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

Лабораторна робота №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.

**Зворотний польський запис**

Звичною формою запису виразів є інфіксна, коли знак бінарної операції записують між позначеннями операндів цієї операції, наприклад, a + b. Розглянемо запис знаків операцій після позначень операндів, тобто постфіксний запис, наприклад, a b +. Такий запис має також назву зворотного польського, бо його запропонував польський логік Ян Лукасевич. Далі словосполучення: «зворотний польський запис» позначатимемо ЗПЗ. Позначення для функції традиційно записують перед аргументами. Природно такий запис назвати префіксним. При описі ЗПЗ переважно обмежуються перетворенням інфіксного запису у ЗПЗ.

Зворотний польський запис має чудові властивості, які перетворюють її на ідеальну проміжну ланку при трансляції коду програми.

Обчислення виразу, записаного в зворотному польському записі, можна проводити шляхом однократного перегляду ЗПЗ.

Зворотний польський запис виразу з арифметичними діями та піднесенням до степеня можна отримати, дотримуючись алгоритму, запропонованого Дейкстpою. Алгоритм отримав назву «сортувальна станція», за подібність його операцій із тим, що відбувається на залізничних сортувальних станціях. Як і алгоритм обчислення ЗПЗ, алгоритм сортувальної станції ґрунтується на стеку. У перетворенні беруть участь дві текстові змінні: вхідний і вихідний рядки. У процесі перетворення використовується стек, що зберігає ще не додані до вихідного рядка операції. Перетворювальна програма читає вхідний рядок послідовно символ за символом (символ – це не обов'язково буква), виконує на кожному кроці деякі дії залежно від того, який символ було прочитано.

**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 виконання завдання:

1. Відкрила ВСкод
2. Прочитала завдання
3. Виконала завдання
4. Затестила
5. Закинула в звіт

**main.py**

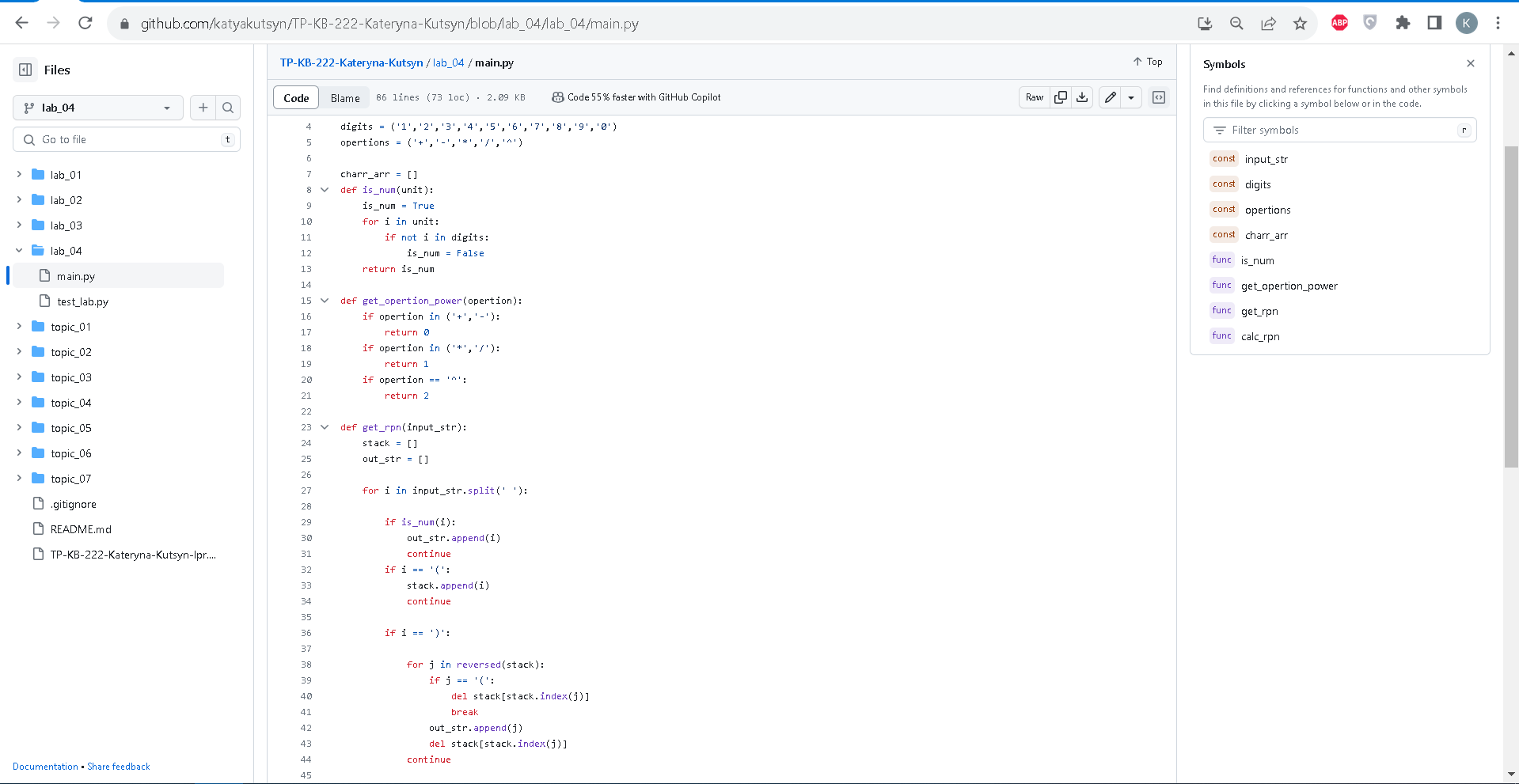
Текст програми:

|  |
| --- |
| input\_str = '( 18 \* 2 ) + ( 64 / 8 ) ^ 4'  digits = ('1','2','3','4','5','6','7','8','9','0')  opertions = ('+','-','\*','/','^')  charr\_arr = []  def is\_num(unit):  is\_num = True  for i in unit:  if not i in digits:  is\_num = False  return is\_num  def get\_opertion\_power(opertion):  if opertion in ('+','-'):  return 0  if opertion in ('\*','/'):  return 1  if opertion == '^':  return 2  def get\_rpn(input\_str):  stack = []  out\_str = []  for i in input\_str.split(' '):    if is\_num(i):  out\_str.append(i)  continue  if i == '(':  stack.append(i)  continue    if i == ')':  for j in reversed(stack):  if j == '(':  del stack[stack.index(j)]  break  out\_str.append(j)  del stack[stack.index(j)]  continue  if i in opertions:  op\_power = get\_opertion\_power(i)  if len(stack) >= 1 and stack[-1] in opertions:  last\_op\_in\_stack\_power = get\_opertion\_power(stack[-1])  if last\_op\_in\_stack\_power >= op\_power:  out\_str.append(stack[-1])  del stack[-1]  stack.append(i)  for i in reversed(stack):  out\_str.append(i)  return out\_str  def calc\_rpn(rpn):  stack = []  for i in rpn:  if is\_num(i):  stack.append(i)  else:  num\_2 = float(stack[-1])  num\_1 = float(stack[-2])  match i:  case '+':  ans = num\_1 + num\_2  case '-':  ans = num\_1 - num\_2  case '/':  ans = num\_1 / num\_2  case '\*':  ans = num\_1 \* num\_2  case '^':  ans = num\_1 \*\* num\_2  del stack[-1]  del stack[-1]  stack.append(ans)  return stack  if \_\_name\_\_ == '\_\_main\_\_':  rpn = get\_rpn(input\_str)  print(rpn)  print(calc\_rpn(rpn)) |

Посилання на github:

<https://github.com/katyakutsyn/TP-KB-222-Kateryna-Kutsyn/blob/lab_04/lab_04/main.py>

Знімок екрану з посилання на github:



**Test\_lab.py**

Текст програми:

|  |
| --- |
| import pytest  import main  import csv  from io import StringIO  import os  def test\_01():    in\_str = '5 \* (3 + 7) / (4 - 2) ^ 2'  rpn = main.get\_rpn(in\_str)  ans = main.calc\_rpn(rpn)  assert ans == [2.5]  def test\_02():    in\_str = '( 100 / 50 - 1 ) / 10'  rpn = main.get\_rpn(in\_str)  ans = main.calc\_rpn(rpn)  assert ans == [0.1]  def test\_03():    in\_str = '12 / 4 + ( 8 / 2 ) ^ 2'  rpn = main.get\_rpn(in\_str)  ans = main.calc\_rpn(rpn)  assert ans == [18]    def test\_04():    in\_str = '9 \* (6 - 2) / (5 + 5) ^ 2'  rpn = main.get\_rpn(in\_str)  ans = main.calc\_rpn(rpn)  assert ans == [0.09]  def test\_05():    in\_str = '( 20 / 10 - 2 ) \* 3'  rpn = main.get\_rpn(in\_str)  ans = main.calc\_rpn(rpn)  assert ans == [-3.0] |

Посилання на github:

<https://github.com/katyakutsyn/TP-KB-222-Kateryna-Kutsyn/blob/lab_04/lab_04/test_lab.py>

Знімок екрану з посилання на github:

