**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування на мові Python від Дизинюрика Гліба Олександровича Кб-241**

**Звіт до Теми №1: Функції та змінні**

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

**Створення репозиторію**

**Репозиторій** — це місце для зберігання й управління файлами проєкту. Найчастіше мається на увазі **git-репозиторій**.

**Перетворення рядка**

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

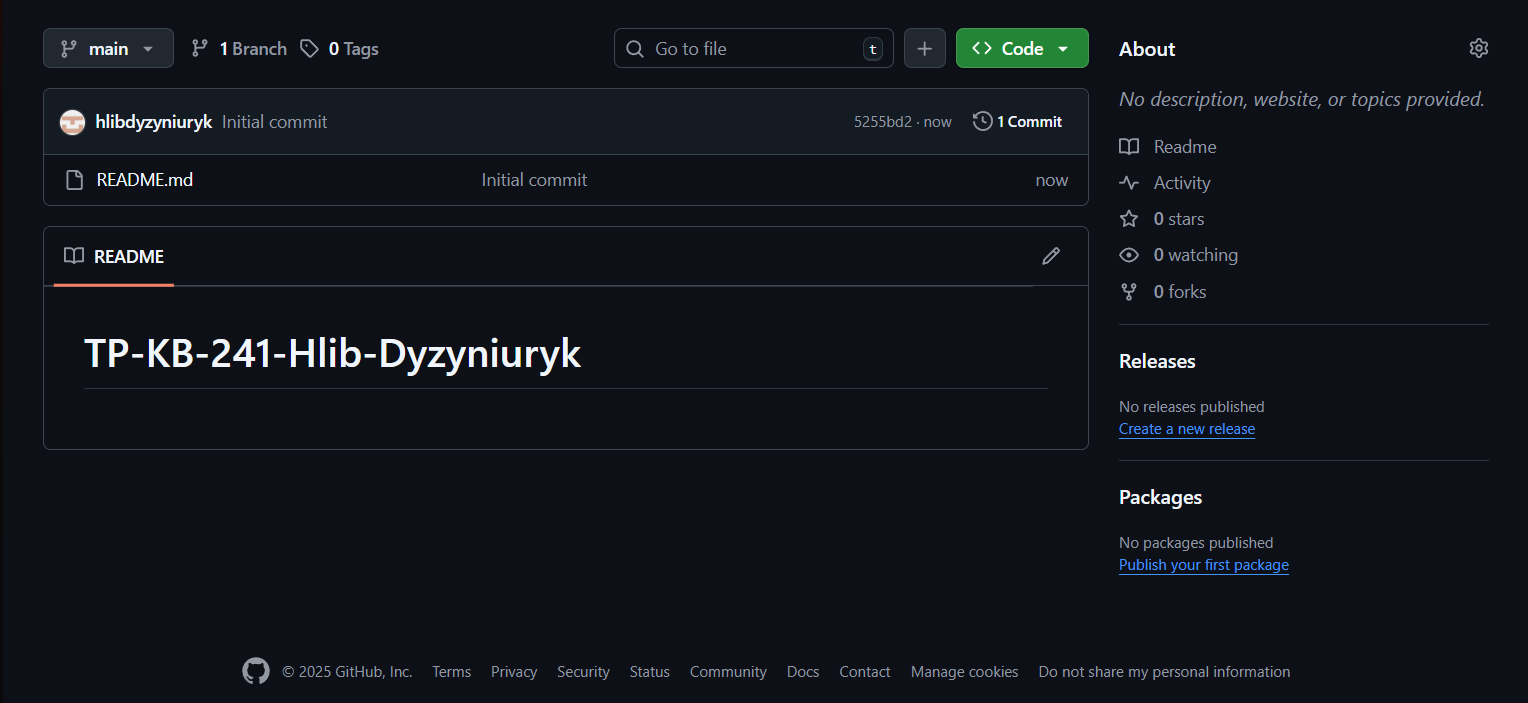
**Написати функцію пошуку дискримінанту квадратного рівняння.**

