**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:

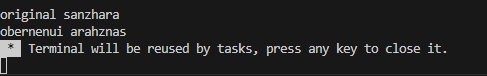
**1)Перетворення рядка**

Необхідно рядок, що має вигляд "sanzhara" перетворити наступним чином "321gfedcba", вважаючи сталою довжину рядку в 10 символів.

Текст програми:

|  |
| --- |
| a = "sanzhara"  b = a[::-1]  print(f"original {a}")  print(f"obernenui {b}") |

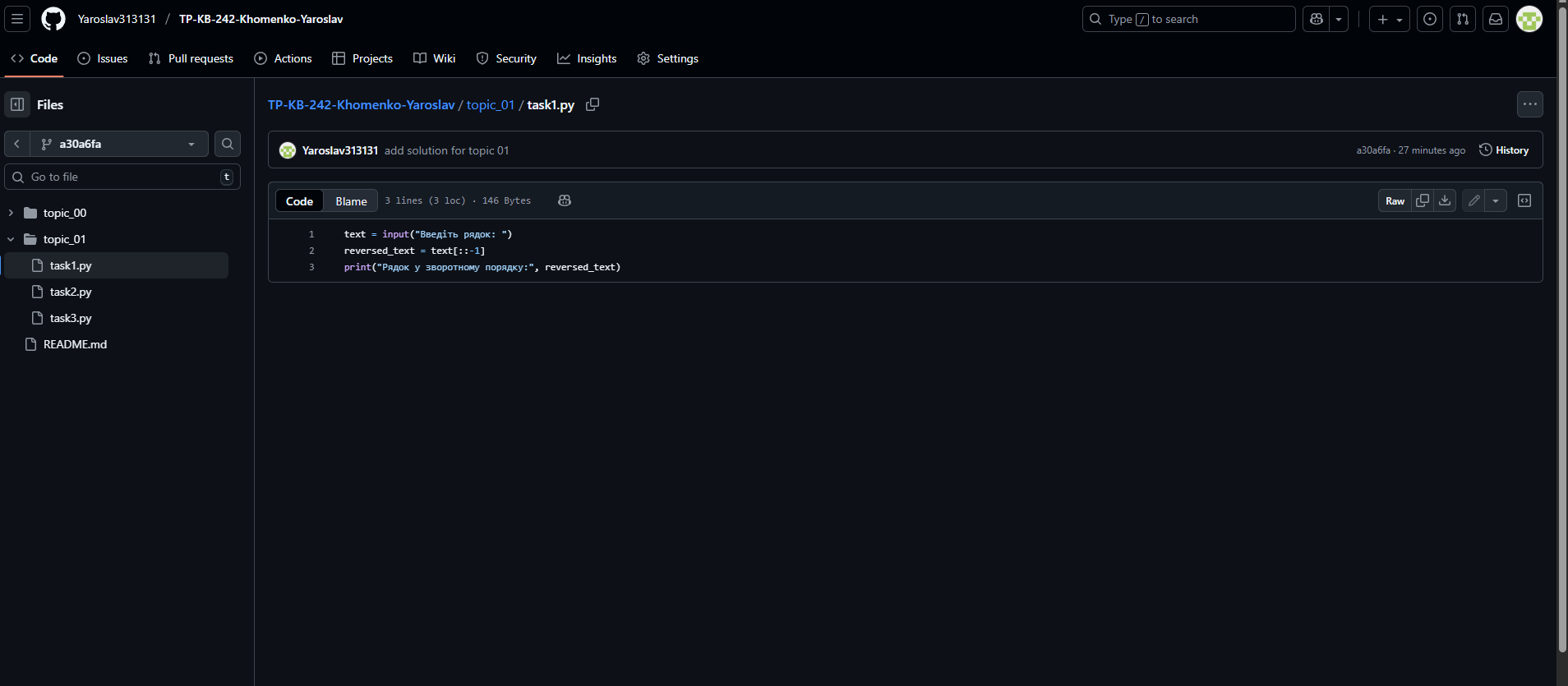
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/a30a6fa7f1dc9a9f66a2de76c18cd2feb5937def/topic_01/task1.py>

Знімок екрану з посилання на github:

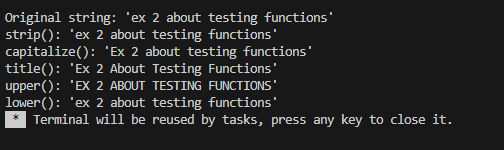


**2)Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().**

Текст програми:

|  |
| --- |
| probilu = "ex 2 about testing functions"  print(f"Original string: '{probilu}'")  # 1  no\_probilu = probilu.strip()  print(f"strip(): '{no\_probilu}'")  # 2  firstBIG = no\_probilu.capitalize()  print(f"capitalize(): '{firstBIG}'")  # 3  everywordBIG = no\_probilu.title()  print(f"title(): '{everywordBIG}'")  # 4  everylitBIG = no\_probilu.upper()  print(f"upper(): '{everylitBIG}'")  # 5  everylitsmall = everylitBIG.lower()  print(f"lower(): '{everylitsmall}'") |

