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

Звіт до Теми №1

Функції та змінні

Під час виконання практичного завдання до Теми №1 було надано варіанти рішення до наступних задач:

1) Обернути рядок в зворотному порядку.

2) Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().

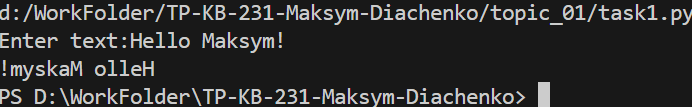
3) Написати функцію пошуку дискримінанту квадратного рівняння.

**Обернення рядка**

Необхідно рядок, наприклад "Hello Maksym!" перетворити наступним чином "!myskaM olleH".

Хід виконання завдання:

У цьому завданні я використав функцію, яка обертає вміст рядка.



Текст програми:

def reverse\_string(text):

    return text[::-1]

text = input("Enter text:")

reversed\_text = reverse\_string(text)

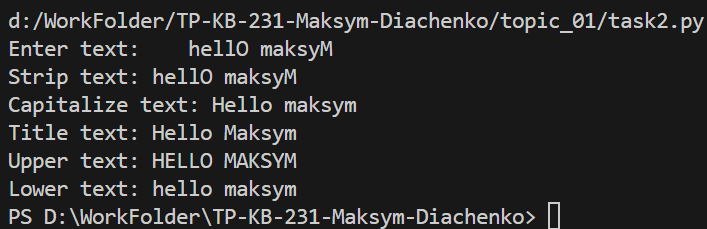
print(reversed\_text)

**Тестування функцій, що працюють з рядками**

Необхідно виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().

Хід виконання завдання:

У цьому завданні я використав вище наведені функції.



Функція Strip прибирає пробіли по краям рядка.

Функція Capitalize робить першу букву першого слова в рядку велику, а всі інші маленькими.

Функція Tittle перетворює регістри перших букв усіх слів рядка великими.

Функція Upper перетворює регістри всіх букв усіх слів рядка великими.

Функція Lower перетворює регістри всіх букв усіх слів рядка маленькими.

Текст програми:

text = input("Enter text:")

text1=text.strip()

print("Strip text: " + text1)

text2=text1.capitalize()

print("Capitalize text: " + text2)

text3=text2.title()

print("Title text: " + text3)

text4=text3.upper()

print("Upper text: " + text4)

text5=text4.lower()

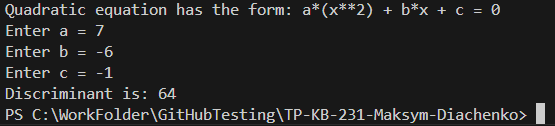
print("Lower text: " + text5)

**Написання функції пошуку дискримінанту квадратного рівняння**

Необхідно написати функцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:

У цьому завданні я написав функцію пошуку дискримінанту квадратного рівняння.



Текст програми:

def discr(a, b, c):

    return b\*\*2 - 4\*a\*c

print("Quadratic equation has the form: a\*(x\*\*2) + b\*x + c = 0")

a = int(input("Enter a = "))

b = int(input("Enter b = "))

c = int(input("Enter c = "))

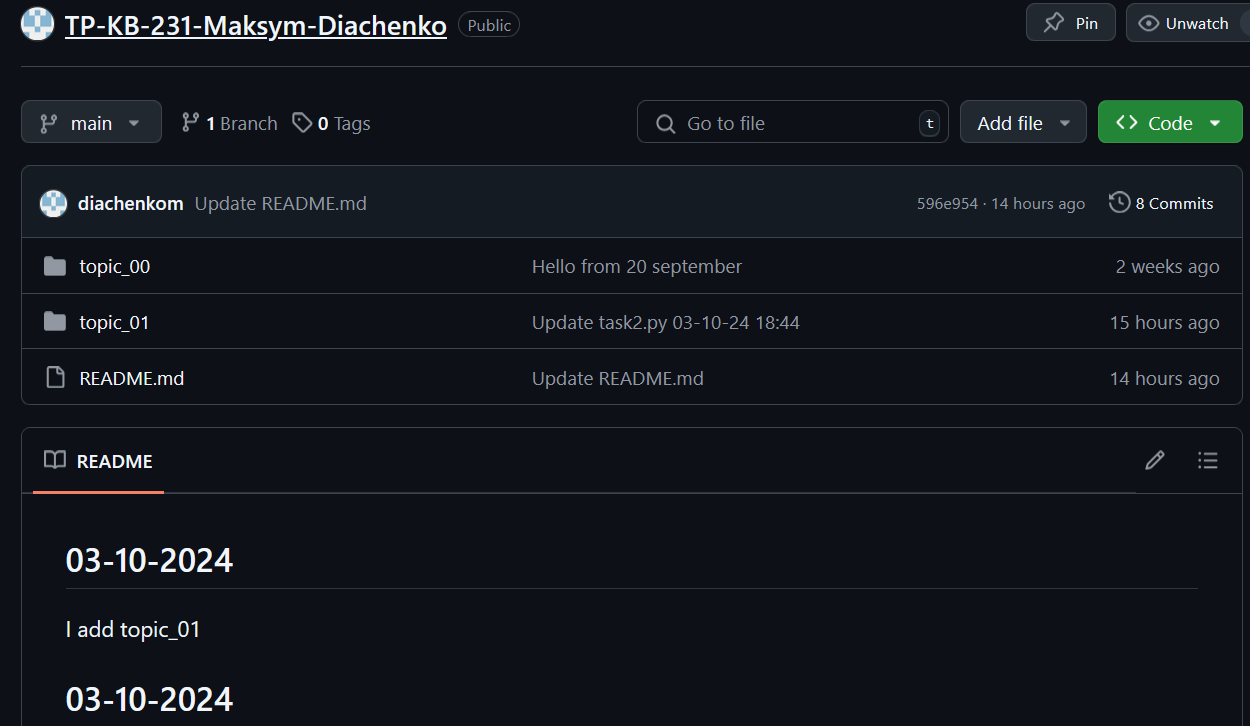
d = discr(a, b, c)

print("Discriminant is: " + str(d))

Посилання на github:

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

Знімок екрану з посилання на github:



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

Звіт до Теми №2

Умовні переходи

Під час виконання практичного завдання до Теми №2 було надано варіанти рішення до наступних задач:

1) Написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи.

2) Написати програму калькулятор використовуючи if else конструкцію. Кожна операція має бути виконана в окремій функції.

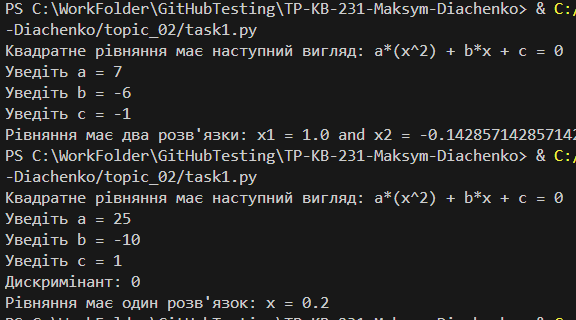
3) Написати програму калькулятор використовуючи match конструкцію. Кожна операція має бути виконана в окремій функції.

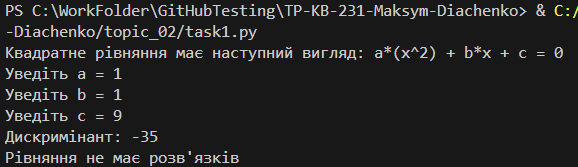
**Функція пошуку коренів квадратного рівняння**

Необхідно написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи.

Хід виконання завдання:

У цьому завданні я, використовуючи функцію розрахунку дискримінанту з попередньої теми, написав функцію пошуку коренів квадратного рівняння з використанням умовних переходів.





Текст програми:

import math

print("Квадратне рівняння має наступний вигляд: a\*(x^2) + b\*x + c = 0")

a = int(input("Уведіть a = "))

b = int(input("Уведіть b = "))

c = int(input("Уведіть c = "))

def discr(a, b, c):

    return b\*\*2 - 4\*a\*c

def equation\_solution(a, b, c):

    d = discr(a, b, c)

    print("Дискримінант: " + str(d))

    if d > 0:

        x1 = (-(b) + math.sqrt(d)) / (2\*a)

        x2 = (-(b) - math.sqrt(d)) / (2\*a)

        return f"Рівняння має два розв'язки: x1 = {x1} and x2 = {x2}"

    elif d == 0:

        x1 = (-b) / (2\*a)

        return f"Рівняння має один розв'язок: x = {x1}"

    else:

        return "Рівняння не має розв'язків"

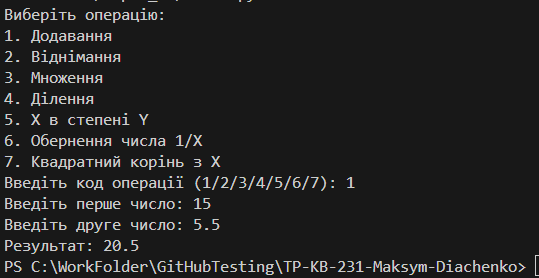
print (equation\_solution(a, b, c))

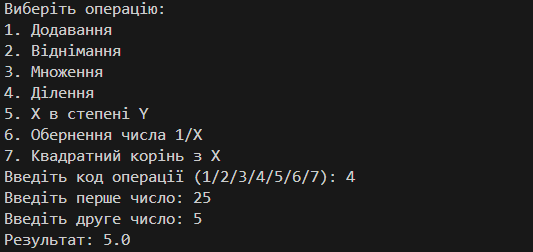
**Программа калькулятор з if else конструкцію**

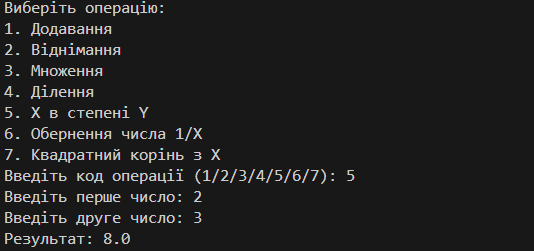
Необхідно написати програму калькулятор використовуючи if else конструкцію.

