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:

ст. гр. ІТ-32

Ольга ЧИГИРИК

Прийняв

доцент каф. ІСМ:

Сергій ЩЕРБАК

|  |  |
| --- | --- |
| **Балів** | **Дата** |
|  |  |

Львів-2023

**Мета**: 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**

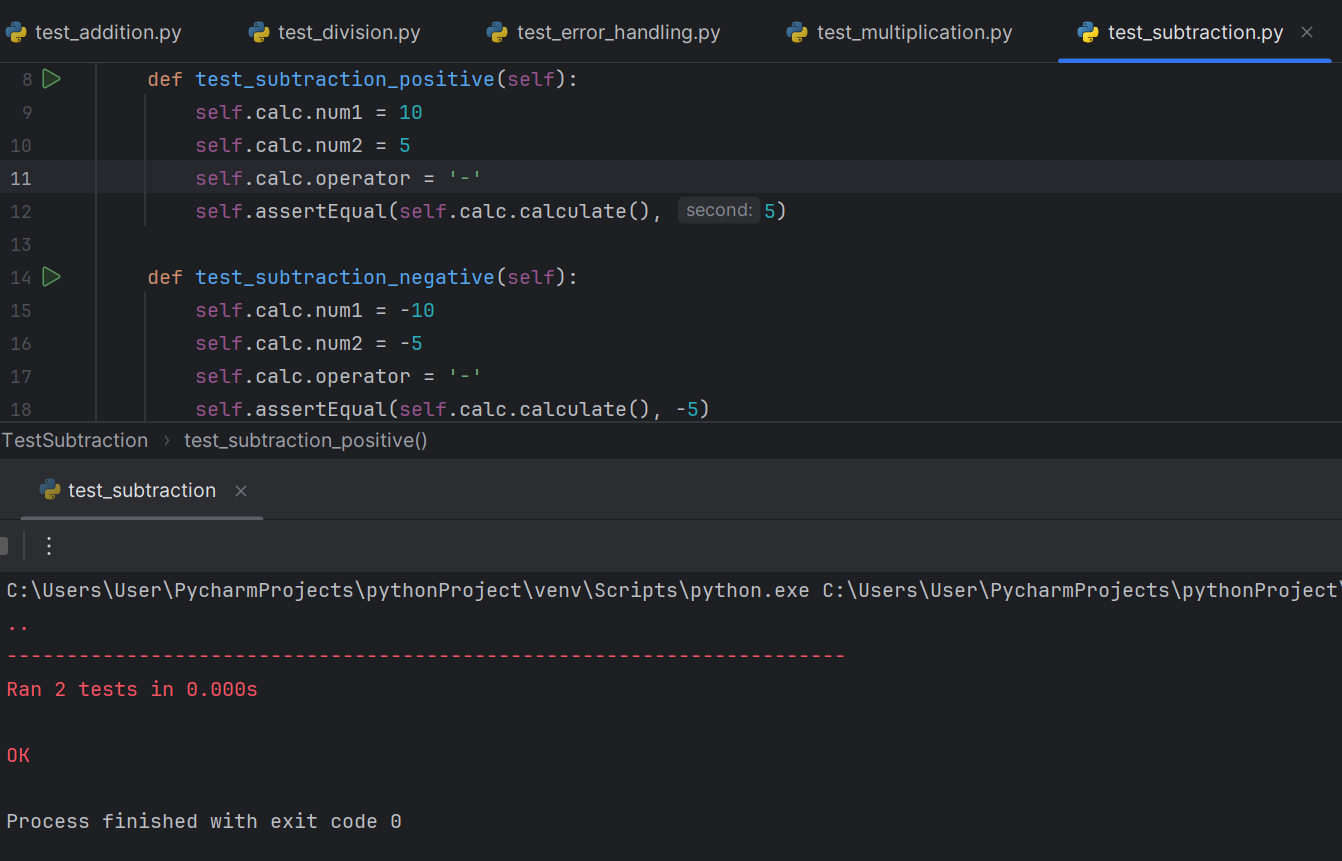
Розробіть юніт-тести для підтвердження точності операції ділення. Тести повинні охоплювати ситуації, пов'язані з діленням на нуль та різними числовими значеннями.