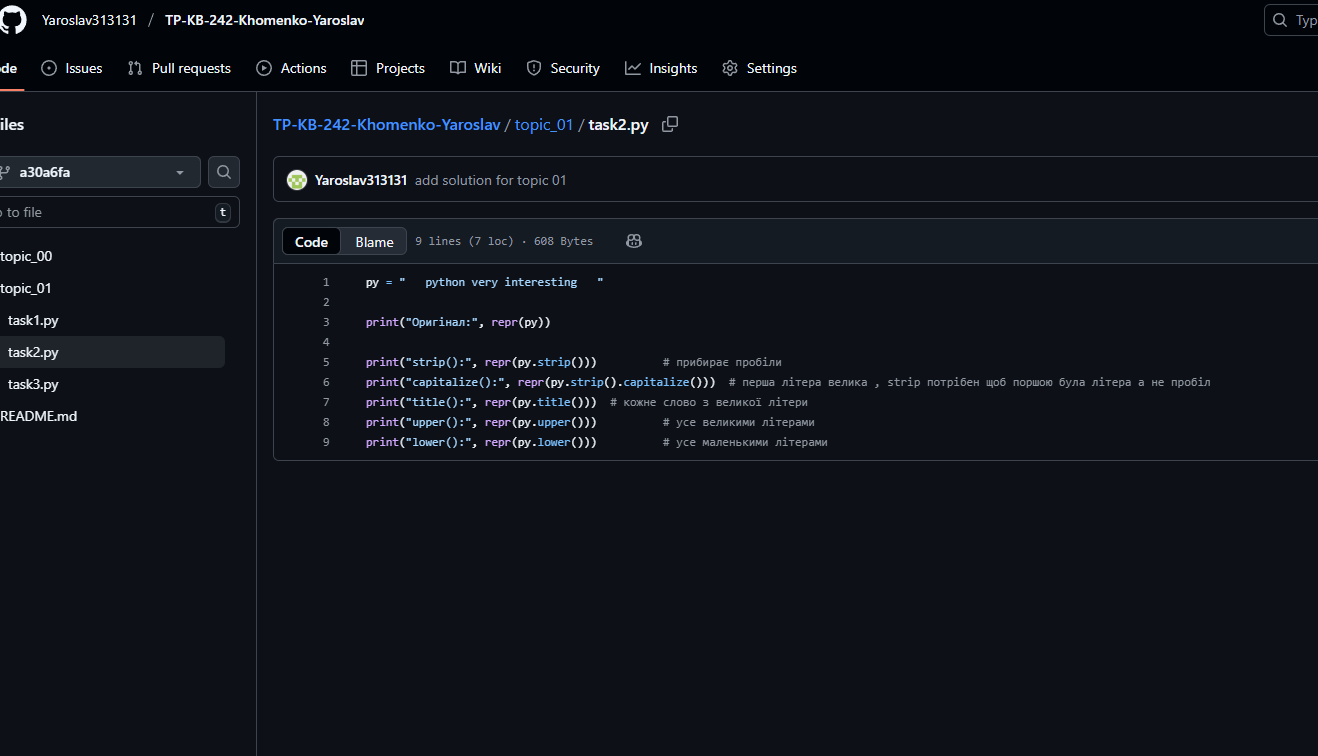
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/a30a6fa7f1dc9a9f66a2de76c18cd2feb5937def/topic_01/task2.py>

Знімок екрану з посилання на github:



1. **Написати функцію пошуку дискримінанту квадратного рівняння.**

Текст програми:

def discr(a, b, c):

    D = b\*\*2 - 4\*a\*c

    return D

a = float(input("Введи a: "))

b = float(input("Введи b: "))

c = float(input("Введи c: "))

if a == 0:

    print("не квадратне рівняння")

else:

    D = discr(a, b, c)

    print("D equals:", D)

    if D > 0:

        print("2 koreni")

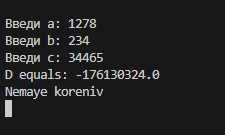
    elif D == 0:

        print("1 korin'")

    else:

        print("Nemaye koreniv")

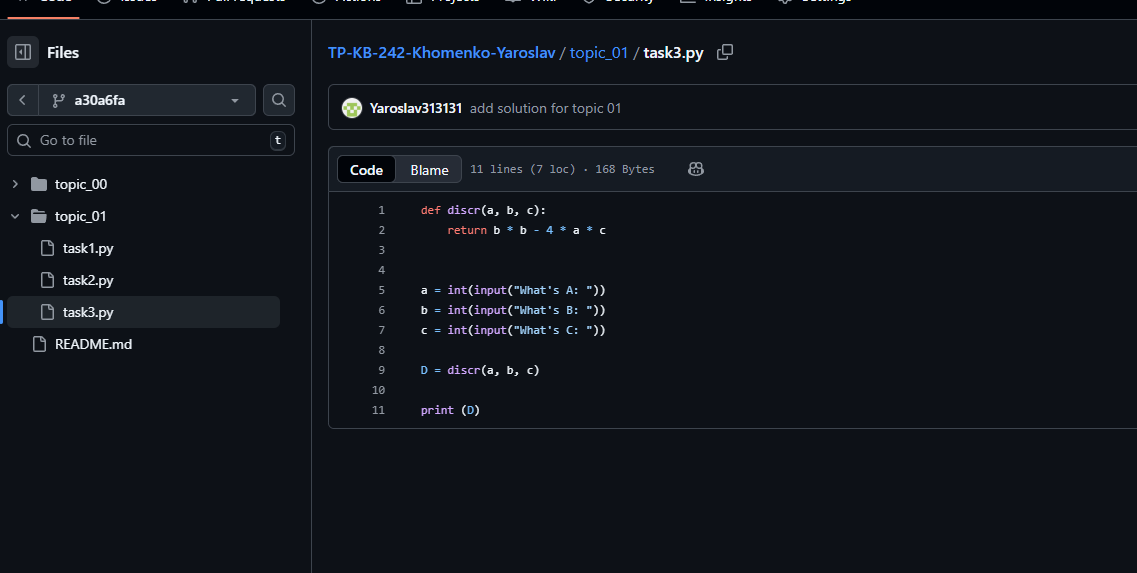
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/a30a6fa7f1dc9a9f66a2de76c18cd2feb5937def/topic_01/task3.py>

Знімок екрану з посилання на github:



**Висновок :** У ході виконання практичної роботи було поглиблено та закріплено знання з теми «Функції та змінні» в Python. Bідпрацьовано навички оголошення змінних, передачі їх як аргументів у функції та отримання результатів за допомогою оператора return. Виконані завдання продемонстрували ключову роль функцій у структуризації коду, покращенні його читабельності та забезпеченні можливості повторного використання програмних блоків.