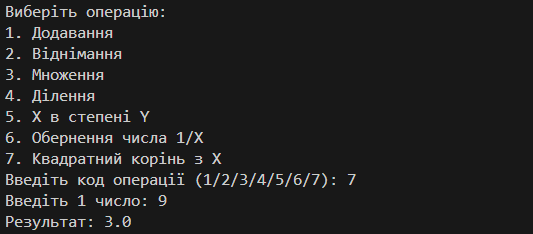
Хід виконання завдання:

У цьому завданні я написав програму калькулятор, що використовує if else конструкції, оформивши кожну операцію в окремій функції.









Текст програми:

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b != 0:

        return a / b

    else:

        return "Помилка: Ділити на нуль не можна!"

# X в степені Y

def step(a,b):

    return a\*\*b

def calculator():

    print("Виберіть операцію:")

    print("1. Додавання")

    print("2. Віднімання")

    print("3. Множення")

    print("4. Ділення")

    print("5. X в степені Y")

    print("6. Обернення числа 1/Х")

    print("7. Квадратний корінь з Х")

    cod = input("Введіть код операції (1/2/3/4/5/6/7): ")

    if (int(cod)<1 or int(cod)>7):

        print("Неправильний ввiд Kоду операції")

        return

    elif (int(cod)>=1 and int(cod)<=5):

        num1 = float(input("Введіть перше число: "))

        num2 = float(input("Введіть друге число: "))

    elif (int(cod)>=6 and int(cod)<=7):

        num1 = float(input("Введіть 1 число: "))

    if cod == '1':

        print(f"Результат: {add(num1, num2)}")

    elif cod == '2':

        print(f"Результат: {subtract(num1, num2)}")

    elif cod == '3':

        print(f"Результат: {multiply(num1, num2)}")

    elif cod == '4':

        print(f"Результат: {divide(num1, num2)}")

    elif cod == '5':

        print(f"Результат: {step(num1, num2)}")

    elif cod == '6':

        print(f"Результат: {1/num1}")

    elif cod == '7':

        print(f"Результат: {num1\*\*0.5}")

    else:

        print("Неправильний код операції.")

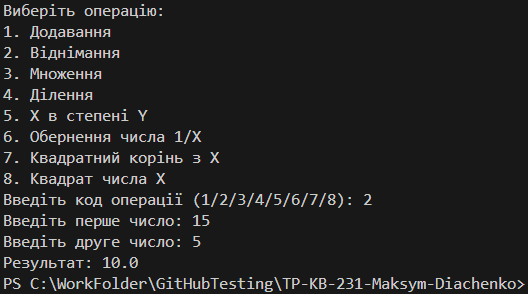
calculator()

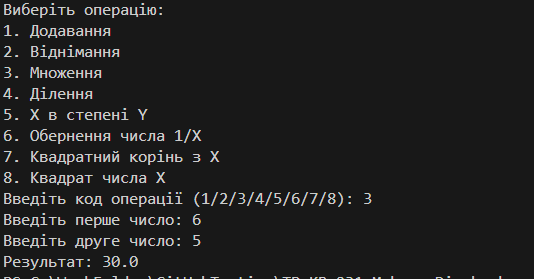
**Програма калькулятор з match конструкцією**

Необхідно написати програму калькулятор використовуючи match конструкцію. Кожна операція має бути виконана в окремій функції.

Хід виконання завдання:

У цьому завданні я написав програму калькулятор, аналогічно до попередньої. Але в цьому завданні вже з використанням match конструкції, оформивши кожну операцію в окремій функції.





Текст програми:

def add(a, b):

    return a + b

def subtract(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b != 0:

        return a / b

    else:

        return "Помилка: Ділити на нуль не можна!"

def step(a,b):

    return a\*\*b

def square(a):

    return a\*\*2

def calculator():

    print("Виберіть операцію:")

    print("1. Додавання")

    print("2. Віднімання")

    print("3. Множення")

    print("4. Ділення")

    print("5. X в степені Y")

    print("6. Обернення числа 1/Х")

    print("7. Квадратний корінь з Х")

    print("8. Квадрат числа Х")

    cod = input("Введіть код операції (1/2/3/4/5/6/7/8): ")

    if (int(cod)<1 or int(cod)>8):

        print("Неправильний ввiд Kоду операції")

        return

    elif (int(cod)>=1 and int(cod)<=5):

        num1 = float(input("Введіть перше число: "))

        num2 = float(input("Введіть друге число: "))

    elif (int(cod)>=6 and int(cod)<=8):

        num1 = float(input("Введіть 1 число: "))

    match cod:

        case '1':

            print(f"Результат: {add(num1, num2)}")

        case '2':

            print(f"Результат: {subtract(num1, num2)}")

        case '3':

            print(f"Результат: {multiply(num1, num2)}")

        case '4':

            print(f"Результат: {divide(num1, num2)}")

        case '5':

            print(f"Результат: {step(num1, num2)}")

        case '6':

            print(f"Результат: {1/num1}")

        case '7':

            print(f"Результат: {num1\*\*0.5}")

        case '8':

            print(f"Результат: {square(num1)}")

        case \_:

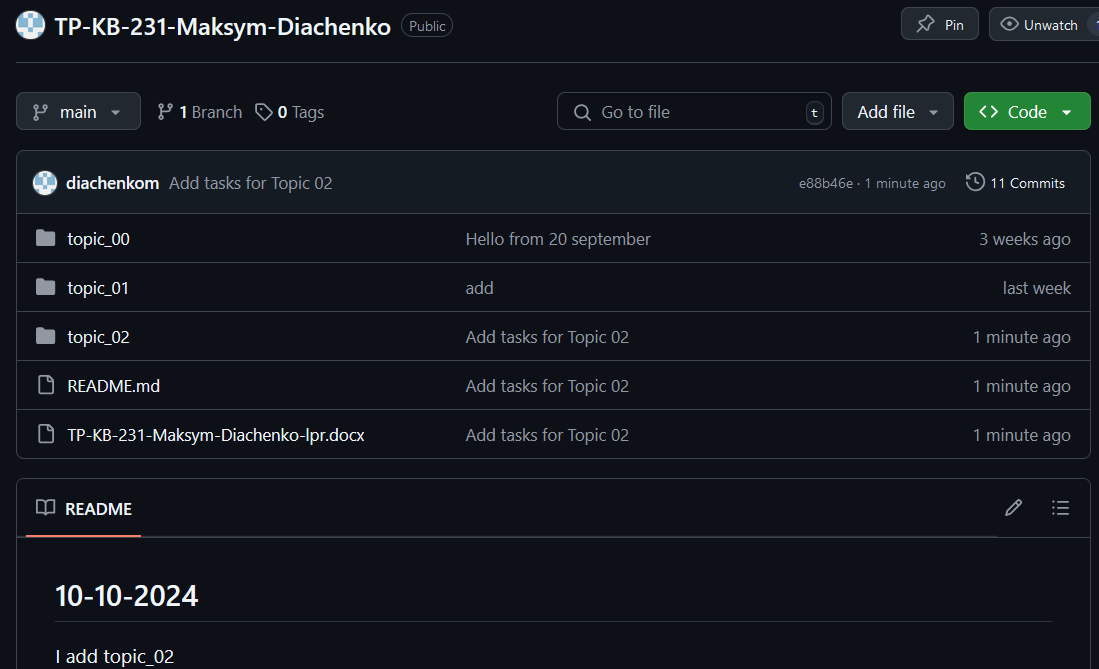
            print("Неправильний код операції.")

calculator()

Посилання на github:

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

Знімок екрану з посилання на github:



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

Звіт до Теми №3

Цикли

Під час виконання практичного завдання до Теми №3 було надано варіанти рішення до наступних задач:

1) Написати програму калькулятор з постійними запитами на введення нових даних та операцій. За основу взяти програму калькулятор з попередньої теми. Реалізувати механізм завершення програми після отримання відповідної команди.

2) Написати програму тестування функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy()

3) Написати програму тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items()

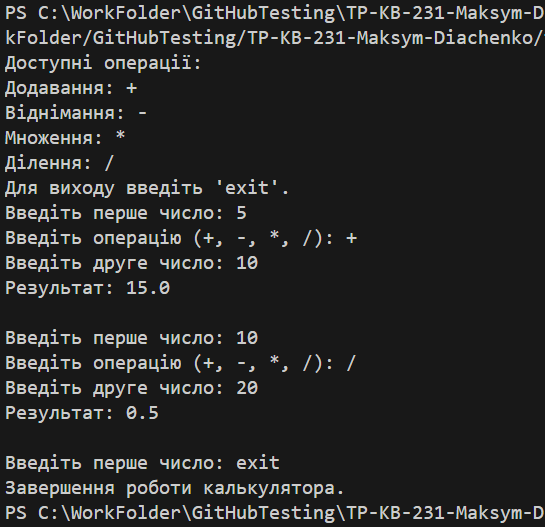
4) Маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.

