МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

**Звіт**

До лабораторної роботи №6

**Розробка та Unit тестування Python додатку**

Виконав:

ст. гр. ІТ-21сп

**Ярослав МАРКІВСЬКИЙ**

Прийняв:

доц. каф. ІСМ

**Сергій ЩЕРБАК**

Львів ‒ 2023

**Лабораторна робота №6**

**Мета:** створення 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.

Файл **testCalculatorAddition.py**

import unittest  
  
from labwork\_2.calculator import Calculator  
  
  
class TestCalculatorAddition(unittest.TestCase):  
  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_positive\_add(self):  
 result = self.calc.\_calculation(3, 5, '+')  
 self.assertEqual(result, 8)  
  
 def test\_negative\_add(self):  
 result = self.calc.\_calculation(-3, -5, '+')  
 self.assertEqual(result, -8)

Файл **testCalculatorDivision.py**

import unittest  
  
from labwork\_2.calculator import Calculator  
  
  
class TestCalculatorDivision(unittest.TestCase):  
  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_positive\_division(self):  
 result = self.calc.\_calculation(25, 5, '/')  
 self.assertEqual(result, 5)  
  
 def test\_negative\_division(self):  
 result = self.calc.\_calculation(-25, 5, '/')  
 self.assertEqual(result, -5)  
  
 def test\_zero\_division(self):  
 result = self.calc.\_calculation(-5, 0, '/')  
 self.assertEqual(result, 0)

Файл **testCalculatorMultiplication.py**

import unittest  
  
from labwork\_2.calculator import Calculator  
  
  
class TestCalculatorMultiplication(unittest.TestCase):  
  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_positive\_multiplication(self):  
 result = self.calc.\_calculation(5, 5, '\*')  
 self.assertEqual(result, 25)  
  
 def test\_negative\_multiplication(self):  
 result = self.calc.\_calculation(-5, 5, '\*')  
 self.assertEqual(result, -25)  
  
 def test\_zero\_multiplication(self):  
 result = self.calc.\_calculation(-5, 0, '\*')  
 self.assertEqual(result, 0)

Файл **testCalculatorSubtract.py**

import unittest  
  
from labwork\_2.calculator import Calculator  
  
  
class TestCalculatorSubtraction(unittest.TestCase):  
  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_positive\_subtract(self):  
 result = self.calc.\_calculation(10, 5, '-')  
 self.assertEqual(result, 5)  
  
 def test\_negative\_subtract(self):  
 result = self.calc.\_calculation(-10, -5, '-')  
 self.assertEqual(result, -5)

Файл **testErrorHandling.py**

import unittest  
  
from labwork\_2.calculator import Calculator  
  
  
class TestErrorHandling(unittest.TestCase):  
  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_error\_division\_by\_zero(self):  
 with self.assertRaises(ZeroDivisionError):  
 self.calc.\_calculation(10, 0, '/')

Файл **variables.py**

choice\_yes = "yes"  
  
user\_input = "Select a group of tests: \n 1. Addition\n 2. Division\n 3. Multiplication\n 4. Subtract\n 5. Error Handling\nPlease choose what you want: "  
  
error\_input = "Error: You made the wrong choice"

Файл **errors.py**

class Error:  
  
 def number\_input\_check(self, number):  
 try:  
 number = int(number)  
 if number < 1 or number > 5:  
 raise ValueError  
 except ValueError:  
 return ValueError()

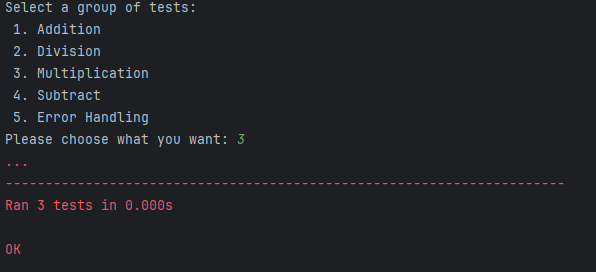
Файл **interface.py**

import unittest  
  
from colorama import Fore, Style  
  
from labwork\_6 import variables  
from labwork\_6.errors import Error  
from labwork\_6.testCalculatorAddition import TestCalculatorAddition  
from labwork\_6.testCalculatorDivision import TestCalculatorDivision  
from labwork\_6.testCalculatorMultiplication import TestCalculatorMultiplication  
from labwork\_6.testCalculatorSubtract import TestCalculatorSubtraction  
from labwork\_6.testErrorHandling import TestErrorHandling  
  
  
class Interface:  
  
 def \_\_init\_\_(self):  
 self.error = Error()  
 self.suite = unittest.TestSuite()  
 self.loader = unittest.TestLoader()  
  
 def user\_interface(self):  
 while True:  
  
 choice = self.get\_input(variables.user\_input, self.error.number\_input\_check, variables.error\_input)  
 if choice == '1':  
 self.suite.addTest(self.loader.loadTestsFromTestCase(TestCalculatorAddition))  
 return self.suite  
 elif choice == '2':  
 self.suite.addTest(self.loader.loadTestsFromTestCase(TestCalculatorSubtraction))  
 return self.suite  
 elif choice == '3':  
 self.suite.addTest(self.loader.loadTestsFromTestCase(TestCalculatorMultiplication))  
 return self.suite  
 elif choice == '4':  
 self.suite.addTest(self.loader.loadTestsFromTestCase(TestCalculatorDivision))  
 return self.suite  
 elif choice == '5':  
 self.suite.addTest(self.loader.loadTestsFromTestCase(TestErrorHandling))  
 return self.suite  
  
 def get\_input(self, number, error\_check, error\_message):  
 while True:  
 value = input(number)  
 if not error\_check(value):  
 return value  
 print(Fore.RED + Style.BRIGHT + error\_message + Style.RESET\_ALL)

Файл **main.py**

import unittest  
  
from labwork\_6.interface import Interface  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 interface = Interface()  
 suite = interface.user\_interface()  
 unittest.TextTestRunner().run(suite)

Результат виконання всіх завдань:



**Рис.1** Результат виконання

**Висновок:** Виконавши ці завдання, я розробив Unit тестування для Python додатку.