**Технології програмування**

**Лабораторна робота №4**

**Мета роботи:** Використовуючи теоретичне підґрунтя про зворотний польський запис розробити програму на вхід якої подається математичний вираз, що має довільний набір операндів, операторів та дужок, на виході програма обчислює результат математичного виразу.

**Завдання:** Використовуючи теоретичне відомості розробити програму яка на вхід отримує математичний вираз з довільною кількістю операндів, операторів та дужок. В першу чергу сформувати послідовність символів у ЗПН. На другому етапі виконання лабораторної роботи вирахувати результат послідовності, що була сформована, використовуючи алгоритм запису математичного виразу у ЗПН.

**Хід роботи**

**is\_number(token)**

Ця функція перевіряє, чи можна токен або значення розглядати як число. Вона намагається перетворити токен у число типу "float" дробове числа. Якщо вона успішно це зробить, то функція поверне значення True, означаючи, що токен є числом. Якщо ж токен не може бути перетворений у число (наприклад, якщо це не число або відсутній числовий формат), то повернеться значення False.

**is\_operator(token)**

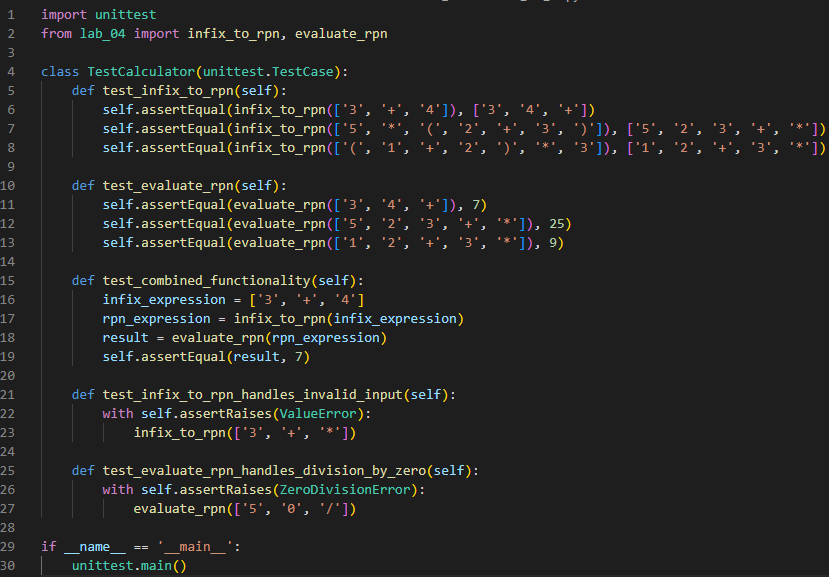
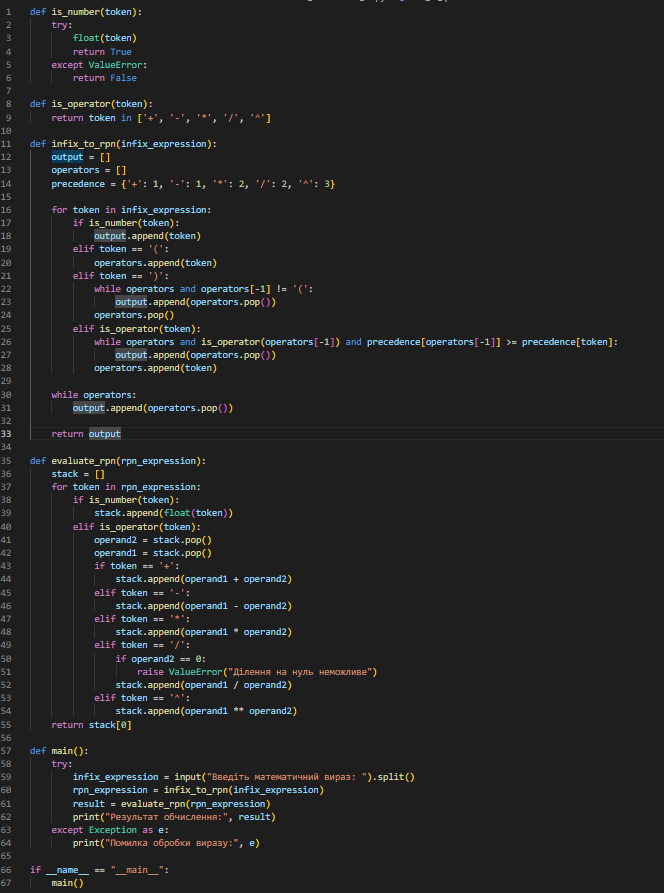
Ця функція перевіряє, чи є токен оператором, якщо порівнювати його зі списком підтримуваних операторів: +, -, \*, /, ^. Якщо токен збігається з будь-яким із цих символів, функція повертає True, показуючи, що це оператор. У випадку, якщо токен не є жодним з цих символів, функція повертає false.

**infix\_to\_rpn(infix\_expression)**

Ця функція перетворює математичний вираз зі звичайної інфіксної форми в особливу форму, яку називають оберненою польською нотацією (RPN). Вона проходить по кожному символу виразу і визначає правильний порядок символів.

**evaluate\_rpn(rpn\_expression)**

Ця функція отримує постфіксний вираз (отриманий після перетворення з інфіксної форми) та обчислює його результат. Вона використовує стек для зберігання чисел і виконання операцій над операндами і операторами у правильному порядку. Кожен оператор застосовується до двох останніх чисел у стеці, отримуючи тим самим результат операції.



**test\_infix\_to\_rpn:** Перевіряє, чи функція ***infix\_to\_rpn*** правильно перетворює інфіксні вирази у постфіксні. Порівнюємо отриманий результат з очікуваним результатом.

**test\_evaluate\_rpn:** Перевіряє, чи функція ***evaluate\_rpn*** правильно обчислює результат постфіксних виразів. Порівнюємо отриманий результат з очікуваним результатом.

**test\_combined\_functionality:** Перевіряє, чи комбіноване використання функцій ***infix\_to\_rpn*** та ***evaluate\_rpn*** дає очікуваний результат.

**test\_infix\_to\_rpn\_handles\_invalid\_input:** Перевіряє, чи функція ***infix\_to\_rpn*** відловлює помилку, якщо отримує некоректний вхідний вираз (наприклад, вираз з оператором без операндів).

**test\_evaluate\_rpn\_handles\_division\_by\_zero:** Перевіряє, чи функція ***evaluate\_rpn*** відловлює помилку, якщо в постфіксному виразі є спроба ділення на нуль.

**Висновок:** Після виконання лабораторної роботи можу зробити висновок що я ще попрактикувався з юніт тестами, а також написав код який обчислює результат мат. виразів.