**Програма калькулятор з постійними запитами на введення нових даних та операцій**

Необхідно написати програму калькулятор з постійними запитами на введення нових даних та операцій.

Хід виконання завдання:

У цьому завданні я, використовуючи цикл while, написав програму калькулятор з постійними запитами на введення нових даних та операцій.



Текст програми:

def add(x, y):

    return x + y

def subtract(x, y):

    return x - y

def multiply(x, y):

    return x \* y

def divide(x, y):

    if y == 0:

        return "Помилка: ділення на нуль не можливе"

    return x / y

def calculator():

    print("Доступні операції:")

    print("Додавання: +")

    print("Віднімання: -")

    print("Множення: \*")

    print("Ділення: /")

    print("Для виходу введіть 'exit'.")

    while True:

        # Запит першого числа

        num1 = input("Введіть перше число: ")

        if num1.lower() in ['exit']:

            print("Завершення роботи калькулятора.")

            break

        try:

            num1 = float(num1)

        except ValueError:

            print("Некоректне число. Спробуйте ще раз.")

            continue

        # Запит операції

        operation = input("Введіть операцію (+, -, \*, /): ")

        if operation.lower() in ['exit']:

            print("Завершення роботи калькулятора.")

            break

        if operation not in ['+', '-', '\*', '/']:

            print("Некоректна операція. Спробуйте ще раз.")

            continue

        # Запит другого числа

        num2 = input("Введіть друге число: ")

        if num2.lower() in ['exit']:

            print("Завершення роботи калькулятора.")

            break

        try:

            num2 = float(num2)

        except ValueError:

            print("Некоректне число. Спробуйте ще раз.")

            continue

        # Виконання обчислення

        if operation == '+':

            result = add(num1, num2)

        elif operation == '-':

            result = subtract(num1, num2)

        elif operation == '\*':

            result = multiply(num1, num2)

        elif operation == '/':

            result = divide(num1, num2)

        print(f"Результат: {result}\n")

calculator()

**Програма тестування функцій списків**

Необхідно написати програму тестування функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy()

Хід виконання завдання:

У цьому завданні я написав програму тестування функцій списків extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy() та з’ясував призначення кожної із цих функцій.

Функція append() виконує додавання елементу в кінець списку.

Функція extend() виконує додавання всіх елементів одного списку до другого.

Функція remove() виконує видалення заданого елементу списку.

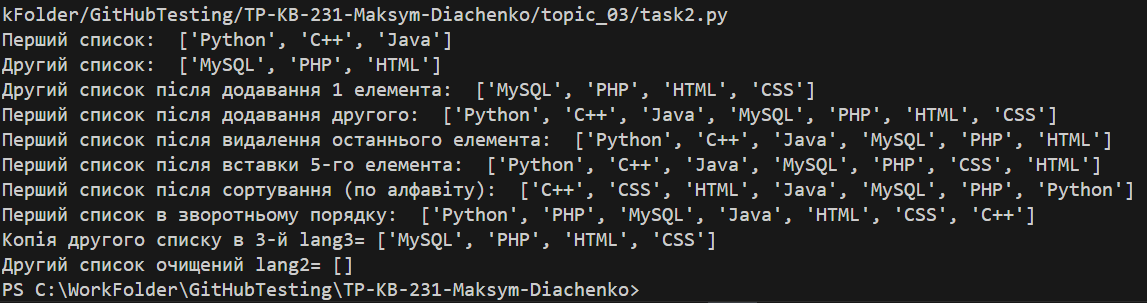
Функція insert() виконує вставку заданого елемента списку у вказану позицію.

Функція sort() виконує сортування всіх елементів списку за алфавітом.

Функція reverse() виконує сортування в зворотньому порядку всіх елементів списку.

Функція copy() робить копію списку.

Функція clear() видаляє всі елементи зі списку.



Текст програми:

lang1 = ["Python", "C++", "Java"]

lang2 = ["MySQL", "PHP", "HTML"]

print("Перший список: ", lang1)

print("Другий список: ", lang2)

# Додавання елементу в кінець списку

lang2.append("CSS")

print("Другий список після додавання 1 елемента: ", lang2)

# Додавання всіх елементів одного списку до другого

lang1.extend(lang2)

print("Перший список після додавання другого: ", lang1)

# Видалення останнього елементу списку

lang1.remove("CSS")

print("Перший список після видалення останнього елемента: ", lang1)

# Вставка передостаннього елементу списку

lang1.insert(5,"CSS")

print("Перший список після вставки 5-го елемента: ", lang1)

# Сортування всіх елементів списку

lang1.sort()

print("Перший список після сортування (по алфавіту): ", lang1)

# Реверс всіх елементів списку

lang1.reverse()

print("Перший список в зворотньому порядку: ", lang1)

# Копія 2-го списку в 3-й список

lang3 = lang2.copy()

print("Копія другого списку в 3-й lang3=", lang3)

# Очищення 2-го списку

lang2.clear()

print("Другий список очищений lang2=", lang2)

**Програма тестування функцій словників**

Необхідно написати програму тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items().

Хід виконання завдання:

У цьому завданні я написав програму тестування функцій словників: update(), del(), clear(), keys(), values(), items() та з’ясував призначення кожної із цих функцій.

Функція update () виконує оновлення елементу словника.

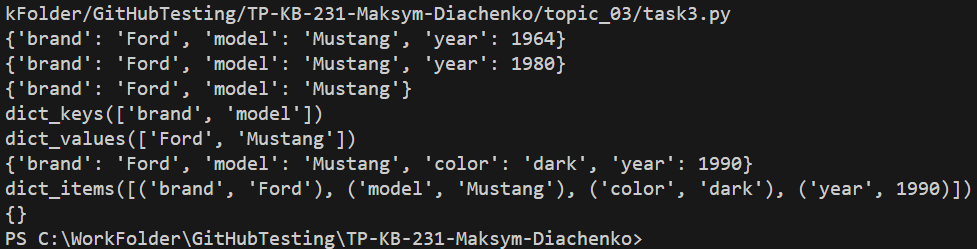
Функція del () виконує видалення елементу словника за вказаним ключем.

Функція clear () виконує очищення всіх елементів словника.

Функція keys () виконує виведення на екран списку усіх ключів словника.

Функція values () виконує виведення на екран списку усіх значень словника.

Функція items () виконує виведення на екран списку усіх елементів словника.



Текст програми:

# Словник для прикладу

my\_dict = {

  "brand": "Ford",

  "model": "Mustang",

  "year": 1964

}

print(my\_dict)

# Оновлення елемента словника

my\_dict.update({"year": 1980})

print(my\_dict)

# Видалення елемента словника по вказаному ключу

del my\_dict["year"]

print(my\_dict)

# Список усіх ключів словника

my\_keys = my\_dict.keys()

print(my\_keys)

# Список усіх значень словника

my\_values = my\_dict.values()

print(my\_values)

# Додати новi елемент до словника

my\_dict["color"] = "dark"

my\_dict["year"] = 1990

print(my\_dict)

# Отримати список усіх елементів словника

my\_items=my\_dict.items()

print(my\_items)

# Очищення усіх елементів словника

my\_dict.clear()

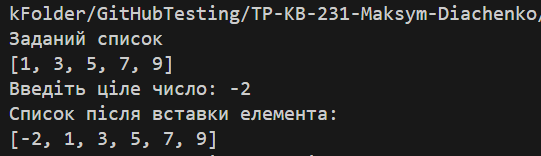
print(my\_dict)

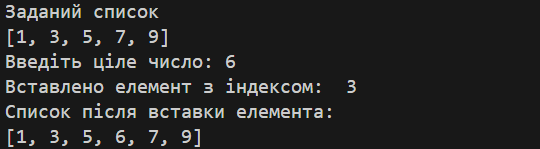
**Функція пошуку позиції для вставки нового елементу в список**

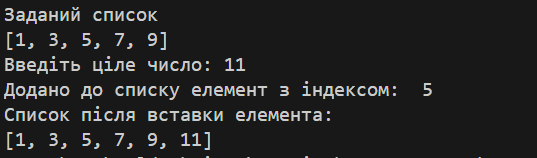
Необхідно маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.

Хід виконання завдання:

У цьому завданні я написав функцію пошуку позиції для вставки нового елементу у відсортований список.







