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:

студент групи РІ-21сп

Андрій ПІГУЛЯК

Львів – 2024

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

**Хід роботи**

1. Написати юніт-тест для перевірки правильності операції додавання. Перевірити роботу з позитивними та негативними числами.

2. Створити юніт-тести для операції віднімання, враховуючи сценарії з від’ємними результатами.

3. Розробити юніт-тести для перевірки операції множення з нулем, позитивними та негативними числами.

4. Написати юніт-тести для підтвердження точності ділення, включаючи перевірку обробки ділення на нуль.

5. Реалізувати юніт-тести для перевірки обробки помилок, включаючи ділення на нуль та інші сценарії. Забезпечити правильне відображення повідомлень про помилки.

1. **Опис і структура програми:**

**1. Програма реалізує калькулятор із такими можливостями:**

* Арифметичні операції: додавання, віднімання, множення, ділення, обчислення модуля та квадратного кореня.
* Робота з пам'яттю (запам'ятовування, очищення, додавання та віднімання з пам'яті).
* Конвертація чисел із рядкового формату в числовий із підтримкою різних форматів запису.

**2. Програма реалізована з використанням об’єктно-орієнтованого підходу. Основними класами є:**

* Calculator — для виконання математичних операцій.
* Memory — для роботи з пам'яттю калькулятора.
* ConvertNumberType — для обробки і переведення чисел.

**3. Юніт-тестування. Для тестування створено набір тестових класів, кожен із яких перевіряє функціональність окремого модуля чи методу:**

* TestCalculatorAddition
* TestCalculatorSubtraction
* TestCalculatorMultiplication
* TestCalculatorDivision
* TestCalculatorMod
* TestCalculatorSquareRoot
* TestMemory
* TestConvertNumberType

**4. Кожен клас містить тести для перевірки:**

1. Коректної роботи з валідними даними.
2. Роботи з граничними випадками.
3. Обробки невалідних вхідних даних (очікуваних помилок).

**Основні функціональні модулі:**

1. test\_prog.py

import unittest  
from labs.lab2.classes.calculator import Calculator, Memory  
from labs.lab2.functions.ConvertNumberType import ConvertNumberType  
  
class TestCalculatorAddition(unittest.TestCase):  
 def setUp(self):  
 self.calc = Calculator()  
  
 def test\_add\_positive\_numbers(self):  
 self.assertEqual(self.calc.Add(5, 10), 15)  
  
 def test\_add\_negative\_numbers(self):  
 self.assertEqual(self.calc.Add(-5, -10), -15)  
  
 def test\_add\_strings(self):  
 with self.assertRaises(ValueError):  
 self.calc.Add("test", 5)  
  
class TestCalculatorSubtraction(unittest.TestCase):  
 def setUp(self):  
 self.calc = Calculator()

**4.** **Приклад роботи:**

**Приклад 1:** Проведення тестування

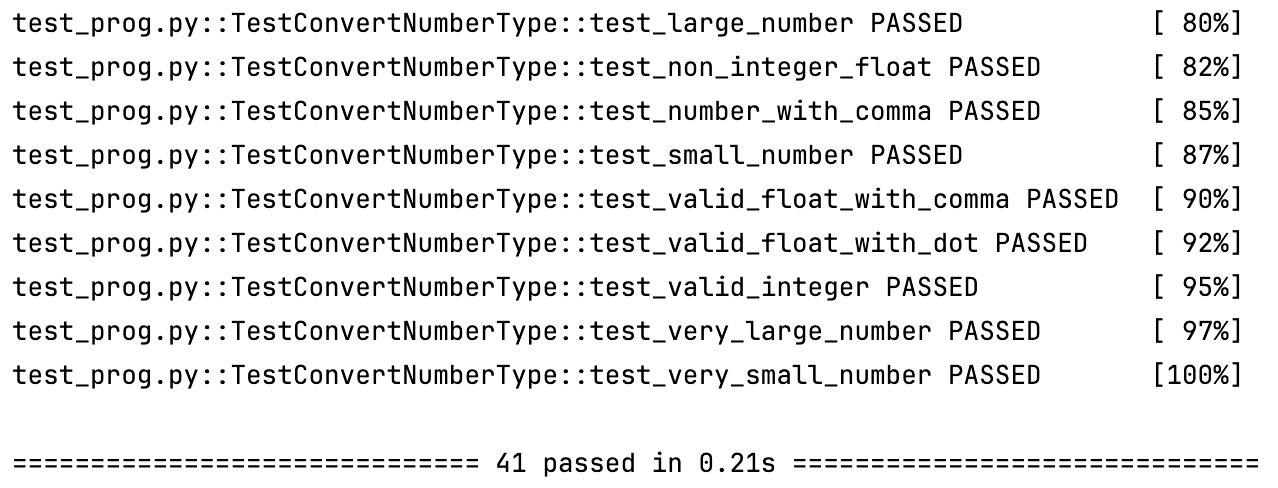


Рис. 1 Відображення процентажу тестування кожної функції

**Висновок:** У ході лабораторної роботи був розроблений калькулятор із підтримкою базових арифметичних операцій, роботи з пам'яттю та конвертації чисел.