**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування на мові Python**

Звіт до Лабораторної роботи №4

Зворотний польський запис та обчислення математичного виразу.

**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.

Під час виконання Лабораторної роботи №4 було надано варіанти рішення до наступних задач:

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.

**Зворотний польський запис та обчислення математичного виразу**

Для виконання завдання цієї лабораторної роботи я написав програму lab4.py, яка спочатку перетворює арифметичний вираз, введений користувачем на Зворотній польський запис. Цей запис формується у вигляді списка, і паралельно створюється список стеку операцій. Наступним етапом програма обробляє цей ЗПЗ та стек операцій і виконує обчислення виразу введеного користувачем. Результат виводиться на екран.

Текст програми lab4.py:

import math

operations = {

    '+': 1,

    '-': 1,

    '\*': 2,

    '/': 2,

    '^': 3

}

def is\_operator(expr):

    return expr in operations

def prior(op):

    return operations[op]

def is\_number(expr):

    try:

        float(expr)

        return True

    except ValueError:

        return False

def to\_zpz(expression):

    output = []

    stack = []

    input\_expr = expression.split()

    for expr in input\_expr:

        if is\_number(expr):

            output.append(expr)

        elif expr == '(':

            stack.append(expr)

        elif expr == ')':

            while stack and stack[-1] != '(':

                output.append(stack.pop())

            stack.pop()

        elif is\_operator(expr):

            while (stack and stack[-1] != '(' and

                   prior(stack[-1]) >= prior(expr)):

                output.append(stack.pop())

            stack.append(expr)

    while stack:

        output.append(stack.pop())

    return output

def evaluate\_zpz(zpz\_expression):

    stack = []

    for expr in zpz\_expression:

        if is\_number(expr):

            stack.append(float(expr))

        elif is\_operator(expr):

            b = stack.pop()

            a = stack.pop()

            if expr == '+':

                stack.append(a + b)

            elif expr == '-':

                stack.append(a - b)

            elif expr == '\*':

                stack.append(a \* b)

            elif expr == '/':

                stack.append(a / b)

            elif expr == '^':

                stack.append(a \*\* b)

    return stack[0] if stack else None

def main():

    print("Введіть математичний вираз (числа, оператори і дужки розділені пробілом):")

    expression = input()

    zpz = to\_zpz(expression)

    print("Зворотний польський запис:", zpz)

    result = evaluate\_zpz(zpz)

    print("Результат:", result)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

На початку роботи модуля lab4.py імпортується бібліотека роботи з математичними виразами. Далі ініціалізується словник пріоритетів математичних операцій. Із списку видно що, найвищий пріоритет має операція приведення до степеня ^ пріоритет 3, далі за пріоритетом іде множення та ділення, пріоритет 2 і найнижчий пріоритет =1 у додавання та віднімання.

Наступна функція is\_operator(expr) перевіряє, чи є символ кодом операції (чи входить він у словник operations). Функція повертає логiчне значення True, якщо символ Є кодом операції.

Наступна функція prior(op) визначає пріоритет операції за її кодом із словника operations та повертає числове значення пріоритету (1, 2 або3).

Наступна функція is\_number(expr) визначає, чи Є символ або послідовність символів числом, пробуючи конвертувати це в формат float і повертає True, якщо це число, False – якщо не число.

Наступна функція to\_zpz(expression) перетворює вираз, введений користувачем у форму ЗПЗ та стеку операцій, ЗПЗ та стек мають форму списків. Функція to\_zpz(expression) отримує в якості параметра математичний вираз введений користувачем і перетворює його в форму Зворотного Польського Запису (ЗПЗ) output[] – у форматі списку python

Спочатку в цій функції ініціалізуються порожні списки для ЗПЗ та стеку:

output = [] # Вихідний список

stack = [] # Стек операторів

Потім введений користувачем вираз розділяється на окремі текстові елементи по розділювачу – пробілу, і зберігається у списку input\_expr[]:

input\_expr = expression.split()