Текст програми:

list1 = [1, 3, 5, 7, 9]

print("Заданий список")

print(list1)

num = int(input("Введіть ціле число: "))

def find\_position():

    for i in range(0, len(list1)-1):

        if num < list1[i]:

            list1.insert(0, num)

            print("Вставлено елемент з індексом: 0")

            break

        if num == list1[i]:

            list1.insert(i, num)

            print("Вставлено елемент з індексом: ",i)

            break

        if (list1[i] < num) and (num < list1[i+1]):

            list1.insert(i+1, num)

            print("Вставлено елемент з індексом: ",i+1)

            break

    else:

        list1.append(num)

        print("Додано до списку елемент з індексом: ",len(list1)-1)

    print("Cписок після вставки елемента:")

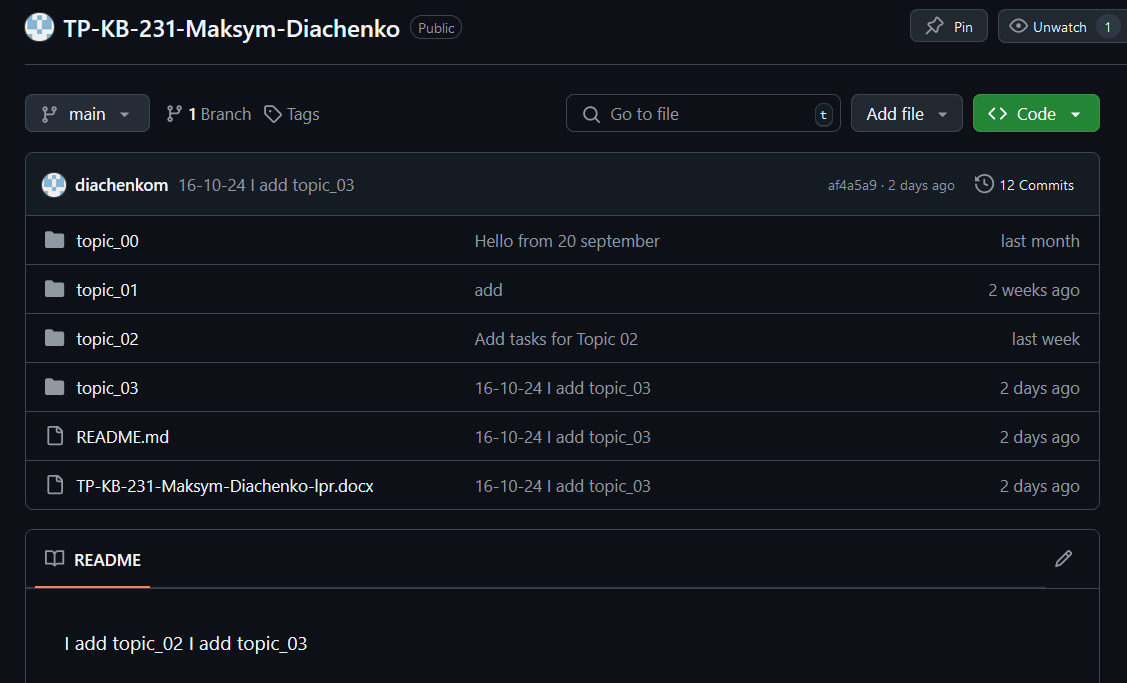
    print(list1)

find\_position()

Посилання на github:

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

Знімок екрану з посилання на github:



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

Звіт до Теми №4

Виняткові ситуації

Під час виконання практичного завдання до Теми №4 було надано варіанти рішення до наступних задач:

1) Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.

2) Розширити функцію ділення обробкою виняткової ситуації ділення на нуль.

**Функція, що обробляє виняткові ситуації**

Необхідно розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.

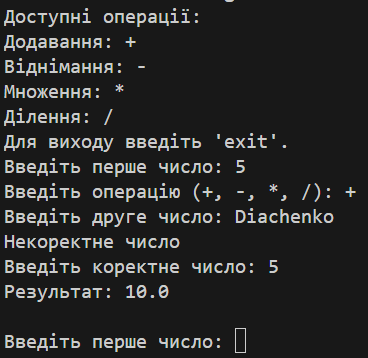
Хід виконання завдання:

У цьому завданні я створив функцію CheckValue(), яка перевіряє коректність вводу чисел користувачем. Перевірка здійснюється за допомогою конструкції Try-Exсept після спроби конвертації строки з цифрами у число з плаваючою точкою. Конструкція Try-Exсept працює в циклі.

Після частини Try здійснюється спроба конвертації строки з цифрами у число з плаваючою точкою.

Якщо конвертація успішна – то відбувається вихід із циклу та повернення сконвертованого числа в команду яка викликала цю функцію.

Якщо при конвертації виникає помилка ValueError – ти видається повідомлення про некоректне число запит повторного вводу цього числа користувачем. Після цього цикл повторюється до тих пір поки користувач не введе коректне число.



Текст функції:

def CheckValue(num):

    while True:

        try:

            num = float(num)

            break

        except ValueError:

            print("Некоректне число")

            num = input("Введіть коректне число: ")

    return num

**Функція ділення з обробкою виняткової ситуації ділення на нуль**

Необхідно розширити функцію ділення обробкою виняткової ситуації ділення на нуль.

Хід виконання завдання:

У цьому завданні я доповнив функцію Divide() обробкою виняткової ситуації ділення на нуль.

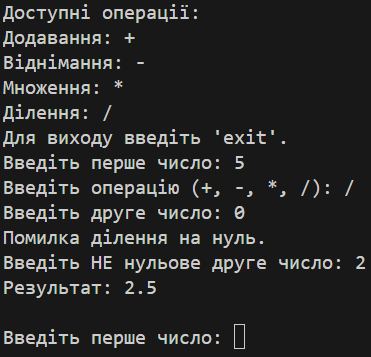
Дана функція отримує в якості параметра два числа «x» та «y», а повертає результат операції «x/y».

В циклі відбувається спроба ділення «х» на «у». Після частини Try: Якщо у=0 – то виникає помилка ділення на нуль «ZeroDivisionError».

Якщо не дорівнює – то відбувається ділення, вихід із циклу та повернення результату ділення в процедуру з якої була викликана функція.

Якщо виникає помилка ділення на нуль «ZeroDivisionError» - то видається повідомлення про помилку, виводиться запит повторного воду дільника «у», після вводу виконується перевірка коректності вводу числа «у» за допомогою функції CheckValue().

Після цього цикл продовжується.



Текст функції:

def divide(x, y):

    while True:

        try:

            a = x / y

            break

        except ZeroDivisionError:

            print("Помилка ділення на нуль.")

            y = input("Введіть НЕ нульове друге число: ")

            y = CheckValue(y)

            continue

    return a

Текст усієї програми:

def add(x, y):

    return x + y

def subtract(x, y):

    return x - y

def multiply(x, y):

    return x \* y

def divide(x, y):

    while True:

        try:

            a = x / y

            break

        except ZeroDivisionError:

            print("Помилка ділення на нуль.")

            y = input("Введіть НЕ нульове друге число: ")

            y = CheckValue(y)

            continue

    return a

def CheckValue(num):

    while True:

        try:

            num = float(num)

            break

        except ValueError:

            print("Некоректне число")

            num = input("Введіть коректне число: ")

    return num

def calculator():

    print("Доступні операції:")

    print("Додавання: +")

    print("Віднімання: -")

    print("Множення: \*")

    print("Ділення: /")

    print("Для виходу введіть 'exit'.")

    while True:

        num1 = input("Введіть перше число: ")

        if num1.lower() in ['exit']:

            print("Завершення роботи калькулятора.")

            break

        else:

            num1=CheckValue(num1)

        operation = input("Введіть операцію (+, -, \*, /): ")

        if operation.lower() in ['exit']:

            print("Завершення роботи калькулятора.")

            break

        if operation not in ['+', '-', '\*', '/']:

            print("Некоректна операція. Спробуйте ще раз.")

            continue

        num2 = input("Введіть друге число: ")

        if num2.lower() in ['exit']:

            print("Завершення роботи калькулятора.")

            break

        else:

            num2=CheckValue(num2)

        if operation == '+':

            result = add(num1, num2)

        elif operation == '-':

            result = subtract(num1, num2)

        elif operation == '\*':

            result = multiply(num1, num2)

        elif operation == '/':

            result = divide(num1, num2)

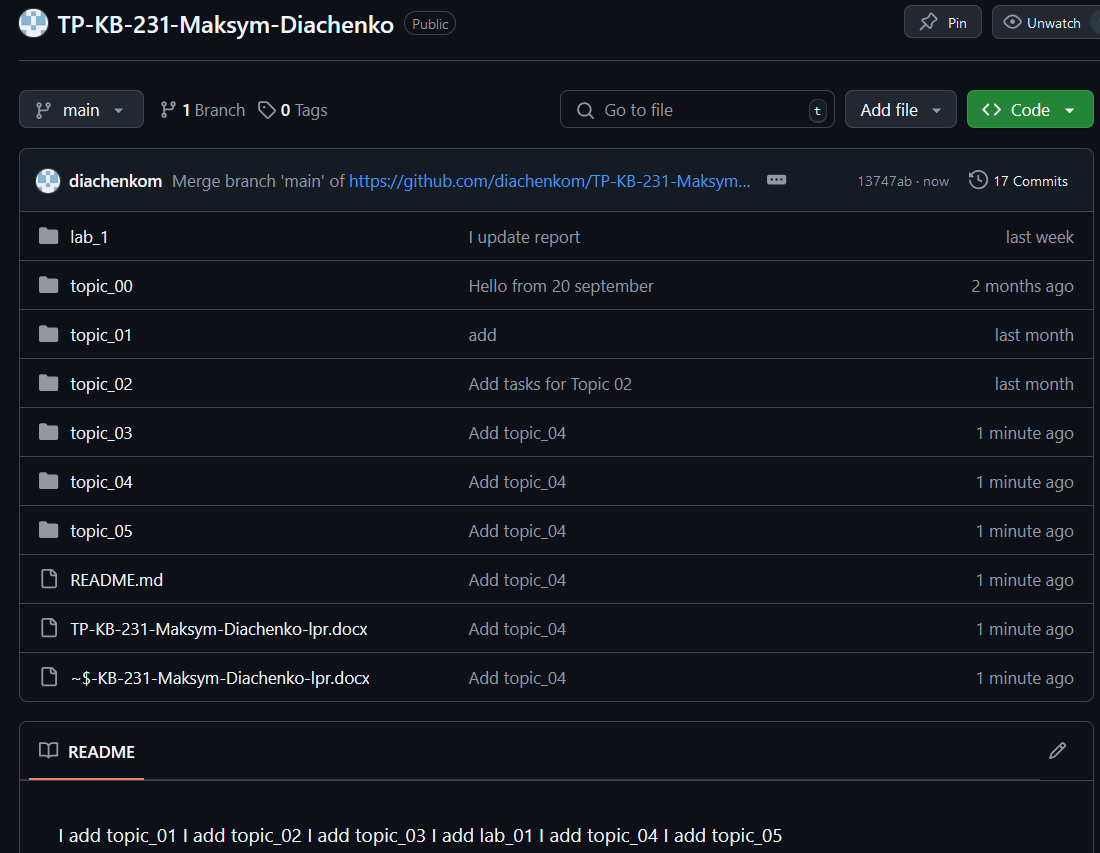
        print(f"Результат: {result}\n")