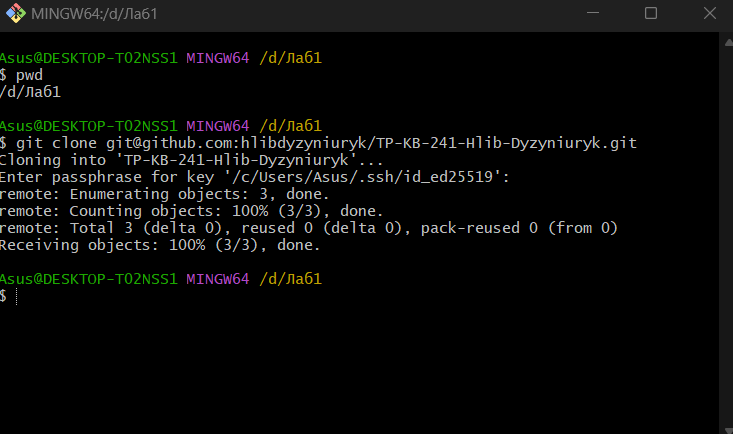
Необхідно написати код, який буде знаходити дискримінанту квадратного рівняння.

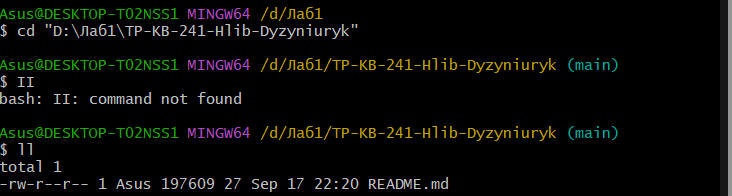
**Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().**

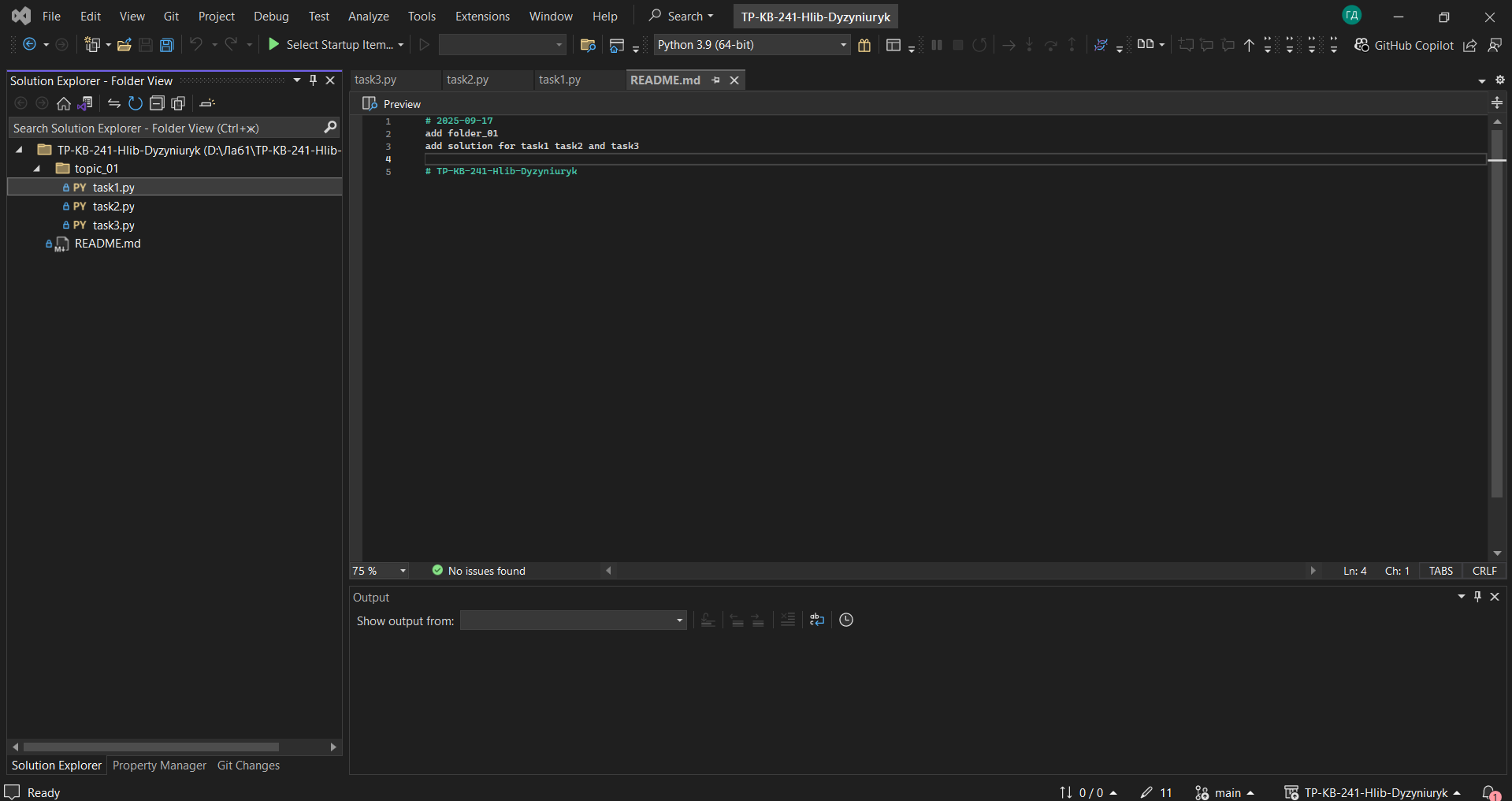
Цей код потрібен, щоб перевірити, як різні функції обробки рядків змінюють текст і чи дають вони очікуваний результат.