В наступному циклі розбирається послідовно зліва направо список input\_expr [] і з його елементів формуються 2 списки: ЗПЗ - output[] та стек stack[]

Якщо поточний елемент списку expr є числом – він додається в список ЗПЗ output[]

Якщо expr=”(“ – вона додається в стек stack[]

Якщо expr=”)“- запускається внутрішній цикл, поки список stack[] не пустий і поки в ньому не знайдено ”(“, із стеку виштовхується (видаляється) поточний елемент і додається в список ЗПЗ output[]. Якщо в стеку знайдена ”(“ – цикл припиняється і дужка видаляється із стеку

Якщо поточний елемент expr є кодом операції – запуск-ся ще 1 внутрішній цикл: поки елементи в стеку не закінчаться і поки попередній елемент стеку не є ”(“ і пріоритет попереднього елемента стеку >= пріоритету поточного елемента expr – поточний елемент виштовхується(видаляється) зі стеку і додається в список ЗПЗ output[]. Після відпрацювання цього внутрішнього циклу до стеку додається поточний елемент: stack.append(expr) – це, як правило код операції

Всі інші елементи expr, що після вищеописаних перевірок залишились у стеку – видаляються зі стеку і додаються в список ЗПЗ output[] внутрішнім циклом:

while stack:

output.append(stack.pop())

В кінці функції to\_zpz(expression) в головну функцію main() повертається вихідний список ЗПЗ output[]

Далі йде функція evaluate\_zpz(zpz\_expression), що обчислює вираз у ЗПЗ. Ця функція отримує як параметр список у формі ЗПЗ, який зробила із введеного виразу функція to\_zpz(expression). Першою командою створюється порожній список stack[].

Далі в циклі перебираються всі елементи expr списку zpz\_expression, порядок перебору елементів ЗПЗ – Зліва -> Направо

Якщо expr – це число, воно конвертується в формат float та додається в стек: stack.append(float(expr))

Якщо expr – код операції – із стеку виштовхується верхній елемент, - це буде число і операнд b математичної операції: b = stack.pop()

Далі із стеку виштовхується наступний елемент, - це буде число і операнд а математичної операції: а = stack.pop()

Далі відповідно коду операції expr розраховується математична операція з операндами а і b, результат операції поміщається в стек, наприклад:

if expr == '+':

stack.append(a + b)

Після відпрацювання циклу по всьому списку ЗПЗ zpz\_expression, останнє число, що залишилося в стеку – це результат математичного виразу.

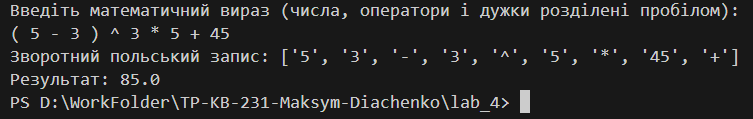
Саме це число – результат виразу, повертається командою:

return stack[0] if stack else None в головну функцію main():

Далі йде функція main():

Спочатку програма запрошує користувача ввести математичний вираз, де всі елементи розділені пробілами, вираз зберігається в змінній expression. Далі введений вираз перетворюється в формат списку ЗПЗ викликом ф-ї: zpz=to\_zpz(expression). Результат перетворення в список ЗПЗ виводиться на екран. Потім по списку ЗПЗ розраховується результат математичної операції rezult=evaluate\_zpz(zpz) Останньою командою в main() на екран виводиться результат математичної операції.

Результат роботи програми lab4.py:



**Висновки.** Під час виконання цієї лабораторної роботи я вивчив алгоритм зворотного польського запису та його застосування для обчислення математичних виразів. Також я створив програму на вхід якої подається математичний вираз, що має довільний набір операндів, операторів та дужок, на виході програма обчислює результат математичного виразу.

Посилання на github:

<https://github.com/diachenkom/TP-KB-231-Maksym-Diachenko>

Знімок екрану з посилання на github:

