self):
        self.memory.store(15)
        self.assertEqual(self.memory.retrieve(), 15)

    def test_history_integration(self):
        self.history.add_entry(5, '*', 5, 25)
        self.assertEqual(len(self.history.entries), 1)
        self.assertEqual(self.history.entries[0], "5 * 5 = 25")

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

Test_main.py

Результат:

```
admin@MacBookAir lab_1_p % python3 -m unittest discover -s tests -p "test_*.py"

.History cleared.
....Error: Division by zero.
...Invalid operator
.Value 15 stored in memory.
Memory: 15
..Error: Cannot calculate square root of a negative number.
.Value 200 stored in memory.
Memory cleared.
No value stored in memory.
.No value stored in memory.
.Value 100 stored in memory.
Memory: 100
.Value 42 stored in memory.
...Error: Division by zero.
....Error: Cannot calculate square root of a negative number.
...
----------------------------------------------------------------------
Ran 25 tests in 0.000s

OK
```

Покликання на виконану роботу в GitHub:
https://github.com/Lilia427/Lab_1_python-calculator/tree/main/tests

Висновок: в ході виконання лабораторної роботи я навчилася як створювати юніт-тести для додатка-калькулятора на основі класів