Міністерство освіти і науки України

Національний університет «Львівська політехніка»



Лабораторна робота №6

з навчальної дисципліни

«Спеціалізовані мови програмування»

на тему

«Розробка та Unit тестування Python додатку»

*Інститут комп’ютерних наук*

*та інформаційних технологій*

*Кафедри інформаційних*

*систем та мереж*

***Виконав:***

*Студент групи РІ-31*

*Олександр ЛУЧКЕВИЧ*

***Прийняла:***

*викладач*

*Cергій ЩЕРБАК*

*Львів – 2024*

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

**План 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.

**Код програми:**

Файл CalculatorAdditionUnitTests.py:

import random

import string

import unittest

from Lab\_2.calculator import Calculator

class CalculatorAdditionUnitTests(unittest.TestCase):

def test\_add\_positive\_numbers\_returns\_correct\_value(self):

# Arrange

calculator = Calculator()

test\_num1 = random.randrange(1, 100) \* 1.0

test\_num2 = random.randrange(1, 100) \* 1.0

operator = "+"

expected = test\_num1 + test\_num2

calc = calculator.calculate(test\_num1, test\_num2, '+')

# Assert

self.assertEqual(expected, calc)

def test\_add\_negative\_numbers\_returns\_correct\_value(self):

# Arrange

calculator = Calculator()

test\_num1 = random.randrange(-100, -1) \* 1.0

test\_num2 = random.randrange(-100, -1) \* 1.0

operator = "+"

expected = test\_num1 + test\_num2

calc = calculator.calculate(test\_num1, test\_num2, '+')

# Assert

self.assertEqual(expected, calc)

def test\_add\_positive\_and\_negative\_numbers\_returns\_correct\_value(self):

# Arrange

calculator = Calculator()

test\_num1 = random.randrange(-100, -1) \* 1.0

test\_num2 = random.randrange(1, 100) \* 1.0

operator = "+"

expected = test\_num1 + test\_num2

calc = calculator.calculate(test\_num1, test\_num2, '+')

# Assert

self.assertEqual(expected, calc)

def test\_add\_positive\_numbers\_returns\_correct\_value(self):

# Arrange

calculator = Calculator()

test\_num1 = random.randrange(1, 100) \* 1.0

test\_num2 = random.randrange(1, 100) \* 1.0

operator = "+"

expected = test\_num1 + test\_num2

calc = calculator.calculate(test\_num1, test\_num2, '+')

# Assert

self.assertEqual(expected, calc)

Файл CalculatorDivisionUnitTests.py:

import random

import string

import unittest

from Lab\_2.calculator import Calculator

class CalculatorDivisionUnitTests(unittest.TestCase):

def test\_division\_of\_two\_positive\_numbers(self):

# Arrange

calculator = Calculator()

number1 = 10

number2 = 2

expected\_result = 5

# Act

actual\_result = calculator.division(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

def test\_division\_of\_two\_negative\_numbers(self):

# Arrange

calculator = Calculator()

number1 = -10

number2 = -2

expected\_result = 5

# Act

actual\_result = calculator.division(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

def test\_division\_of\_positive\_and\_negative\_numbers(self):

# Arrange

calculator = Calculator()

number1 = 10

number2 = -2

expected\_result = -5

# Act

actual\_result = calculator.division(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

def test\_division\_of\_two\_random\_numbers(self):

# Arrange

calculator = Calculator()

number1 = random.randint(1, 100)

number2 = random.randint(1, 100)

expected\_result = number1 / number2

# Act

actual\_result = calculator.division(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

def test\_division\_of\_two\_random\_float\_numbers(self):

# Arrange

calculator = Calculator()

number1 = random.uniform(1, 100)

number2 = random.uniform(1, 100)

expected\_result = number1 / number2

# Act

actual\_result = calculator.division(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

Файл CalculatorMultiplicationUnitTests.py:

import random

import string

import unittest

from Lab\_2.calculator import Calculator

class CalculatorMultiplicationUnitTests(unittest.TestCase):

def test\_multiplication\_of\_two\_positive\_numbers(self):

# Arrange

calculator = Calculator()

number1 = 10

number2 = 20

expected\_result = 200

# Act

actual\_result = calculator.multiplication(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

def test\_multiplication\_of\_two\_negative\_numbers(self):

# Arrange

calculator = Calculator()

number1 = -10

number2 = -20

expected\_result = 200

# Act

actual\_result = calculator.multiplication(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

def test\_multiplication\_of\_positive\_and\_negative\_numbers(self):

# Arrange

calculator = Calculator()

number1 = 10

number2 = -20

expected\_result = -200

# Act

actual\_result = calculator.multiplication(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

def test\_multiplication\_of\_two\_random\_numbers(self):

# Arrange

calculator = Calculator()

number1 = random.randint(1, 100)

number2 = random.randint(1, 100)

expected\_result = number1 \* number2

# Act

actual\_result = calculator.multiplication(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

def test\_multiplication\_of\_two\_random\_float\_numbers(self):

# Arrange

calculator = Calculator()

number1 = random.uniform(1, 100)

number2 = random.uniform(1, 100)

expected\_result = number1 \* number2

# Act

actual\_result = calculator.multiplication(number1, number2)

# Assert

self.assertEqual(expected\_result, actual\_result)

**Висновок.**

Виконавши ці завдання, у мене є 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