МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІСМ

ЗВІТ

про виконання лабораторної роботи №6

з дисципліни

«Спеціалізовані мови програмування»

студента групи ІТ-32

Ткачишина Юрія

Прийняв Щербак С. С.

Львів - 2023

Мета роботи: Створення юніт-тестів для додатка-калькулятора на основі класів.

Індивідуальне завдання

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.

Хід виконання:

На рис. 1 зображено знімок екрану із середовища розробки.

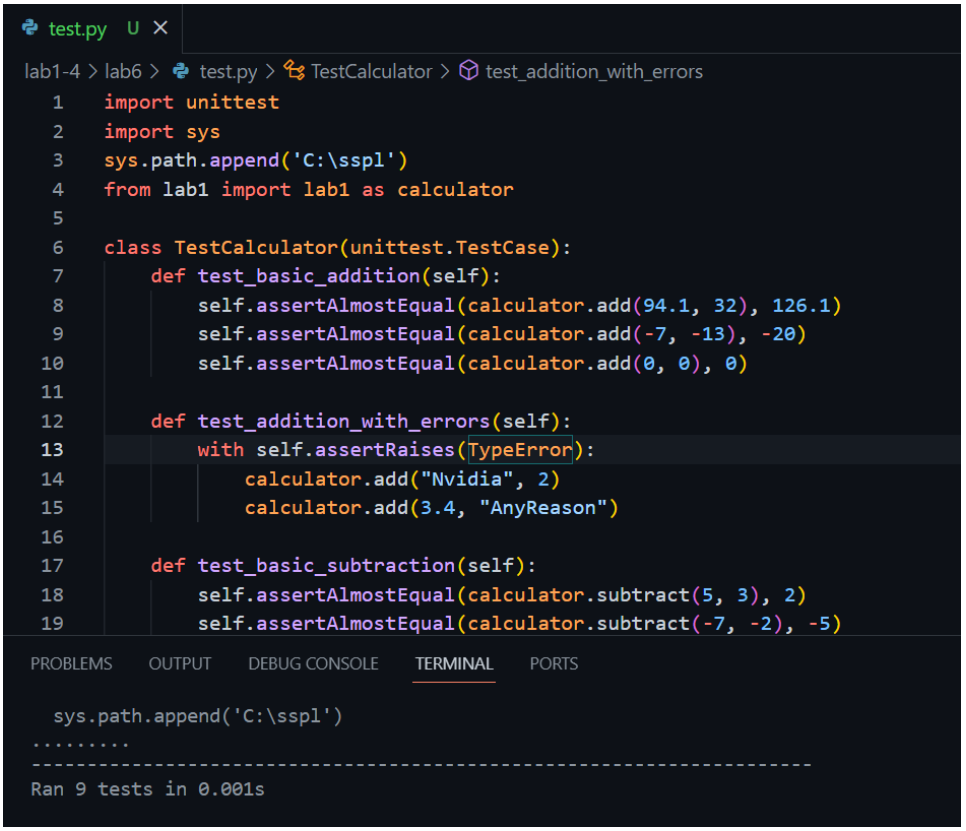

Рис. 1 Виконання програми

Код:

```
import unittest

import sys

sys.path.append('C:\sspl')

from lab1 import lab1 as calculator


class TestCalculator(unittest.TestCase):

    def test_basic_addition(self):

    self.assertAlmostEqual(calculator.add(94.1, 32), 126.1)

    self.assertAlmostEqual(calculator.add(-7, -13), -20)

    self.assertAlmostEqual(calculator.add(0, 0), 0)


    def test_addition_with_errors(self):

    with self.assertRaises(TypeError):
```

```
calculator.add("Nvidia", 2)
calculator.add(3.4, "AnyReason")

def test_basic_subtraction(self):
self.assertAlmostEqual(calculator.subtract(5, 3), 2)
self.assertAlmostEqual(calculator.subtract(-7, -2), -5)
self.assertAlmostEqual(calculator.subtract(8, 10), -2)

def test_subtraction_with_errors(self):
with self.assertRaises(TypeError):
calculator.subtract("Amd", 4)
calculator.subtract(3.6, "Zeon")

def test_basic_multiplication(self):
self.assertAlmostEqual(calculator.multiply(6, 7), 42)
self.assertAlmostEqual(calculator.multiply(-3, 4), -12)
self.assertAlmostEqual(calculator.multiply(0, 5), 0)

def test_multiplication_with_errors(self):
with self.assertRaises(TypeError):
calculator.multiply("Apollon", 8)
calculator.multiply(3.14, "Zen")

def test_basic_division(self):
self.assertAlmostEqual(calculator.divide(15, 3), 5)
self.assertAlmostEqual(calculator.divide(-10, 5), -2)
self.assertAlmostEqual(calculator.divide(5, 2), 2.5)
```

```
    def test_division_by_zero(self):
    with self.assertRaises(ZeroDivisionError):
    calculator.divide(7, 0)
    calculator.divide(-4, 0)

    def test_division_with_errors(self):
    with self.assertRaises(TypeError):
    calculator.divide("Devolto", 3)
    calculator.divide(95, "Maximus")

if __name__ == '__main__':
    unittest.main()
```

Висновок: Я створив тести, які перевіряють правильність основних арифметичних операцій у вашому додатку-калькуляторі. Ці тести допомогли виявити та виправити будь-які проблеми або помилки, які можуть виникнути під час розробки чи обслуговування вашого додатку, забезпечуючи його надійність і точність.