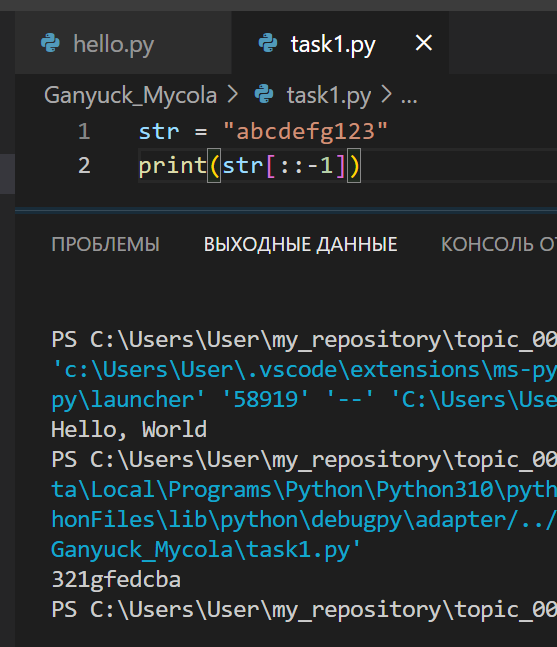
Ганюк Микола КБ-221

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

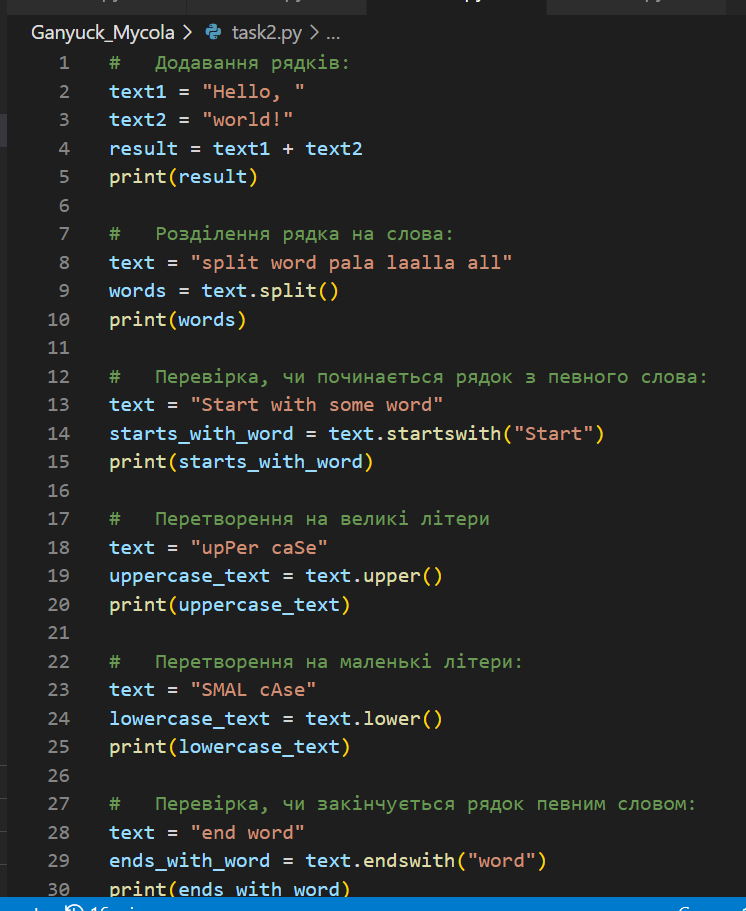
Тема 01

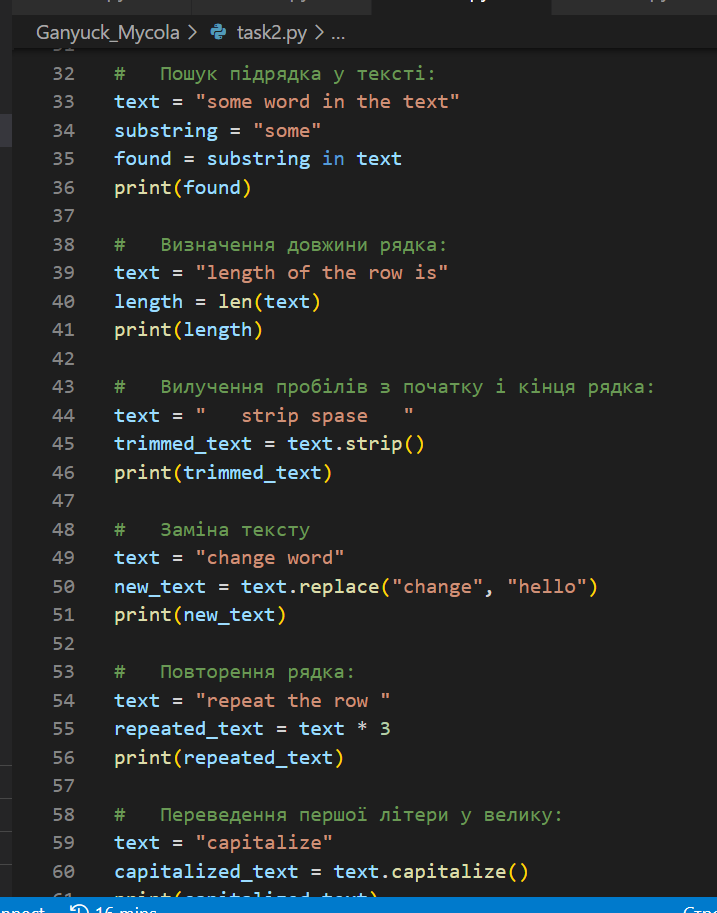
1. Перетворення рядка

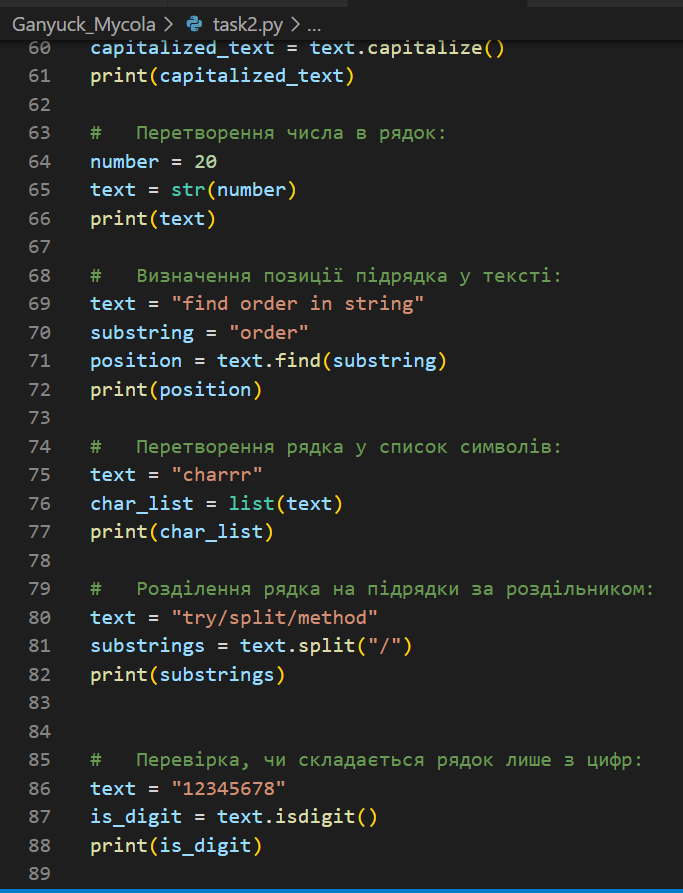
Необхідно взяти рядок, що має вигляд «abcdefg123» та перетворити на «321gfedcba»

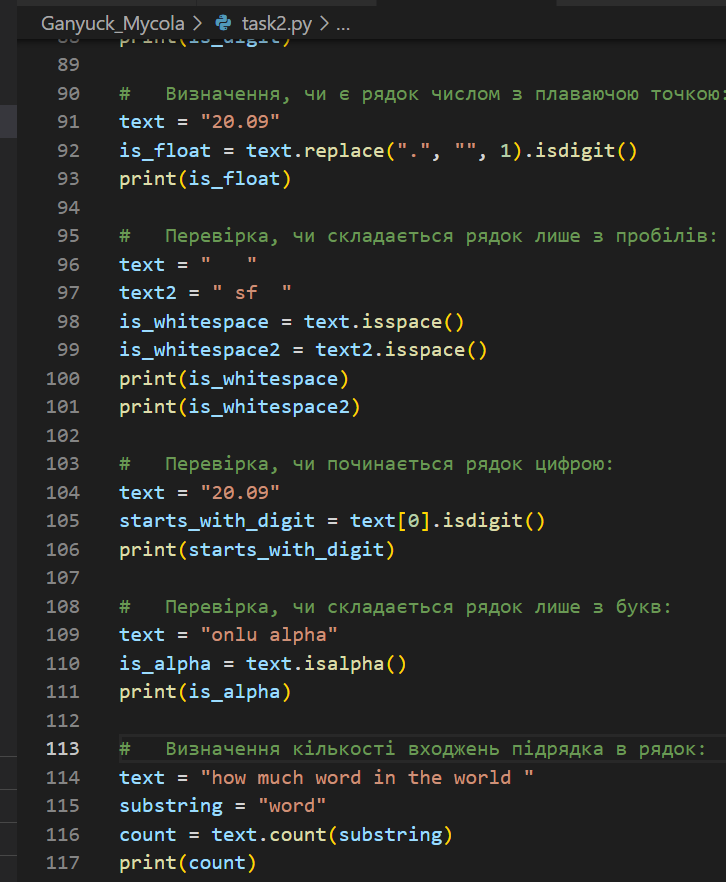


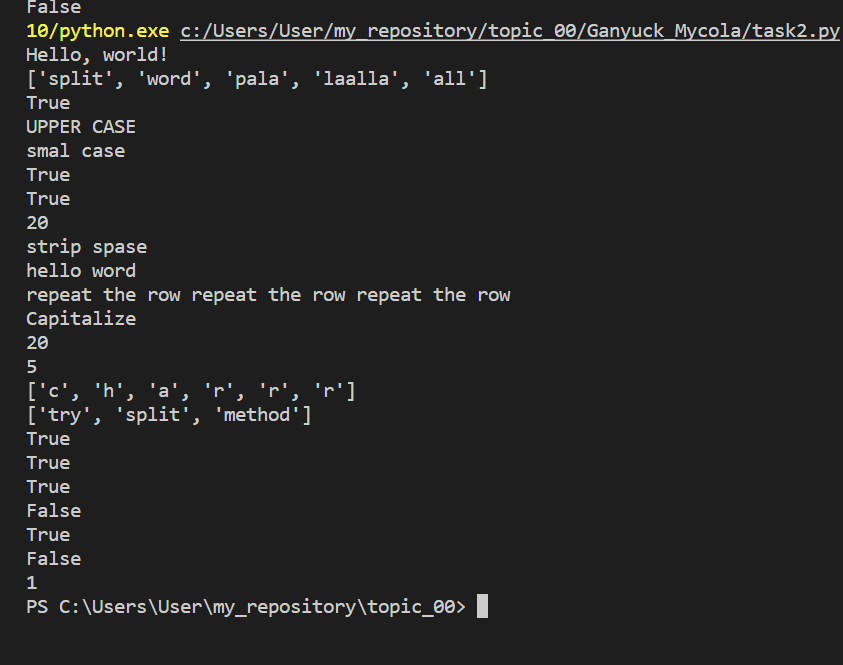
1. Тестування базових функцій для рядків
2. **split -** Розділення рядка на слова Розділення рядка на підрядки за роздільником
3. **replace -**Заміна тексту Видалення символу з рядка
4. **upper -** Перетворення на великі літери
5. **lower -** Перетворення на маленькі літери
6. **startwith -** Перевірка, чи починається рядок з певного слова
7. **endwith -** Перевірка, чи закінчується рядок певним словом
8. **substring in -** Пошук підрядка у тексті
9. **len -** Визначення довжини рядка
10. **strip -** Вилучення пробілів з початку і кінця рядка
11. **\* -** Повторення рядка
12. **f -** Форматування рядка
13. **str -** Перетворення числа в рядок
14. **find -** Визначення позиції підрядка у тексті
15. **list -** Перетворення рядка у список символів
16. **capitalize -** Переведення першої літери у велику
17. **isalpha -** Перевірка, чи складається рядок лише з букв
18. **isdigit -** Перевірка, чи складається рядок лише з цифр
19. **isspace -** Перевірка, чи складається рядок лише з пробілів
20. **count -** Визначення кількості входжень підрядка в рядок



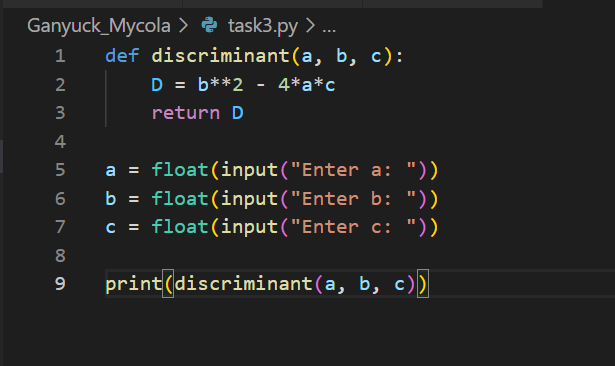


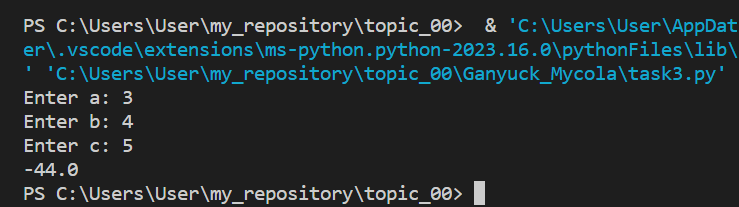




Термінал: 

**3.** Функція пошуку Дискримінанту

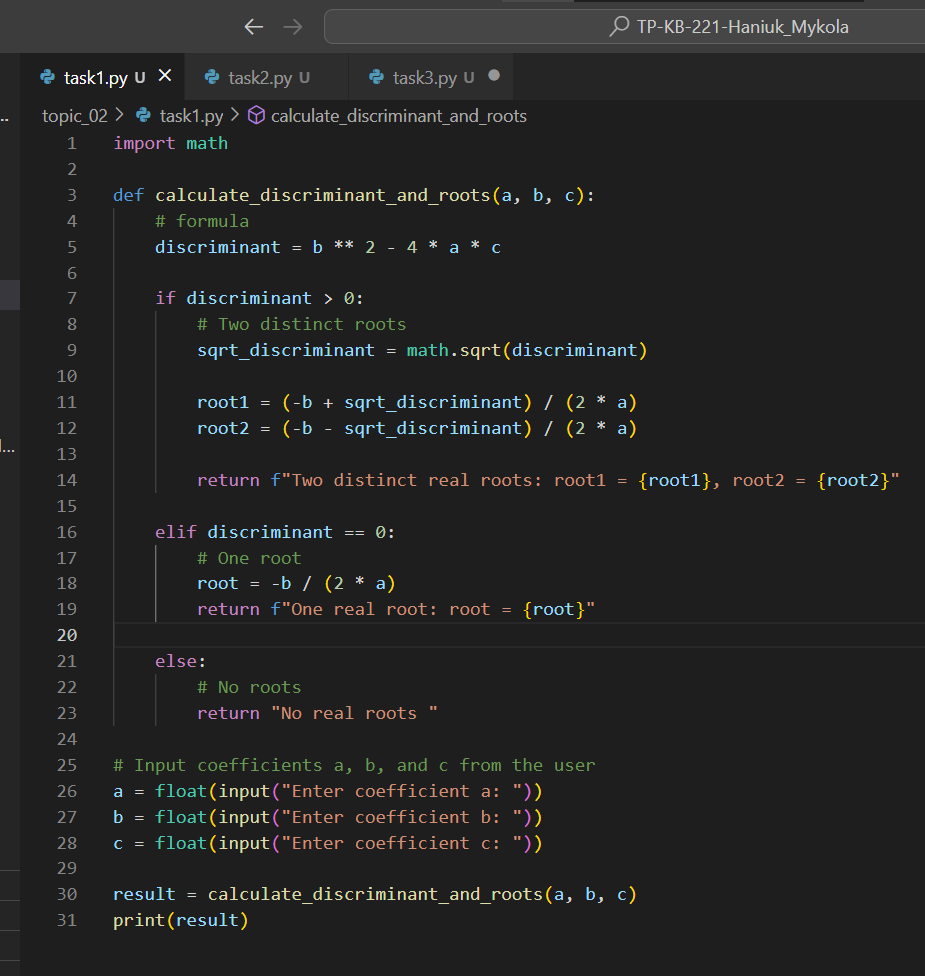


Результат роботи 

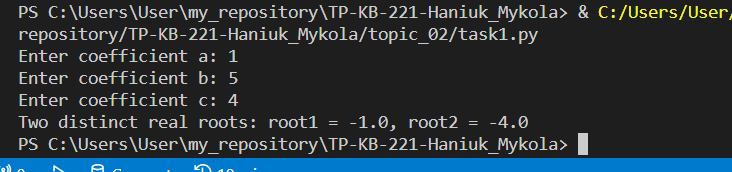
**Висновок:** в ході виконання практичного завдання я ознайомився з вбудованими функціями мови python та створив власну для знаходження дискримінанту.

**Тема 02**

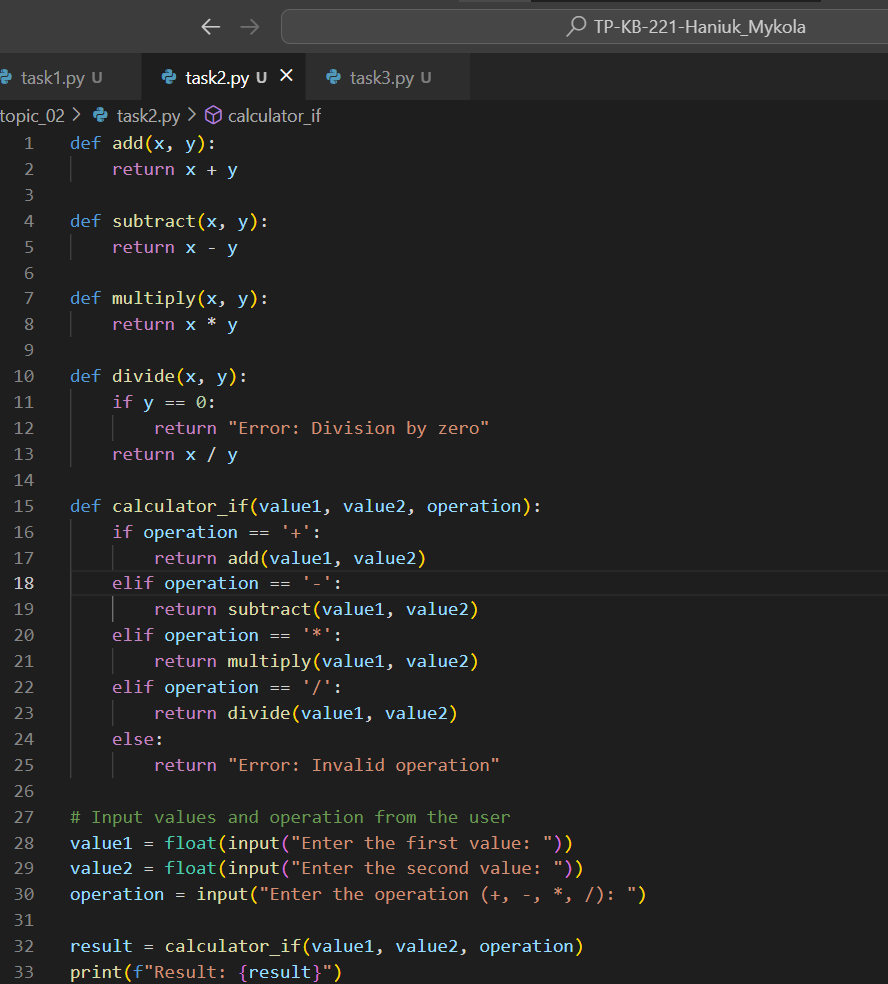
1. **Пошук дискримінанту та коренів.**
2. Код



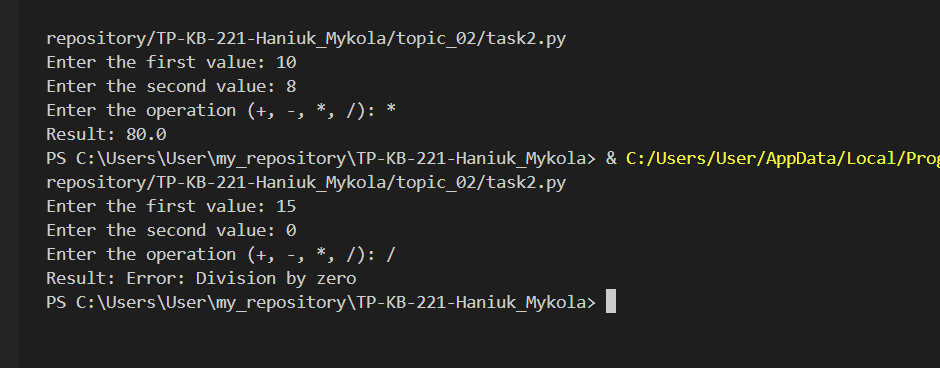
1. Результат виконання:



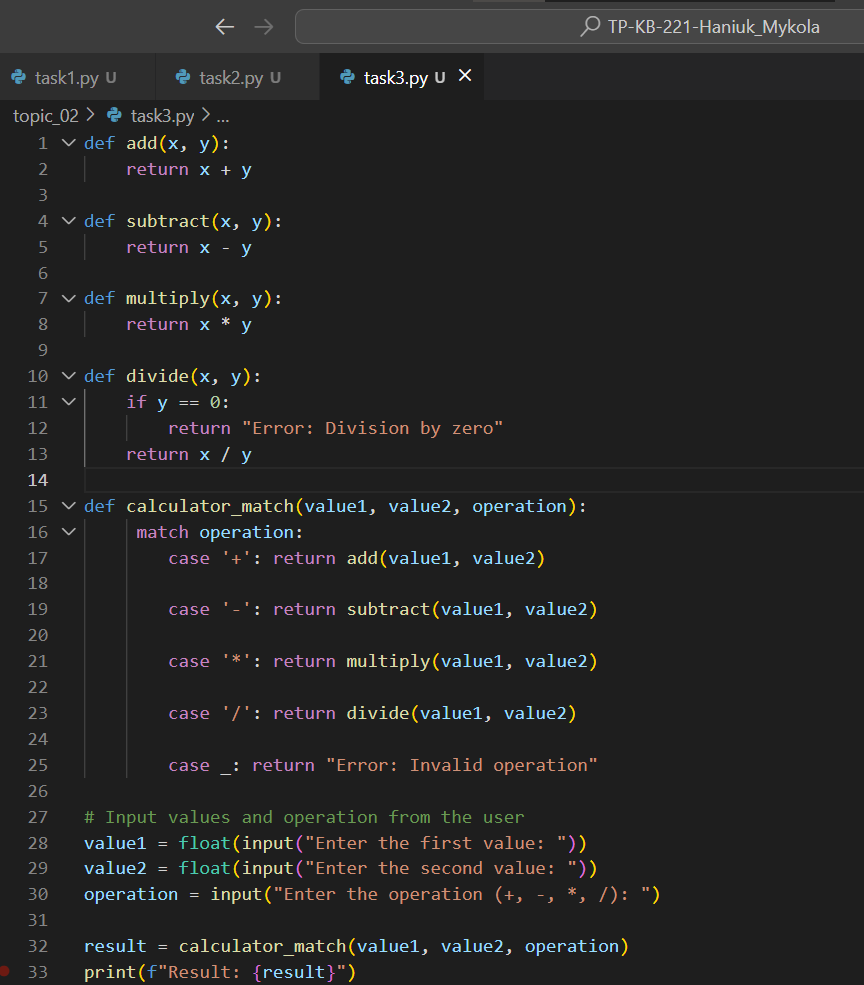
1. Калькулятор (if-elif)
2. Код



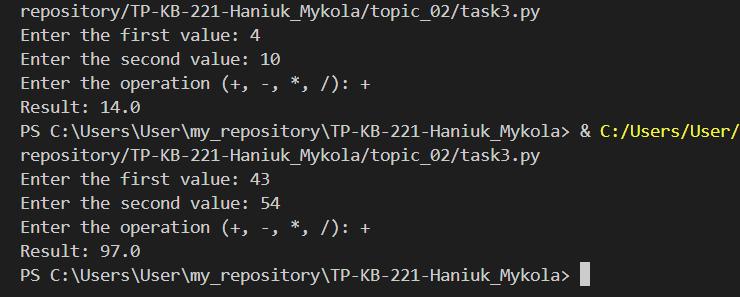
1. Результат виконання:



1. Калькулятор (case)
2. Код



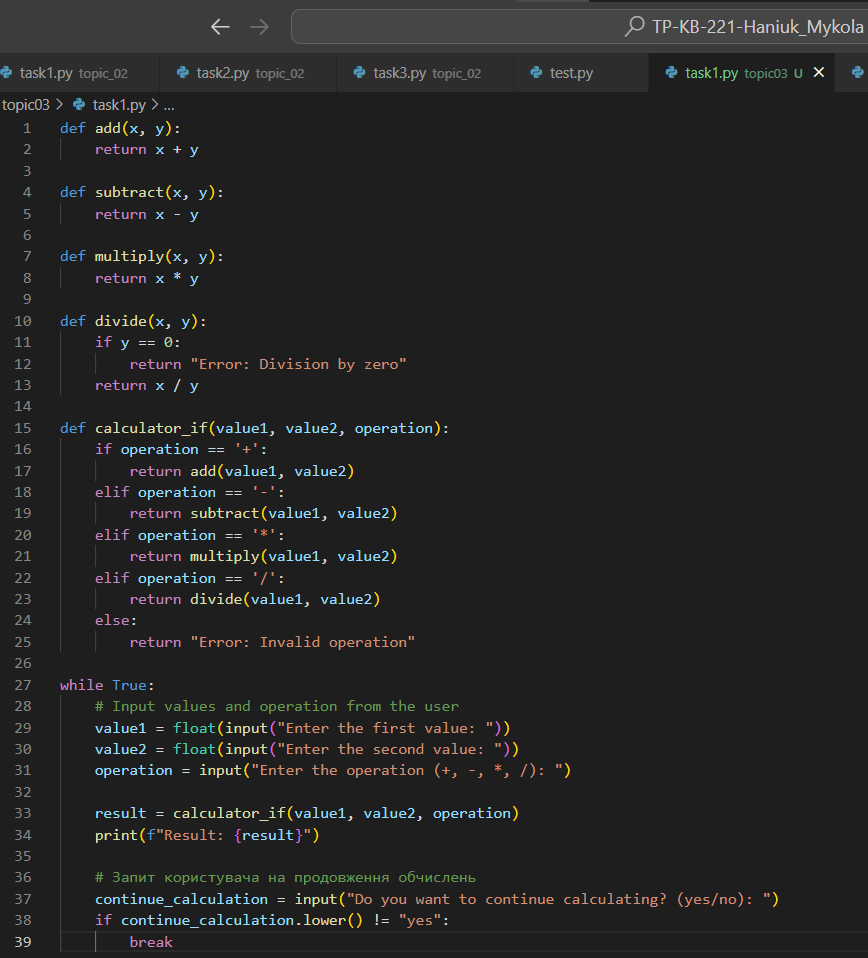
2)Результат виконання:



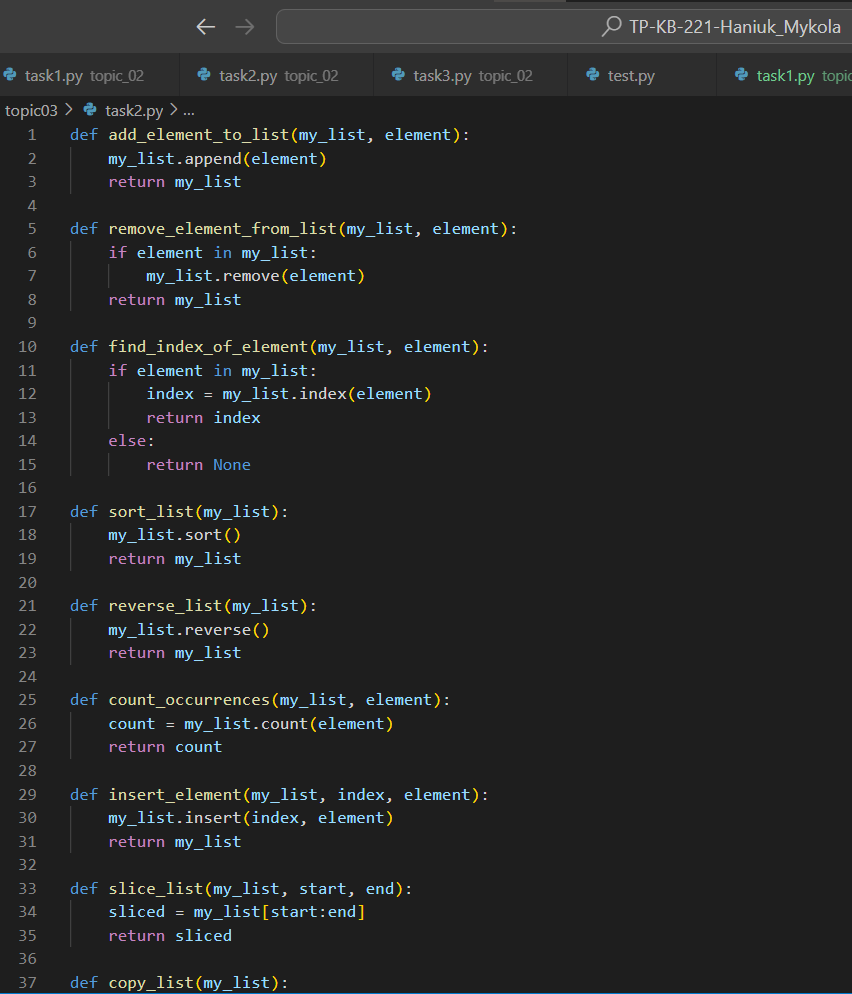
**Висновок:** в ході виконання практичного завдання я дописав функцію дискримінанту та створив 2 базових калькулятори .

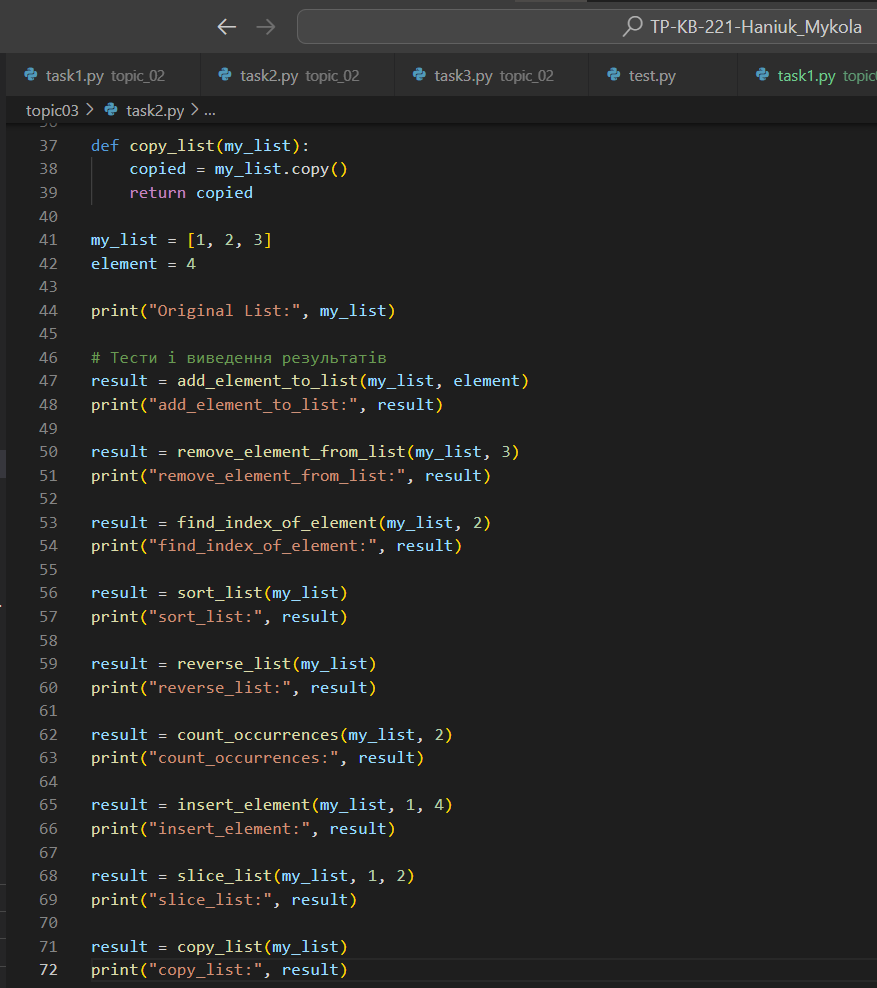
**Тема 03**

1. Написати програму калькулятор з постійними запитами на введення нових даних та операцій:

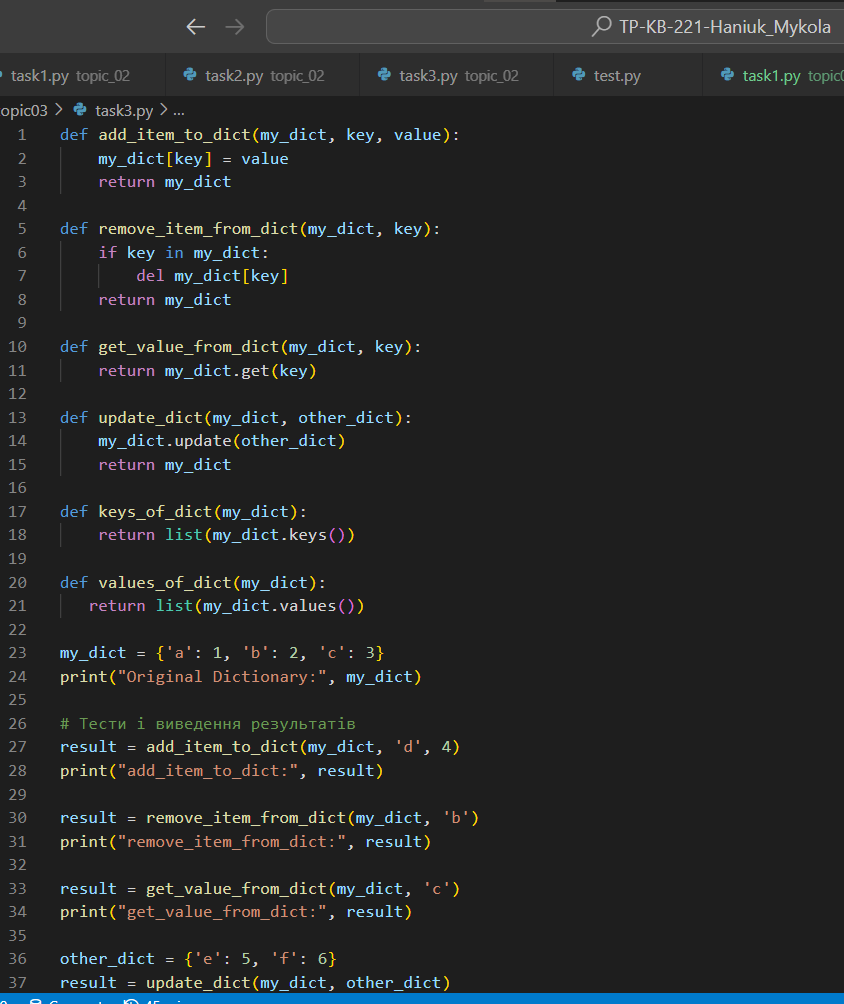


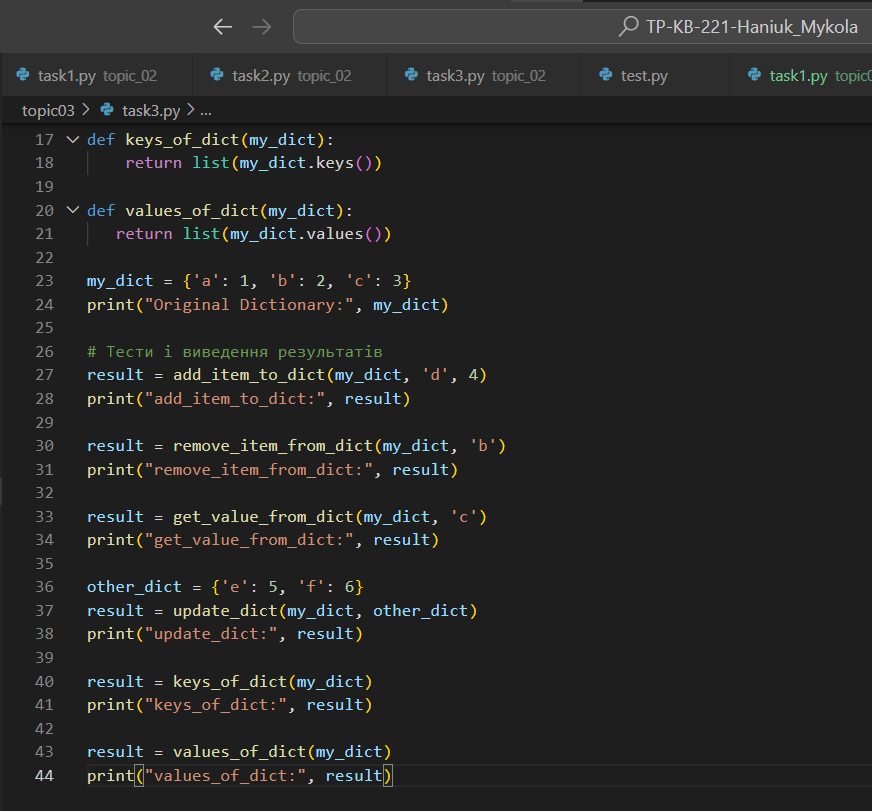
1. Написати програму тестування функцій списків:



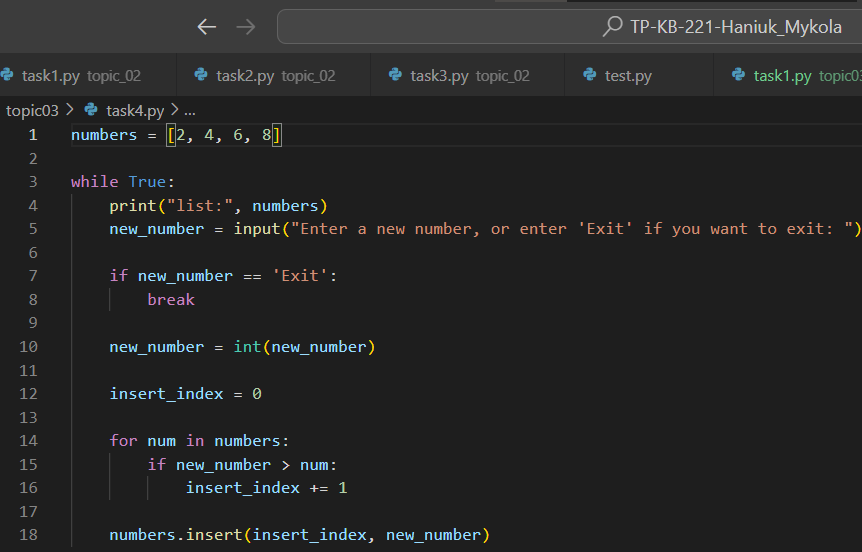


1. Написати програму тестування функцій словників:





4. Написати функцію пошуку позиції для вставки нового елементу у відсортований список

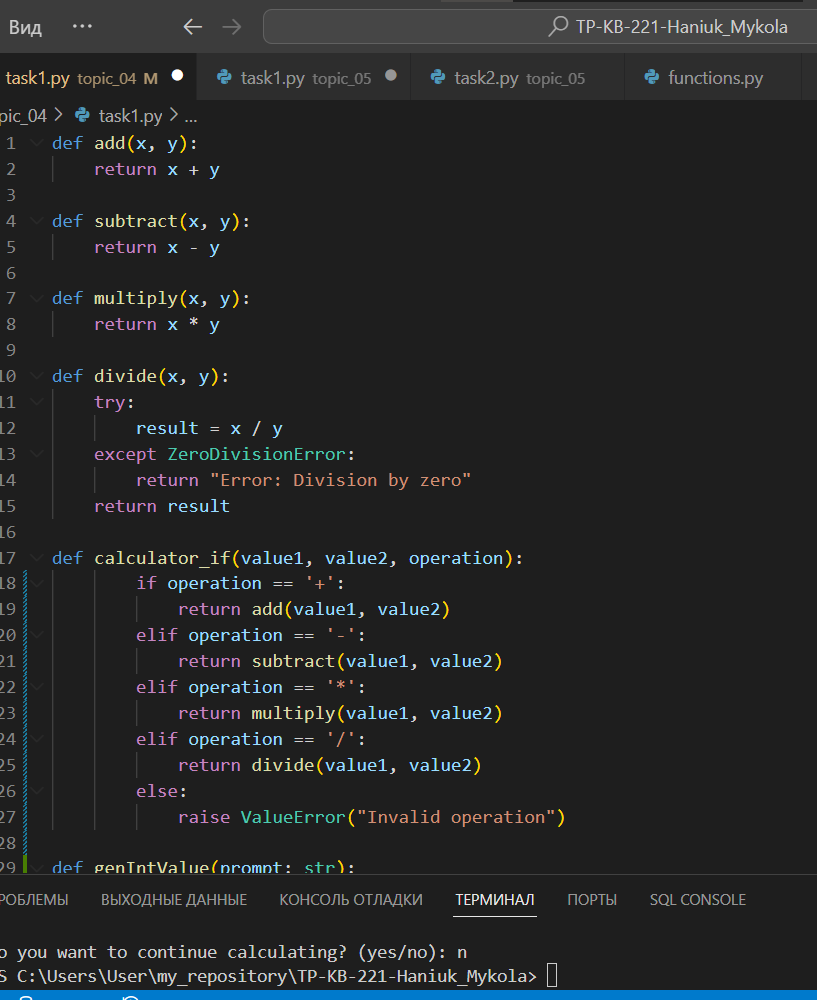


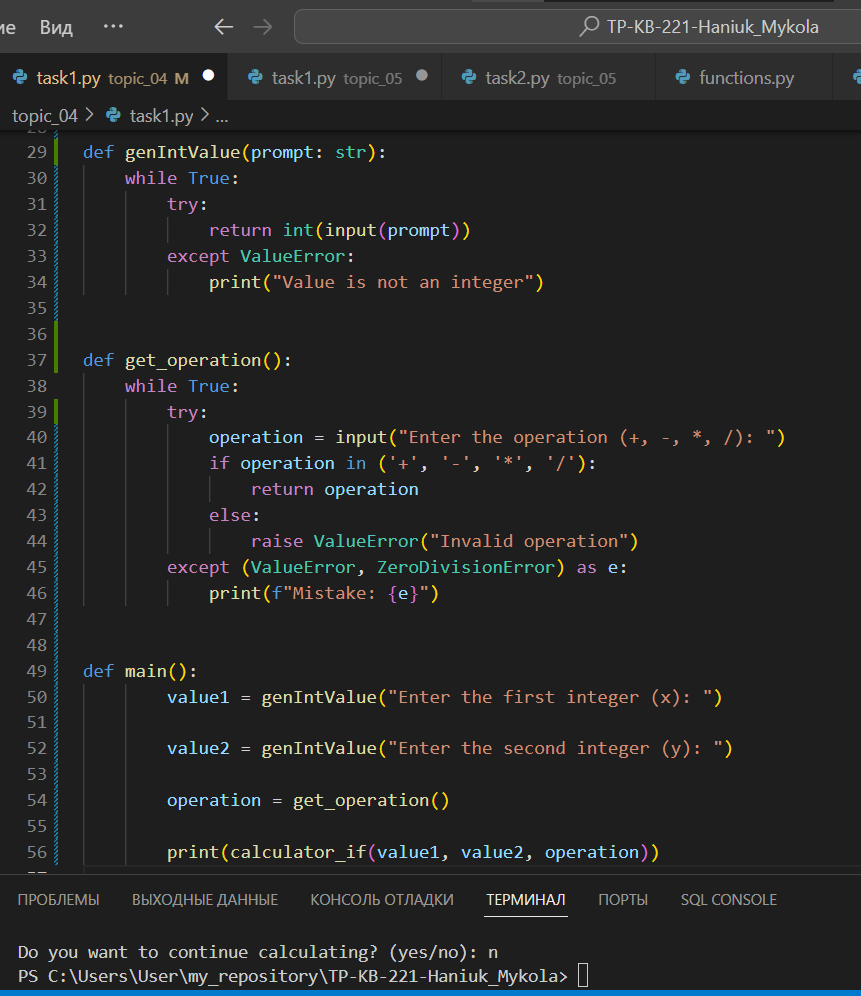
**Висновок:** в ході виконання практичного завдання я попрацював зі словниками, масивами, допрацював калькулятор та створив функцію для пошуку індексу числа в масиві.

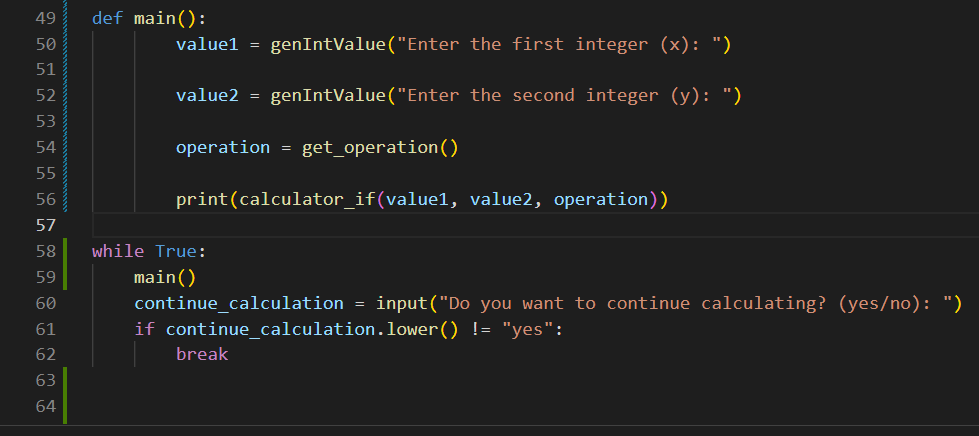
**Тема 04**

Завдання

Розширити програму калькулятор функцією запитів від користувача, що обробляє виняткові ситуації та функцію ділення обробкою виняткової ситуації ділення но нуль.



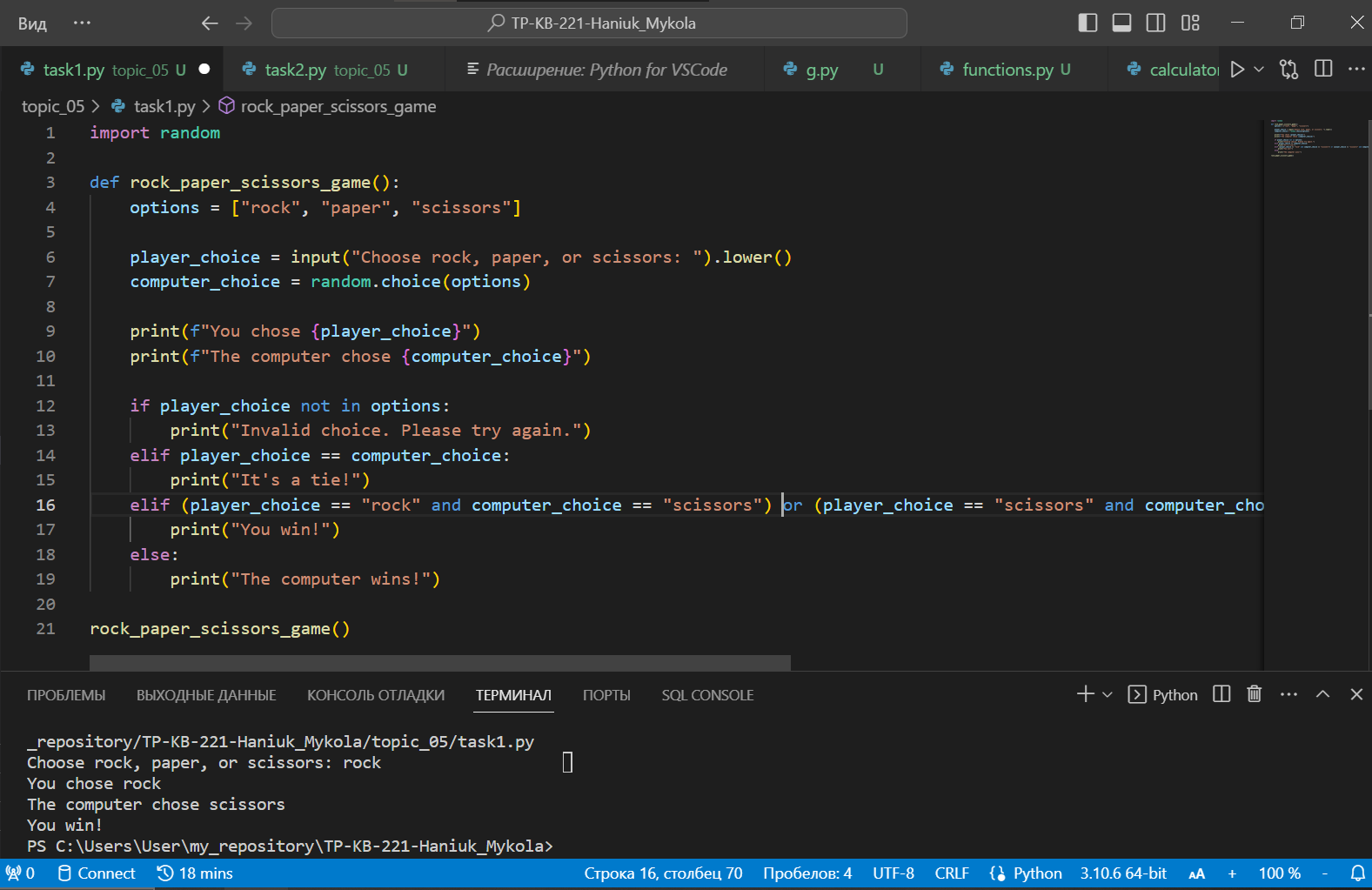




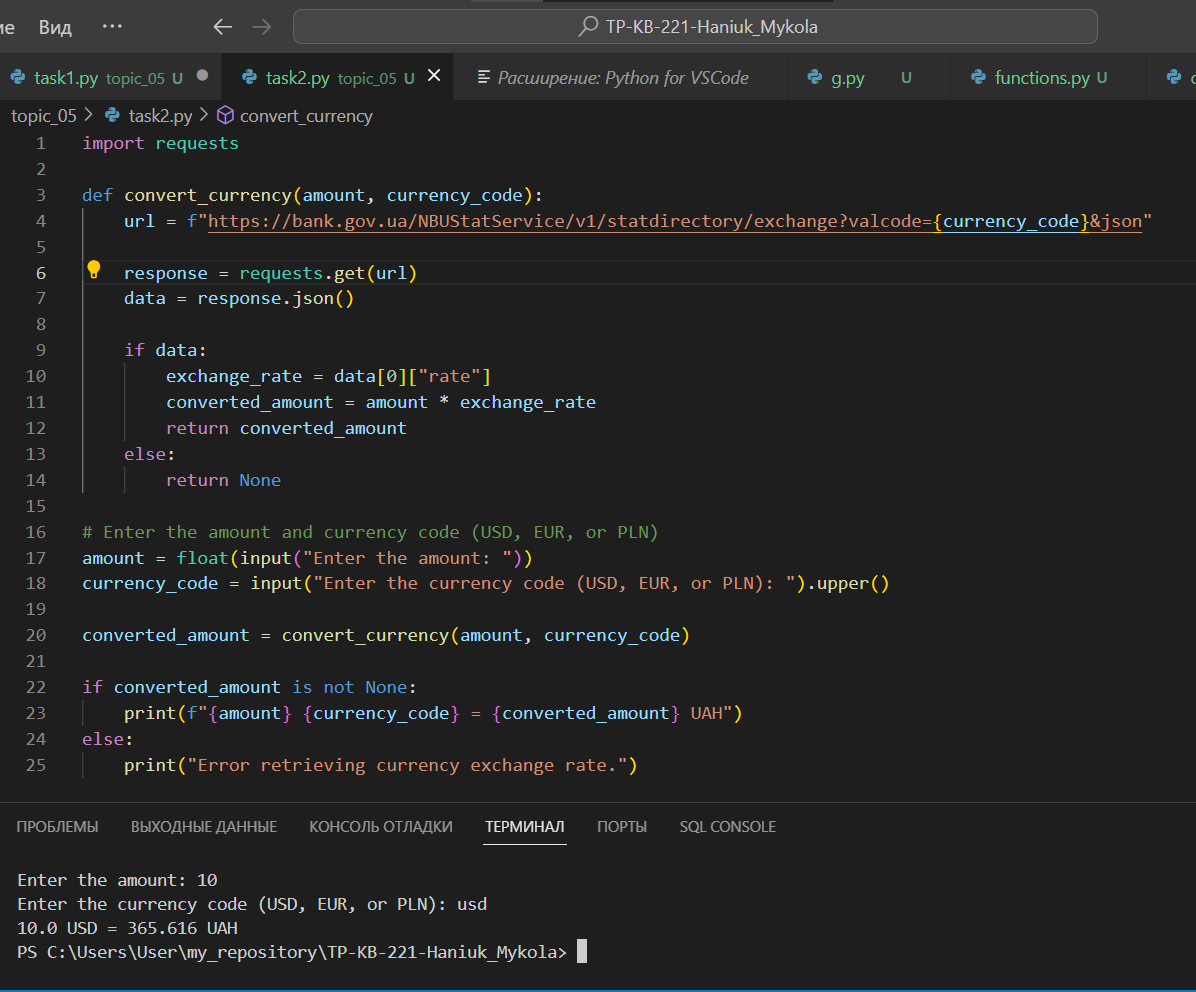
**Висновок:** в ході виконання практичної роботи я додав до калькулятора обробку виключень для вхідного потоку та ділення на нуль.

**Тема 5**

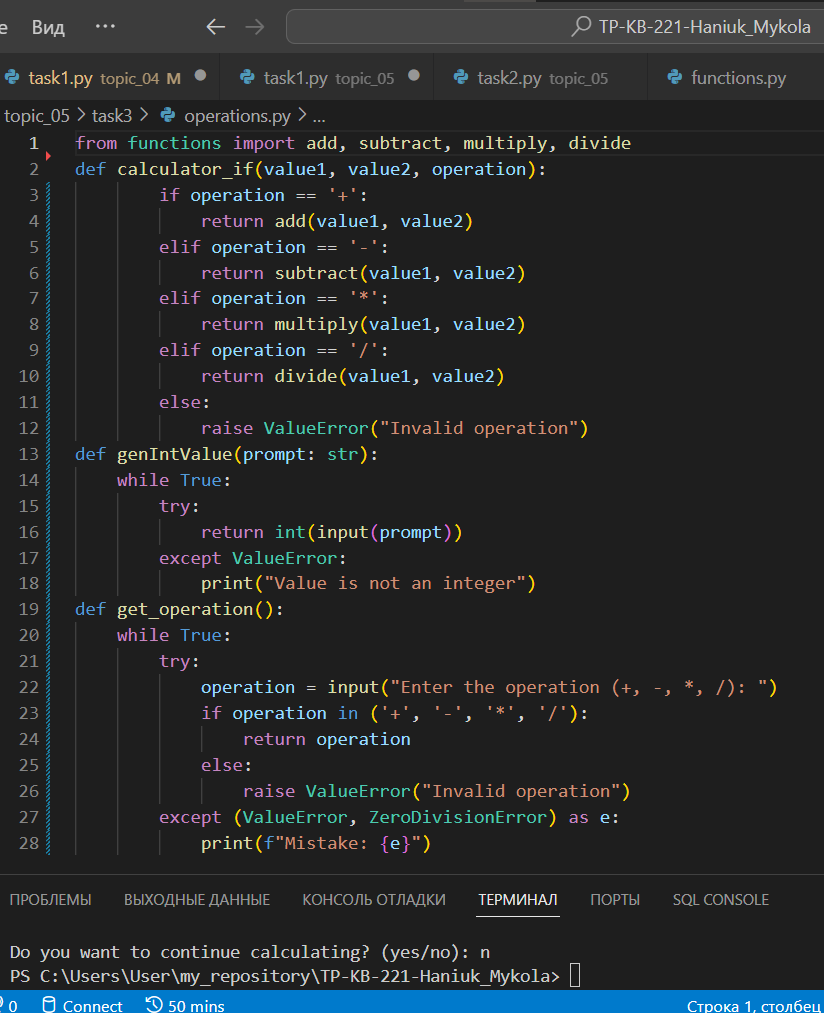
1. Завдання: гра Камінь Ножиці Папер

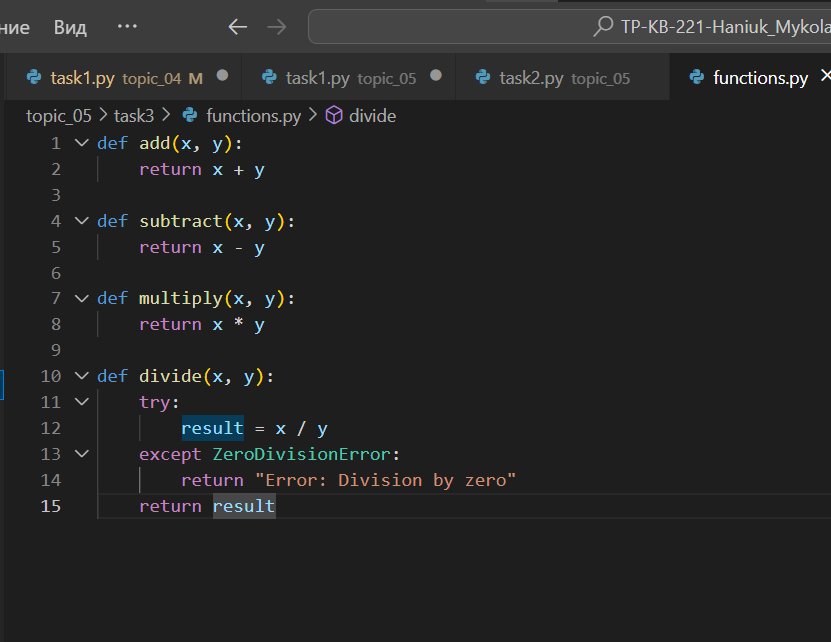


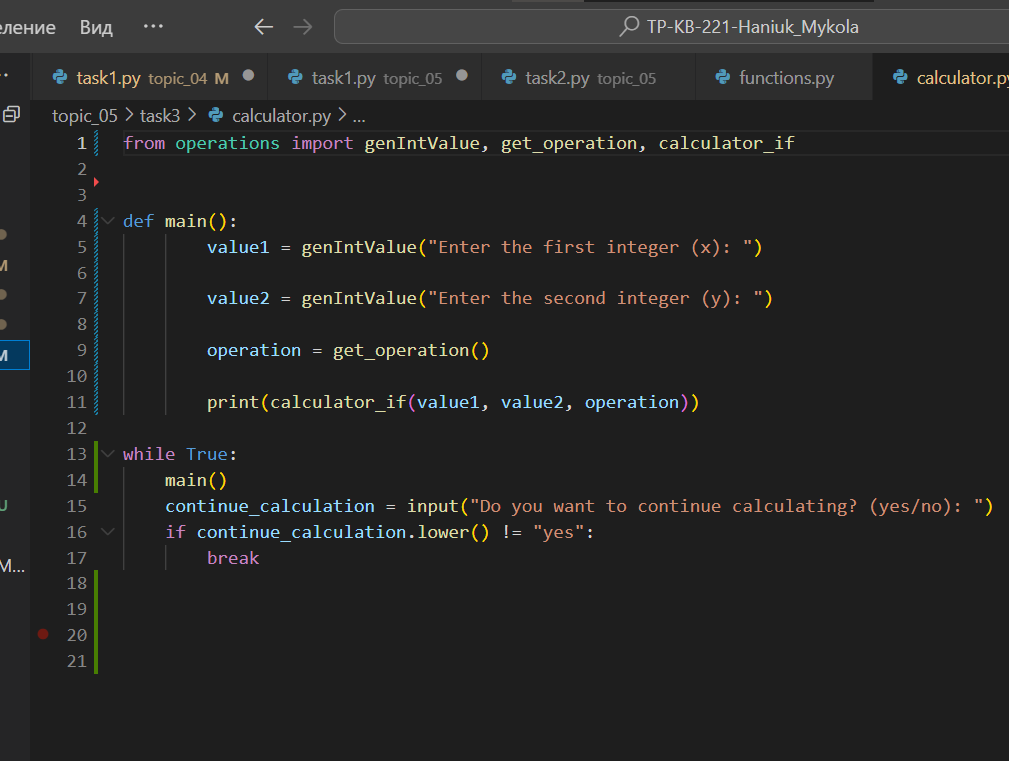
2 Завдання : Конвертор валют

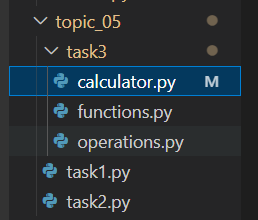


3. Завдання: модулі для програми калькулятор



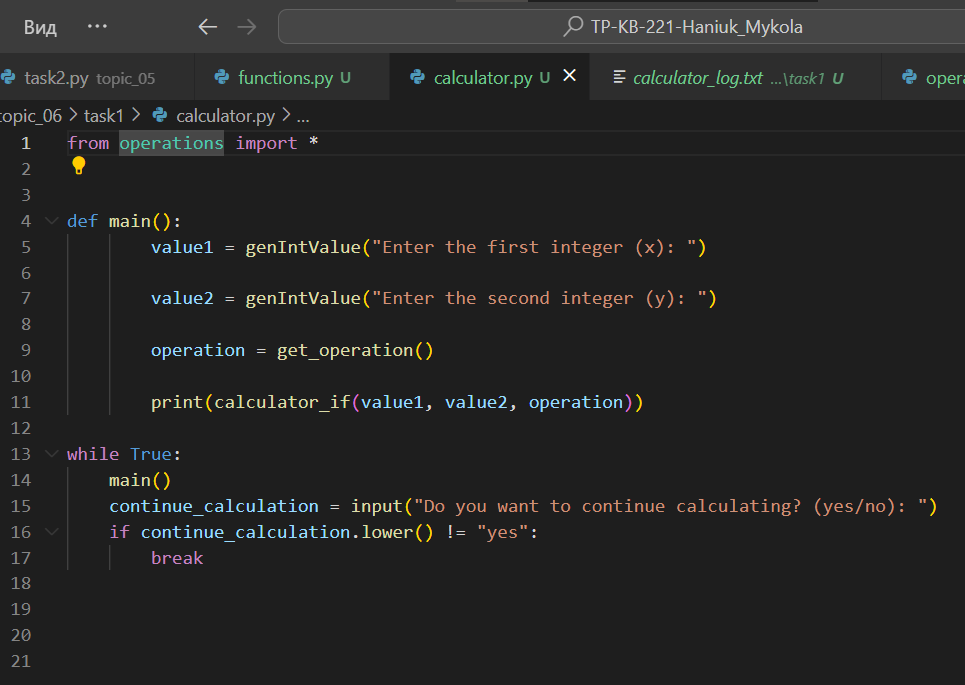


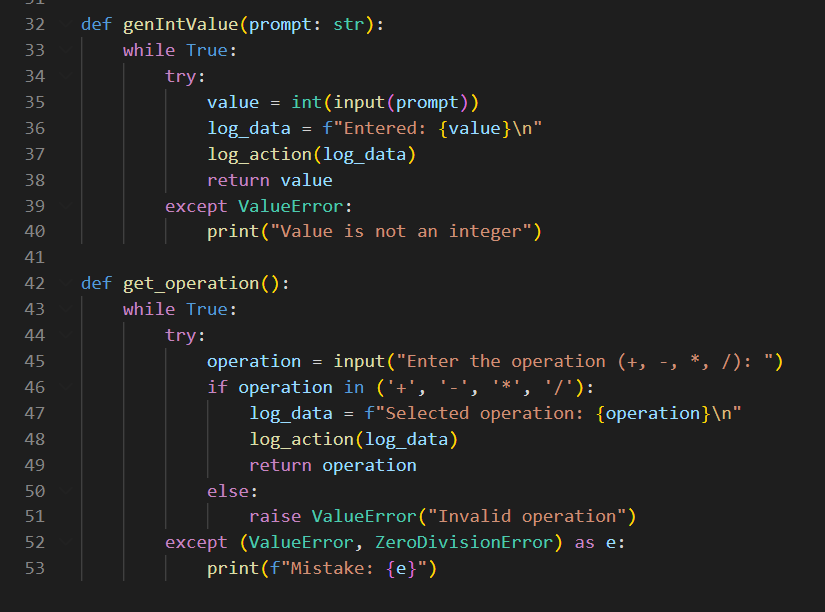
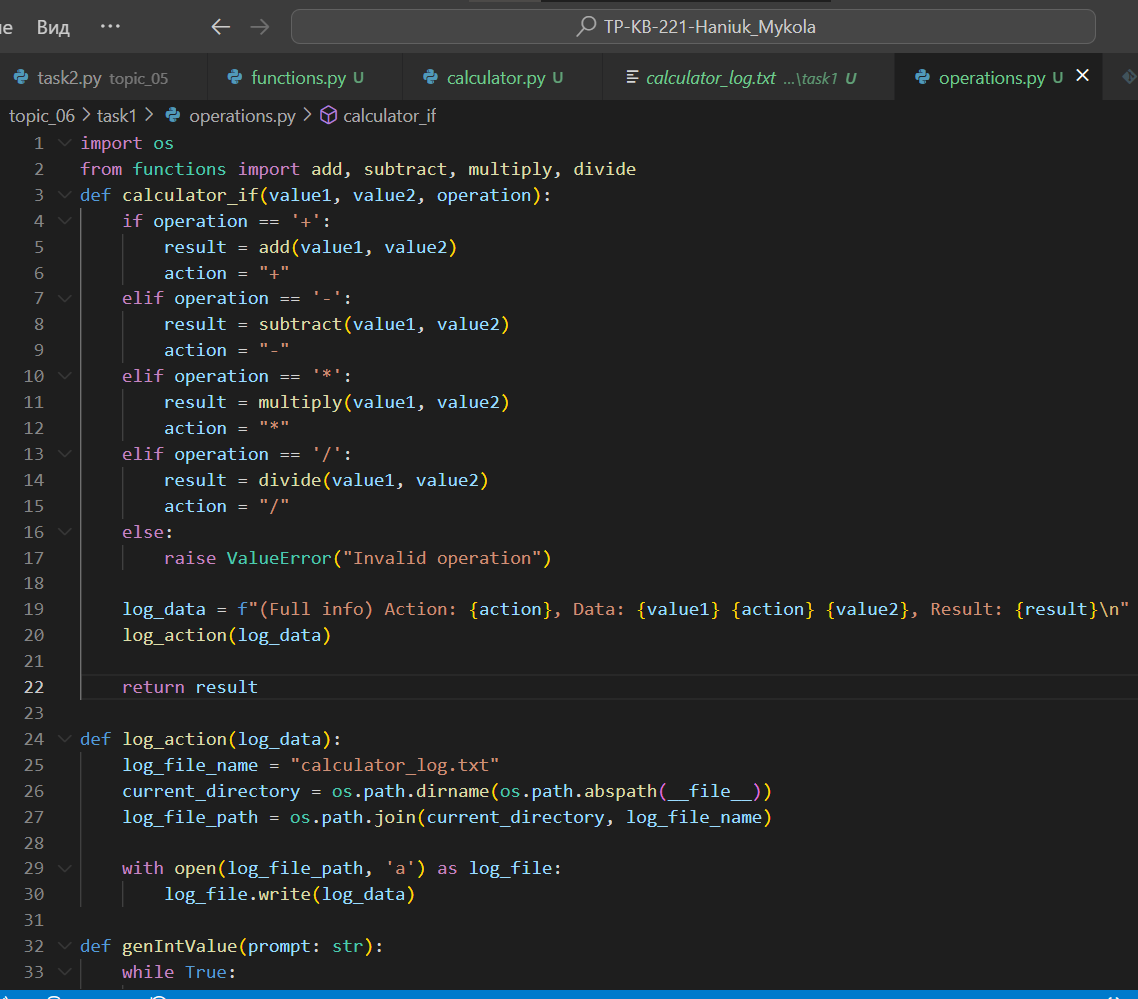
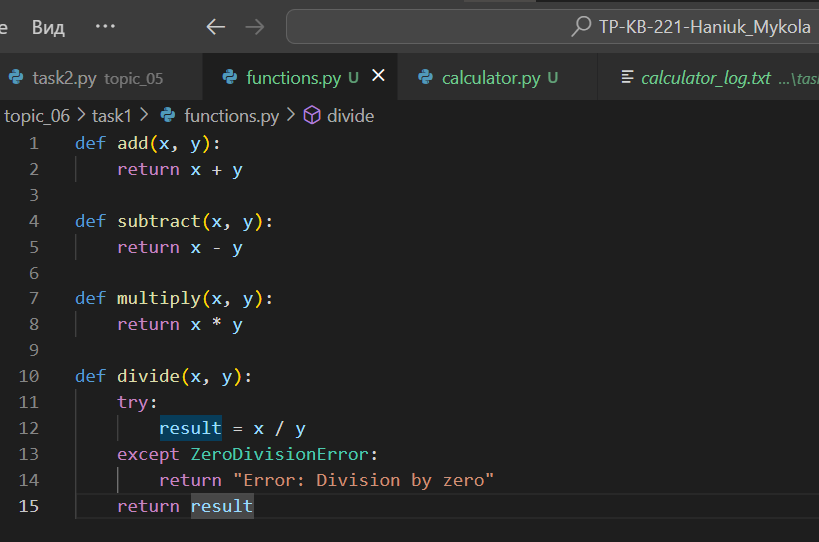


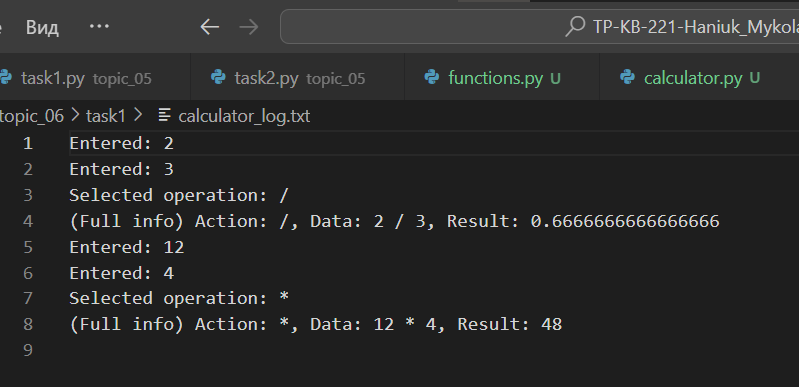


**Висновок:** в ході виконання практичної роботи я створив консольну гру камінь ножиці папір, конвертор валют, та виокремив функції калькулятора в модулі

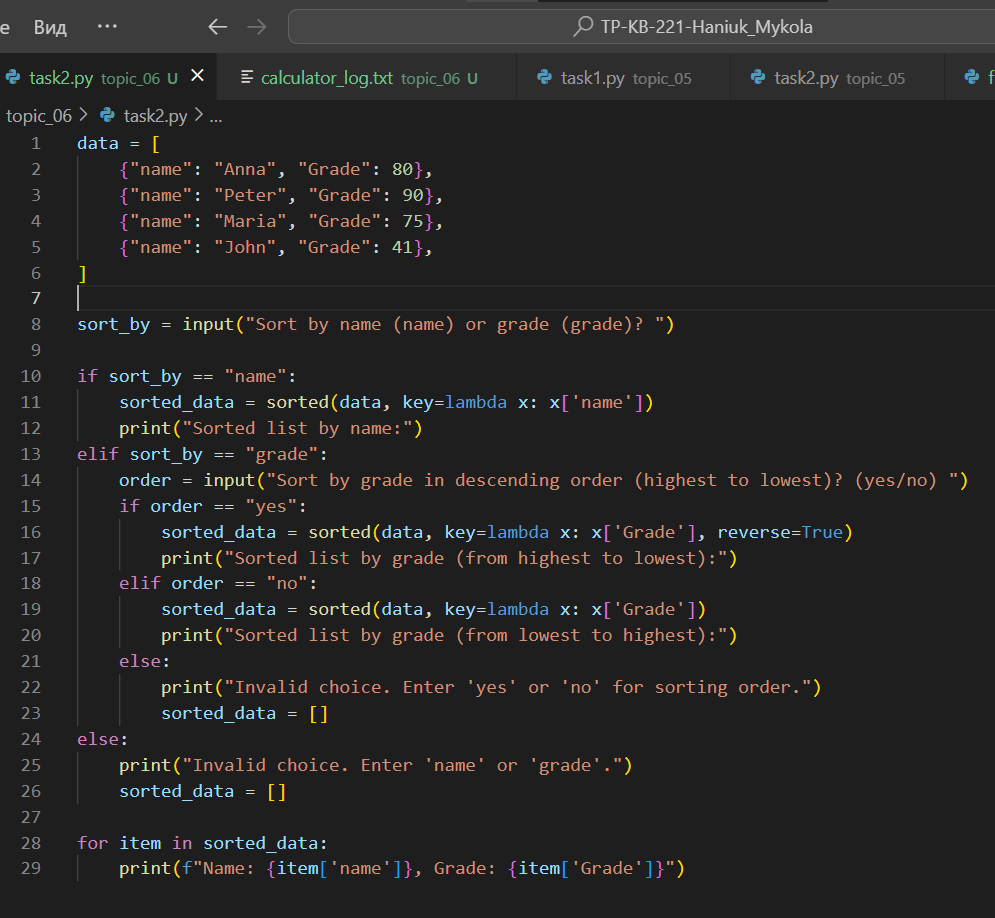
**Тема 6**

Завдання : Логування всіх дій в застосунку Калькулятор





2 Завдання: Використання lambda функцій для функції сортування



