**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 "abcdefg123" перетворити наступним чином "321gfedcba", вважаючи сталою довжину рядку в 10 символів.

Текст програми:

|  |
| --- |
| text = input("Введіть рядок: ")  reversed\_text = text[::-1]  print("Рядок у зворотному порядку:", reversed\_text) |

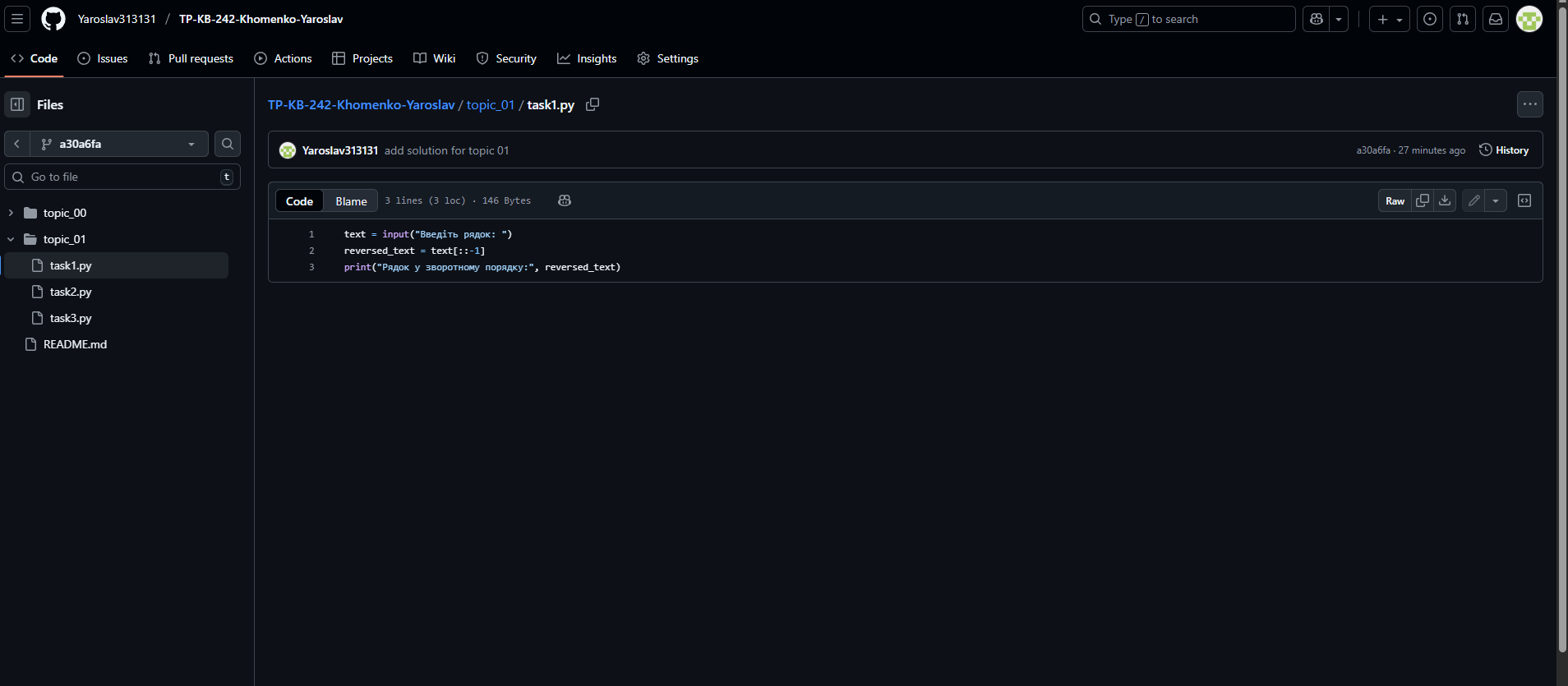
Результат виконання програми:



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:

py = "   python very interesting   "

print("Оригінал:", repr(py))

print("strip():", repr(py.strip()))          # прибирає пробіли

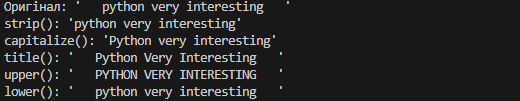
print("capitalize():", repr(py.strip().capitalize()))  # перша літера велика , strip потрібен щоб поршою була літера а не пробіл

print("title():", repr(py.title()))  # кожне слово з великої літери

print("upper():", repr(py.upper()))          # усе великими літерами

print("lower():", repr(py.lower()))          # усе маленькими літерами

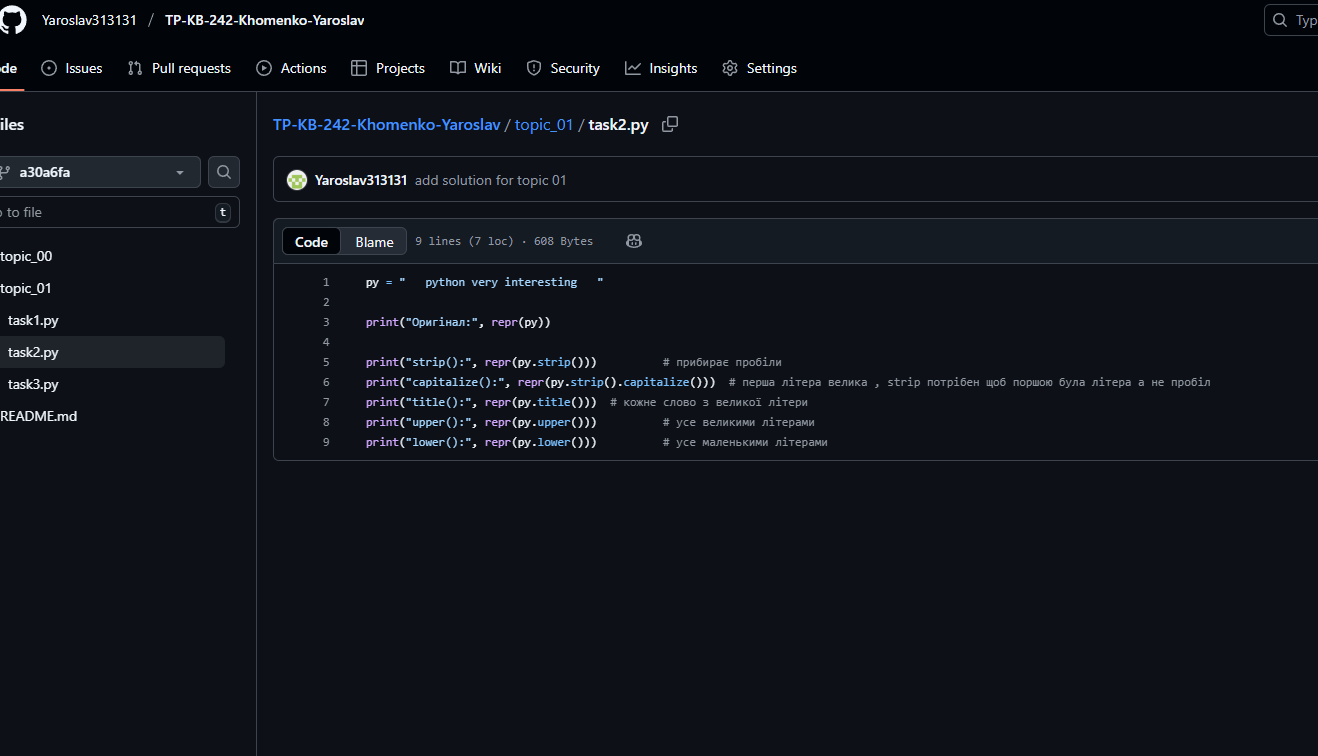
Результат виконання програми:



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):

    return b \* b - 4 \* a \* c

a = int(input("What's A: "))

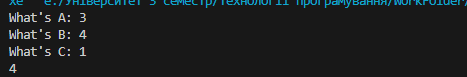
b = int(input("What's B: "))

c = int(input("What's C: "))

D = discr(a, b, c)

print (D)

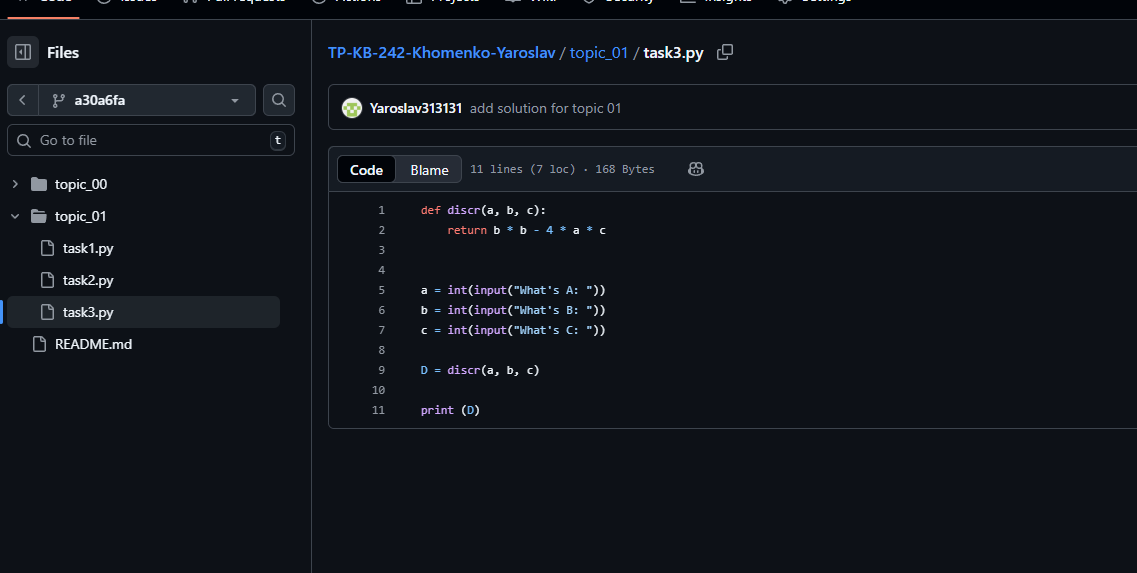
Результат виконання програми:



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 :** Під час виконання практичного завдання я закріпив знання з теми «Функції та змінні» у мові Python. Навчився оголошувати змінні, передавати їх у функції та використовувати оператор для повернення результатів. Практичні вправи дозволили зрозуміти важливість функцій для структуризації програми, підвищення її читабельності та повторного використання коду.

**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):

    return b \* b - 4 \* a \* c

def solve\_quadratic(a, b, c):

    D = discr(a, b, c)

    print("Дискримінант:", D)

    if D > 0:

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

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

        print("Два різних корені: x1 =", x1, ", x2 =", x2)

    elif D == 0:

        x = -b / (2 \* a)

        print("Один корінь: x =", x)

    else:

        print("Коренів немає (D < 0)")

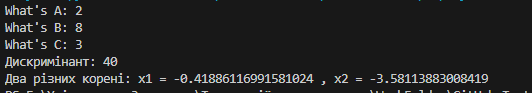
a = int(input("What's A: "))

b = int(input("What's B: "))

c = int(input("What's C: "))

solve\_quadratic(a, b, c)

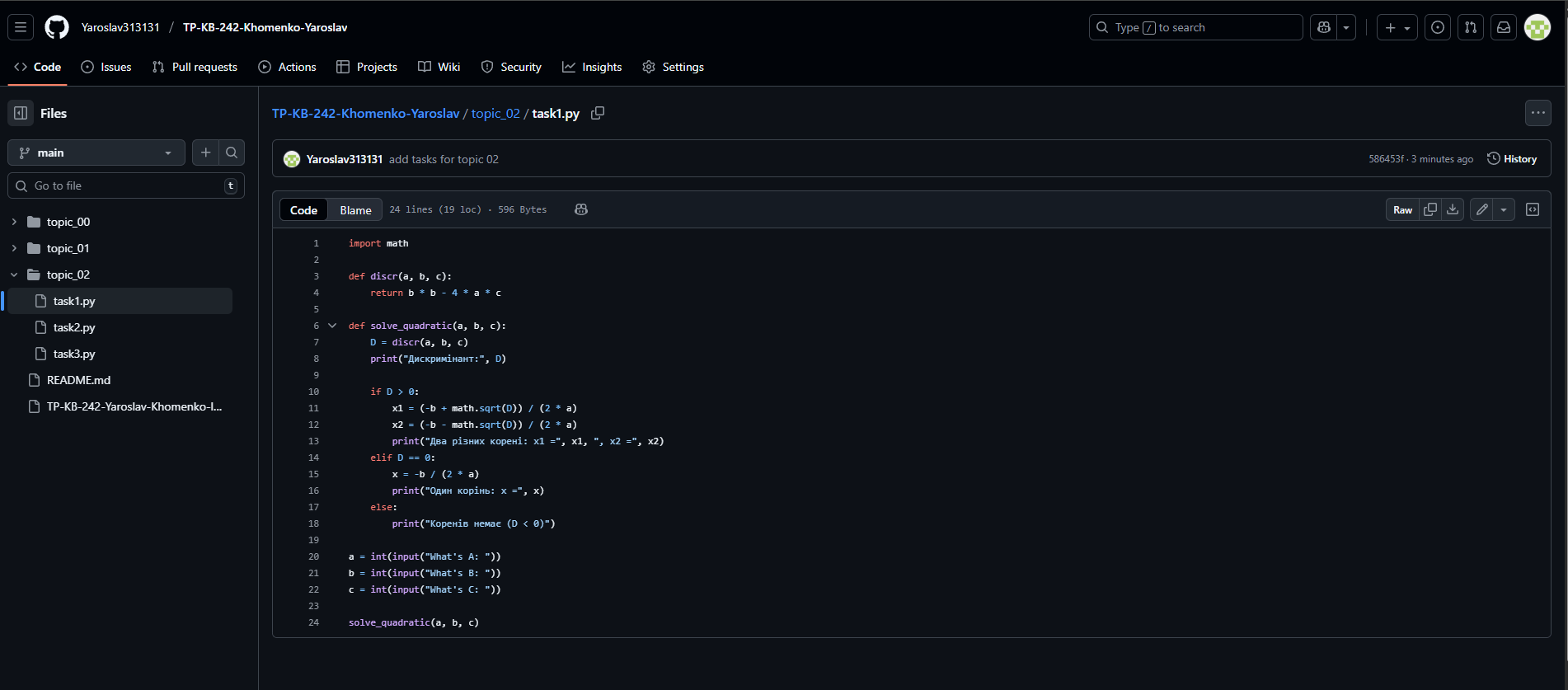
Результат виконання програми:



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 sub(x, y):

    return x - y

def mul(x, y):

    return x \* y

def div(x, y):

    if y != 0:

        return x / y

    else:

        return "Ділення на нуль неможливе!"

op = input("Оберіть операцію: +, -, \*, /: ")

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

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

if op == "+":

    print("Результат:", add(a, b))

elif op == "-":

    print("Результат:", sub(a, b))

elif op == "\*":

    print("Результат:", mul(a, b))

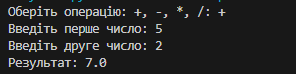
elif op == "/":

    print("Результат:", div(a, b))

else:

    print("Помилка! Невідома операція.")