**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.

**Test\_subtraction.py**

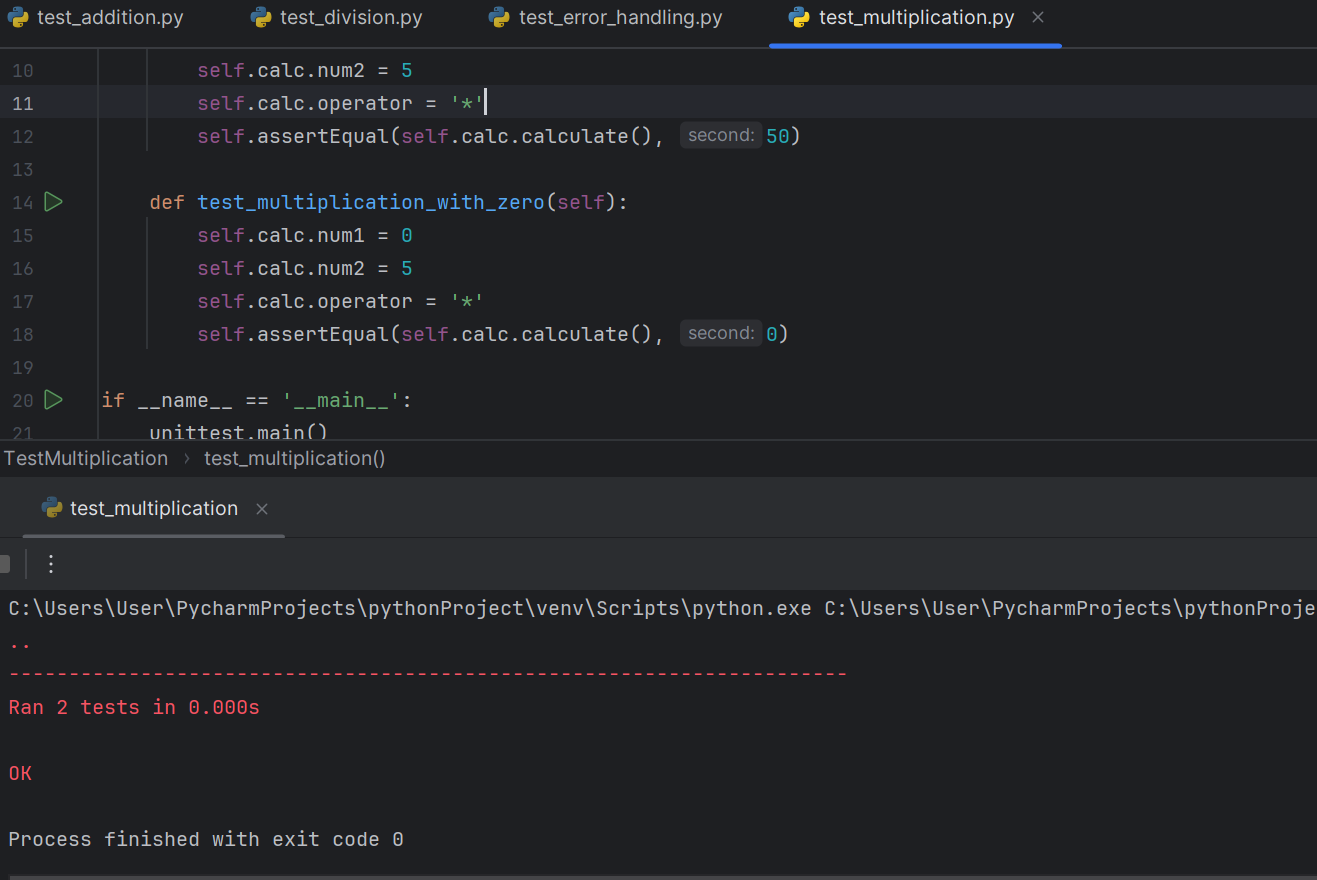
import unittest  
from Scripts.all\_python\_labs.lab\_python\_2.functions2 import Calculator  
  
class TestSubtraction(unittest.TestCase):  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_subtraction\_positive(self):  
 self.calc.num1 = 10  
 self.calc.num2 = 5  
 self.calc.operator = '-'  
 self.assertEqual(self.calc.calculate(), 5)  
  
 def test\_subtraction\_negative(self):  
 self.calc.num1 = -10  
 self.calc.num2 = -5  
 self.calc.operator = '-'  
 self.assertEqual(self.calc.calculate(), -5)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()



*Рис. 1 Результат виконання юніттесту*

**Test\_multiplication.py**

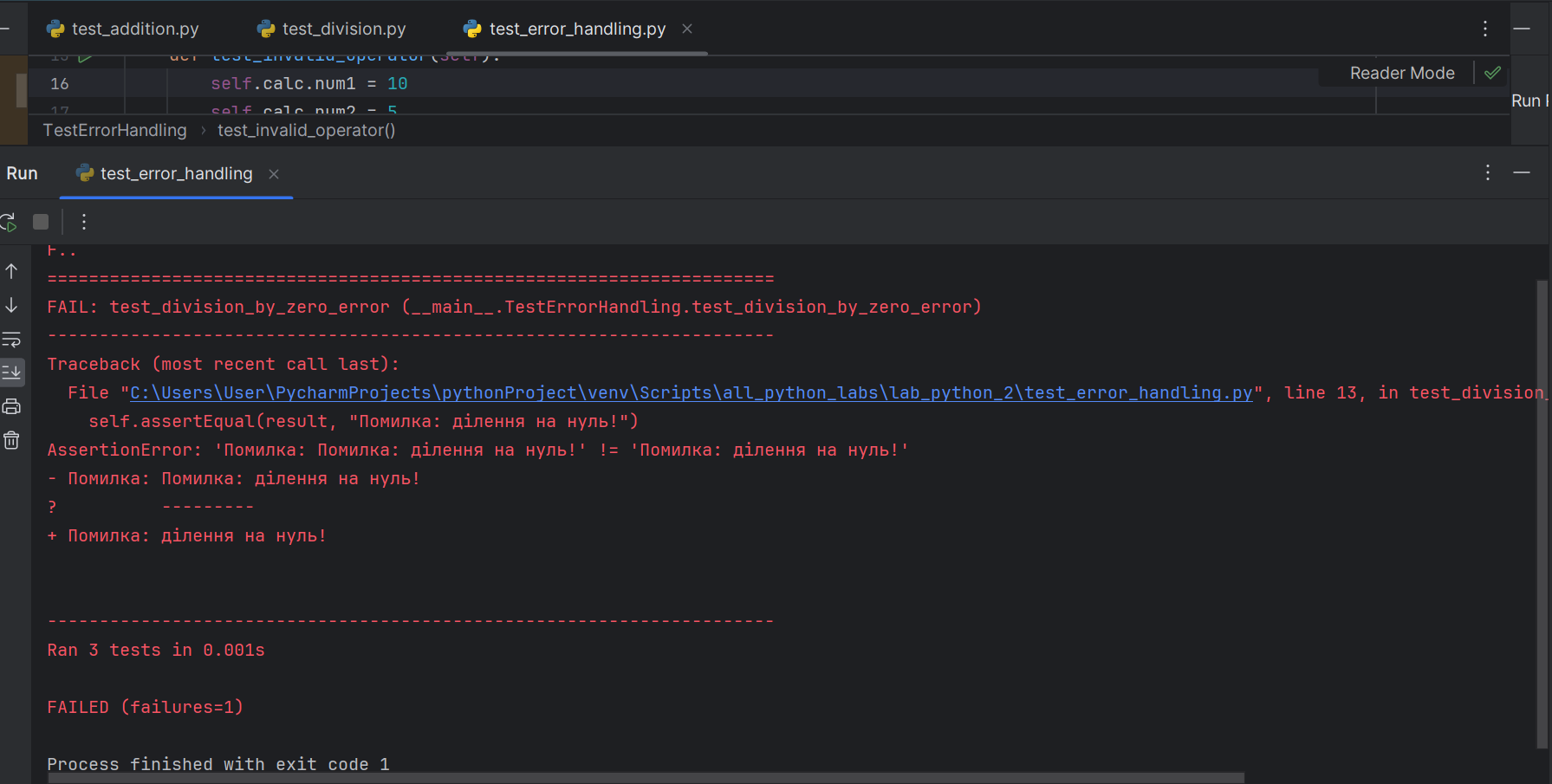
import unittest  
from Scripts.all\_python\_labs.lab\_python\_2.functions2 import Calculator  
  
class TestMultiplication(unittest.TestCase):  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_multiplication(self):  
 self.calc.num1 = 10  
 self.calc.num2 = 5  
 self.calc.operator = '\*'  
 self.assertEqual(self.calc.calculate(), 50)  
  
 def test\_multiplication\_with\_zero(self):  
 self.calc.num1 = 0  
 self.calc.num2 = 5  
 self.calc.operator = '\*'  
 self.assertEqual(self.calc.calculate(), 0)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()