**Висновок:** в ході практичного завдання я додав логування до калькулятора та написав функцію сортування використовуючи lambda.

**Тема 7**

2. Завдання 1)\_\_init\_\_(self): Цей метод викликається автоматично при створенні нового об'єкта класу. Він ініціалізує атрибути об'єкта і може використовуватися для встановлення початкових значень атрибутів.

Наприклад:

class Person:

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

self.name = name

self.age = age

person = Person("Oleg", 18)

2) \_\_str\_\_(self): Цей метод повертає рядок, який представляє об'єкт у зрозумілому для людини форматі. Він викликається, коли ми використовуємо функцію print() для об'єкта або коли ми перетворюємо об'єкт на рядок за допомогою str(). Наприклад:

class Person:

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

self.name = name

self.age = age

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

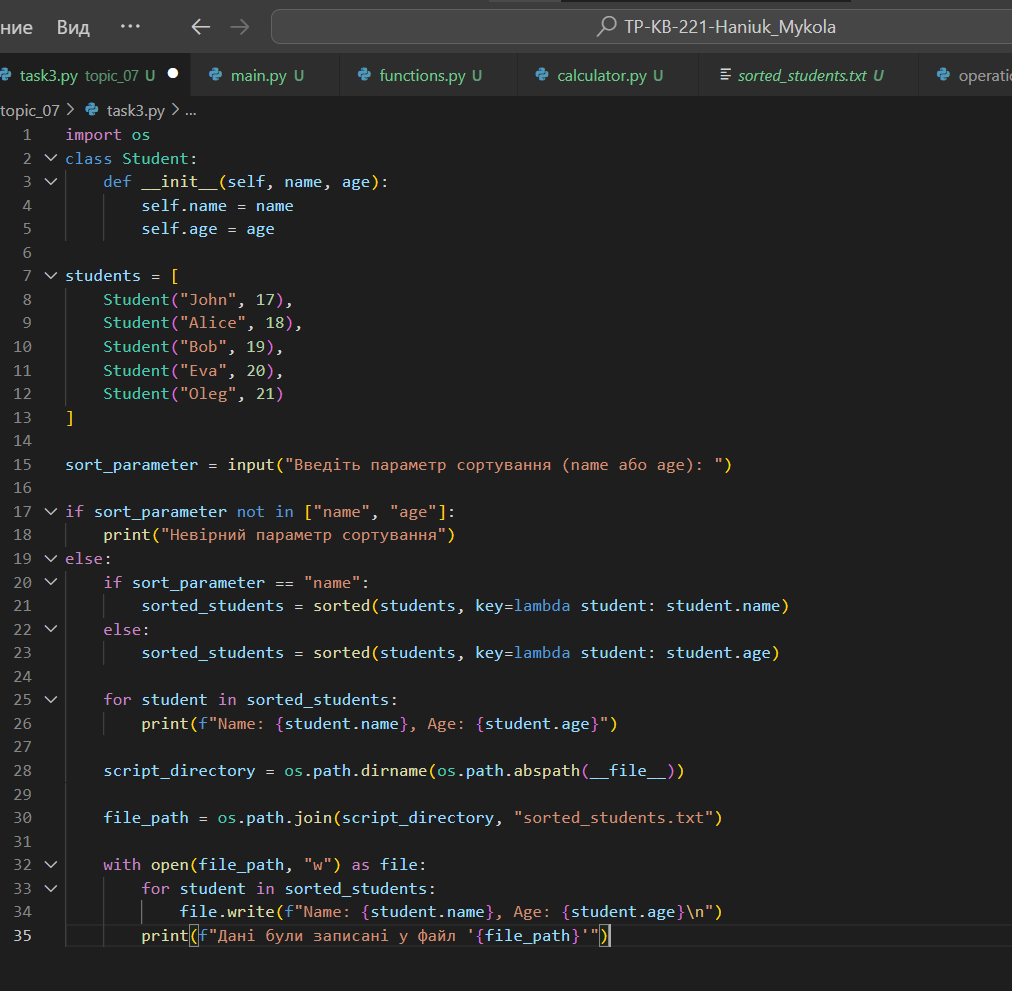
return f"Person(name={self.name}, age={self.age})"

person = Person("Oleg", 18)