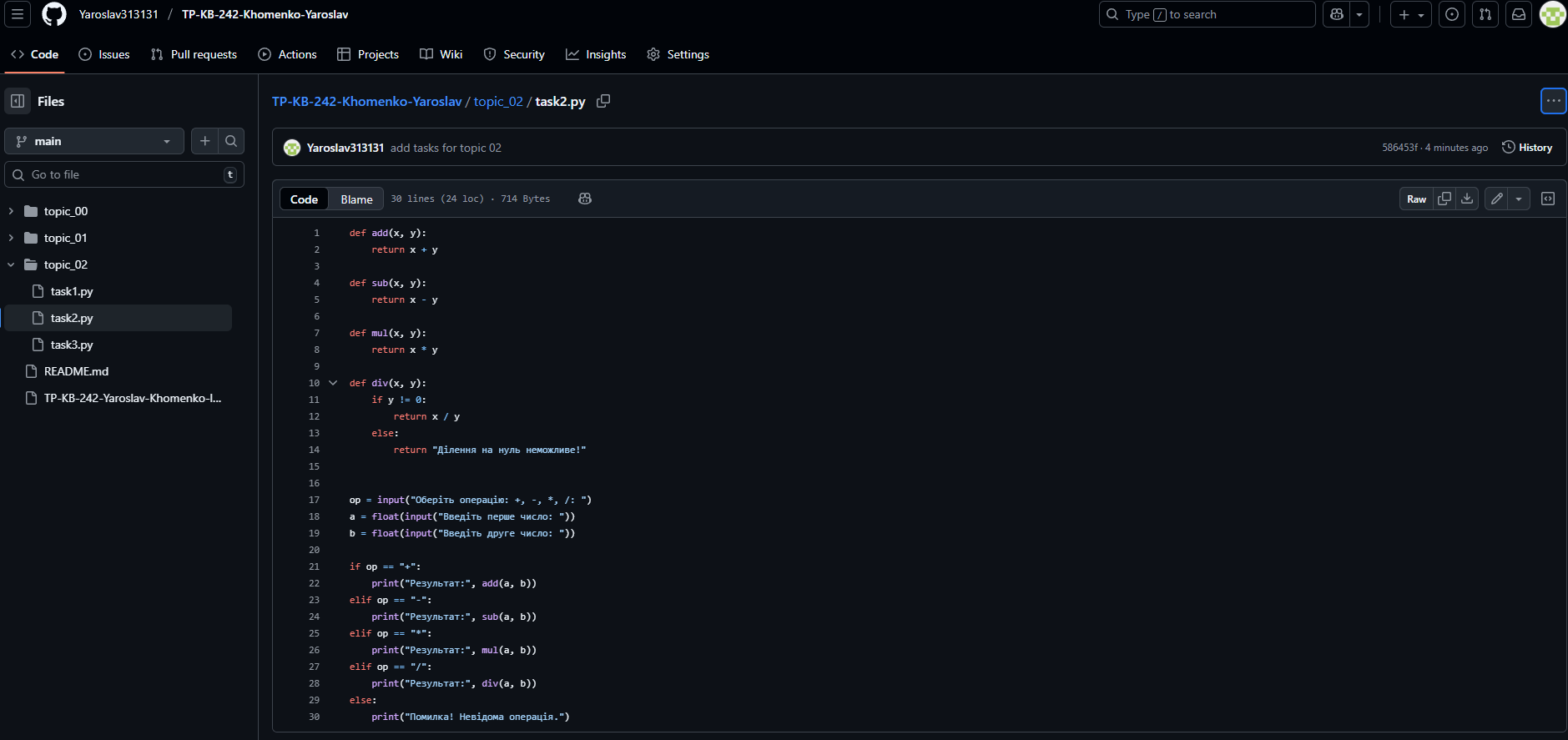
Результат виконання програми:



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):

    return x + y

def sub(x, y):

    return x - y

def mul(x, y):

    return x \* y

def div(x, y):

    if y != 0:

        return x / y

    else:

        return "Ділення на нуль неможливе!"

op = input("Оберіть операцію: +, -, \*, /: ")

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

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

match op:

    case "+":

        print("Результат:", add(a, b))

    case "-":

        print("Результат:", sub(a, b))

    case "\*":

        print("Результат:", mul(a, b))

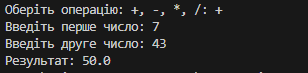
    case "/":

        print("Результат:", div(a, b))

    case \_:

        print("Помилка! Невідома операція.")

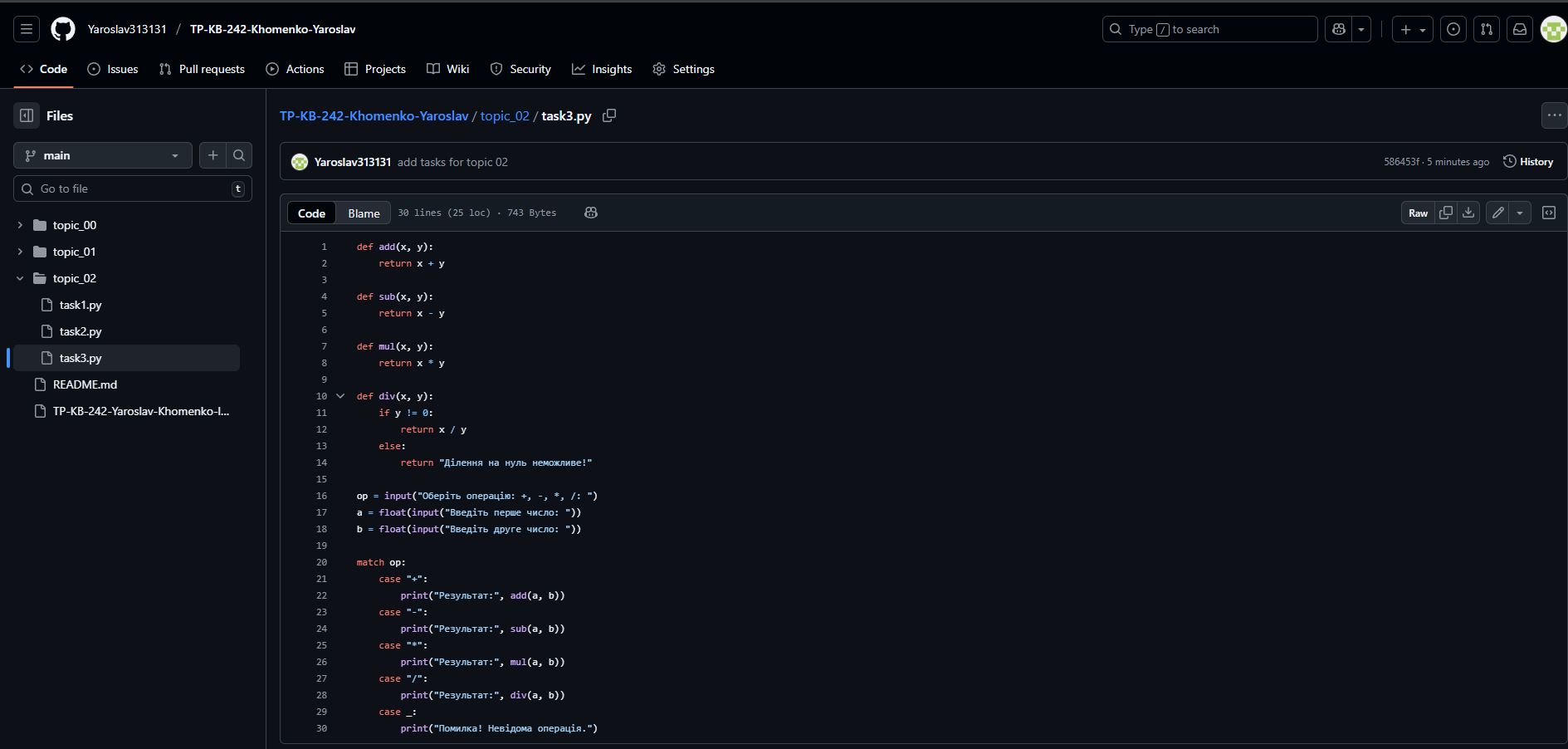
Результат виконання програми:



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** : у ході виконання практичного завдання було закріплено навички роботи з функціями, умовними конструкціями if-else та match-case у Python. Створено програму для обчислення коренів квадратного рівняння з використанням функції розрахунку дискримінанта, а також реалізовано калькулятор двома способами: через умовні переходи if-else та через конструкцію match-case. Отриманий результат показав, що застосування функцій дозволяє зробити код більш структурованим і зручним для використання, а сучасна конструкція match-case спрощує вибір дій у програмі. Таким чином, було досягнуто мети роботи — набуття практичних умінь програмування з використанням базових операторів розгалуження та функцій у 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.**

Текст програми:

def add(x, y):

    return x + y

def sub(x, y):

    return x - y

def mul(x, y):

    return x \* y

def div(x, y):

    if y != 0:

        return x / y

    else:

        return "Ділення на нуль неможливе!"

print("Для завершення введіть: exit")

while True:

    op = input("\nОберіть операцію: +, -, \*, /: ")

    if op == "exit":

        print("Програма завершена.")

        break  # вихід з циклу

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

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

    if op == "+":

        print("Результат:", add(a, b))

    elif op == "-":

        print("Результат:", sub(a, b))

    elif op == "\*":

        print("Результат:", mul(a, b))

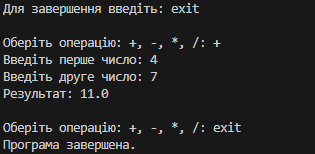
    elif op == "/":

        print("Результат:", div(a, b))

    else:

        print("Помилка! Невідома операція.")

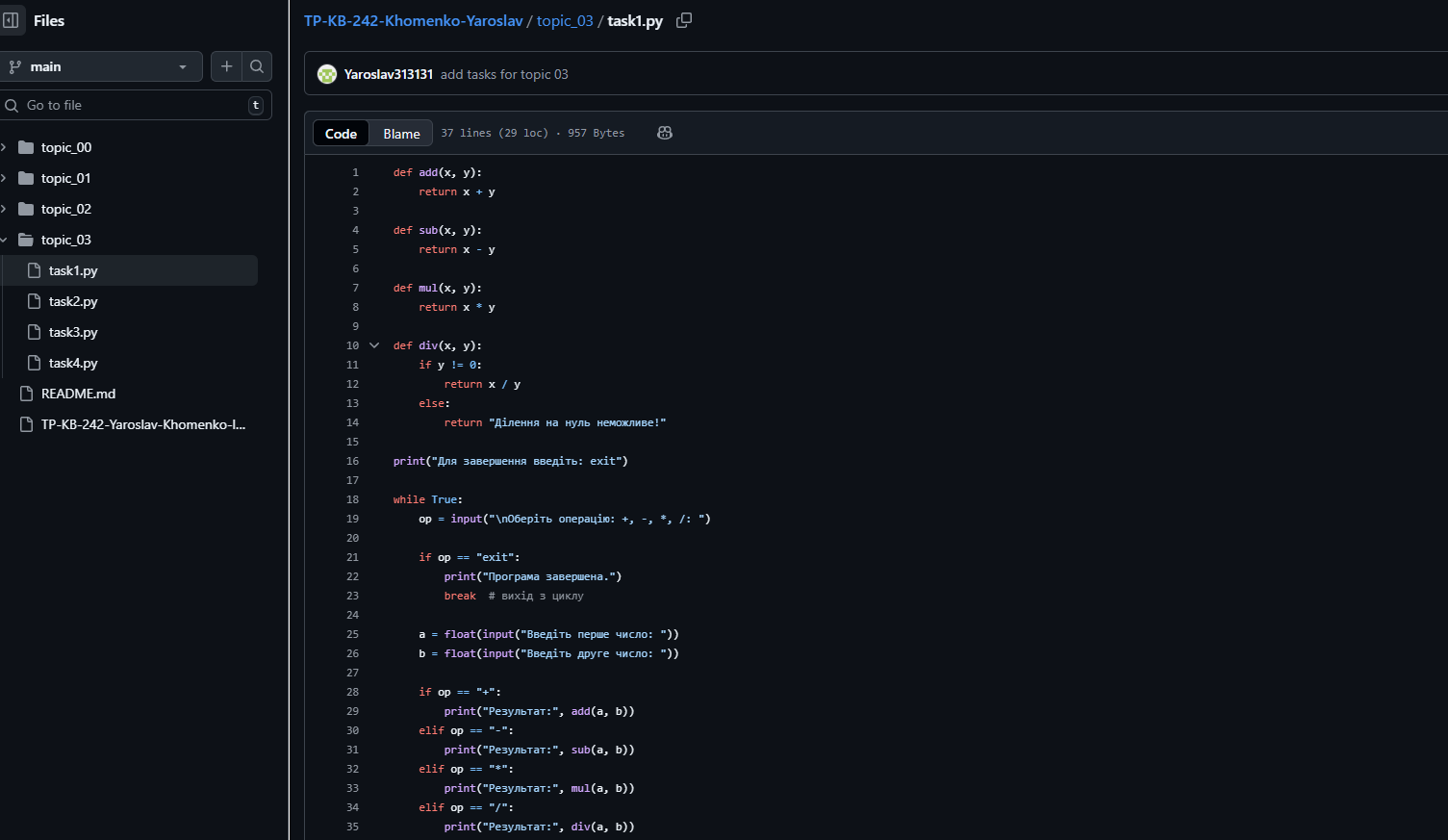
Результат виконання програми:



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:

print("Тестування методів списків \n")

# Початковий список

lst = [5, 2, 8]

print("Початковий список:", lst)

# append() – додає один елемент у кінець

lst.append(10)

print("Після append(10):", lst)

# extend() – додає кілька елементів

lst.extend([20, 30])

print("Після extend([20, 30]):", lst)

# insert() – вставляє елемент у вказану позицію

lst.insert(1, 99)

print("Після insert(1, 99):", lst)

# remove() – видаляє перше входження значення

lst.remove(8)

print("Після remove(8):", lst)

# copy() – створює копію списку

lst\_copy = lst.copy()

print("Копія списку (copy()):", lst\_copy)

# sort() – сортує список за зростанням

lst.sort()

print("Після sort():", lst)

# reverse() – розвертає список

lst.reverse()

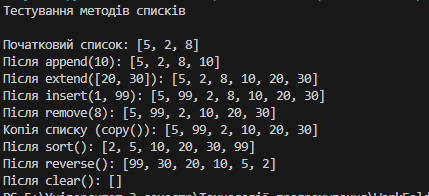
print("Після reverse():", lst)

