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»

на тему «Розробка та Unit тестування Python додатку»

Виконала:

студентка гр. ІТ-31

Проців Р.В.

Прийняв:

доц. каф. ІСМ

Щербак С.С.

Львів – 2023

**Мета роботи**: Cтворення юніт-тестів для додатка-калькулятора на основі класів

**Хід 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.

**Програмний код**:

import unittest

from unittest.mock import patch

import sys

from importlib.machinery import SourceFileLoader

sys.path.append(r"D:\Лаби\3 курс\Лаби\Пітон\python")

lab2\_module = SourceFileLoader(

"calculator",

r"D:\Лаби\3 курс\Лаби\Пітон\python\lab\_2\lab\_2.py",

).load\_module()

class CalculatorTests(unittest.TestCase):

def setUp(self):

self.calculator = lab2\_module.HistoryCalculator()

@patch("builtins.input", side\_effect=[5, 3, "+"])

def test\_addition(self, mock\_input):

self.calculator.calculate()

self.assertEqual(self.calculator.result, 8)

@patch("builtins.input", side\_effect=[5, 3, "-"])

def test\_subtraction(self, mock\_input):

self.calculator.calculate()

self.assertEqual(self.calculator.result, 2)

@patch("builtins.input", side\_effect=[5, 3, "\*"])

def test\_multiplication(self, mock\_input):

self.calculator.calculate()

self.assertEqual(self.calculator.result, 15)

@patch("builtins.input", side\_effect=[6, 2, "/"])

def test\_division(self, mock\_input):

self.calculator.calculate()

self.assertEqual(self.calculator.result, 3)

@patch("builtins.input", side\_effect=[6, 0, "/"])

@patch("builtins.print")

def test\_division\_by\_zero(self, mock\_print, mock\_input):

self.calculator.calculate()

mock\_print.assert\_called\_with("Division by zero is not allowed.")

@patch("builtins.input", side\_effect=[5, 7, "+"])

def test\_add\_to\_history(self, mock\_input):

self.calculator.calculate()

self.assertEqual(self.calculator.history, ["5.0 + 7.0 = 12.0"])

def main ():

unittest.main()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**GitHub Repository:** <https://github.com/fxxwol/python>

**Висновок:**  в результаті даної лабораторної роботи було створено набір юніт-тестів, які перевіряють правильність основних арифметичних операцій у додатку-калькуляторі. Ці тести допомогли виявити та виправити будь-які проблеми або помилки, які могли виникнути під час розробки чи обслуговування додатку, забезпечуючи його надійність і точність.