**Лабораторна робота №4**

**Тема**: Зворотний польський запис та обчислення математичного виразу.

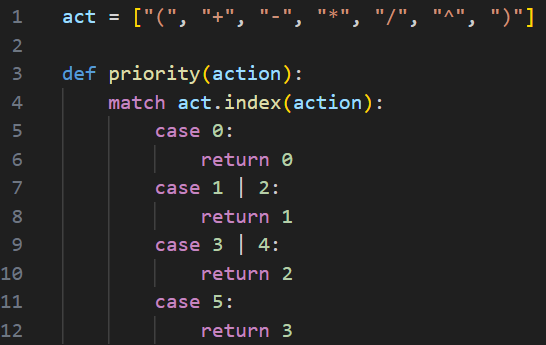
**Мета**: Використовуючи теоретичне підґрунтя про зворотний польський запис розробити програму на вхід якої подається математичний вираз, що має довільний набір операндів, операторів та дужок, на виході програма обчислює результат математичного виразу.

**Завдання до лабораторної роботи**

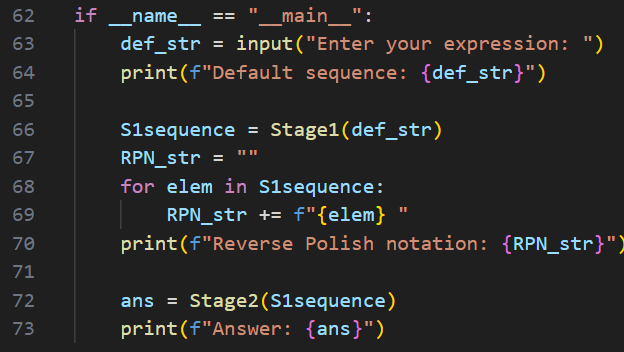
Використовуючи теоретичне відомості розробити програму яка на вхід отримує математичний вираз з довільною кількістю операндів, операторів та дужок. В першу чергу сформувати послідовність символів у ЗПН. На другому етапі виконання лабораторної роботи вирахувати результат послідовності, що була сформована,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

**Хід роботи**

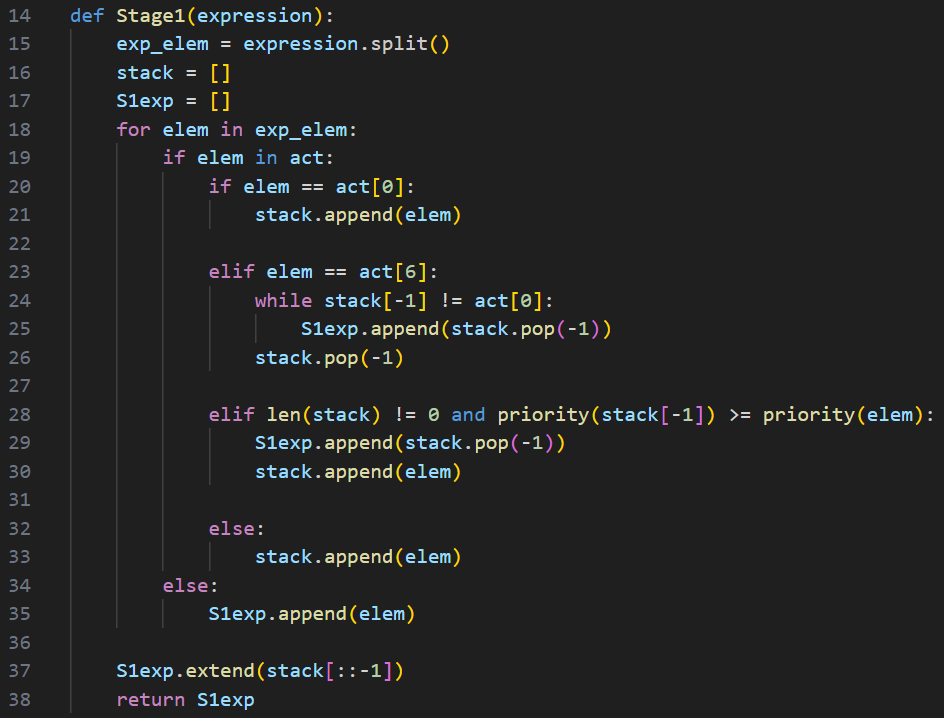
Для початку я створив список дій та функцію для визначення пріоритета дії:



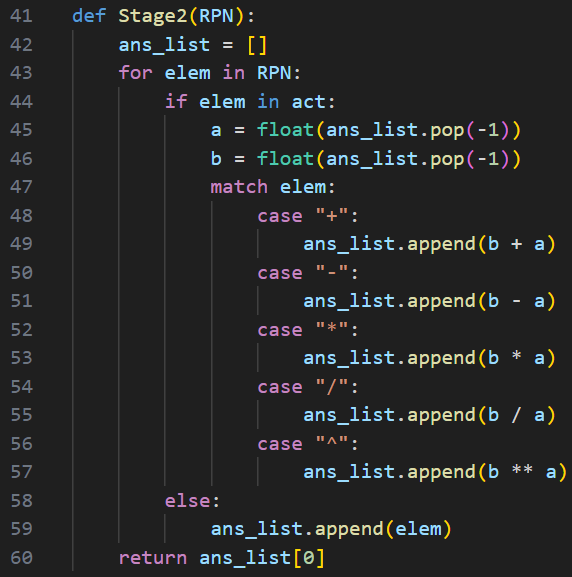
Одразу покажу послідовність виконання програми, щоб було легше зрозуміти структуру, спочатку користувач вводить свій рядок, від друкується, потім виконується функція 1 етапу, результат (список) записується в змінну, після цього за допомогою циклу створюється рядок з пробілами між діями та числами вже готового ЗПЗ, щоб вивести його, останнім йде 2 етап, результат записується в змінну та виводиться користувачу:



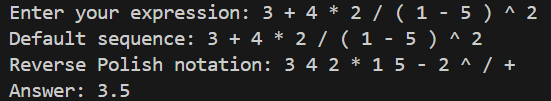
Тепер розглянемо 1 етап: в ньому введений користувачем рядок ділиьтся на елементи списку за пробілами, створюється стек та вихідний вираз 1 стадії, далі йде цикл, що перебирає всі елементи списку з числами та діями, якщо елемента немає в списку дій, то він просто додається до вихідного виразу, якщо ж є, то спочатку перевіряється, чи це відкриваюча дужка, якщо так, то елемент додається в стек. Наступною перевіркою буде закриваюча дужка, якщо це вона, то зі стеку виштовхуються всі елементи з кінці до відкриваючої дужки, інакше буде перевірка на пріоритет останньої дії в стеку, якщо він більший, або такий самий, то цей елемент виштовхується у вихідний список, якщо ні, то елемент просто додається в стек. Після виконання циклу залишаться дії в стеку і їх потрібно буде виштовхнути в вихідний список за порядком з кінця:



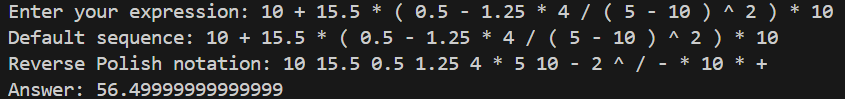
На останок розглянемо 2 етап: вводиться ЗПЗ, створюється список відповіді, та починається цикл, який перебирає всі елементи ЗПЗ, якщо елемент не дія, то він просто записується в список відповіді, якщо ж це дія, то останні 2 числа в списку відповіді видаляються, а їхні значення записуються в змінні, після чого в залежності від поточного елемента виконується операція над цими змінними, та значення цієї операції додається у список відповіді, після завершення циклу залишиться лише 1 елемент у списку відповіді, він і повертається з функції:



Ось приклади вирішення виразу 1 буде вираз з прикладу в завданні, там була відповідь 3.5:



2 виразом буде придуманий мною, я порахував і його відповіддю повинно бути 56.5:



Текст програми:

|  |
| --- |
| act = ["(", "+", "-", "\*", "/", "^", ")"]  def priority(action):      match act.index(action):          case 0:              return 0          case 1 | 2:              return 1          case 3 | 4:              return 2          case 5:              return 3  def Stage1(expression):      exp\_elem = expression.split()      stack = []      S1exp = []      for elem in exp\_elem:          if elem in act:              if elem == act[0]:                  stack.append(elem)              elif elem == act[6]:                  while stack[-1] != act[0]:                      S1exp.append(stack.pop(-1))                  stack.pop(-1)              elif len(stack) != 0 and priority(stack[-1]) >= priority(elem):                  S1exp.append(stack.pop(-1))                  stack.append(elem)              else:                  stack.append(elem)          else:              S1exp.append(elem)      S1exp.extend(stack[::-1])      return S1exp    def Stage2(RPN):      ans\_list = []      for elem in RPN:          if elem in act:              a = float(ans\_list.pop(-1))              b = float(ans\_list.pop(-1))              match elem:                  case "+":                      ans\_list.append(b + a)                  case "-":                      ans\_list.append(b - a)                  case "\*":                      ans\_list.append(b \* a)                  case "/":                      ans\_list.append(b / a)                  case "^":                      ans\_list.append(b \*\* a)          else:              ans\_list.append(elem)      return ans\_list[0]  if \_\_name\_\_ == "\_\_main\_\_":      def\_str = input("Enter your expression: ")      print(f"Default sequence: {def\_str}")      S1sequence = Stage1(def\_str)      RPN\_str = ""      for elem in S1sequence:          RPN\_str += f"{elem} "      print(f"Reverse Polish notation: {RPN\_str}")        ans = Stage2(S1sequence)      print(f"Answer: {ans}") |

**Висновок:** під час виконання роботи я використовуючи теоретичне підґрунтя про зворотний польський запис розробив програму на вхід якої подається математичний вираз, що має довільний набір операндів, операторів та дужок, на виході програма обчислює результат математичного виразу.