# clear() – очищає список

lst.clear()

print("Після clear():", lst)

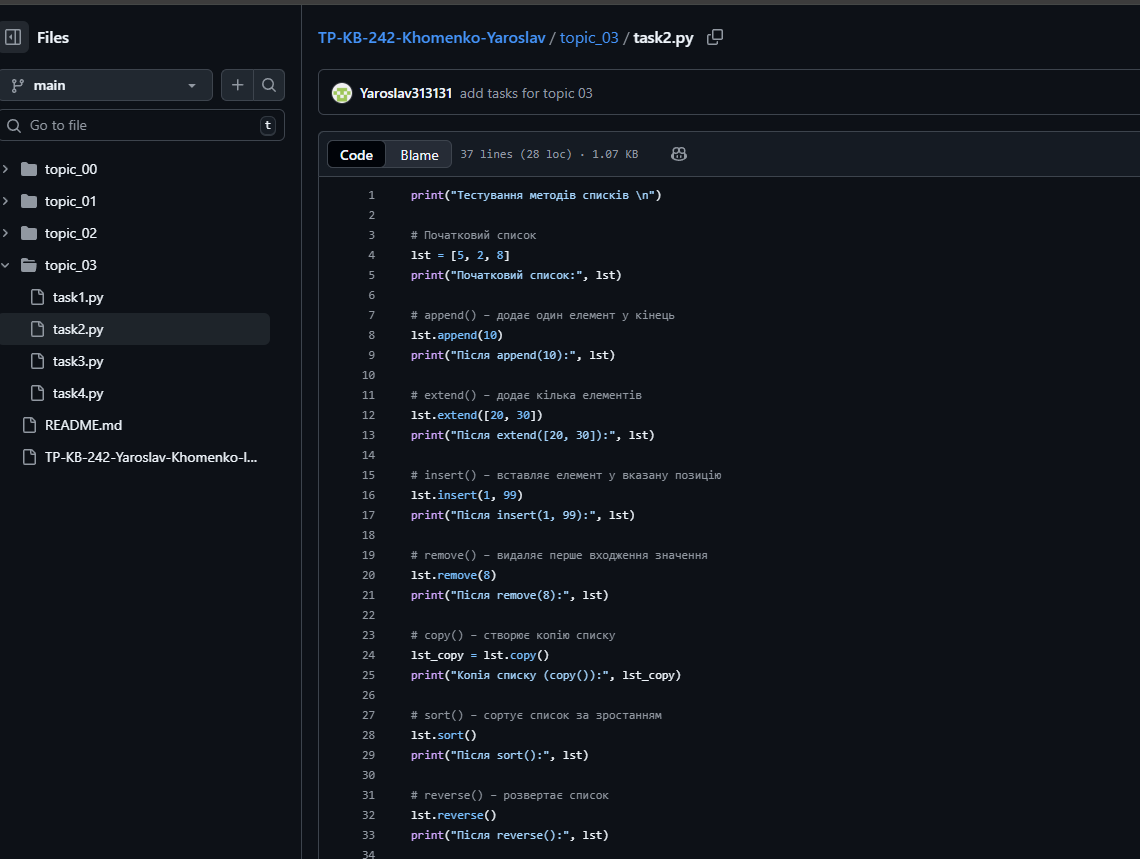
Результат виконання програми:



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:

print("Тестування методів словників \n")

# Початковий словник

d = {"name": "Yaroslav", "age": 19, "city": "Chernihiv"}

print("Початковий словник:", d)

# update() – додає або змінює елементи

d.update({"age": 21, "country": "Ukraine"})

print("Після update({'age': 21, 'country': 'Ukraine'}):", d)

# del – видаляє елемент за ключем

del d["city"]

print("Після del d['city']:", d)

# keys() – повертає всі ключі

print("Ключі словника (keys()):", d.keys())

# values() – повертає всі значення

print("Значення словника (values()):", d.values())

# items() – повертає пари (ключ, значення)

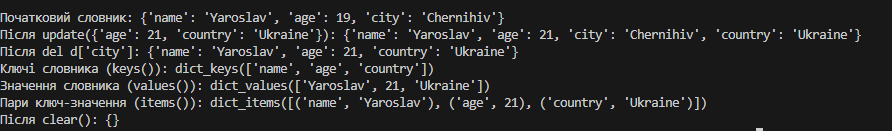
print("Пари ключ-значення (items()):", d.items())

# clear() – очищає словник

d.clear()

print("Після clear():", d)

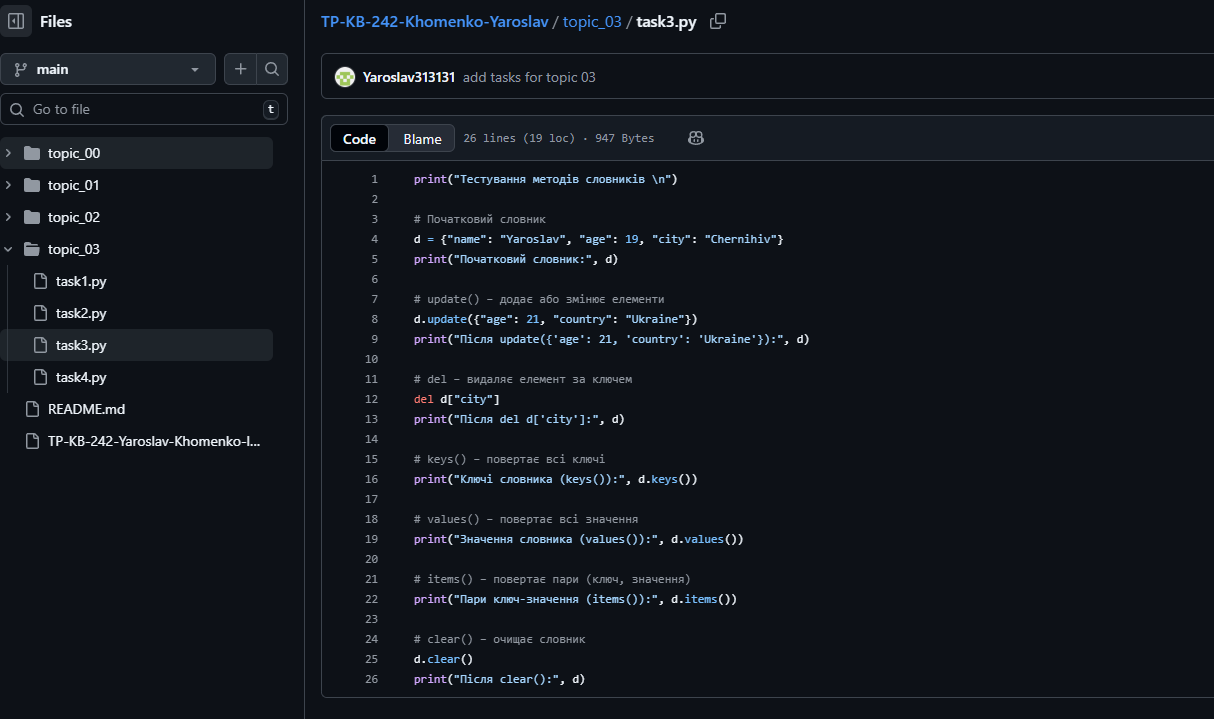
Результат виконання програми:



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

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

print("Список:", lst)

val = int(input("Введіть число для вставки: "))

pos = bisect.bisect\_left(lst, val)  # знаходимо позицію для вставки

print(f"Елемент {val} слід вставити на позицію {pos}")

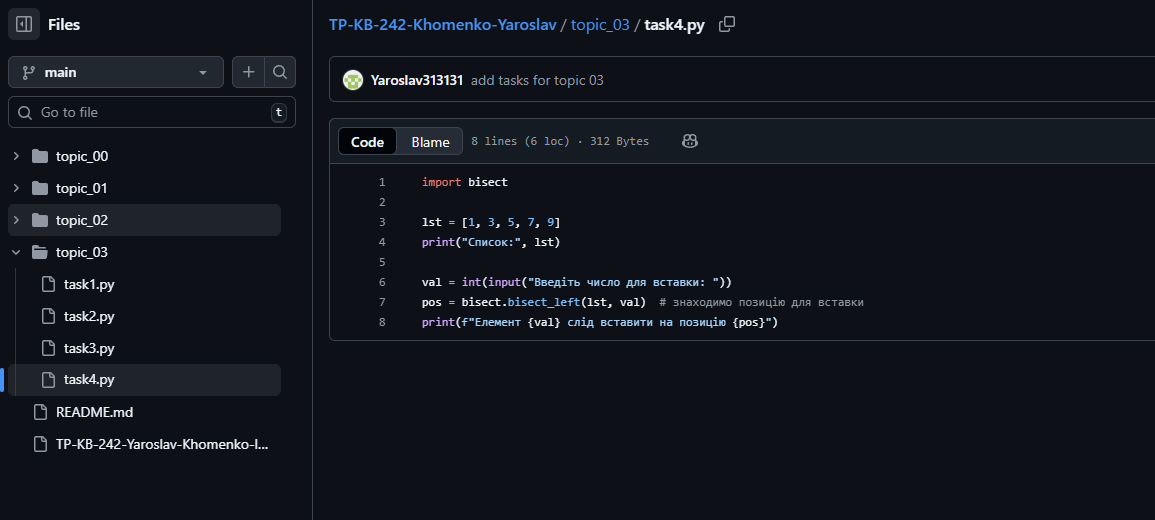
Результат виконання програми:



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** : у ході виконання завдань було закріплено навички роботи з Python: створення інтерактивного калькулятора з циклом та завершенням за командою користувача, тестування основних методів списків (append(), extend(), insert(), remove(), clear(), sort(), reverse(), copy()) та словників (update(), del, clear(), keys(), values(), items()), а також реалізацію функції пошуку позиції для вставки нового елемента у відсортований список. Завдання сприяли розвитку логічного мислення та практичних навичок роботи з базовими структурами даних.