МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ



Лабораторна робота №6

з дисципліни “Спеціалізовані мови програмування”

на тему

“озробка та Unit тестування Python додатку .”

Виконала:

студентка групи РІ-32

Ангеліна Мацкула

Прийняв:

к.т.н.,

доц. кафедри ІСМ

Сергій ЩЕРБАК

Львів-2023

**Мета роботи:** Створення Генератора ASCII-арту без використання зовнішніх бібліотек

**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.

Програмний код:

from lab2 import Calculator

import unittest

class CalculatorTest(unittest.TestCase):

def setUp(self):

self.calc = Calculator()

def test\_add\_positive(self):

self.calc.x = 5

self.calc.y = 3

self.calc.operator = '+'

self.assertEqual(self.calc.calculation(), 8)

def test\_add\_negative(self):

self.calc.x = -5

self.calc.y = -3

self.calc.operator = '+'

self.assertEqual(self.calc.calculation(), -8)

def test\_sub\_positive(self):

self.calc.x = 5

self.calc.y = 3

self.calc.operator = '-'

self.assertEqual(self.calc.calculation(), 2)

def test\_sub\_negative(self):

self.calc.x = -5

self.calc.y = -3

self.calc.operator = '-'

self.assertEqual(self.calc.calculation(), -2)

def test\_mult\_positive(self):

self.calc.x = 5

self.calc.y = 3

self.calc.operator = '\*'

self.assertEqual(self.calc.calculation(), 15)

def test\_mult\_negative(self):

self.calc.x = -5

self.calc.y = -3

self.calc.operator = '\*'

self.assertEqual(self.calc.calculation(), 15)

def test\_mult\_zero(self):

self.calc.x = 5

self.calc.y = 0

self.calc.operator = '\*'

self.assertEqual(self.calc.calculation(), 0)

def test\_div(self):

self.calc.x = 5

self.calc.y = 3

self.calc.operator = '/'

self.assertEqual(self.calc.calculation(), 5/3)

class CalculatorErrorTest(unittest.TestCase):

def setUp(self):

self.calc = Calculator()

def test\_wrong\_operator(self):

self.calc.x = 10

self.calc.y = 5

self.calc.operator = 'wrong'

self.assertFalse(self.calc.validate\_operator())

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

self.calc.x = 10

self.calc.y = 0

self.calc.operator = '/'

self.assertEqual(self.calc.calculation(), "You can't divide by 0!")

def test\_wrong\_first\_num(self):

self.calc.x = "wrong\_num"

self.calc.y = 5

self.calc.operator = '+'

self.assertFalse(self.calc.user\_input(), "Invalid number input")

def test\_wrong\_second\_num(self):

self.calc.x = 10

self.calc.y = "wrong\_num"

self.calc.operator = '+'

self.assertFalse(self.calc.user\_input(), "Invalid number input")

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

unittest.main()

Висновок: Виконуючи ці завдання, я створила набір юніт-тестів, які перевіряють правильність основних арифметичних операцій у вашому додатку-калькуляторі. Ці тести допоможуть виявити та виправити будь-які проблеми або помилки, які можуть виникнути під час розробки чи обслуговування вашого додатку, забезпечуючи його надійність і точність