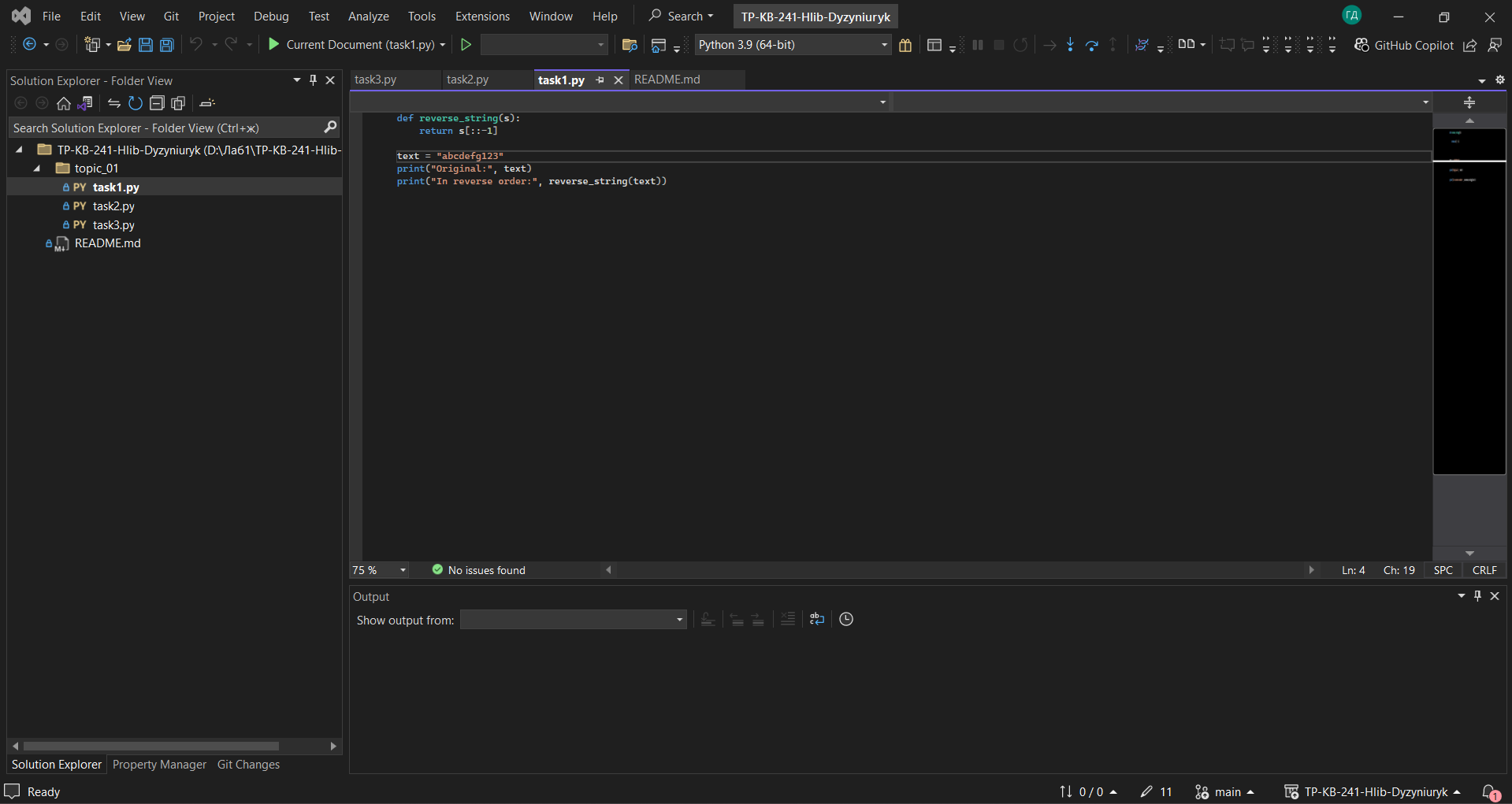
**Хід роботи**