*Рис. 2 Результат виконання юніттесту*

**Test\_error\_handling.py**

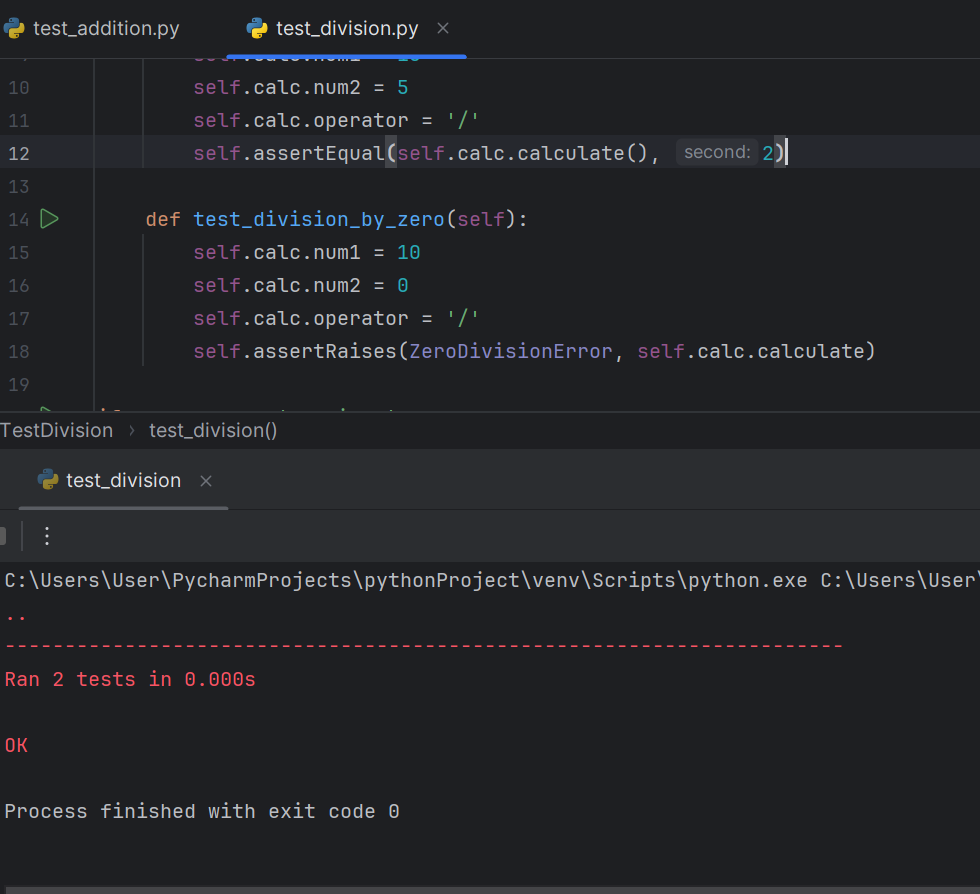
import unittest  
from Scripts.all\_python\_labs.lab\_python\_2.functions2 import Calculator  
  
class TestErrorHandling(unittest.TestCase):  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_division\_by\_zero\_error(self):  
 self.calc.num1 = 10  
 self.calc.num2 = 0  
 self.calc.operator = '/'  
 result = self.calc.safe\_calculate()  
 self.assertEqual(result, "Помилка: ділення на нуль!")  
  
 def test\_invalid\_operator(self):  
 self.calc.num1 = 10  
 self.calc.num2 = 5  
 self.calc.operator = '@' # Invalid operator  
 result = self.calc.safe\_calculate()  
 self.assertEqual(result, "Помилка: Недійсний оператор")  
  
 def test\_sqrt\_of\_negative\_number(self):  
 self.calc.num1 = -4  
 self.calc.num2 = 2  
 self.calc.operator = '√'  
 result = self.calc.safe\_calculate()  
 self.assertEqual(result, "Помилка: Корінь з від'ємного числа не дозволений")  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()