**Звіт до Теми №2**

Умовний перехід

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.**

Текст програми:

import math

def discr(a, b, c):

    """Обчислює дискримінант D."""

    D = b\*\*2 - 4\*a\*c

    return D

try:

    a = float(input("Введи a: "))

    b = float(input("Введи b: "))

    c = float(input("Введи c: "))

except ValueError:

    print("Введено некоректне число")

    exit()

if a == 0:

    print("Це не квадратне рівняння")

else:

    D = discr(a, b, c)

    print(f"\nДискримінант дорівнює: {D}")

    if D > 0:

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

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

        print("Маємо 2 різні дійсні корені:")

        print(f"x1 = {x1:.2f}")

        print(f"x2 = {x2:.2f}")

    elif D == 0:

        x = -b / (2 \* a)

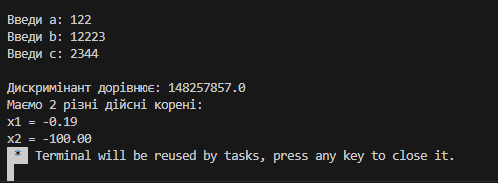
        print("Маємо 1 дійсний корінь ")

        print(f"x = {x:.2f}")

    else:

        print("Дискримінант від'ємний ")

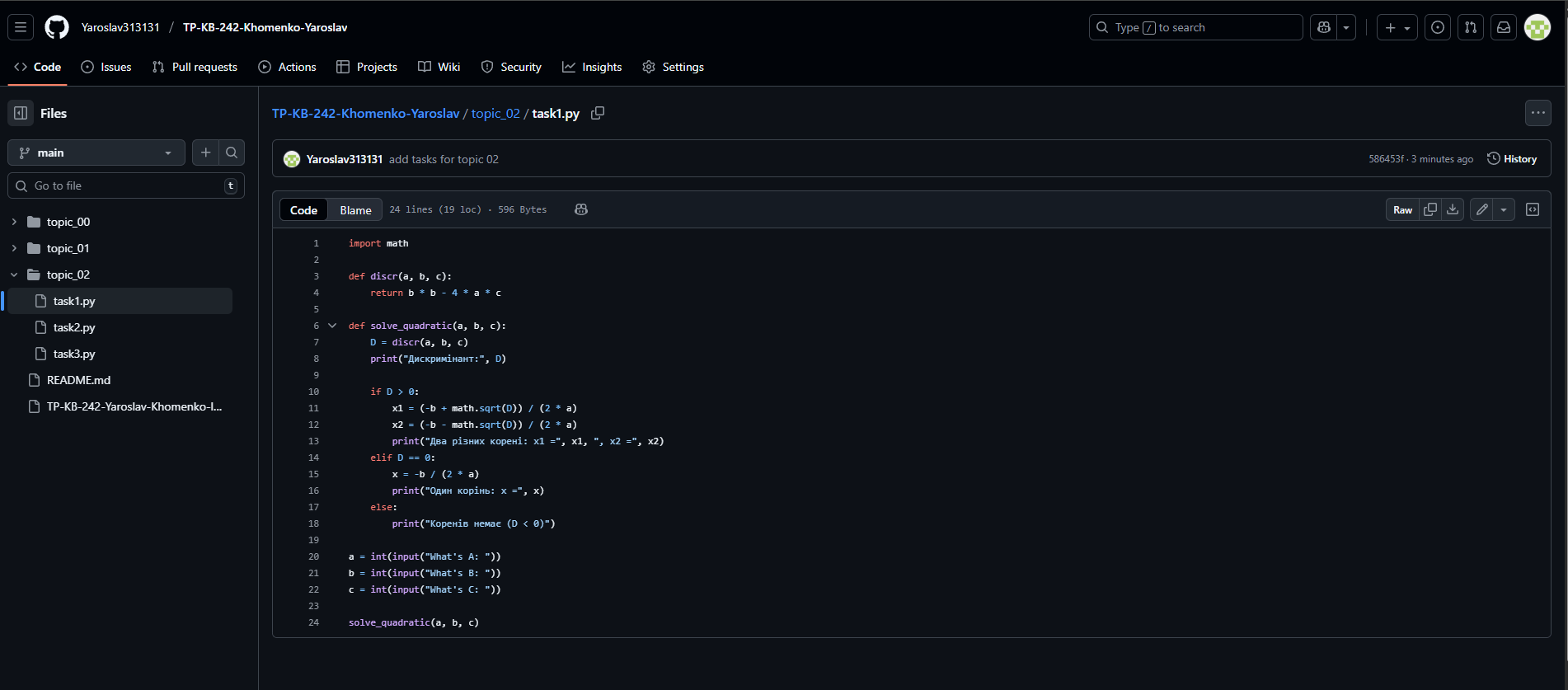
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/586453f1c77ff641bc016693805bf2aa8b3fcd5d/topic_02/task1.py>

Знімок екрану з посилання на github:



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

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 "На нуль не ділять"

    return x / y

def calc\_if\_else():

    print("==============================")

    print("==========КАЛЬКУЛЯТОР=========")

    print("==============================")

    try:

        x = float(input("Введіть число x: "))

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

        y = float(input("Введіть число y: "))

    except ValueError:

        print("Точно числа ввели?")

        return

    результат = None

    if операція == '+':

        результат = add(x, y)

    elif операція == '-':

        результат = subtract(x, y)

    elif операція == '\*':

        результат = multiply(x, y)

    elif операція == '/':

        результат = divide(x, y)

    else:

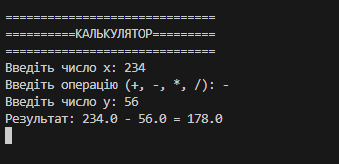
        print("Невідома операція. Спробуйте +, -, \*, або /.")

        return

    print(f"Результат: {x} {операція} {y} = {результат}")

calc\_if\_else()

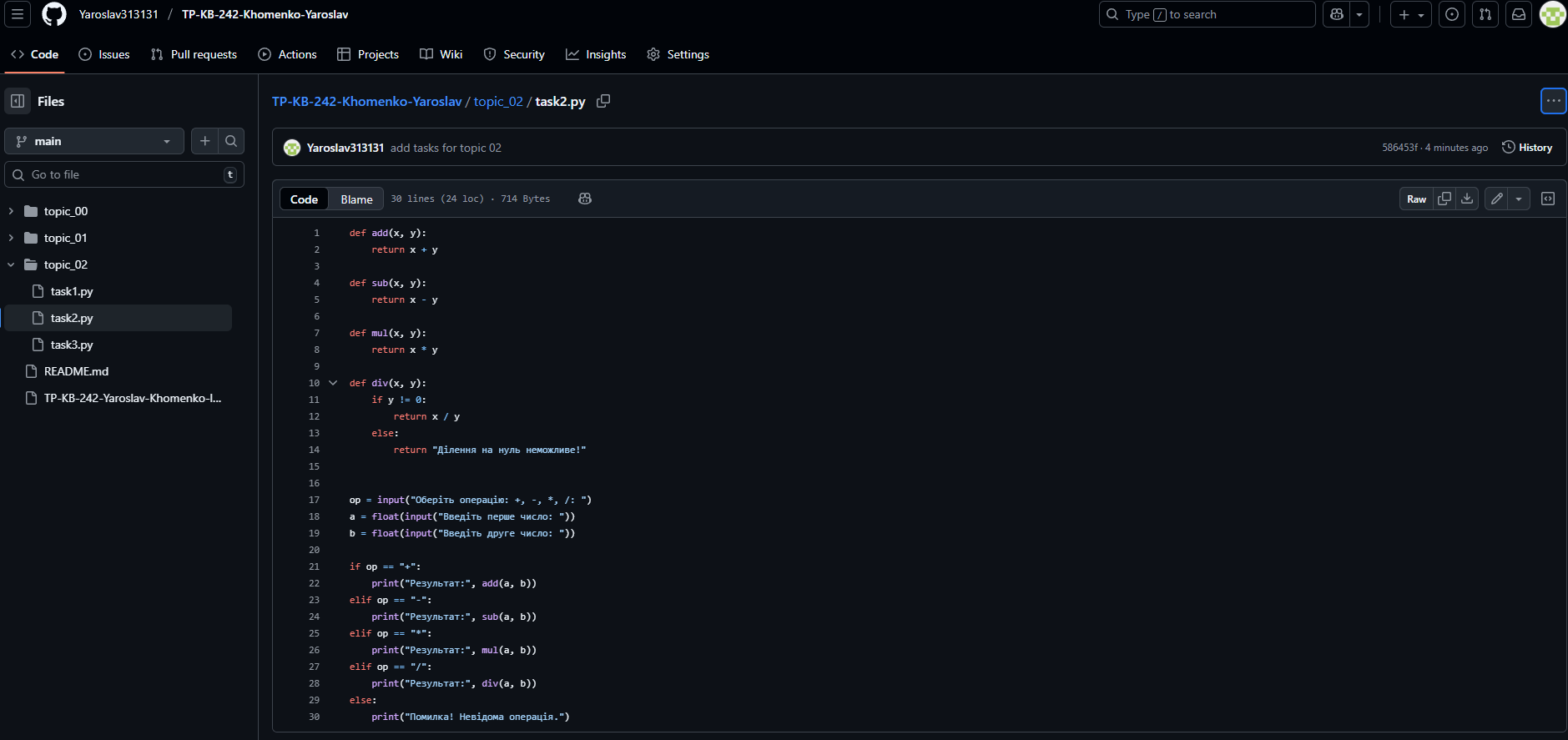
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/586453f1c77ff641bc016693805bf2aa8b3fcd5d/topic_02/task2.py>

Знімок екрану з посилання на github:



1. **Написати програму калькулятор використовуючи match конструкцію. Кожна операція має бути виконана в окремій функції.**

Текст програми:

def add(x, y):

    """Повертає суму двох чисел."""

    return x + y

def subtract(x, y):

    """Повертає різницю двох чисел."""

    return x - y

def multiply(x, y):

    """Повертає добуток двох чисел."""

    return x \* y

def divide(x, y):

    """Повертає частку. Перевіряє ділення на нуль."""

    if y == 0:

        return "На нуль не ділять"

    return x / y

def calc\_match\_case():

    print("==============================")

    print("=====КАЛЬКУЛЯТОР (MATCH)======")

    print("==============================")

    try:

        x = float(input("Введіть число x: "))

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

        y = float(input("Введіть число y: "))

    except ValueError:

        print("Точно числа ввели?")

        return

    результат = None

    match операція:

        case '+':

            результат = add(x, y)

        case '-':

            результат = subtract(x, y)

        case '\*':

            результат = multiply(x, y)

        case '/':

            результат = divide(x, y)

        case \_:

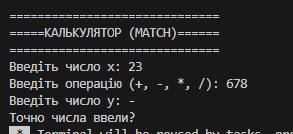
            print("Невідома операція. Спробуйте +, -, \*, або /.")

            return

    print(f"Результат: {x} {операція} {y} = {результат}")

calc\_match\_case()

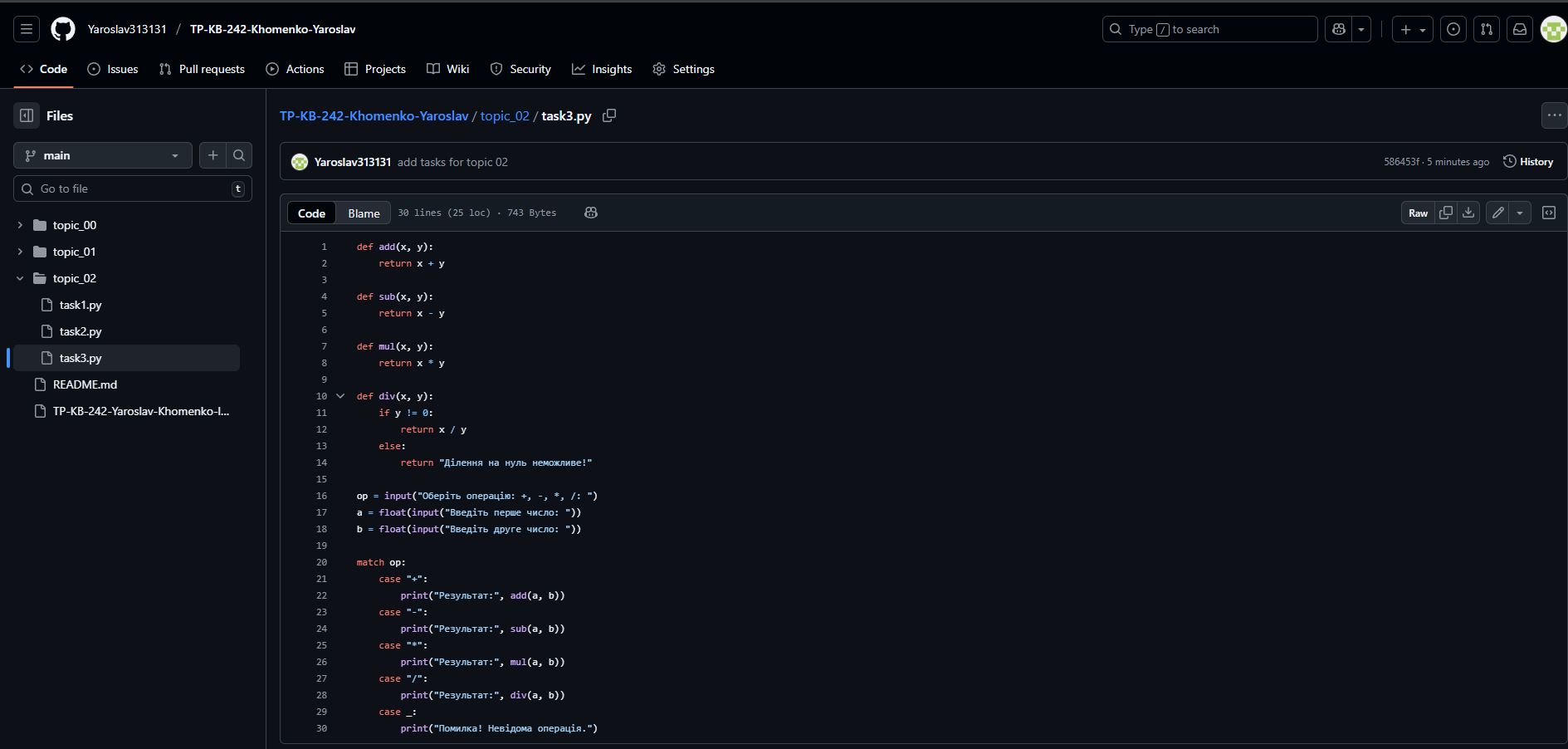
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/586453f1c77ff641bc016693805bf2aa8b3fcd5d/topic_02/task3.py>

Знімок екрану з посилання на github:



**Висновок** : Під час виконання цього завдання я закріпив навички роботи з функціями та умовними конструкціями в Python. Було розроблено дві програми: одна для знаходження коренів квадратного рівняння через функцію дискримінанта, а інша — калькулятор, реалізований як за допомогою if-else, так і match-case.

Робота показала, що функції роблять код організованішим, а match-case є зручним та сучасним інструментом для розгалуження логіки. Таким чином, я успішно набув практичного досвіду у використанні цих фундаментальних інструментів програмування.

**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.**

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 "На нуль не ділять"

    return x / y

def calculator():

    print("==============================")

    print("==========КАЛЬКУЛЯТОР=========")

    print("==============================")

    while True:

        try:

            x\_str = input("Введіть перше число (або 'q' для виходу): ")

            if x\_str.lower() == 'q':

                break

            x = float(x\_str)

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

            y\_str = input("Введіть друге число: ")

            y = float(y\_str)

        except ValueError:

            print("Точно числа ввели? Спробуйте ще раз.")

            continue

        результат = None

        if операція == '+':

            результат = add(x, y)

        elif операція == '-':

            результат = subtract(x, y)

        elif операція == '\*':

            результат = multiply(x, y)

        elif операція == '/':

            результат = divide(x, y)

        else:

            print("Невідома операція. Спробуйте +, -, \*, або /.")

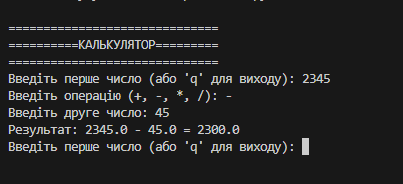
            continue

        print(f"Результат: {x} {операція} {y} = {результат}")

    print("Калькулятор завершує роботу.")

calculator()

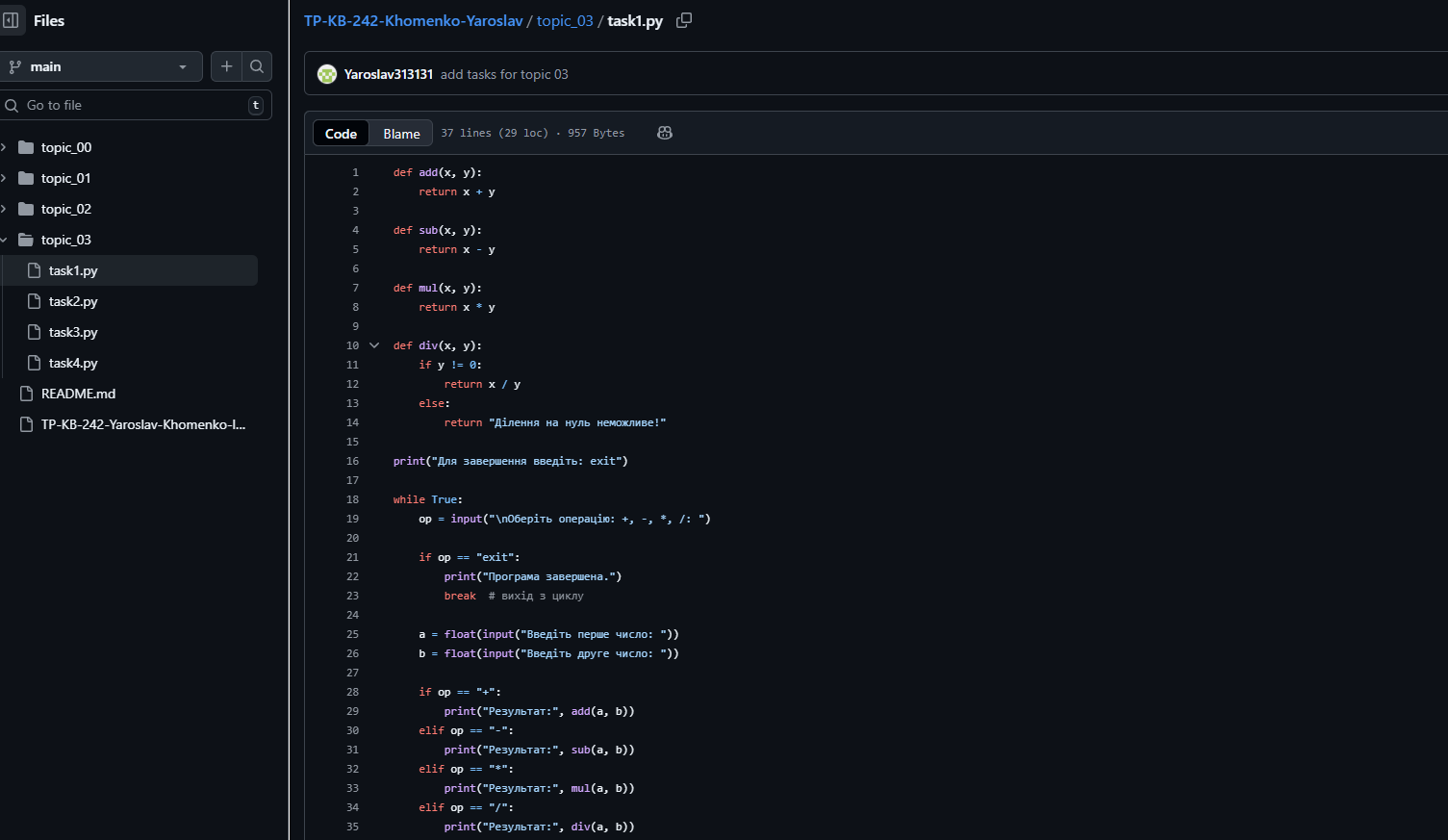
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/a964cd050725edb29dd8c468aa1aa15850fb5987/topic_03/task1.py>

Знімок екрану з посилання на github:



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

Текст програми:

def test\_list\_functions():

    my\_list = [1, 2, 3, 4, 5]

    print(f"Початковий: {my\_list}")

    my\_list.append(6)

    print(f"Після append(6): {my\_list}")

    my\_list.extend([7, 8])

    print(f"Після extend([7, 8]): {my\_list}")

    my\_list.insert(0, 0)

    print(f"Після insert(0, 0): {my\_list}")

    my\_list.remove(4)

    print(f"Після remove(4): {my\_list}")

    my\_list.reverse()

    print(f"Після reverse(): {my\_list}")

    my\_list.sort()

    print(f"Після sort(): {my\_list}")

    new\_list = my\_list.copy()

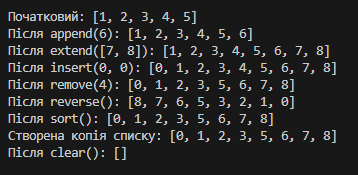
    print(f"Створена копія списку: {new\_list}")

    my\_list.clear()

    print(f"Після clear(): {my\_list}")

test\_list\_functions()

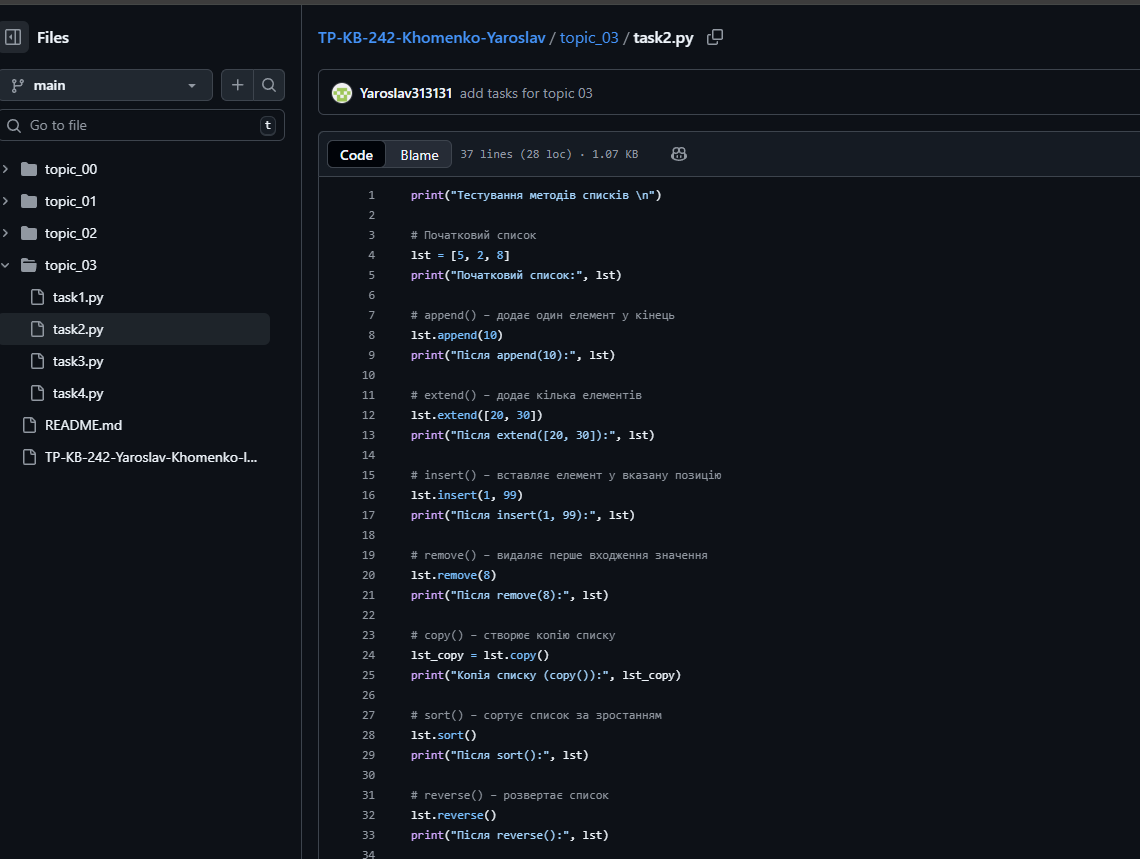
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/a964cd050725edb29dd8c468aa1aa15850fb5987/topic_03/task2.py>

Знімок екрану з посилання на github:



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

Текст програми:

def test\_dict\_functions():

    my\_dict = {'a': 1, 'b': 2, 'c': 3}

    print(f"Початковий словник: {my\_dict}")

    print("\n--- Тестування update ---")

    my\_dict.update({'d': 4, 'b': 22})

    print(f"Після update({{'d': 4, 'b': 22}}): {my\_dict}")

    print("\n--- Тестування del ---")

    del my\_dict['a']

    print(f"Після del ['a']: {my\_dict}")

    print("\n--- Тестування keys ---")

    keys = my\_dict.keys()

    print(f"Ключі словника: {list(keys)}")

    print("\n--- Тестування values ---")

    values = my\_dict.values()

    print(f"Значення словника: {list(values)}")

    print("\n--- Тестування items ---")

    items = my\_dict.items()

    print(f"Пари 'ключ-значення': {list(items)}")

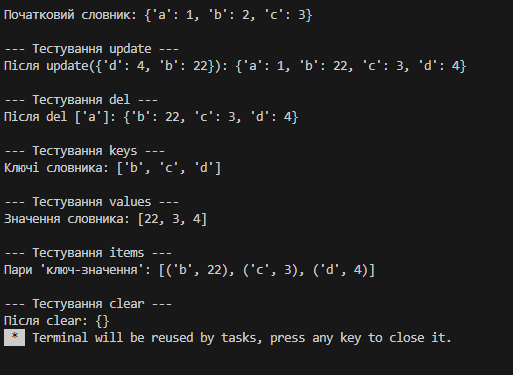
    print("\n--- Тестування clear ---")

    my\_dict.clear()

    print(f"Після clear: {my\_dict}")

test\_dict\_functions()

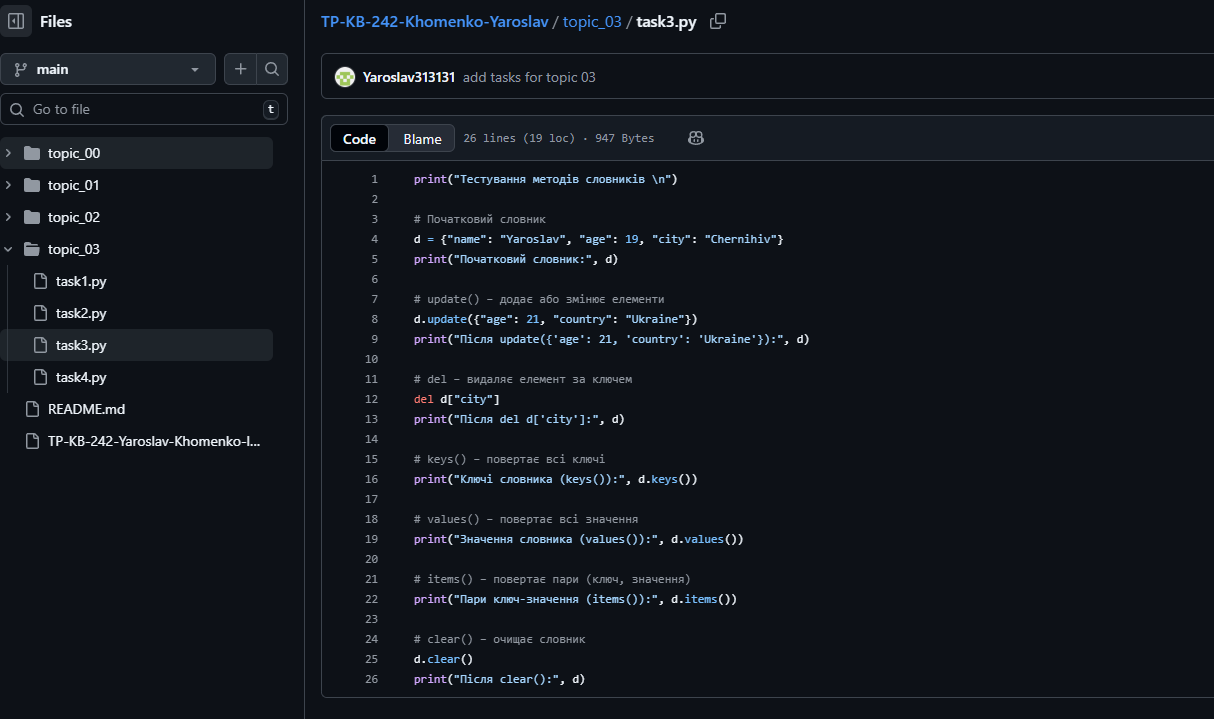
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/a964cd050725edb29dd8c468aa1aa15850fb5987/topic_03/task3.py>

Знімок екрану з посилання на github:



1. **Маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.**

Текст програми:

import bisect

def find\_insertion\_position(sorted\_list, new\_element):

    position = bisect.bisect\_left(sorted\_list, new\_element)

    return position

my\_sorted\_list = [10, 20, 30, 40, 50, 60]

element\_to\_add = 35

pos = find\_insertion\_position(my\_sorted\_list, element\_to\_add)

print(f"Відсортований список: {my\_sorted\_list}")

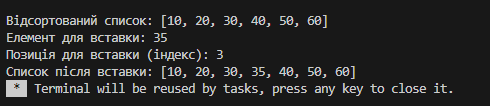
print(f"Елемент для вставки: {element\_to\_add}")

print(f"Позиція для вставки (індекс): {pos}")

my\_sorted\_list.insert(pos, element\_to\_add)

print(f"Список після вставки: {my\_sorted\_list}")

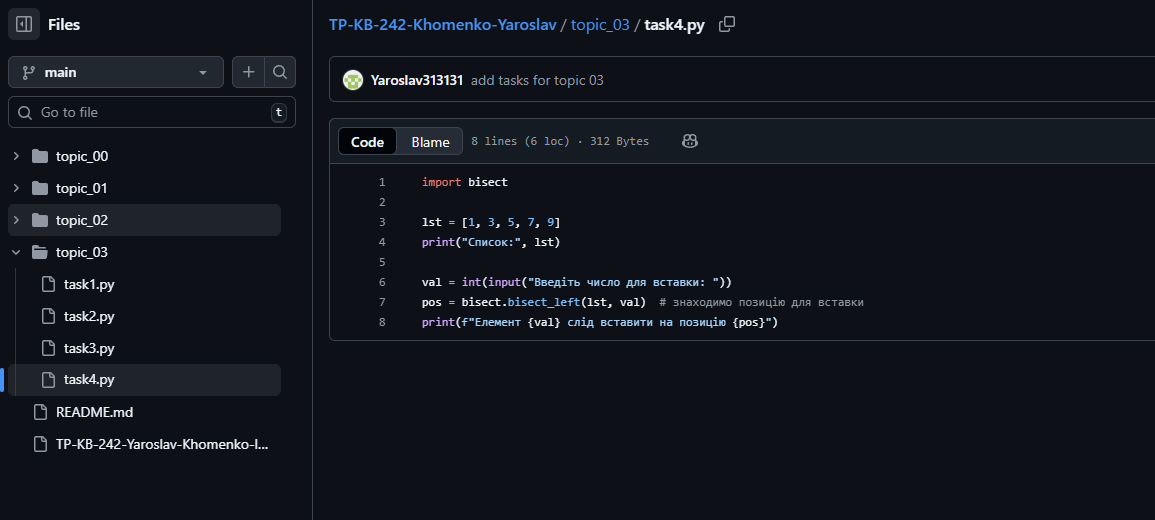
Результат виконання програми:



Посилання на github:

<https://github.com/Yaroslav313131/TP-KB-242-Khomenko-Yaroslav/blob/a964cd050725edb29dd8c468aa1aa15850fb5987/topic_03/task4.py>

Знімок екрану з посилання на github:



**Висновок** : Під час виконання завдань було вдосконалено практичні навички програмування на Python. Зокрема, було розроблено інтерактивний калькулятор з циклічним виконанням та можливістю виходу за командою. Також було опрацьовано ключові методи для роботи зі списками (такі як append(), insert(), sort()) та словниками (update(), keys(), items()). Крім того, було реалізовано функцію для визначення індексу вставки нового елемента у відсортований список. Виконані вправи сприяли поглибленню знань про базові структури даних та розвитку алгоритмічного мислення.