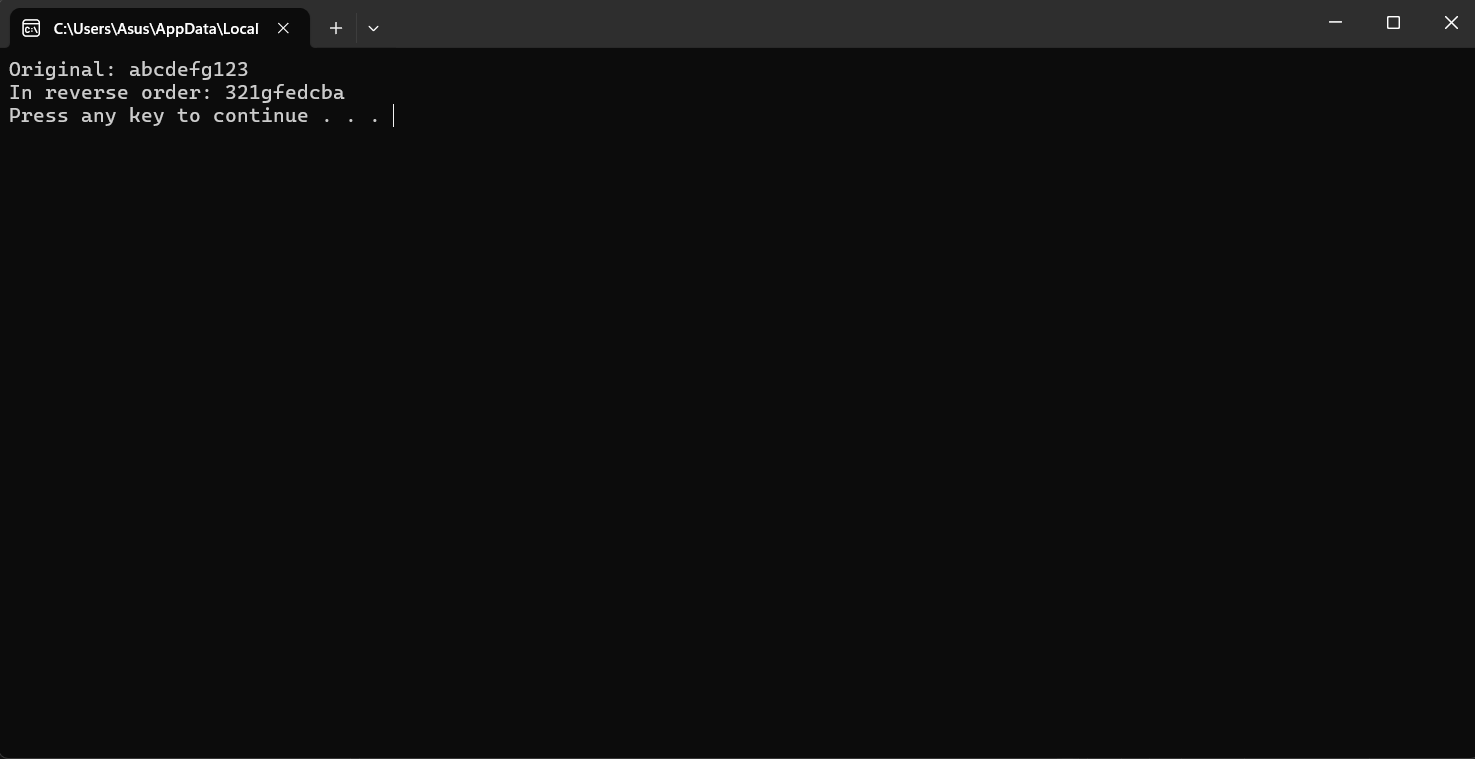
Рис. 1. Створений репозиторій у GitHub