calculator()

Посилання на github:

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

Знімок екрану з посилання на github:



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

Звіт до Теми №5

Бібліотеки

Під час виконання практичного завдання до Теми №5 було надано варіанти рішення до наступних задач:

1) Гра з комп’ютером: камінь, ножиці, папір.

2) Програма конвертування іноземної валюти в українську гривню.

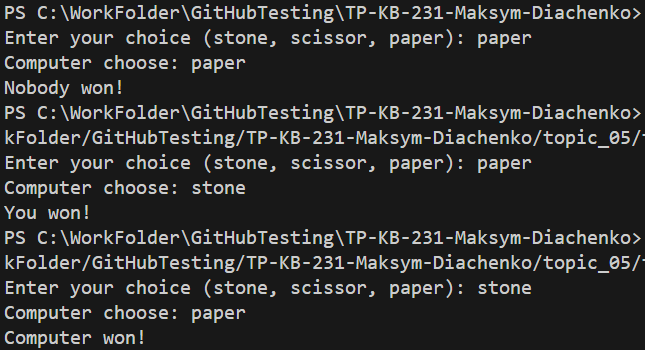
3) Використання модулів для програми калькулятор.

**Гра камінь, ножиці, папір.**

Необхідно зробити програму за допомогою якої можна грати з комп’ютером у гру: камінь, ножиці, папір. Програма виконує запит від користувача на введення одного із значень ["stone", "scissor", "paper"]. Наступним кроком, використовуючи модуль random, програма у випадковому порядку вибирає одне із значень ["stone", "scissor", "paper"]. В залежності від умови, що камінь перемагає ножиці, ножиці перемагають папір, а папір перемагає камінь визначити переможця.

Хід виконання завдання:

У цьому завданні я створив гру з комп’ютером: камінь, ножиці, папір.



Перша команда задає список варіантів вибору для РС та користувача.

Далі запит вводу користувача.

Якщо користувач увів некоректне слово – то друкується повідомлення про помилку та завершення роботи програми.

Якщо ввід користувача коректний – генерується випадковий вибір комп’ютера.

Якщо вибір користувача та РС співпадають – то друкується повідомлення про нічию.

Далі йде перелік комбінацій при яких користувач виграє в комп’ютера.

Якщо так – друкується повідомлення про виграш користувача.

Текст програми:

import random

options = ["stone", "scissor", "paper"]

my\_option = input("Enter your choice (stone, scissor, paper): ")

if my\_option not in options:

    print("Wrong value, try again.")

else:

    pc\_option = random.choice(options)

    print(f"Computer choose: {pc\_option}")

    if my\_option == pc\_option:

        print("Nobody won!")

    elif (my\_option == "stone" and pc\_option == "scissor") or (my\_option == "scissor" and pc\_option == "paper") or (my\_option == "paper" and pc\_option == "stone"):

        print("You won!")

    else:

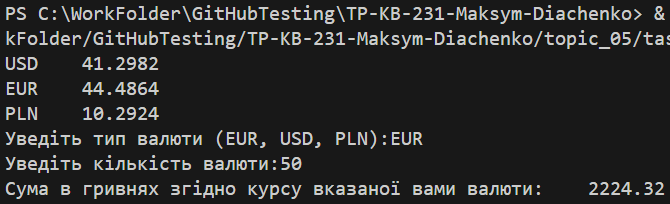
        print("Computer won!")

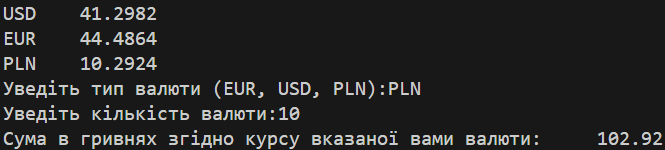
**Програма конвертування іноземної валюти в українську гривню**

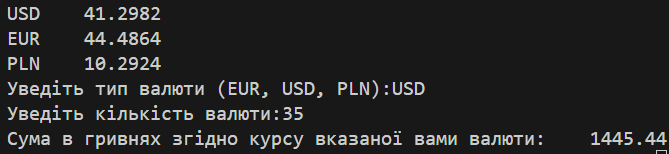
Необхідно створити програму для конвертування іноземної валюти в українську гривню. Для отримання актуальних курсів валют необхідно використовувати API НБУ та модуль, що надає можливість виконувати запити до сторонніх сервісів requests. Достатня умова роботи – можливість конвертації для трьох іноземних валют EUR, USD, PLN. Користувачу надається можливість введення кількості та типу валюти, результат роботи програми – конвертоване значення в українських гривнях.

Хід виконання завдання:

У цьому завданні я створив програму для конвертування іноземної валюти в українську гривню.







Спочатку імпортується бібліотека requests.

Далі змінна отримає перелік курсів валют із вказаного сайту. У списку вказані лише ті валюти які можна вводити.

Наступний цикл виводить список валют та їх курсів, які може вводити користувач.

Наступний цикл виконує введення типу валюти та перевірку правильності вводу.

Наступний цикл виконує введення та перевірку кількості обраної валюти.

Останній цикл шукає в списку всіх валют строку із типом валюти, вказаної користувачем, обраховує суму в гривнях та виводить її на екран з двома знаками після коми.

Текст програми:

import requests

response = requests.get("https://bank.gov.ua/NBUStatService/v1/statdirectory/exchangenew?json")

currency\_all = ["EUR", "USD", "PLN"]

for elem in response.json():

    if elem['cc'] in currency\_all:

        print(elem['cc'], "  ", elem['rate'])

while True:

    currency = input("Уведіть тип валюти (EUR, USD, PLN):")

    currency = currency.upper()

    if currency not in currency\_all:

        print("Неправильно введений тип валюти. Уведіть EUR, USD або PLN: ")

        continue

    else:

        break

while True:

    amount = input("Уведіть кількість валюти:")

    try:

        amount=float(amount)

        if amount <= 0:

            print("Неправильне значення. Уведіть ненульове додатне число")

            continue

        else:

            break

    except ValueError:

        print("Неправильне значення. Уведіть додатне число")

        continue

for elem in response.json():

    if elem['cc'] == currency:

        sum = elem['rate'] \* amount

        print(f"Сума в гривнях згідно курсу вказаної вами валюти: {sum:10.2f}")

**Використання модулів для програми калькулятор**

Необхідно використати модулі для програми калькулятор. Функції додавання, віднімання, множення та ділення перенести в файл functions.py. Функції запиту на введення даних для операцій та самих операцій перемістити в файл operations.py. Програму калькулятор реалізувати в файлі calc.py, до якого підключають файл functions.py та operations.py.

Хід виконання завдання:

У цьому завданні я розділив програму калькулятор з минулих завдань на модулі. Я розділив програму на три файли:

functions.py — функції додавання, віднімання, множення, ділення.

operations.py — обробляє взаємодію з користувачем, отримання введення та виконання операцій.

calc.py — головний файл, який запускає програму калькулятора і імпортує функції з інших файлів.

У файлі functions.py реалізовані чотири функції для основних операцій: додавання, віднімання, множення та ділення.

Текст програми functions.py:

def add(x, y):

    return x + y

def subtract(x, y):

    return x - y

def multiply(x, y):

    return x \* y

def divide(x, y):

    if y == 0:

        raise ValueError("На нуль ділити не можна!")

    return x / y

У файлі operations.py реалізовані функції для отримання введення користувача та вибору операції.

Спочатку імпортуються функції із файлу functions.py.

Далі йде функція get\_numbers() яка здійснює ввід двох чисел та перевірку правильності вводу. Вона повертає введені числа типу float().

Наступна функція choose\_operation(), яка друкує перелік можливих арифметичних операцій, запитує введення користувачем операції та дозволяє вийти з калькулятора.

Наступна функція perform\_operation() отримує в якості параметрів код операції та введені числа та викликає відповідну функцію арифметичної операції.

