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

Посилання на github   
<https://github.com/TP-KB-242-Vadim-Hovdiy/TP-KB-242-Vadim-Hovdiy.git>

Звіт до Теми №1

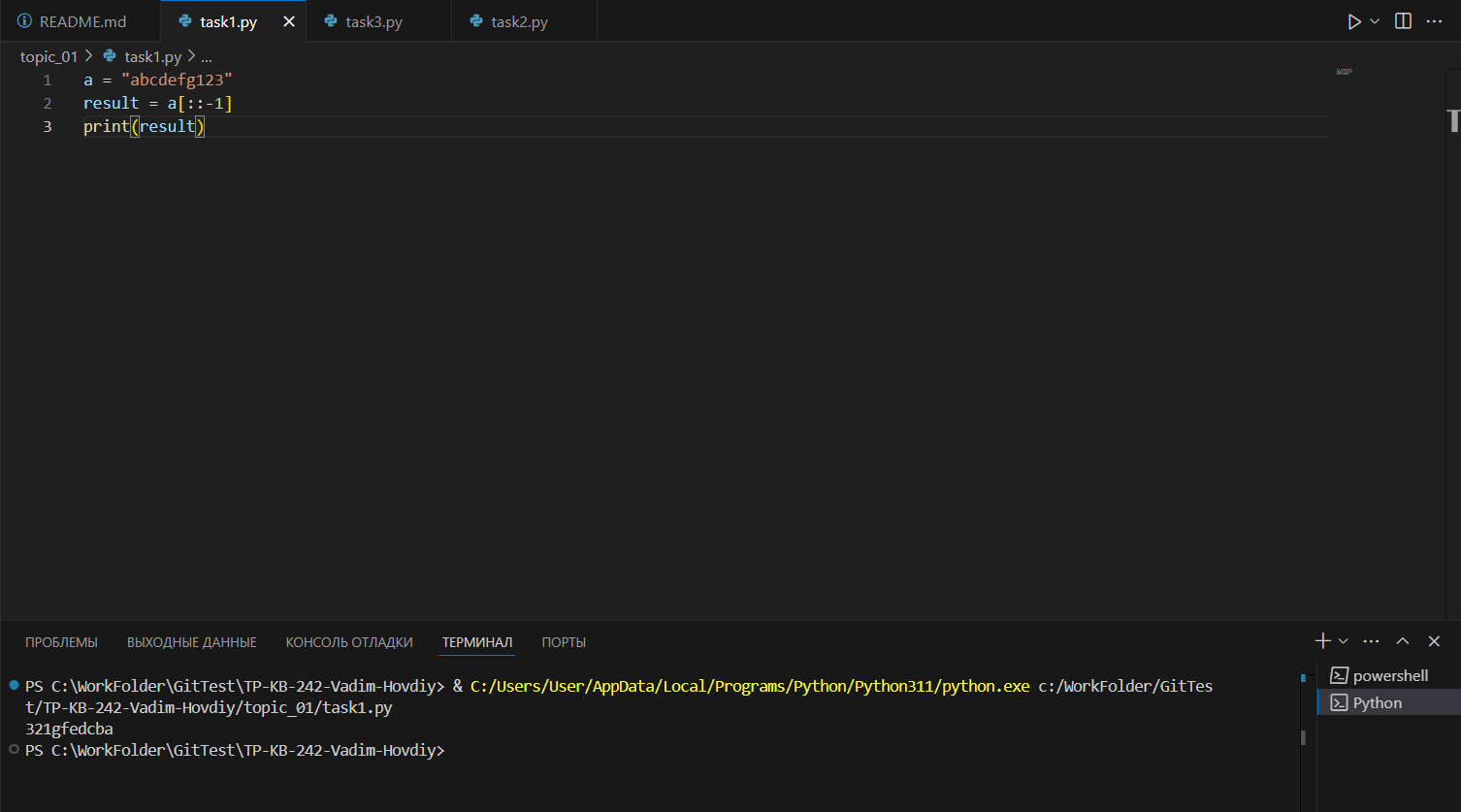
Функції та змінні

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

1) Необхідно рядок, що має вигляд "abcdefg123" перетворити наступним чином "321gfedcba", вважаючи сталою довжину рядку в 10 символів.

Код виконання завдання:

|  |
| --- |
| a = "abcdefg123"  result = a[::-1]  print(result) |

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

# strip() – прибирає пробіли на початку та в кінці

text1 = "   hello world   "

print("text1():", text1.strip())

# capitalize() – робить першу літеру великою, решта стають маленькими

text2 = "hello world"

print("text2():", text2.capitalize())

# title() – робить першу літеру кожного слова великою

text3 = "hello world"

print("text3():", text3.title())

# upper() – всі букви стають великими

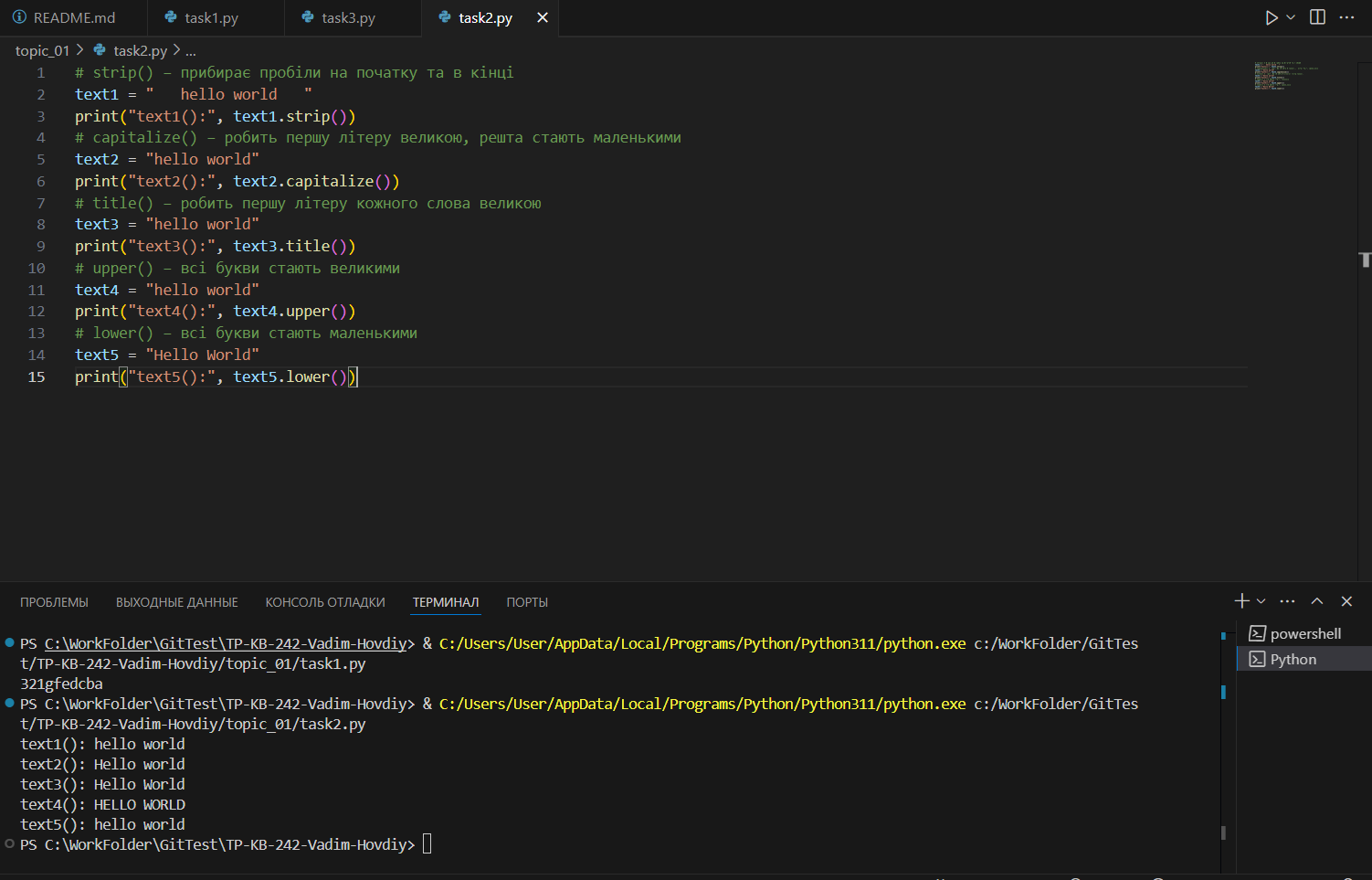
text4 = "hello world"

print("text4():", text4.upper())

# lower() – всі букви стають маленькими

text5 = "Hello World"

print("text5():", text5.lower())

Результат коду: 

3) Написати функцію пошуку дискримінанту квадратного рівняння.

Код завдання:

def disc(a, b, c):

    return b\*b - 4\*a\*c

a = int(input("Значення A: "))

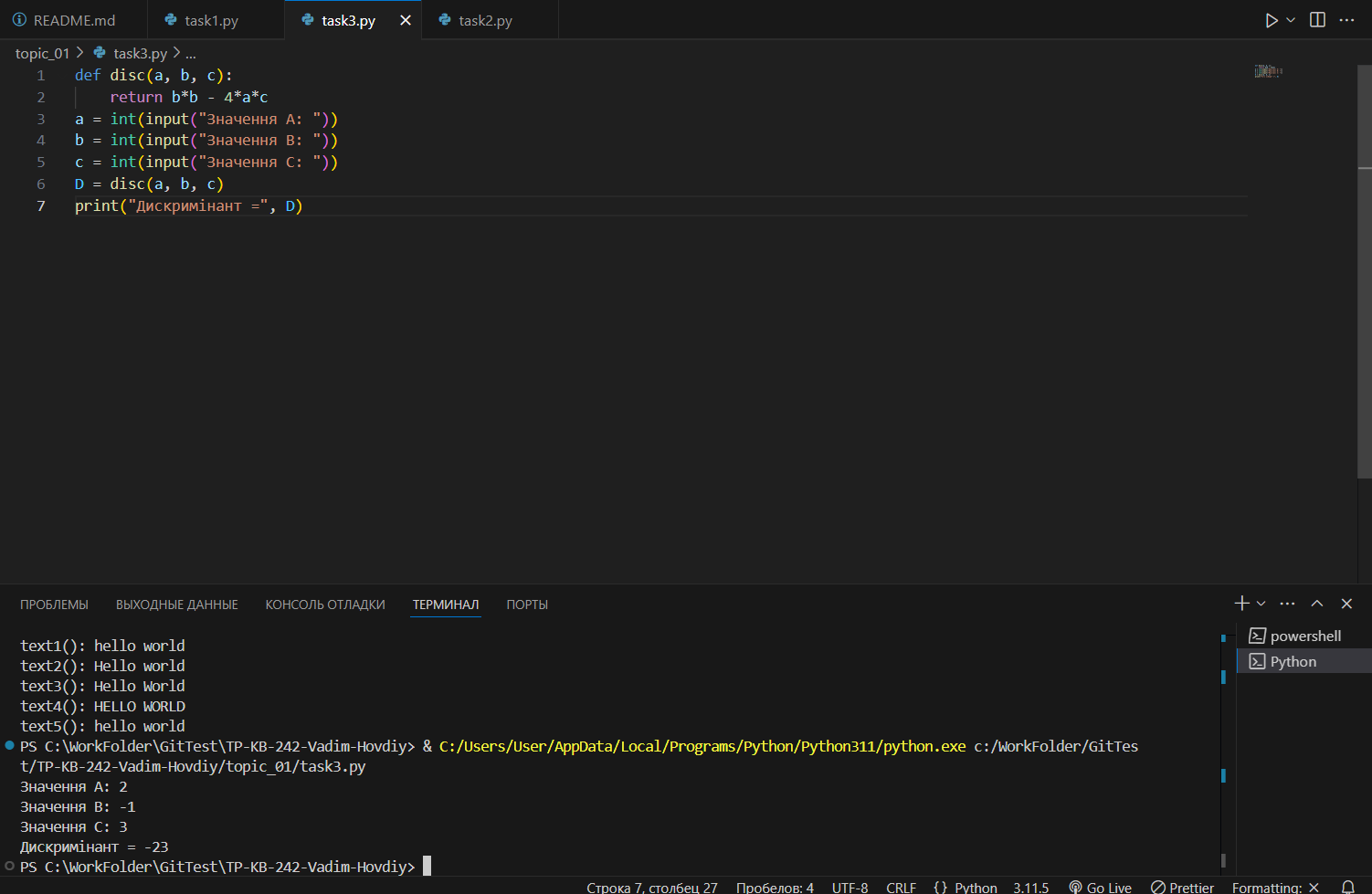
b = int(input("Значення B: "))

c = int(input("Значення C: "))

D = disc(a, b, c)

print("Дискримінант =", D)

Результат коду:



Звіт до Теми №2

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

Код функції:

def disc(a, b, c):

    return b\*b - 4\*a\*c

a = int(input("Значення A: "))

b = int(input("Значення B: "))

c = int(input("Значення C: "))

D = disc(a, b, c)

print("Дискримінант =", D)

def roots(a, b, c):

    D = disc(a, b, c)

    if D > 0:

        x1 = (-b + D\*\*0.5) / (2\*a)

        x2 = (-b - D\*\*0.5) / (2\*a)

        return (x1, x2)

    elif D == 0:

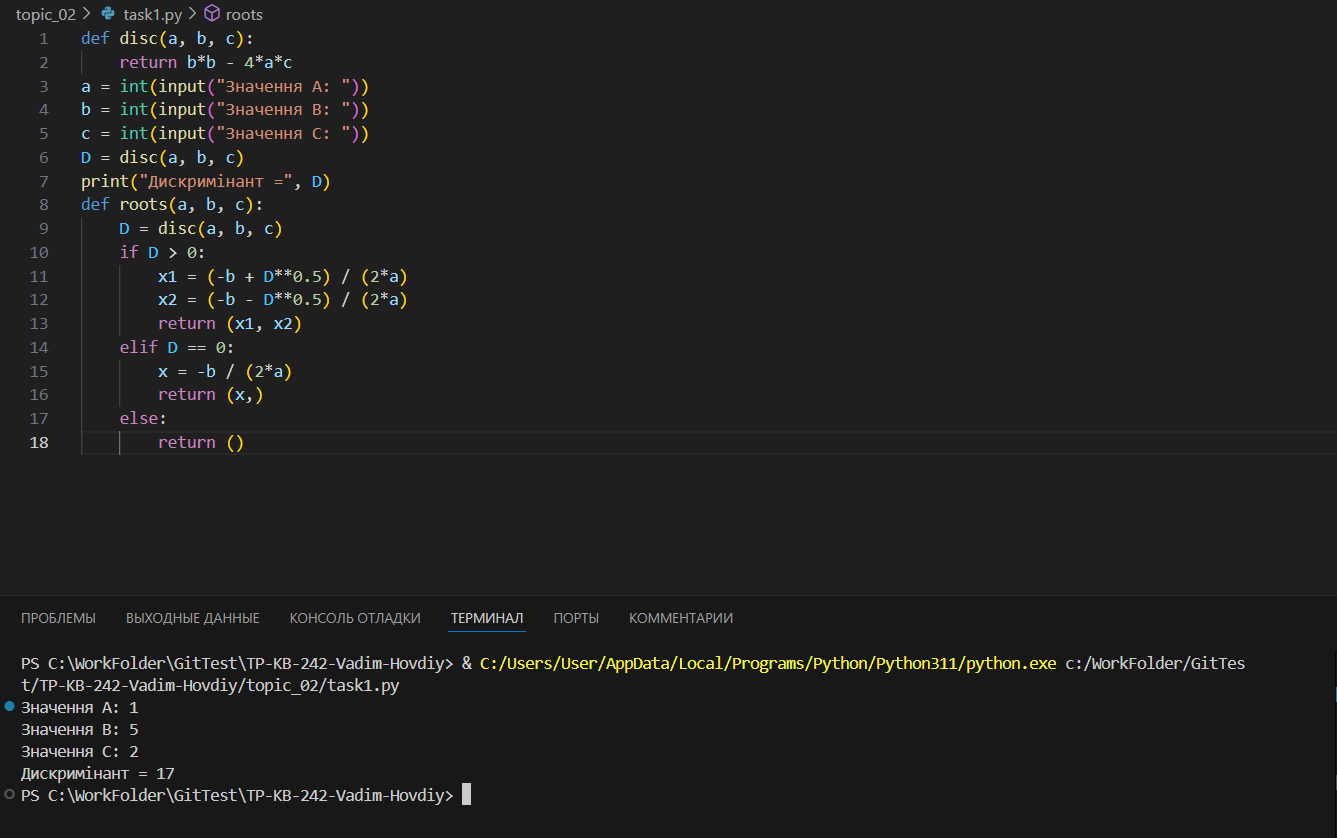
        x = -b / (2\*a)

        return (x,)

    else:

        return ()

Результат функції:



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

Код програми:

def plus(x, y):

    return x + y

def minus(x, y):

    return x - y

def multiply(x, y):

    return x \* y

def divide(x, y):

    if y != 0:

        return x / y

    else:

        return "На нуль ділити не можна"

print("Оберіть операцію:")

print("1 - Додавання (+)")

print("2 - Віднімання (-)")

print("3 - Множення (\*)")

print("4 - Ділення (/)")

choice = input("Введіть номер операції (1/2/3/4): ")

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

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

if choice == "1":

    print("Результат:", plus(a, b))

elif choice == "2":

    print("Результат:", minus(a, b))

elif choice == "3":

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

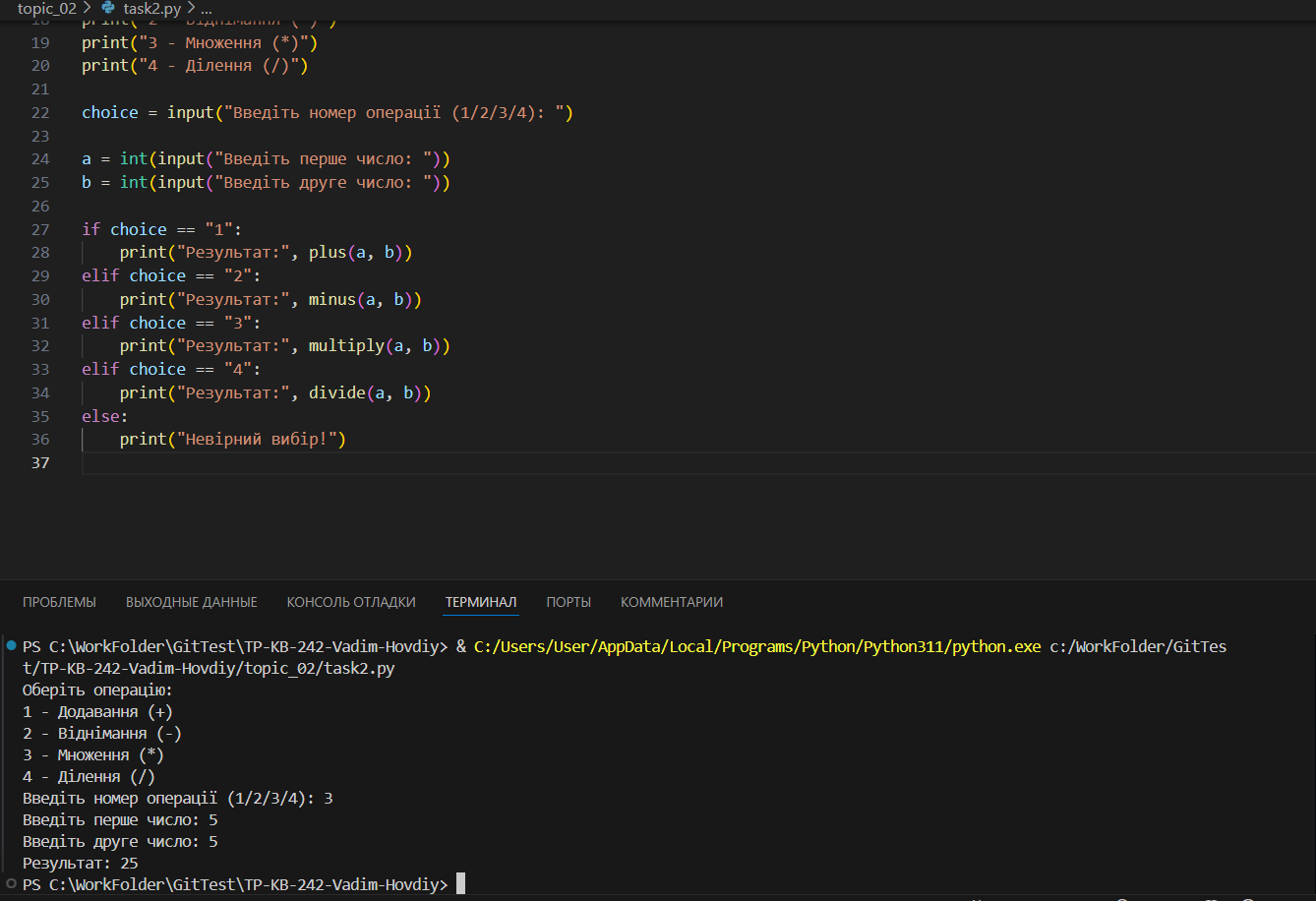
elif choice == "4":

    print("Результат:", divide(a, b))

else:

    print("Невірний вибір!")

Результат програми:



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

Код програми:

def plus(a, b):

    return a + b

def minus(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b == 0:

        return "Помилка: ділення на нуль!"

    return a / b

def calculator():

    print("Доступні операції: +, -, \*, /")

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

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

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

    match op:

        case "+":

            result = plus(a, b)

        case "-":

            result = minus(a, b)

        case "\*":

            result = multiply(a, b)

        case "/":

            result = divide(a, b)

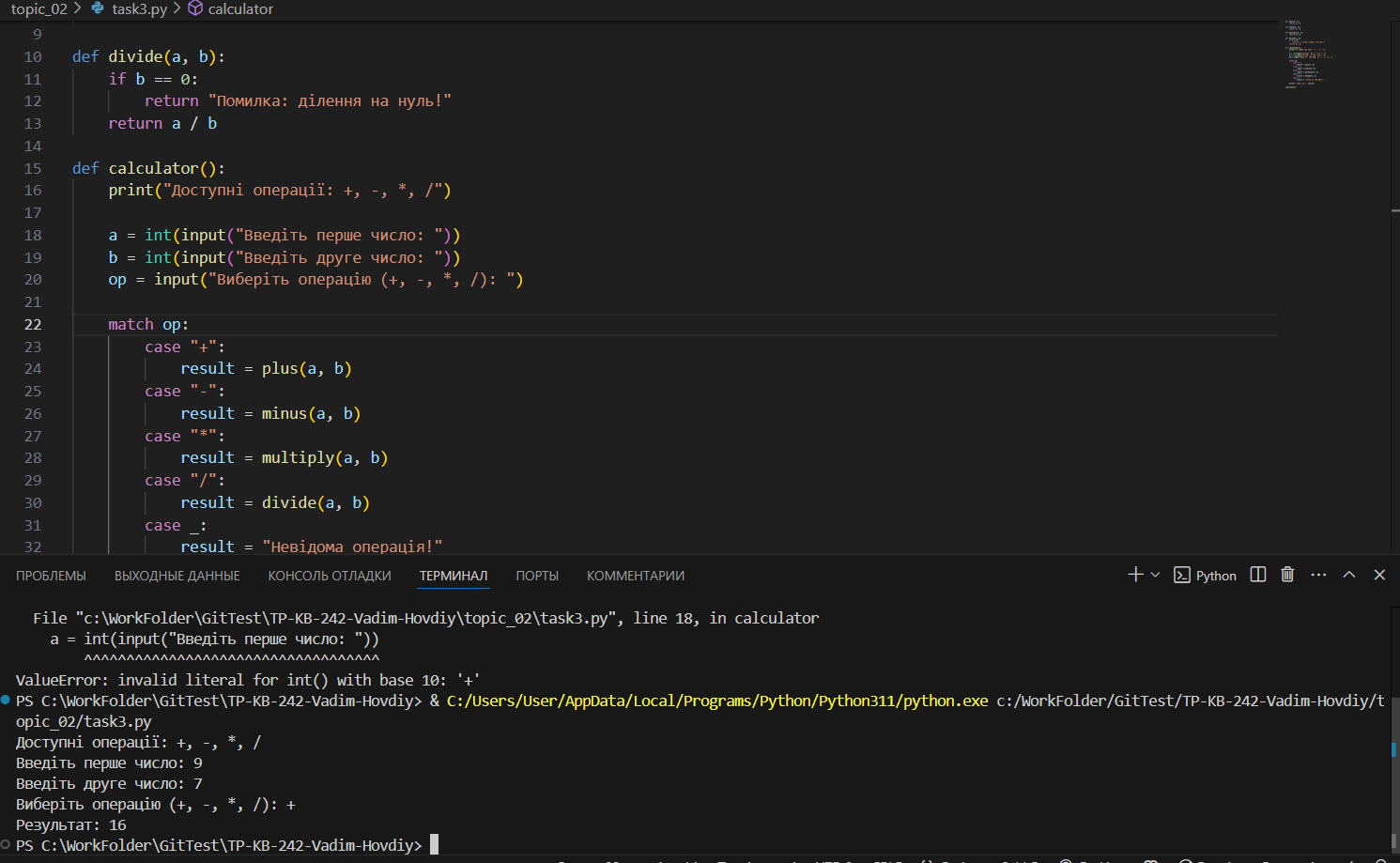
        case \_:

            result = "Невідома операція!"

    print("Результат:", result)

calculator()

Результат коду:



Висновок: У ході виконання практичних завдань було створено три програми на мові Python. У першому завданні реалізовано функцію пошуку коренів квадратного рівняння з використанням допоміжної функції для обчислення дискримінанта та умовних переходів для визначення кількості коренів. У другому завданні розроблено простий калькулятор з використанням конструкцій if...else, де кожна арифметична операція винесена в окрему функцію. У третьому завданні було використано сучасну конструкцію match...case, що зробило програму більш структурованою та зручною для розширення.

Звіт до Теми №3

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.

Код програми:

def plus(a, b):

    return a + b

def minus(a, b):

    return a - b

def multiply(a, b):

    return a \* b

def divide(a, b):

    if b == 0:

        return "Помилка: ділення на нуль!"

    return a / b

def calculator():

    print("Доступні операції: +, -, \*, /")

    print("Введіть 'exit' для завершення програми.")

    while True:

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

        if op == "exit":

            print("Програма завершена.")

            break

        try:

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

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

        except ValueError:

            print("Помилка: введено не число!")

            continue

        match op:

            case "+":

                result = plus(a, b)

            case "-":

                result = minus(a, b)

            case "\*":

                result = multiply(a, b)

            case "/":

                result = divide(a, b)

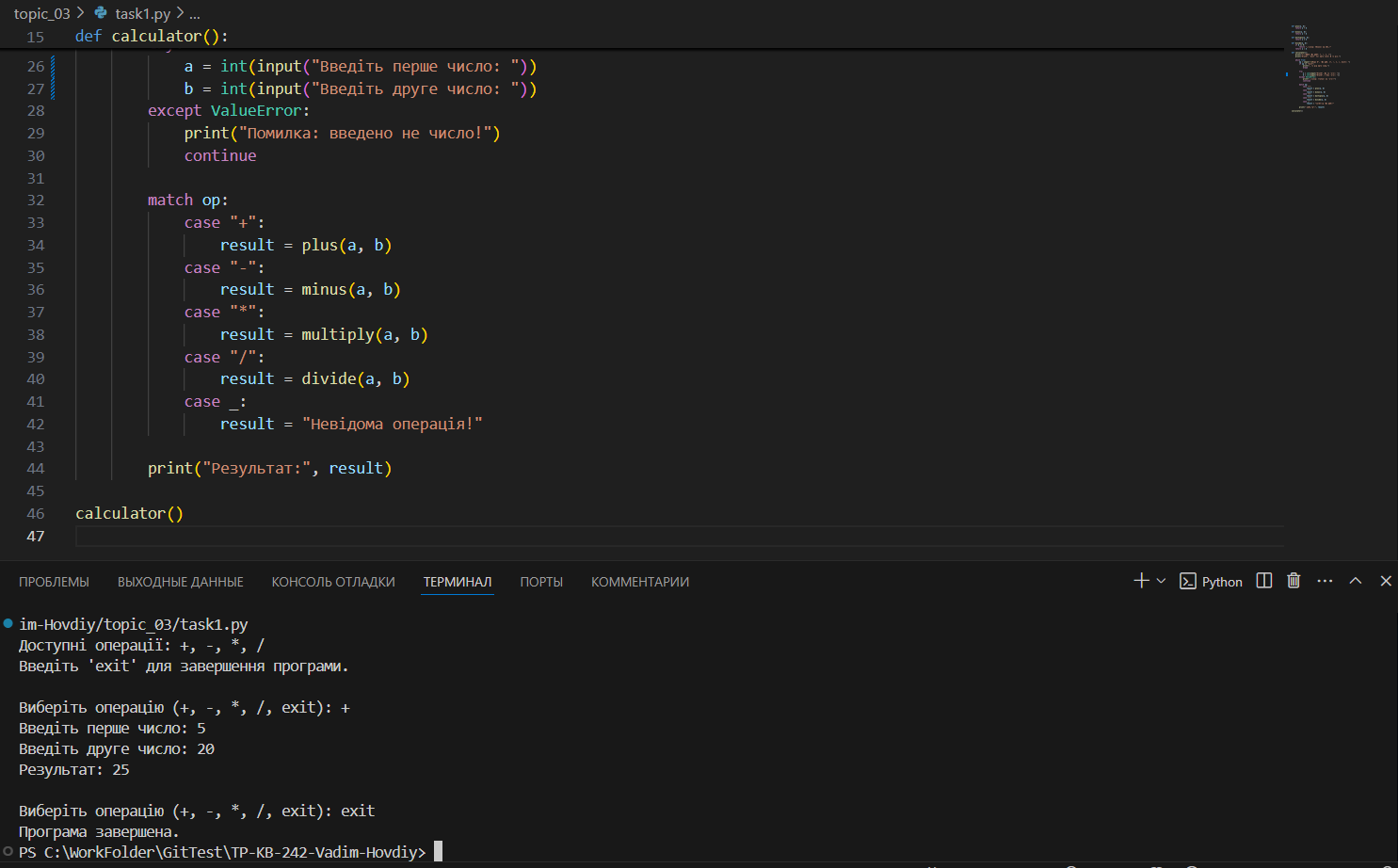
            case \_:

                result = "Невідома операція!"

        print("Результат:", result)

calculator()

Результат коду:



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

Код завдання:

def test():

    my\_list = [1, 2, 3]

    print("Початковий список:", my\_list)

    my\_list.append(4)

    print("append(4):", my\_list)

    my\_list.extend([5, 6])

    print("extend([5, 6]):", my\_list)

    my\_list.insert(2, 99)

    print("insert(2, 99):", my\_list)

    my\_list.remove(99)

    print("remove(99):", my\_list)

    copy\_list = my\_list.copy()

    print("copy():", copy\_list)

    my\_list.sort(reverse=True)

    print("sort(reverse=True):", my\_list)

    my\_list.reverse()

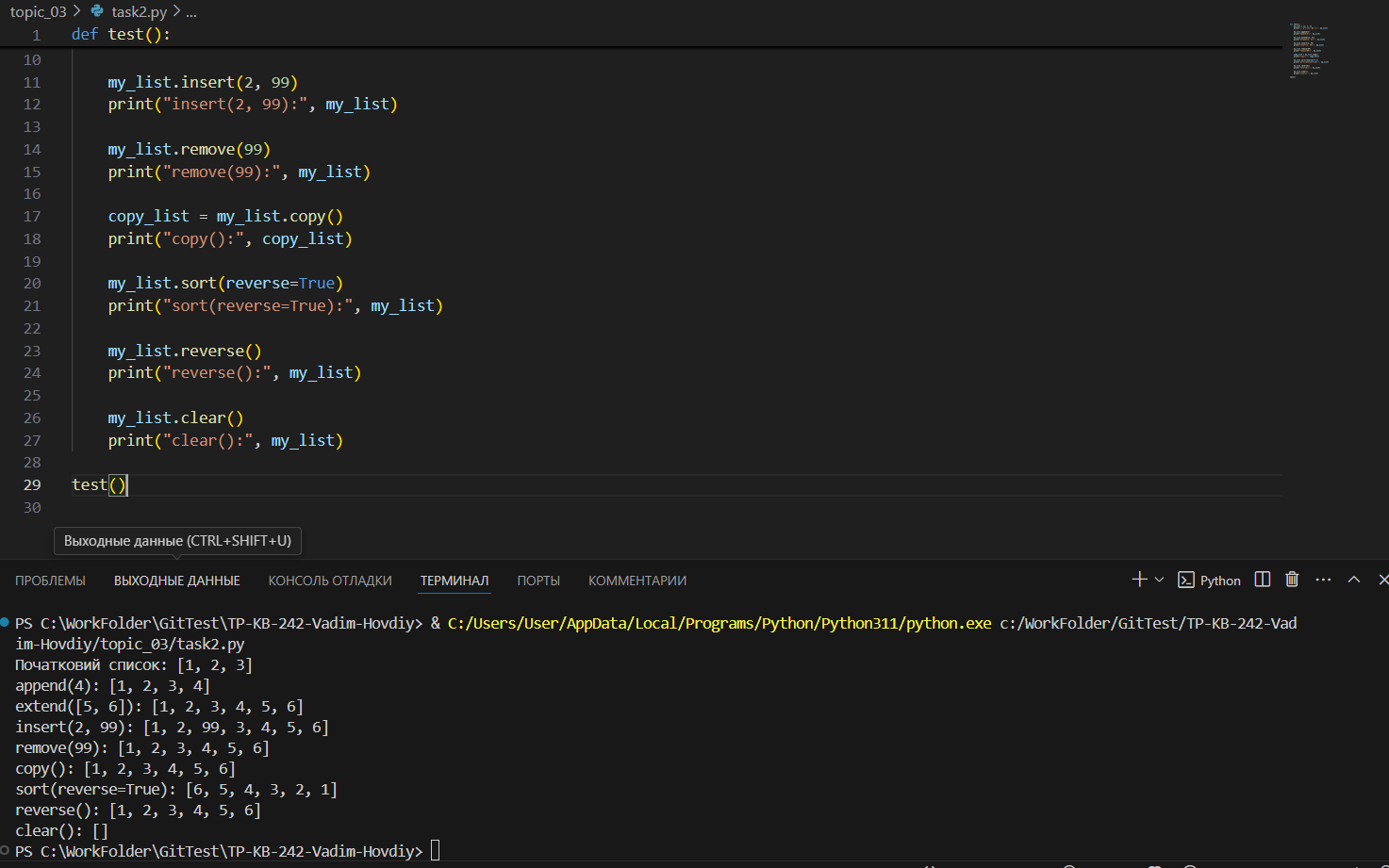
    print("reverse():", my\_list)

    my\_list.clear()

    print("clear():", my\_list)

test()

Результат коду:



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

Код програми:

def test\_dict\_functions():

    first\_function = {"a": 1, "b": 2, "c": 3}

    print("Початковий словник:", first\_function)

    first\_function.update({"d": 4})

    print("update({'d': 4}):", first\_function)

    del first\_function["a"]

    print("del first\_function['a']:", first\_function)

    print("keys():", list(first\_function.keys()))

    print("values():", list(first\_function.values()))

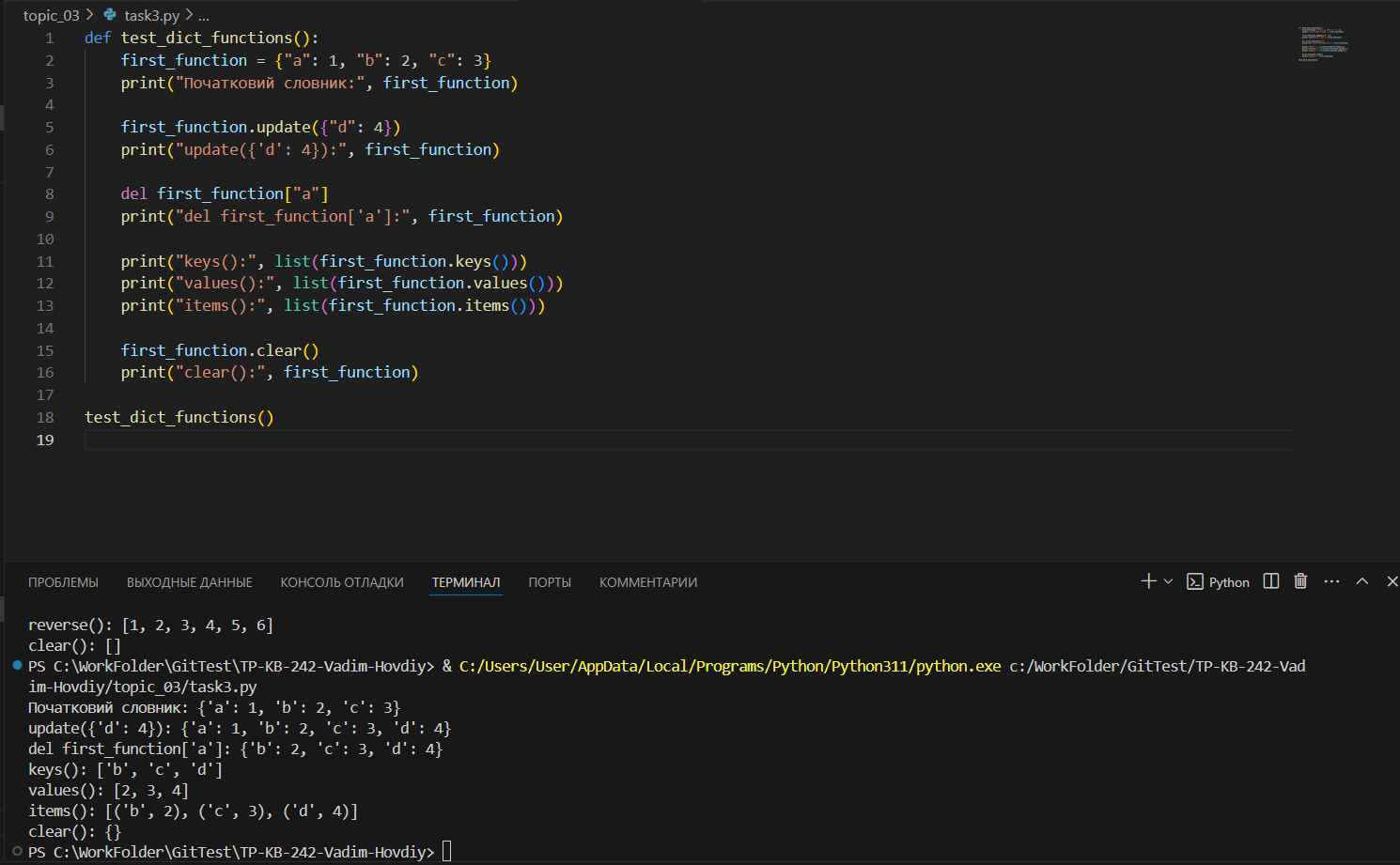
    print("items():", list(first\_function.items()))

    first\_function.clear()

    print("clear():", first\_function)

test\_dict\_functions()

Результат коду:



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

Код функції:

def find(sorted\_list, element):

    for i, val in enumerate(sorted\_list):

        if element < val:

            return i

    return len(sorted\_list)  # вставка в кінець

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

print("Список:", nums)

x = 6

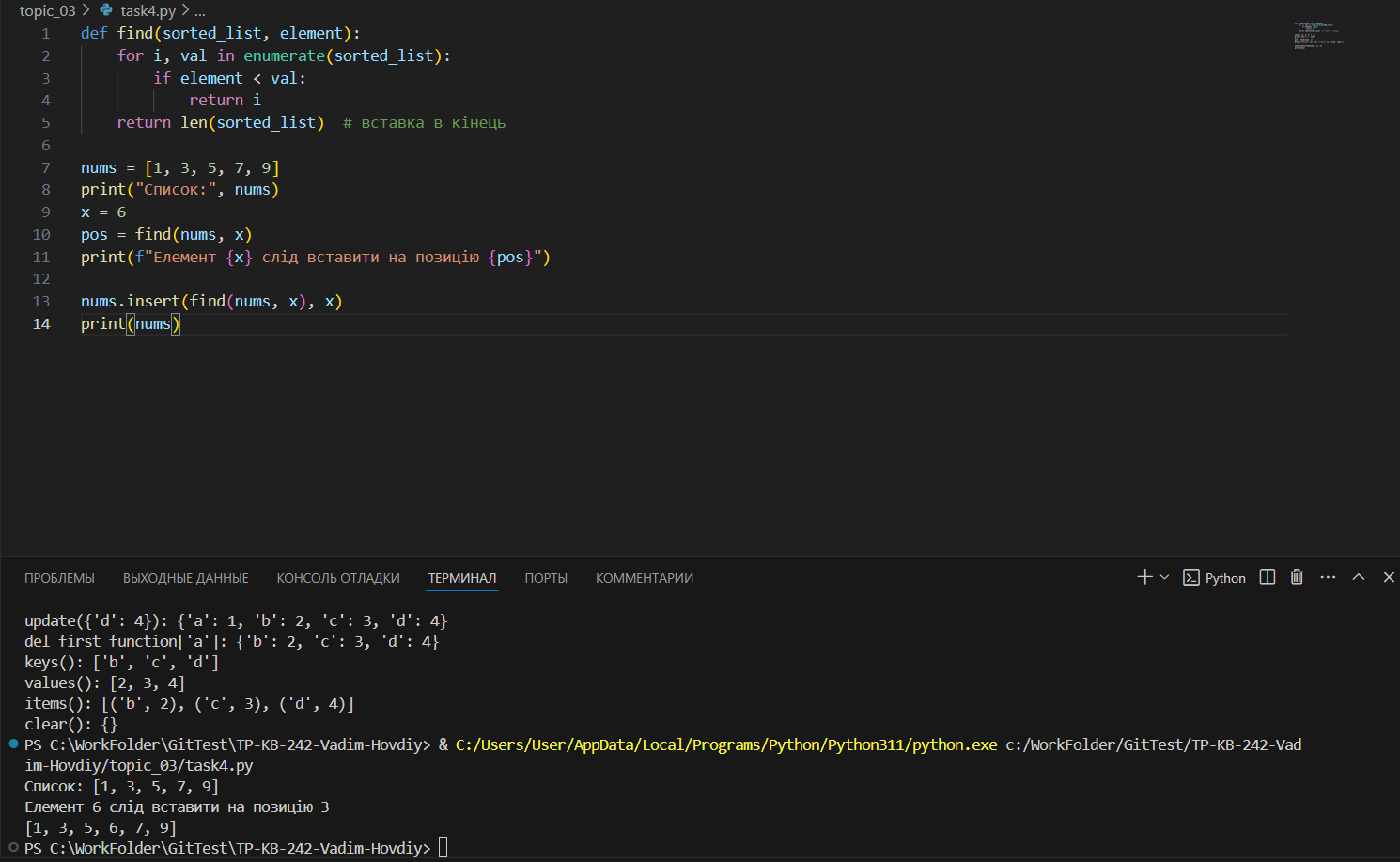
pos = find(nums, x)

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

nums.insert(find(nums, x), x)

print(nums)

Результат коду:



Висновок: у процесі виконання роботи було закріплено практичні навички роботи з циклами, умовними операторами, списками, словниками та алгоритмами пошуку, що є важливим для подальшого вивчення програмування.