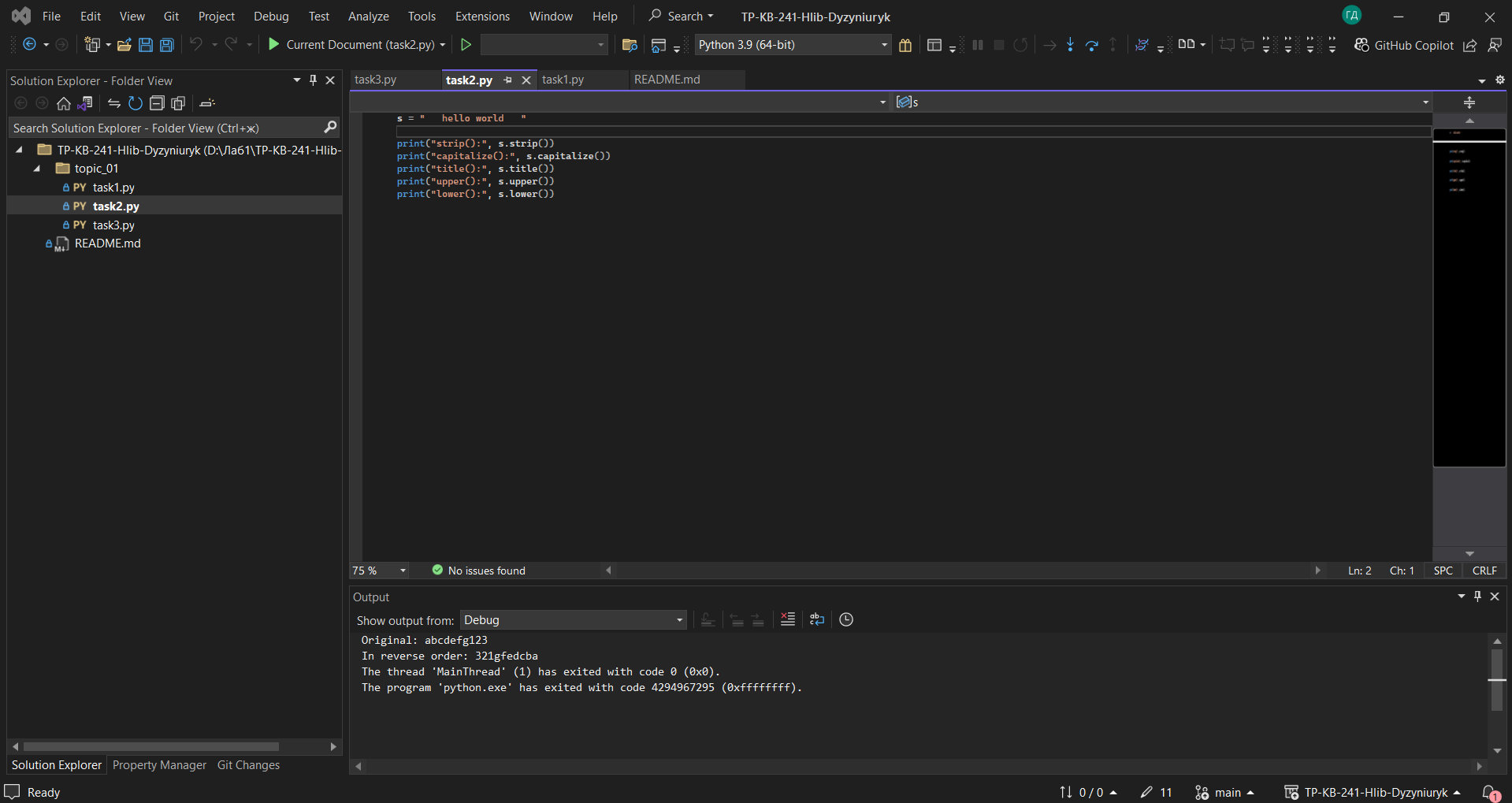
Рис. 2. Клонування репозиторію у директорій

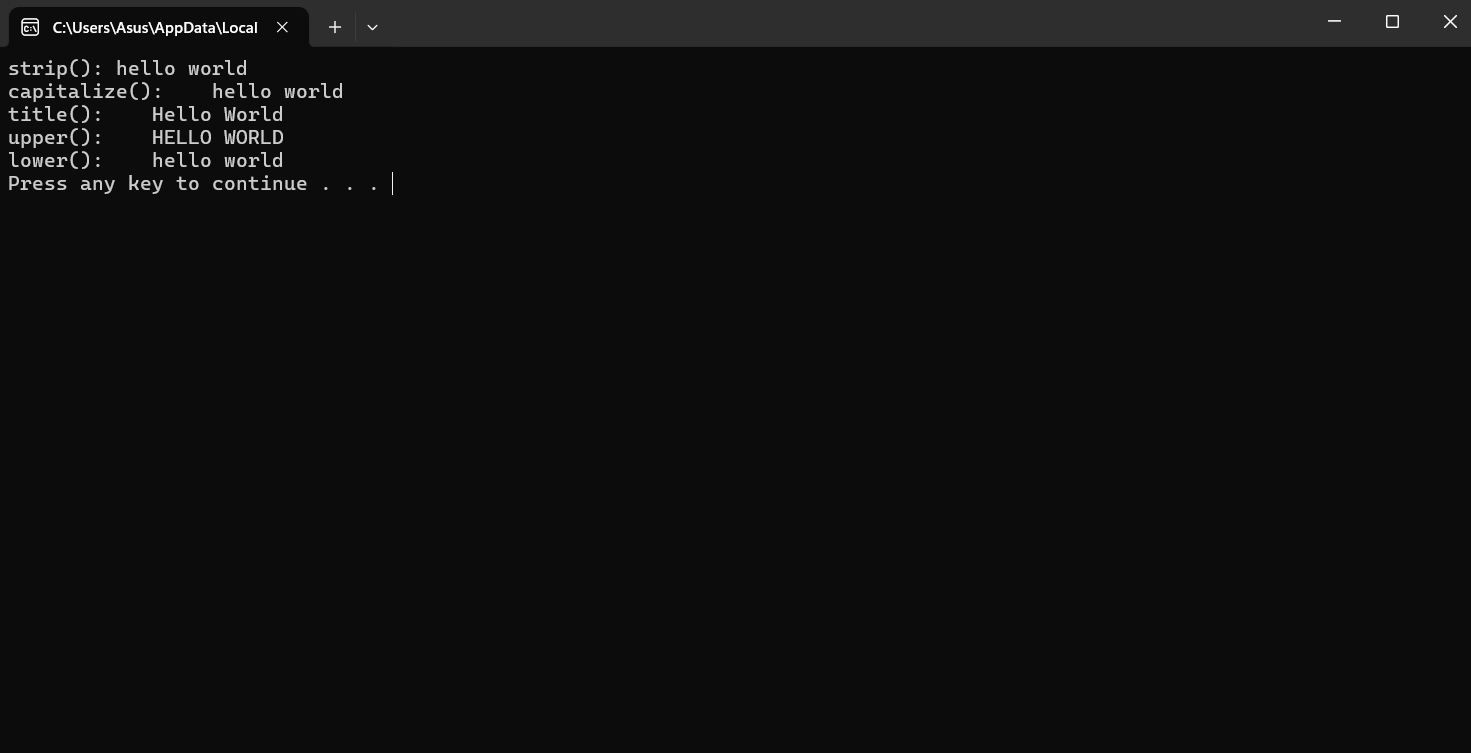
Рис. 3. Перевірка вмісту каталога завдяки команди II