Текст програми operations.py:

from functions import add, subtract, multiply, divide

def get\_numbers():

    try:

        a = float(input("Введіть перше число: "))

        b = float(input("Введіть друге число: "))

        return a, b

    except ValueError:

        print("Будь ласка, введіть правильне число.")

        return get\_numbers()

def choose\_operation():

    print("Оберіть тип операції:")

    print("Додавання +")

    print("Віднімання -")

    print("Множення \*")

    print("Ділення /")

    choice = input("Ваш вибір (+, -, \*, /) для виходу - exit: ")

    return choice

def perform\_operation(choice, a, b):

    if choice == '+':

        return add(a, b)

    elif choice == '-':

        return subtract(a, b)

    elif choice == '\*':

        return multiply(a, b)

    elif choice == '/':

        return divide(a, b)

    else:

        print("Невірний вибір операції.")

        return None

У файлі calc.py знаходиться головна функція роботи калькулятора main().

Спочатку виконується імпорт функцій get\_numbers, choose\_operation, perform\_operation з файлу operations.py.

У циклі головної функції maіn() викликаються функції get\_numbers() та choose\_operation() які присвоюють значення змінним a, b (уведені числа) та змінній choice (тип операції).

Якщо тип операції дорівнює exit – то відбувається вихід з калькулятора.

Далі після частини Try: викликається функція perform\_operation(choice, a, b) яка аналізує числа та тип операції.

Якщо вони не коректні – то змінна result отримує значення None, друкується повідомлення про помилку і цикл повторюється.

Якщо результат дорівнює якомусь числу – то друкується цей результат роботи калькулятор та повторюється цикл запиту першого числа.

В кінці викликається головна функція main().

Текст програми calc.py:

from operations import get\_numbers, choose\_operation, perform\_operation

def main():

    print("Ласкаво просимо до калькулятора!")

    while True:

        a, b = get\_numbers()

        choice = choose\_operation()

        if choice.lower() == 'exit':

            break

        try:

            result = perform\_operation(choice, a, b)

            if result is not None:

                print("Результат:", result)

        except ValueError as e:

            print(e)

        continue

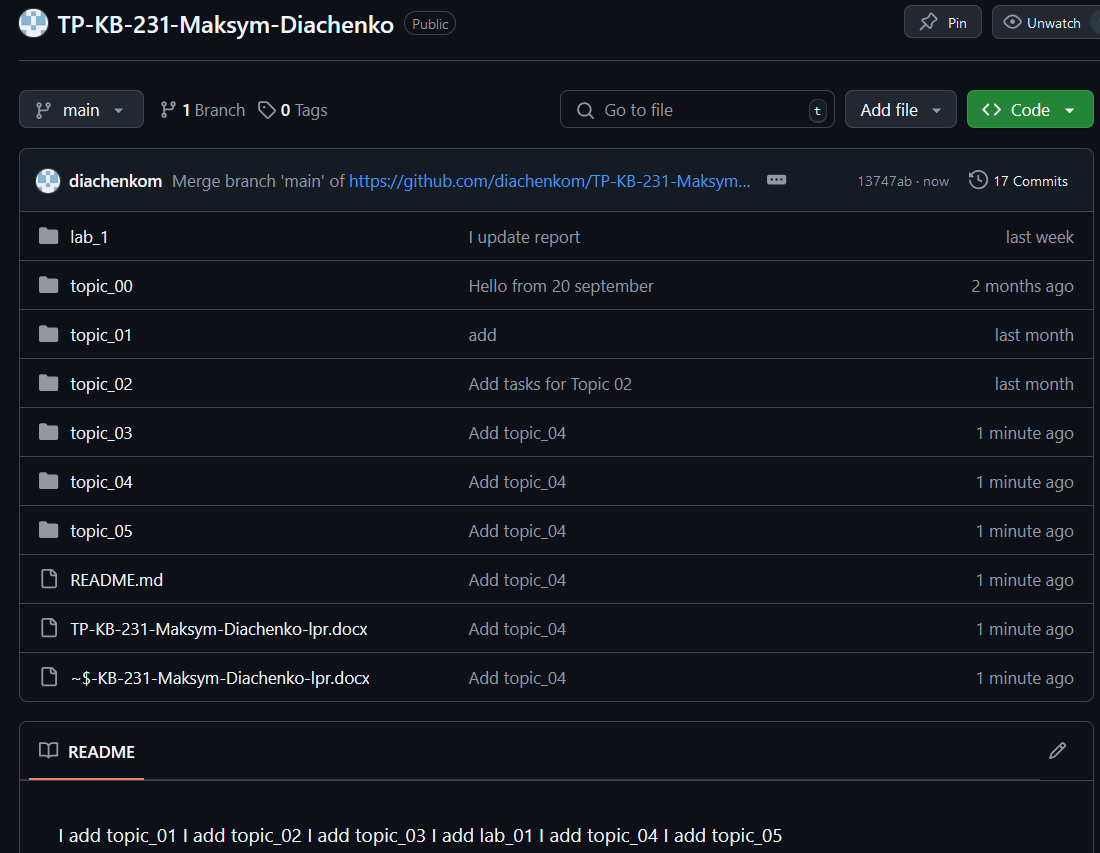
if \_\_name\_\_ == "\_\_main\_\_":

    main()

Посилання на github:

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

Знімок екрану з посилання на github:



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

Звіт до Теми №6

Робота з файлами

Під час виконання практичного завдання до Теми №6 було надано варіанти рішення до наступних задач:

Попередні умови: Реалізована програма калькулятор в файлі calc.py, до якого підключають файл functions.py та operations.py. Інструкції до оформлення вмісту файлів розміщенні в завданні 3 до теми 5.

1) Розробити механізм логування всіх дій, що виконує програма. Забезпечити зберігання інформації про введені данні, виконану операцію та результат виконання операції над даними.

2) Маючи не відсортований список, елементами якого є словники з двома параметрами (ім’я та оцінка) виконати сортування списку, використовуючи стандартну функцію sorted(). Другим параметром для функції sorted() має бути lambda функція, що повертає ім’я або оцінку із елемента словника.

**Логування всіх дій програми калькулятора**

Необхідно розробити механізм логування всіх дій, що виконує програма. Забезпечити зберігання інформації в текстовому файлі про введені данні, виконану операцію та результат виконання операції над даними.

Хід виконання завдання:

У цьому завданні я вніс наступні зміни в програму калькулятор calc.py:

Після імпорту функцій з моєї бібліотеки operations.py за допомогою кострукції Try:…finally: відкривається текстовий файл Test.txt в режимі «а» - тобто додавання нових записів. Цей режим відкриття файлу потрібен для збереження даних, які вже записані у файл Test.txt і дописування нових записів в кінець цього файлу-журналу.

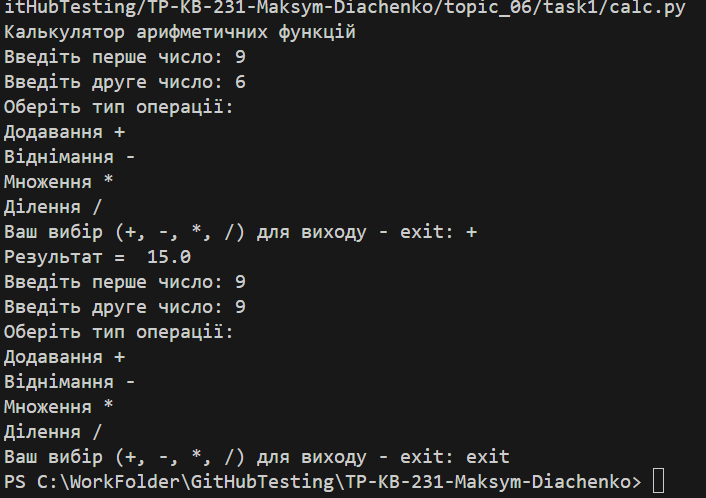
Далі в файл вводиться рядок тексту «Журнал роботи калькулятора», який свідчить про новий сеанс роботи програми calc.py і відповідно - нові записи журналу її роботи в файлі Test.txt.

Далі в циклі, після введення чисел «а», «b» та коду операції «+, -, \*, /» - записуються в файл-журнал Test.txt в 1 строку спочатку число «а», потім код операції, далі число «b»

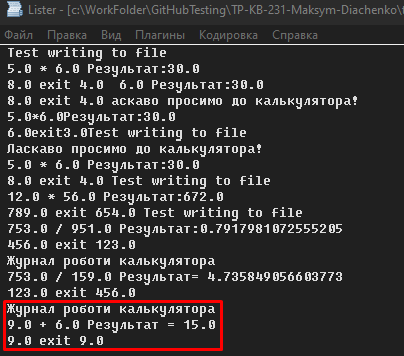
Після розрахунку результату операції і присвоєння його змінній result – в файл-журнал Test.txt в наступну строку виводиться результат арифметичної операції, і т.д. в циклі.

Після завершення роботи програми калькулятора calc.py (коли користувач на етапі вводу арифметичної операції набрав «exit») – спрацьовує конструкція Finally: - після якої файл Test.txt закривається командою file1.close() і робота програми припиняється.