*Рис. 3 Результат виконання юніттесту, показує помилку, адже в калькуляторі не передбачено ділення на 0*

**Test\_division.py**

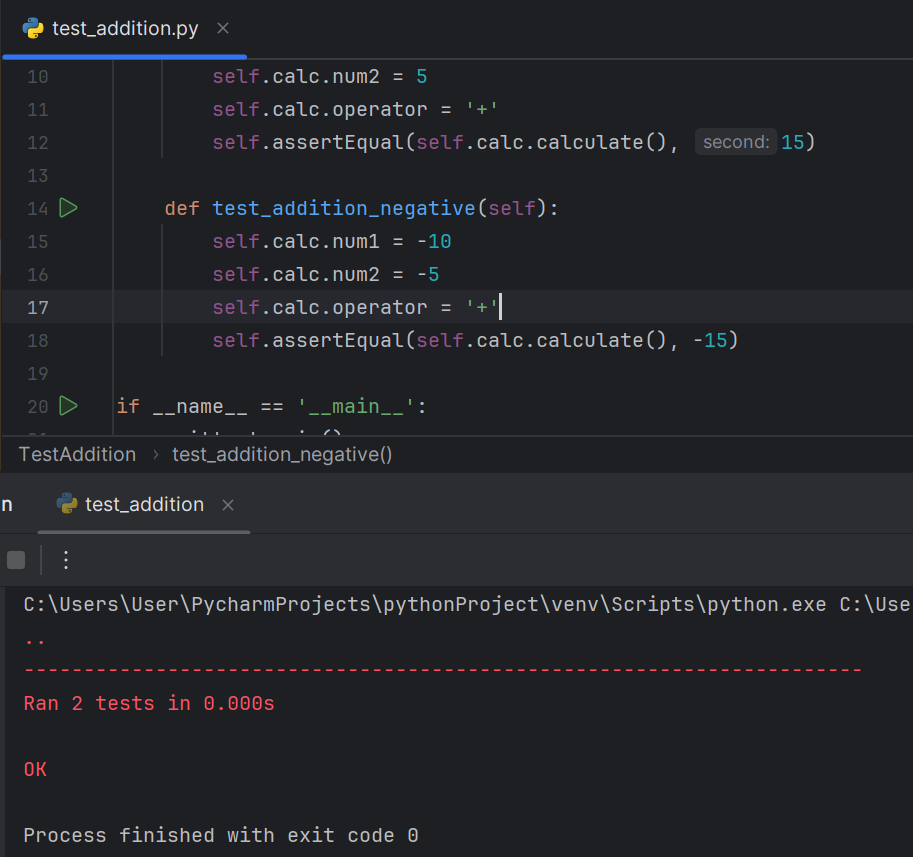
import unittest  
from Scripts.all\_python\_labs.lab\_python\_2.functions2 import Calculator  
  
class TestDivision(unittest.TestCase):  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_division(self):  
 self.calc.num1 = 10  
 self.calc.num2 = 5  
 self.calc.operator = '/'  
 self.assertEqual(self.calc.calculate(), 2)  
  
 def test\_division\_by\_zero(self):  
 self.calc.num1 = 10  
 self.calc.num2 = 0  
 self.calc.operator = '/'  
 self.assertRaises(ZeroDivisionError, self.calc.calculate)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()



*Рис. 4 Результат виконання юніттесту*

**Test\_additiun.py**

import unittest  
from Scripts.all\_python\_labs.lab\_python\_2.functions2 import Calculator  
  
class TestAddition(unittest.TestCase):  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_addition\_positive(self):  
 self.calc.num1 = 10  
 self.calc.num2 = 5  
 self.calc.operator = '+'  
 self.assertEqual(self.calc.calculate(), 15)  
  
 def test\_addition\_negative(self):  
 self.calc.num1 = -10  
 self.calc.num2 = -5  
 self.calc.operator = '+'  
 self.assertEqual(self.calc.calculate(), -15)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()



*Рис. 5 Результат виконання юніттесту*

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