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»

Виконав:

ст. гр. РІ-32

Гірняк О. О.

Прийняв:

доц. каф. ІСМ

Щербак С. С.

Львів - 2024

**Мета**

Cтворення юніт-тестів для додатка-калькулятора на основі класів

**Хід виконання роботи**

**Завдання 1: Тестування Додавання**

Було розроблено юніт-тести, які перевіряють правильність операції додавання в додатку-калькуляторі. Тести охоплюють різні випадки, включаючи додавання як позитивних, так і негативних чисел.

**Завдання 2: Тестування Віднімання**

Було створено юніт-тести для перевірки коректності операції віднімання. Тести включають сценарії з різними результатами, в тому числі з від'ємними значеннями.

**Завдання 3: Тестування Множення**

Було написано юніт-тести, що підтверджують правильність операції множення. Враховано множення з нульовим, позитивними та негативними числами.

**Завдання 4: Тестування Ділення**

Розроблено юніт-тести для перевірки точності операції ділення. Тести охоплюють сценарії з діленням на нуль та різними числовими значеннями.

**Завдання 5: Тестування Обробки Помилок**

Створено юніт-тести для перевірки того, як додаток-калькулятор обробляє помилки. Тести включають ділення на нуль та інші потенційні помилки, перевіряючи, чи відображаються відповідні повідомлення про помилки.

**Код програмного продукту**

import unittest

from labs.lab2.bll.Operation import Operation

class CalculatorBllTest(unittest.TestCase):

"""

Unit tests for the CalculatorBll operations.

This test class validates the core mathematical operations

(addition, subtraction, multiplication, and division) as implemented

in the Operation class. For each operation, multiple tests are set

up to ensure the accuracy and correctness of the results.

Methods:

- setUp: Initializes a set of pre-defined operations with known results.

- test\_operation\_add: Verifies the addition operation is correct.

- test\_operation\_subtraction: Verifies the subtraction operation is correct.

- test\_operation\_multiply: Verifies the multiplication operation is correct.

- test\_operation\_divide: Verifies the division operation is correct.

- test\_operation\_divide\_by\_zero: Ensures division by zero results in an appropriate error.

"""

def setUp(self):

self.operation\_add = Operation("+", 5, 3, 8)

self.operation\_add\_2 = Operation("+", 10, 15, 25)

self.operation\_subtract = Operation("-", 7, 2, 5)

self.operation\_subtract\_2 = Operation("-", 20, 5, 15)

self.operation\_multiply = Operation("\*", 4, 3, 12)

self.operation\_multiply\_2 = Operation("\*", 6, 7, 42)

self.operation\_divide = Operation("/", 10, 2, 5)

self.operation\_divide\_2 = Operation("/", 20, 4, 5)

self.operation\_divide\_by\_zero = Operation("/", 10, 0, None)

def test\_operation\_add(self):

self.assertEqual(self.operation\_add.get\_first\_number() + self.operation\_add.get\_second\_number(),

self.operation\_add.get\_result())

self.assertEqual(self.operation\_add\_2.get\_first\_number() + self.operation\_add\_2.get\_second\_number(),

self.operation\_add\_2.get\_result())

def test\_operation\_subtraction(self):

self.assertEqual(self.operation\_subtract.get\_first\_number() - self.operation\_subtract.get\_second\_number(),

self.operation\_subtract.get\_result())

self.assertEqual(self.operation\_subtract\_2.get\_first\_number() - self.operation\_subtract\_2.get\_second\_number(),

self.operation\_subtract\_2.get\_result())

def test\_operation\_multiply(self):

self.assertEqual(self.operation\_multiply.get\_first\_number() \* self.operation\_multiply.get\_second\_number(),

self.operation\_multiply.get\_result())

self.assertEqual(self.operation\_multiply\_2.get\_first\_number() \* self.operation\_multiply\_2.get\_second\_number(),

self.operation\_multiply\_2.get\_result())

def test\_operation\_divide(self):

self.assertEqual(self.operation\_divide.get\_first\_number() / self.operation\_divide.get\_second\_number(),

self.operation\_divide.get\_result())

self.assertEqual(self.operation\_divide\_2.get\_first\_number() / self.operation\_divide\_2.get\_second\_number(),

self.operation\_divide\_2.get\_result())

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

self.assertFalse(self.operation\_divide\_by\_zero.is\_complete())

with self.assertRaises(ZeroDivisionError):

self.operation\_divide\_by\_zero.get\_first\_number() / self.operation\_divide\_by\_zero.get\_second\_number()

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

unittest.main()

**Висновки**

Я створив набір юніт-тестів, що перевіряють мій додаток-калькулятор. При тестуванні я помітив що помилка при діленні з використанням % не відбувалась, що я зміг виправити, що допомогло мені забезпечити надійність і точність додатку.