Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Лабораторна роботи №6

з дисципліни «Спеціалізовані мови програмування»

на тему

«Розробка та Unit тестування Python додатку»

Виконав:

Костик В. Ю.

Перевірив:

Щербак С. С.

Львів 2024

**Мета:** 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.

**Реалізація:**

**Файл AppSettings.py:**  
import lab1.constants.constants as const

class AppSettings:

    def \_\_init\_\_(self):

        self.decimal\_places = const.DECIMALS

        self.console\_color = "white"

    def change\_decimals(self, decimals):

        self.decimal\_places = decimals

    def change\_color(self, color):

        self.console\_color = color

**Папка functions, файл calcFunctions.py:**import math

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    try:

        return a / b

    except ZeroDivisionError:

        return "Error: Division by zero"

def power(a, b):

    return a \*\* b

def sqrt(a):

    return math.sqrt(a)

def modulus(a, b):

    return a % b

**Папка interface, файл userInterface.py:**

def get\_user\_input(prompt):

return input(prompt)

def display\_output(message):

print(message)

**Папка functions, файл Calculator.py:**

import math

class Calculator:

    def \_\_init\_\_(self):

        self.num1 = 0.0

        self.num2 = 0.0

        self.operator = ''

        self.valid\_operators = ['+', '-', '\*', '/', '^', '√', '%']

    def get\_user\_input(self):

        expression = input("Enter expression (e.g. '5 + 3' or '√9'): ").strip()

        return self.parse\_expression(expression)

    def parse\_expression(self, expression):

        if '√' in expression:

            self.operator = '√'

            self.num1 = float(expression[1:])

            self.num2 = None

        else:

            for operator in self.valid\_operators:

                if operator in expression:

                    self.num1, self.num2 = map(float, expression.split(operator))

                    self.operator = operator

                    break

        if self.operator not in self.valid\_operators:

            raise ValueError("Invalid operator!")

    def check\_operator(self):

        if self.operator not in self.valid\_operators:

            raise ValueError(f"Invalid operator: {self.operator}")

    def calculate(self):

        try:

            if self.operator == '+':

                return self.num1 + self.num2

            elif self.operator == '-':

                return self.num1 - self.num2

            elif self.operator == '\*':

                return self.num1 \* self.num2

            elif self.operator == '/':

                if self.num2 == 0:

                    raise ZeroDivisionError("Cannot divide by zero!")

                return self.num1 / self.num2

            elif self.operator == '^':

                return self.num1 \*\* self.num2

            elif self.operator == '√':

                if self.num1 < 0:

                    raise ValueError("Cannot take square root of negative number!")

                return math.sqrt(self.num1)

            elif self.operator == '%':

                return self.num1 % self.num2

        except (ValueError, ZeroDivisionError) as e:

            print(f"Error: {e}")

            return None

    def display\_result(self, result):

        if result is not None:

            print(f"Result: {result:.2f}")

        else:

            print("Error occurred during calculation.")

    def run(self):

        while True:

            try:

                self.get\_user\_input()

                result = self.calculate()

                self.display\_result(result)

            except ValueError as e:

                print(e)

            choice = input("Do you want to perform another calculation? (y/n): ").lower()

            if choice != 'y':

                break

**Файл main.py:**

from lab6.functions.Calculator import Calculator

def main():

    calc = Calculator()

    calc.perform\_calculation()

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

    main()

**Файл AdditionTests.py**import unittest

from lab6.functions.calcFunctions import AddOperation as add

class TestAddition(unittest.TestCase):

    def test\_add\_positive\_numbers(self):

        self.assertEqual(add(2, 3), 5)

    def test\_add\_negative\_numbers(self):

        self.assertEqual(add(-2, -3), -5)

    def test\_add\_mixed\_numbers(self):

        self.assertEqual(add(2, -3), -1)

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

    unittest.main()

**Файл DivisionTests.py**

import unittest

from lab6.functions.calcFunctions import SubtractOperation as divide

class TestDivision(unittest.TestCase):

    def test\_divide\_positive\_numbers(self):

        self.assertEqual(divide(6, 2), 3)

    def test\_divide\_negative\_numbers(self):

        self.assertEqual(divide(-6, 2), -3)

    def test\_divide\_by\_zero(self):

        with self.assertRaises(ZeroDivisionError):

            divide(5, 0)

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

    unittest.main()

**Файл ErrorHandlingTests.py**

import unittest

from lab6.functions.calcFunctions import add, subtract, multiply, divide, power, sqrt, modulus

class TestErrorHandling(unittest.TestCase):

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

        with self.assertRaises(ZeroDivisionError):

            divide(5, 0)

    def test\_invalid\_input(self):

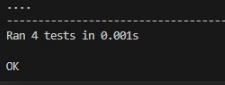
        with self.assertRaises(TypeError):

            add('string', 5)

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

    unittest.main()

**Результат виконання (test\_calculator.py):**

****

**Висновок:** на цій лабораторній роботі я додав набір юніт-тестів, які перевіряють правильність основних арифметичних операцій у 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