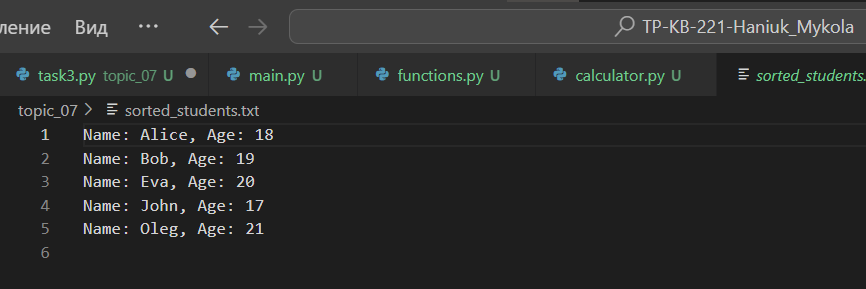
print(person)

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.

Код

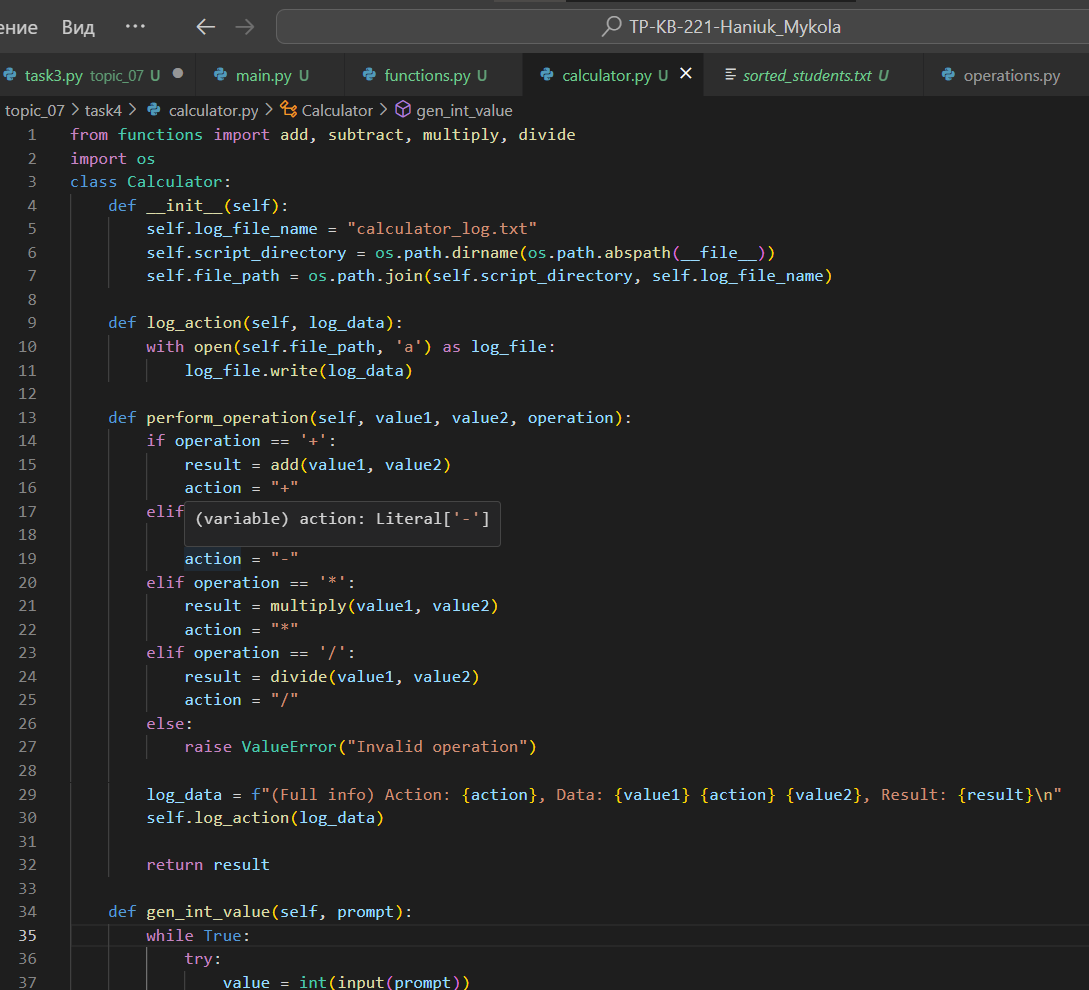
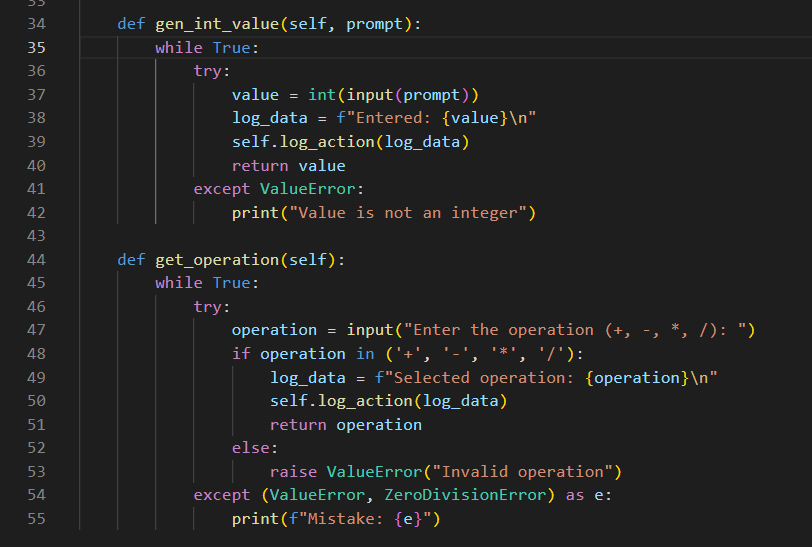


Результат:

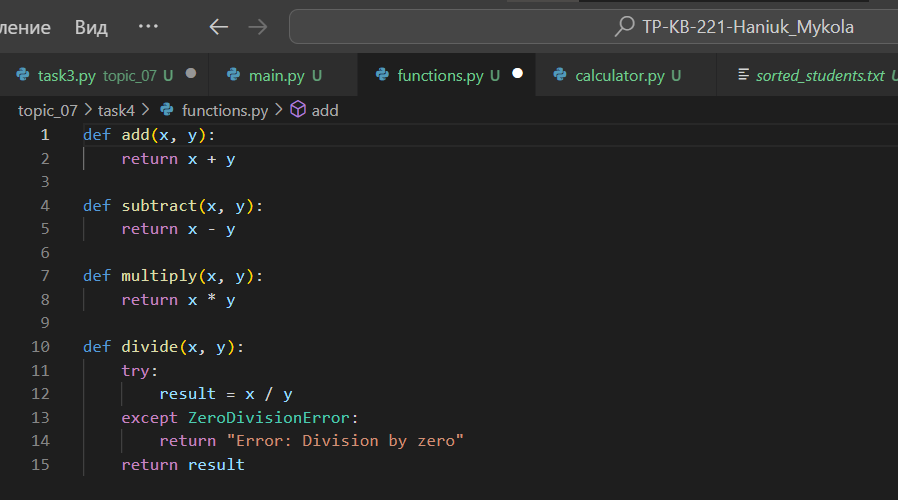


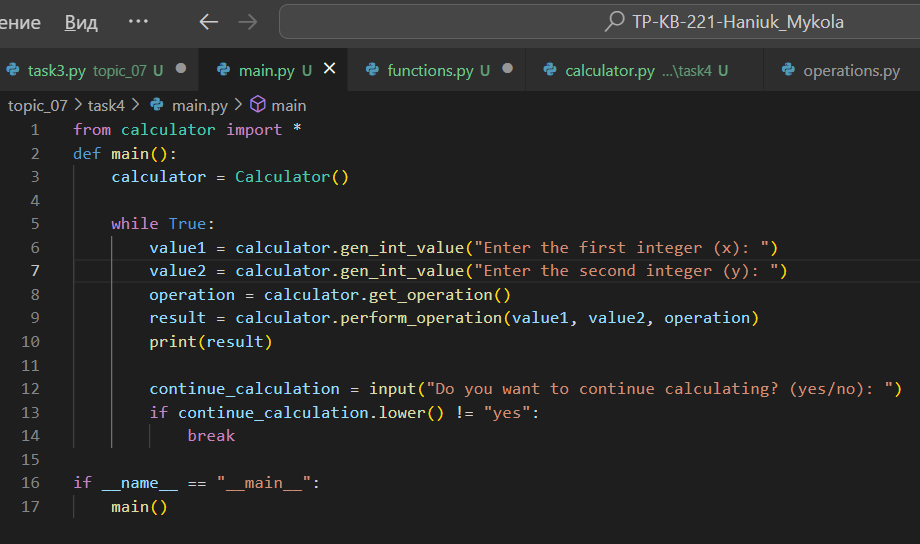
4. Використовуючи принципи ООП переписати програму Калькулятор. Завдання має бути виконано використовуючи модульний підхід.

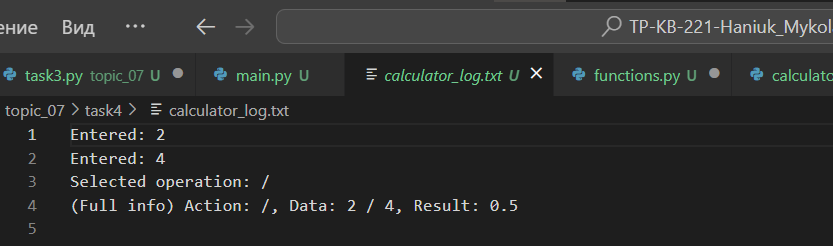
За основу було взято код який був створений в практичному завданні до попередньої теми.

Class calculator 

Functions



Main 

Результат 

**Висновок:** в ході практичної роботи я ознайомився з класами в мові пайтон, та створив власний також переписав калькулятор опираючись на принципи ООП.