МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ



Лабораторна робота №6

з дисципліни “Спеціалізовані мови програмування”

Виконала:

студентка групи ІТ-31

Юлія ДОБОШ

Прийняв:

доцент кафедри ІСМ

Сергій ЩЕРБАК

Львів-2023

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

**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.

Детально код можна переглянути тут : <https://github.com/YuliiaDobosh/Python.git>

Код програми :

import unittest

from sources.lab2.calc import Calculator

class TestCalculatorMethods(unittest.TestCase):

def test\_init(self):

calc = Calculator()

self.assertEqual(0, calc.\_result)

# Перевірка введених чисел - \_num1 та \_num2 повинні бути або типу float, або int.

def test\_input\_nums(self):

calc = Calculator()

calc.input\_numbers()

self.assertIsInstance(calc.\_num1, float | int)

self.assertIsInstance(calc.\_num2, float | int)

# Перевірка введеного оператора - \_operator повинен бути типу str.

def test\_input\_operator(self):

calc = Calculator()

calc.input\_operator()

self.assertIsInstance(calc.\_operator, str)

# Перевірка функціоналу додавання.

def test\_add(self):

calc = Calculator()

calc.\_num1 = 3

calc.\_num2 = 11

calc.\_operator = '+'

calc.calculate()

self.assertEqual(calc.\_result, 14)

# Перевірка функціоналу віднімання.

def test\_subtract(self):

calc = Calculator()

calc.\_num1 = 15

calc.\_num2 = 5

calc.\_operator = '-'

calc.calculate()

self.assertEqual(calc.\_result, 10)

# Перевірка функціоналу множення.

def test\_multiply(self):

calc = Calculator()

calc.\_num1 = 4

calc.\_num2 = 5

calc.\_operator = '\*'

calc.calculate()

self.assertEqual(calc.\_result, 20)

# Перевірка функціоналу ділення.

def test\_divide(self):

calc = Calculator()

calc.\_num1 = 20

calc.\_num2 = 5

calc.\_operator = '/'

calc.calculate()

self.assertEqual(calc.\_result, 4)

# Тест на випадок ділення на нуль - має викликати ZeroDivisionError.

def test\_divide\_by\_zero(self):

calc = Calculator()

calc.\_num1 = 10

calc.\_num2 = 0

calc.\_operator = '/'

with self.assertRaises(ZeroDivisionError):

calc.calculate()

# Запуск тестів

def main():

suite = unittest.TestLoader().loadTestsFromTestCase(TestCalculatorMethods)

runner = unittest.TextTestRunner()

runner.run(suite)

input()

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

main()

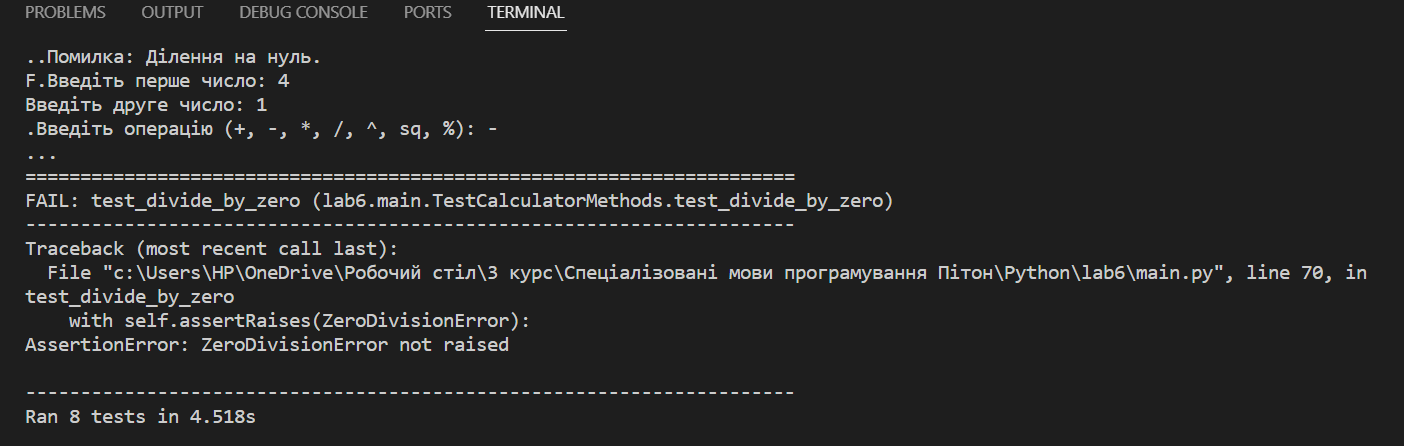


Рис.1 Результат виконання роботи

**Висновок**: Виконуючи лабораторну роботу було створено набір юніт-тестів, які перевіряють правильність основних арифметичних операцій у вашому 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