Виконані дії в калькуляторі:



Видно, що попередні дії записалися до лог-файлу:



Текст програми calc.py:

from operations import get\_numbers, choose\_operation, perform\_operation

try:

    file1 = open("test.txt", "a")

    file1.write("\nЖурнал роботи калькулятора\n")

    print("Калькулятор арифметичних функцій")

    while True:

        a, b = get\_numbers()

        choice = choose\_operation()

        file1.write(str(a) + " " + choice + " " + str(b) + " ")

        if choice.lower() == 'exit':

            break

        try:

            result = perform\_operation(choice, a, b)

            if result is not None:

                print("Результат = ", result)

                file1.write("Результат = " + str(result) +"\n")

        except ValueError as e:

            print(e)

        continue

finally:

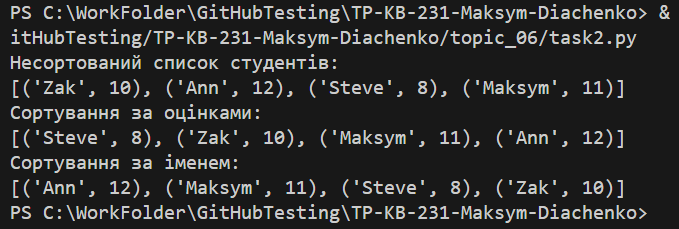
    file1.close()

**Сортування списку за допомогою функції sorted()**

Необхідно виконати сортування списку, використовуючи стандартну функцію sorted().

Хід виконання завдання:

У цьому завданні я Маючи не відсортований список, елементами якого є словники з двома параметрами (ім’я та оцінка) виконав сортування списку, використовуючи стандартну функцію sorted(). Другим параметром для функції sorted() має бути lambda функція, що повертає ім’я або оцінку із елемента словника.



Спочатку задається не відсортований список студентів який складається з двох полів: ім’я студента та його оцінка.

Наступними двома командами виводиться на екран цей не відсортований список.

Далі за допомогою функції sorted() виконується сортування заданого списку: спочатку за оцінками студентів, а далі – за ім’ям студента. Ключем key=lambda elem: elem[] задається поле за яким треба сортувати список. Номер поля вказується в квадратних дужках.

Після цього сортований список виводиться на екран.

Текст програми:

student\_grades = [

    ('Zak', 10),

    ('Ann', 12),

    ('Steve', 8),

    ('Maksym', 11),

]

print("Несортований список студентів:")

print(student\_grades)

a = sorted(student\_grades, key=lambda elem: elem[1])

print("Сортування за оцінками:")

print(a)

b = sorted(student\_grades, key=lambda elem: elem[0])

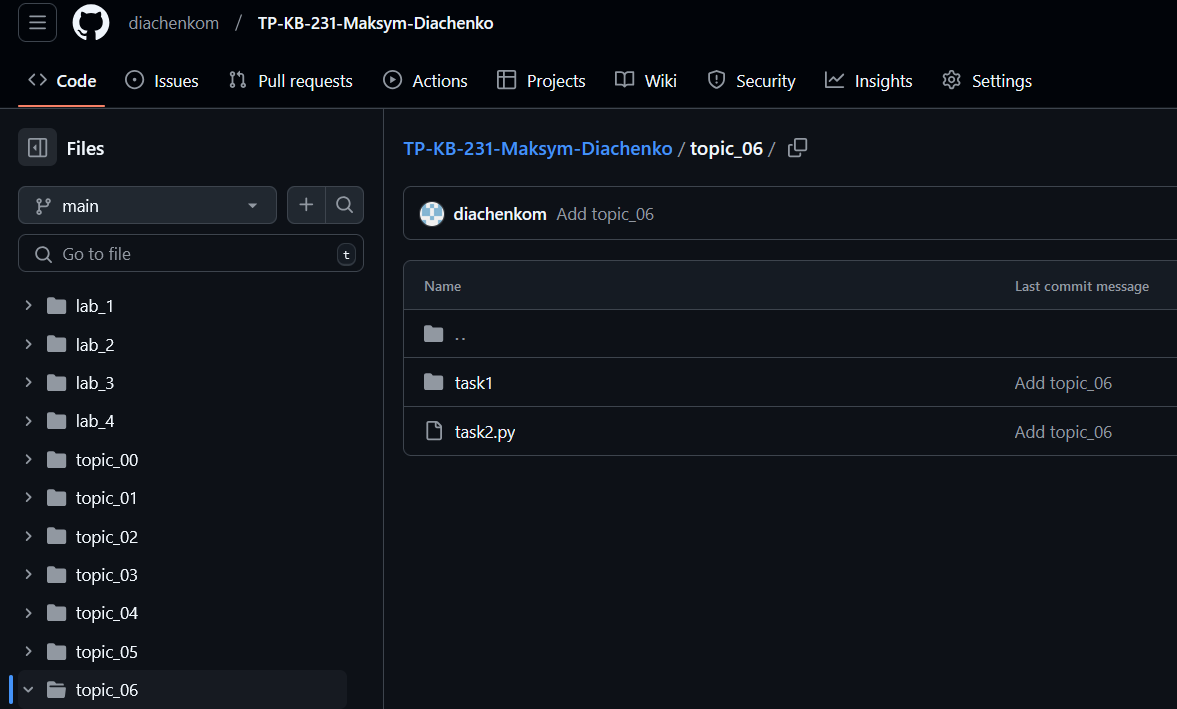
print("Сортування за іменем:")

print(b)

Посилання на github:

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

Знімок екрану з посилання на github:



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

Звіт до Теми №7

Об'єктно-орієнтоване програмування

Під час виконання практичного завдання до Теми №7 було надано варіанти рішення до наступних задач:

1) Ознайомитись з документацією що описує можливості використання класів у мові Python https://docs.python.org/3/tutorial/classes.html

2) Ознайомитись з існуючими за замовченням методами класу по типу \_\_init\_\_(self) \_\_str\_\_(self)\_\_ та надати приклади використання.

3) Розробити клас Student атрибутами якого э два параметра name та age. Створити список елементами якого є об'єкти класу Student. Написати цикл який виводить на екран елементи списку у відсортованому порядку. Для сортування використати стандартну функцію sorted. Функція sorted має використовувати lambda функцію для визначення ключа сортування.

4) Використовуючи принципи ООП переписати програму Калькулятор. Завдання має бути виконано використовуючи модульний підхід.

**Ознайомитись з документацією що описує можливості використання класів у мові Python**

Під час виконання цього завдання я ознайомився з поняттями: клас, об’єкт класу, атрибут об’єкта, конструктор класу, метод класу. Навчився створювати класи та об’єкти, присвоювати значення атрибутам об’єктів, створювати та використовувати методи класів.

**Приклади використання \_\_init\_\_(self) та \_\_str\_\_(self)**

Під час виконання цього завдання я ознайомився х конструктором класу \_\_init\_\_(self) та методом відображення об’єкту класу \_\_str\_\_(self).

Для створення нового об'єкта класу використовується конструктор. Ми можемо явно визначити в класах конструктор за допомогою спеціального методу, який називається \_\_init\_\_().

В Лабораторній роботі №3, де в коді програм використовувались класи застосовувався також і конструктор \_\_init\_\_(), наприклад в модулі Student.py, при створенні класу Student()

class Student:

    def \_\_init\_\_(self, name: str, phone: str, kurs: str, group: str):

        self.name = name

        self.phone = phone

        self.kurs = kurs

        self.group = group

При створенні студента – об’єкта класу Student йому присвоюються ряд атрибутів: Ім’я: атрибут «name», Телефон: атрибут «phone», Курс: атрибут «kurs», Група: атрибут «group».

Цей метод викликається автоматично кожного разу при створенні об’єкта класу Student.

В цьому ж класі Student є також метод \_\_str\_\_(self), який дозволяє виводити всю інформацію про студента (об’єкта класу Student), тобто всі його атрибути, у вигляді строки:

def \_\_str\_\_(self):

        return f"Name: {self.name}, Phone: {self.phone}, Course: {self.kurs}, Group: {self.group}"

Так інформацію про студента в тексті програми можна вивести на екран командою: print(Student), а за допомогою нескладної функції в модулі StudentList.py реалізовано вивід списку всіх студентів на екран:

 def printAllList(self):

        if not self.students:

            print("Student list is empty.")

        else:

            for student in self.students:

                print(student)

Таким чином метод \_\_str\_\_(self) дозволяє 1 раз задати формат виводу об’єкта класу на екран і далі виводити об’єкт простою командою print(Student), незалежно від переліку атрибутів цього об’єкта.

**Клас Student атрибутами якого э два параметра name та age**

Під час виконання цього завдання я написав програмний модуль task\_3.py, в якому створюється клас Student з атрибутами об’єктів класу name, age. Далі в цьому модулі створюється несортований список об’єктів класу Student. Потім об’єкти цього списку сортуються за значенням атрибута age командою sorted. Ключ сортування задається в параметрах функції sorted після lambda-функції. Після чого відсортований список об’єктів виводиться на екран.

