

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:  
студент групи РІ-21сп  
Владислав РИБАК

Львів – 2024

**Мета виконання лабораторної 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.

### Текст програмної реалізації:

#### **test\_calculator.py:**

```
import unittest

from bll.operations import Calculator

class TestCalculator(unittest.TestCase):

    def setUp(self):

        self.calc = Calculator()

    def test_addition(self):

        self.assertEqual(self.calc.perform_calculation(2, 3, '+'), 5)

    def test_subtraction(self):

        self.assertEqual(self.calc.perform_calculation(5, 3, '-'), 2)

    def test_multiplication(self):

        self.assertEqual(self.calc.perform_calculation(2, 3, '*'), 6)

    def test_division(self):
```

```

self.assertEqual(self.calc.perform_calculation(6, 3, '/'), 2)

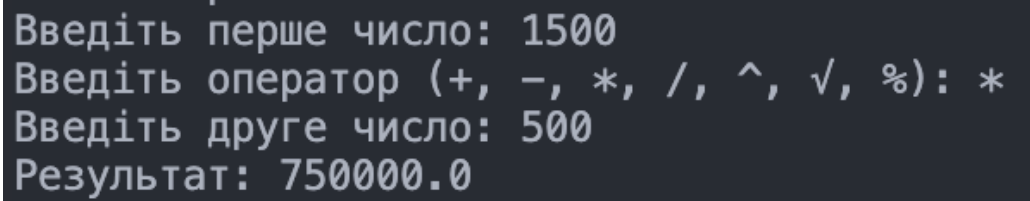
def test_division_by_zero(self):
    with self.assertRaises(ValueError):
        self.calc.perform_calculation(6, 0, '/')

def test_square_root(self):
    self.assertEqual(self.calc.perform_calculation(9, None, '√'), 3)

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

```

### Результати тестування:

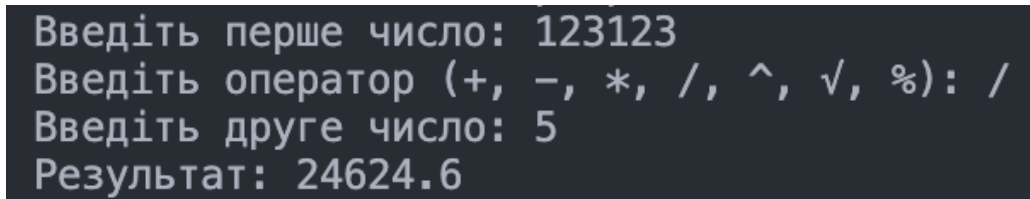


```

Введіть перше число: 1500
Введіть оператор (+, -, *, /, ^, √, %): *
Введіть друге число: 500
Результат: 750000.0

```

Рис. 1. Результати множення



```

Введіть перше число: 123123
Введіть оператор (+, -, *, /, ^, √, %): /
Введіть друге число: 5
Результат: 24624.6

```

Рис. 2. Результати ділення

```

> python3 test_calculator.py
.....
-----
Ran 6 tests in 0.000s

OK

```

Рис. 3. Результат запуску тестів

```

> python3 run_tests.py
.....
-----
Ran 6 tests in 0.000s

OK
Name                               Stmt  Miss  Cover
-----
bll/operations.py                   28     7   75%
test_calculator.py                 20     1   95%
-----
TOTAL                               48     8   83%

```

Рис. 4. Покриття тестами застосунок

**Висновки:** на цій лабораторній роботі я імплементував юніт-тести для додатка-калькулятора на основі класів.