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»

Виконала студентка

групи РІ-32

Титиш О.І.

Прийняв:

Щербак С.С

Львів – 2024

**Мета:** Cтворення 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.

**Код програми**

class\_calc.py

from Shared.AppSettings\_LAB1 import decimal\_places  
from Shared.logs.logger\_LAB1 import log\_operation, log\_history, show\_history  
from functions\_LAB1 import calculate  
  
class Calculator:  
 def \_\_init\_\_(self):  
 self.memory = None  
 self.decimal\_places = decimal\_places  
  
 def get\_input(self):  
 try:  
 num1 = float(input("Введіть перше число: "))  
 operator = input("Введіть оператор (+, -, \*, /, ^, %, √): ")  
 num2 = None  
 if operator != '√':  
 num2 = float(input("Введіть друге число: "))  
 return num1, operator, num2  
 except ValueError:  
 print("Неправильний ввід. Спробуйте знову.")  
 return self.get\_input()  
  
 def is\_valid\_operator(self, operator):  
 return operator in ['+', '-', '\*', '/', '^', '%', '√']  
  
 def perform\_calculation(self, num1, operator, num2):  
 try:  
 result = calculate(num1, operator, num2)  
 return round(result, self.decimal\_places)  
 except ZeroDivisionError:  
 print("Помилка: Ділення на нуль неможливе.")  
 except ValueError as e:  
 print(f"Помилка: {e}")  
 except Exception as e:  
 print(f"Виникла непередбачена помилка: {e}")  
 return None  
  
 def ask\_to\_continue(self):  
 return input("Бажаєте виконати ще одне обчислення? (так/ні): ").lower() == 'так'  
  
 def store\_in\_memory(self, result):  
 self.memory = result  
 print(f"Результат {result} збережений у пам'яті.")  
  
 def recall\_memory(self):  
 if self.memory is not None:  
 print(f"Збережене значення: {self.memory}")  
 return self.memory  
 else:  
 print("Пам'ять порожня.")  
 return None  
  
 def run(self):  
 print(f"Результати відображатимуться з {self.decimal\_places} десятковими знаками.")  
 while True:  
 num1, operator, num2 = self.get\_input()  
  
 if not self.is\_valid\_operator(operator):  
 print("Недійсний оператор. Спробуйте ще раз. Ви можете використовувати тільки +, -, \*, /, ^, %, √")  
 continue  
  
 result = self.perform\_calculation(num1, operator, num2)  
 if result is not None:  
 print(f"Результат: {result}")  
 self.store\_in\_memory(result)  
  
 expression = f"{num1} {operator} {num2 if operator != '√' else ''}"  
 log\_operation(f"{expression} = {result}")  
 log\_history(expression, result)  
  
 if input("Бажаєте переглянути історію розрахунків? (так/ні): ").lower() == 'так':  
 show\_history()  
  
 if not self.ask\_to\_continue():  
 break

test\_calculator.py

import unittest  
from classes.class\_calc\_LAB2 import Calculator  
  
class TestCalculator(unittest.TestCase):  
  
 def setUp(self):  
 self.calculator = Calculator()  
  
 *# Завдання 1: Тестування Додавання* def test\_addition\_positive\_numbers(self):  
 result = self.calculator.perform\_calculation(3, '+', 5)  
 self.assertEqual(result, 8)  
  
 def test\_addition\_negative\_numbers(self):  
 result = self.calculator.perform\_calculation(-3, '+', -5)  
 self.assertEqual(result, -8)  
  
 *# Завдання 2: Тестування Віднімання* def test\_subtraction\_positive\_numbers(self):  
 result = self.calculator.perform\_calculation(10, '-', 5)  
 self.assertEqual(result, 5)  
  
 def test\_subtraction\_negative\_result(self):  
 result = self.calculator.perform\_calculation(5, '-', 10)  
 self.assertEqual(result, -5)  
  
 *# Завдання 3: Тестування Множення* def test\_multiplication\_positive\_numbers(self):  
 result = self.calculator.perform\_calculation(4, '\*', 5)  
 self.assertEqual(result, 20)  
  
 def test\_multiplication\_with\_zero(self):  
 result = self.calculator.perform\_calculation(0, '\*', 10)  
 self.assertEqual(result, 0)  
  
 def test\_multiplication\_negative\_numbers(self):  
 result = self.calculator.perform\_calculation(-4, '\*', 5)  
 self.assertEqual(result, -20)  
  
 *# Завдання 4: Тестування Ділення* def test\_division\_positive\_numbers(self):  
 result = self.calculator.perform\_calculation(10, '/', 2)  
 self.assertEqual(result, 5)  
  
 def test\_division\_negative\_numbers(self):  
 result = self.calculator.perform\_calculation(-9, '/', 3)  
 self.assertEqual(result, -3)  
  
 def test\_division\_by\_zero(self):  
 result = self.calculator.perform\_calculation(10, '/', 0)  
 self.assertIsNone(result) *# Перевіряємо, що повертається None у випадку ділення на нуль  
  
 # Завдання 5: Тестування Обробки Помилок* def test\_invalid\_operator(self):  
 result = self.calculator.perform\_calculation(10, '#', 2)  
 self.assertIsNone(result) *# Перевірка на некоректний оператор*if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

**Результат виконання програми**

**Зображення, що містить текст, знімок екрана, Шрифт, програмне забезпечення

Автоматично згенерований опис**

*Рисунок 1. Результат виконання*

Посилання на GitHub:

**Висновок:** При виконанні цієї лабораторної робити, я створила набір юніт-тестів, які перевіряють правильність основних арифметичних операцій у моєму додатку-калькуляторі. Ці тести допомогли виявити та виправити будь-які проблеми або помилки, які могли виникнути під час розробки чи обслуговування мого додатку, забезпечуючи його надійність і точність.