Текст програми task\_3.py:

class Student:

    def \_\_init\_\_(self, name, age):

        self.name = name

        self.age = age

    def \_\_str\_\_(self):

        return f"Name: {self.name}, Age: {self.age}"

students = [

    Student("Tom", 20),

    Student("Bob", 18),

    Student("Yan", 17),

    Student("Zak", 23),

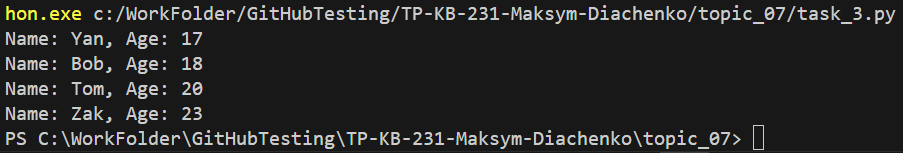
]

sorted\_students = sorted(students, key=lambda student: student.age)

for student in sorted\_students:

    print(student)

Результат роботи програми task\_3.py:



**Програма Калькулятор, яка переписана використовуючи принципи ООП**

Під час виконання цього завдання я переробив програму Калькулятор, розбивши її на окремі модулі та використовуючи принципи Об’єктно-орієнтованого програмування. Для вирішення цього завдання я розділив програму Калькулятор на 3 модулі: calc7.py – головний модуль, oper7.py, - арифметичні операції, які виконує калькулятор, func7.py – основні функції, які викликаються з головної програми calc7.py під час роботи Калькулятора.

Текст програми oper7.py:

class Operations:

    def add(x, y):

        return x + y

    def subtract(x, y):

        return x - y

    def multiply(x, y):

        return x \* y

    def divide(x, y):

        if y == 0:

            raise ValueError("Zero Division Error")

        return x / y

В модулі oper7.py оголошується клас Operations(), в якому описані у вигляді методів всі арифметичні операції, які може виконувати Калькулятор:

додавання – метод аdd(x,y), повертає результат виразу (x+y)

віднімання – метод subtract(x,y), повертає результат виразу (x-y)

множення – метод multiply(x,y), повертає результат виразу (x\*y)

ділення – метод divide(x,y), повертає результат виразу (x/y)

Кожний з методів отримує в якості аргументів 2 числа х, y – операнди арифметичних операцій. Після цього методи аdd(x,y), subtract(x,y), multiply(x,y) – виконують відповідну операцію та повертають результат в процедуру, з якої вони викликані.

В методі divide(x,y) спочатку виконується перевірка чи аргумент y не рівен 0, якщо так – видається повідомлення про помилку і ділення не відбувається:

Raise ValueErryor(“На 0 ділити не можна!”)

Відбувається вихід в головну програму calc7.py та продовження циклу її роботи.

В модулі func7.py зібрані основні функції необхідні для роботи Калькулятора.

Текст програми func7.py:

from oper7 import Operations

class Functions:

    def get\_numbers():

        try:

            a = float(input("Input first number: "))

            b = float(input("Input second number: "))

            return a, b

        except ValueError:

            print("Wrong input. Please repeat")

            return Functions.get\_numbers()

    def choose\_operation():

        print("Choose type operation:")

        print("add +")

        print("subtract -")

        print("multiply \*")

        print("divide /")

        choice = input("Choose (+, -, \*, /) to leave enter 'exit': ")

        return choice

    def perform\_operation(choice, a, b):

        if choice == '+':

            return Operations.add(a, b)

        elif choice == '-':

            return Operations.subtract(a, b)

        elif choice == '\*':

            return Operations.multiply(a, b)

        elif choice == '/':

            return Operations.divide(a, b)

        else:

            print("Wrong choose.")

            return None

class Logger:

    def log(text1):

        log\_file="calc\_log.txt"

        with open(log\_file, "a") as file:

            file.write(text1 + "\n")

Першою строкою відбувається імпорт класу арифметичних операцій Operations() з модуля oper7.py.

Далі оголошується клас Functions(), в якому зібрані всі основні функції, необхідні для роботи Калькулятора. Кожна фунція – це метод класу Functions().

Першою оголошується Метод get\_numbers(), яка очікує від користувача ввід 2-х чисел для арифметичної операції, пробує конвертувати введені числа в формат float. Якщо конвертація виконана успішно, функція повертає 2 числа типу float у головну процедуру calc7, з якої вона викликалась.

Якщо конвертація введених користувачем чисел невдала – виводиться повідомлення про помилку і запит вводу чисел повторюється.

Метод choose\_operation() – створює діалог з користувачем для вводу коду арифметичної операції. Після вводу коду операції у вигляді символа: “+”, “-“, “\*”, “/” він зберігається в змінній choice і повертається в процедуру, яка викликала цей метод. На етапі цього методу користувач може також коректно завершити роботу Калькулятора, набравши слово «exit».

Перевірка змінної choice на слово «exit» здійснюється в головній процедурі calc7.py

Наступний метод: perform\_operation() – виконує арифметичну операцію відповідно обраного коду операції та повертає її результат в головну процедуру calc7. Він отримує в якості параметрів операнди “а” та “b” та викликає потрібний метод класу Operations() відповідно обраного коду операції, переданого в цей метод параметром “choice”. Викликаний метод класу Operations() власне і виконує потрібну користувачу операцію, результат якої повертається командою return в головну процедуру calc7.

Якщо код операції не входить у визначений перелік: “+”, “-“, “\*”, “/”,

Виводиться повідомлення про помилковий код операції і в головну процедуру повертається порожній результат «None»

Далі оголошується клас Logger, який призначений для логування роботи калькулятора в текстовий файл. Цей клас має тільки один метод Log, в який передається в якості параметру строка тексту text1. Першою командою в змінній задається ім’я файлу журналу. Наступна команда відкриває файл для запису в режимі додавання записів: якщо файл не існує – то він створюється, а якщо файл вже існує – то нові записи додаються в кінець файлу. Останньою командою передана строка тексту записується в файл журнал.

Головна процедура Калькулятора – сalc7.py.

Текст програми calc7.py:

from func7 import Functions, Logger

print("Start calculator")

while True:

    a, b = Functions.get\_numbers()

    choice = Functions.choose\_operation()

    Logger.log(f"{a} {choice} {b}")

    if choice.lower() == 'exit':

        break

    try:

        result = Functions.perform\_operation(choice, a, b)

        if result is not None:

            print("Result = ", result)

            str\_res="Result = " + str(result) +"\n"

            Logger.log(f"{str\_res}")

    except ValueError as e:

        print(e)

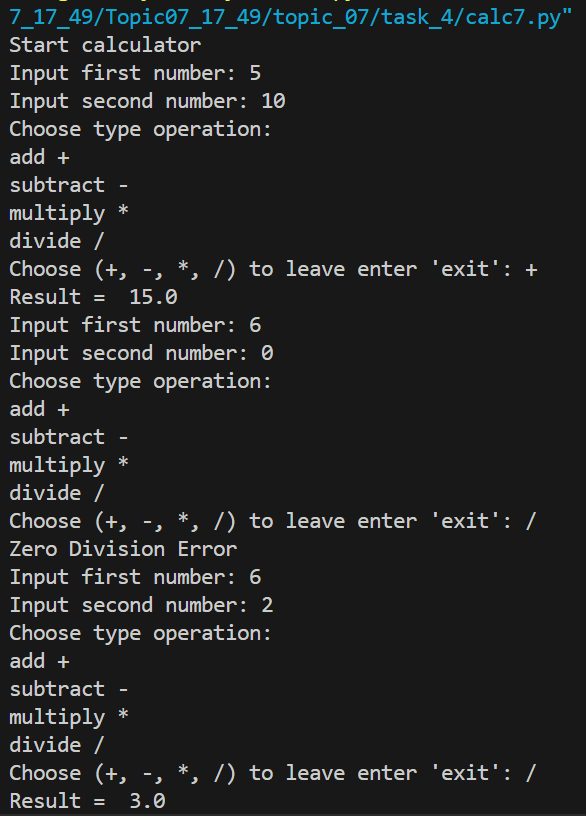
    continue

Спочатку в головній процедурі виконується імпорт класів Functions та Logger із модуля func7.

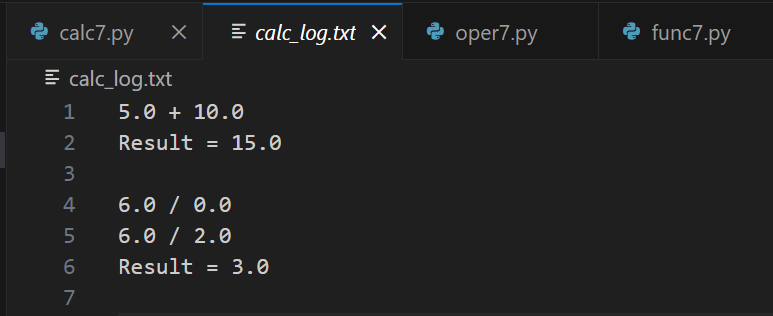
Далі запускається бескінечний цикл. Спочатку викликається метод get\_numbers класу Functions, який запезпечує введення користувачем двох операндів та конвертує їх до типу float. Далі викликається метод choose\_operation класу Functions, який забезпечує введення користувачем типу арифметичної операції (+, -, \*, /). Далі викликається метод log класу Logger, який забезпечує збереження записів арифметичної операції у файлі журналу. Далі перевіряється змінна choiсe: якщо користувач набрав 'exit' – то калькулятор завершає роботу.

Далі викликається метод perform\_operation класу Functions, який намагається виконати операцію введену користувачем та повертає результат опереції в змінну result. Якщо змінна result = None – операція не виконується та запитується повторне введення операндів користувачем, якщо змінна result має числове значення – то результат виводиться на екран та зберігається у файлі журналу.

Результат роботи програми calc7.py:



Записи у журналі роботи калькулятора calc\_log.txt:



Посилання на github:

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

Знімок екрану з посилання на github:

