мІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

нАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «лЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНОРМАЦІЙНИХ ТЕХНОЛОГІЙ

|  |
| --- |
|  |



ЗВІТ

Про виконання лабораторної роботи № 6

З дисципліни

«Спеціалізовані мови програмування»

Виконала:

студентка гр. ІТ-31

Катерина ПОПОВА

Прийняв:

Сергій ЩЕРБАК

Львів – 2023

**Тема роботи :** Розробка та Unit тестування Python додатку

**Мета роботи:** 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**

**main.py**

import unittest

from calculator import Calculator

class TestCalculator(unittest.TestCase):

def setUp(self):

self.calculator = Calculator()

# Тест додавання

def test\_addition\_positive(self):

self.assertEqual(self.calculator.perform\_calculation(10, 5, '+'), 15)

def test\_addition\_negative(self):

self.assertEqual(self.calculator.perform\_calculation(-10, -5, '+'), -15)

# Тест віднімання

def test\_subtraction\_positive(self):

self.assertEqual(self.calculator.perform\_calculation(10, 5, '-'), 5)

def test\_subtraction\_negative(self):

self.assertEqual(self.calculator.perform\_calculation(5, 10, '-'), -5)

# Тести множення

def test\_multiplication\_positive(self):

self.assertEqual(self.calculator.perform\_calculation(10, 5, '\*'), 50)

def test\_multiplication\_negative(self):

self.assertEqual(self.calculator.perform\_calculation(10, -5, '\*'), -50)

def test\_multiplication\_zero(self):

self.assertEqual(self.calculator.perform\_calculation(10, 0, '\*'), 0)

# Тести ділення

def test\_division(self):

self.assertEqual(self.calculator.perform\_calculation(10, 5, '/'), 2)

def test\_division\_zero(self):

with self.assertRaises(ZeroDivisionError):

self.calculator.perform\_calculation(10, 0, '/')

def test\_invalid\_operator(self):

with self.assertRaises(ValueError):

self.calculator.perform\_calculation(10, 5, 'invalid\_operator')

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

**calculator.py**

class Calculator:

def \_\_init\_\_(self):

self.memory = []

def get\_user\_input(self):

try:

num1 = float(input("Введіть перше число: "))

operator = input("Введіть оператор (+, -, \*, /, ^, sq, %): ")

if operator not in ['+', '-', '\*', '/', '^', 'sq', '%']:

raise ValueError("Невідомий оператор")

num2 = float(input("Введіть друге число: "))

return num1, num2, operator

except (ValueError, ZeroDivisionError) as error:

raise ValueError(f"Помилка: {error}")

def perform\_calculation(self, num1, num2, operator):

if operator == '+':

result = num1 + num2

elif operator == '-':

result = num1 - num2

elif operator == '\*':

result = num1 \* num2

elif operator == '/':

if num2 == 0:

raise ZeroDivisionError("Ділення на нуль неможливе")

result = num1 / num2

elif operator == '^':

result = num1 \*\* num2

elif operator == 'sq':

result = num1 \*\* (1 / num2)

elif operator == '%':

result = num1 % num2

else:

raise ValueError("Невідомий оператор")

self.memory.append(result)

return result

def run\_calculator(self):

while True:

try:

num1, num2, operator = self.get\_user\_input()

self.perform\_calculation(num1, num2, operator)

except ValueError as error:

print(f"Помилка: {error}")

finally:

another\_calculation = input("Бажаєте виконати ще одне обчислення? (так/ні): ")

if another\_calculation.lower() != 'так':

self.show\_calculation\_history()

break

def show\_calculation\_history(self):

print("Історія обчислень:")

for idx, item in enumerate(self.memory, start=1):

print(f"{idx}: {item}")

calc = Calculator()

calc.run\_calculator()

**Результат виконання**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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