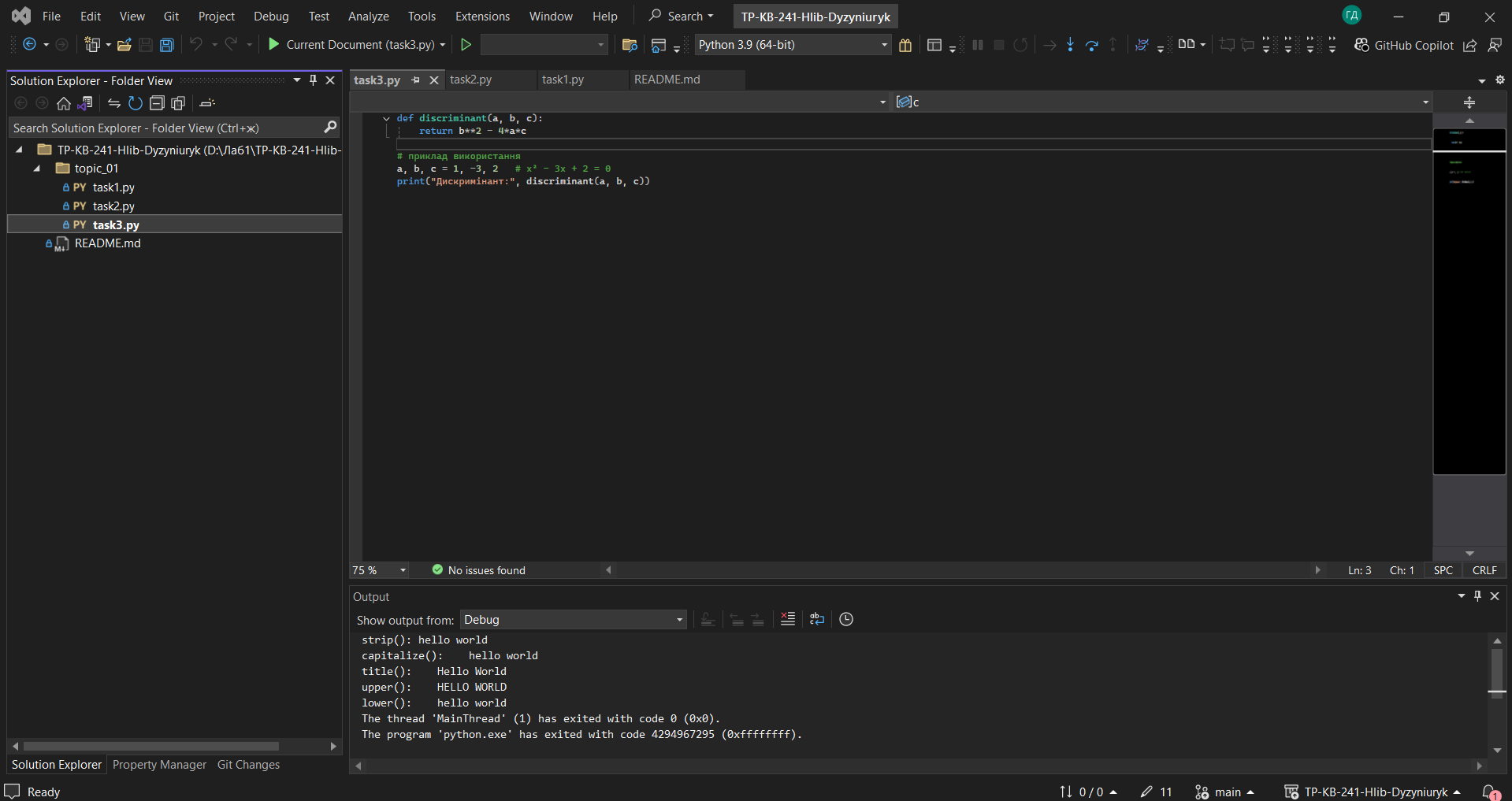
Рис. 4. Скриншот README файлу

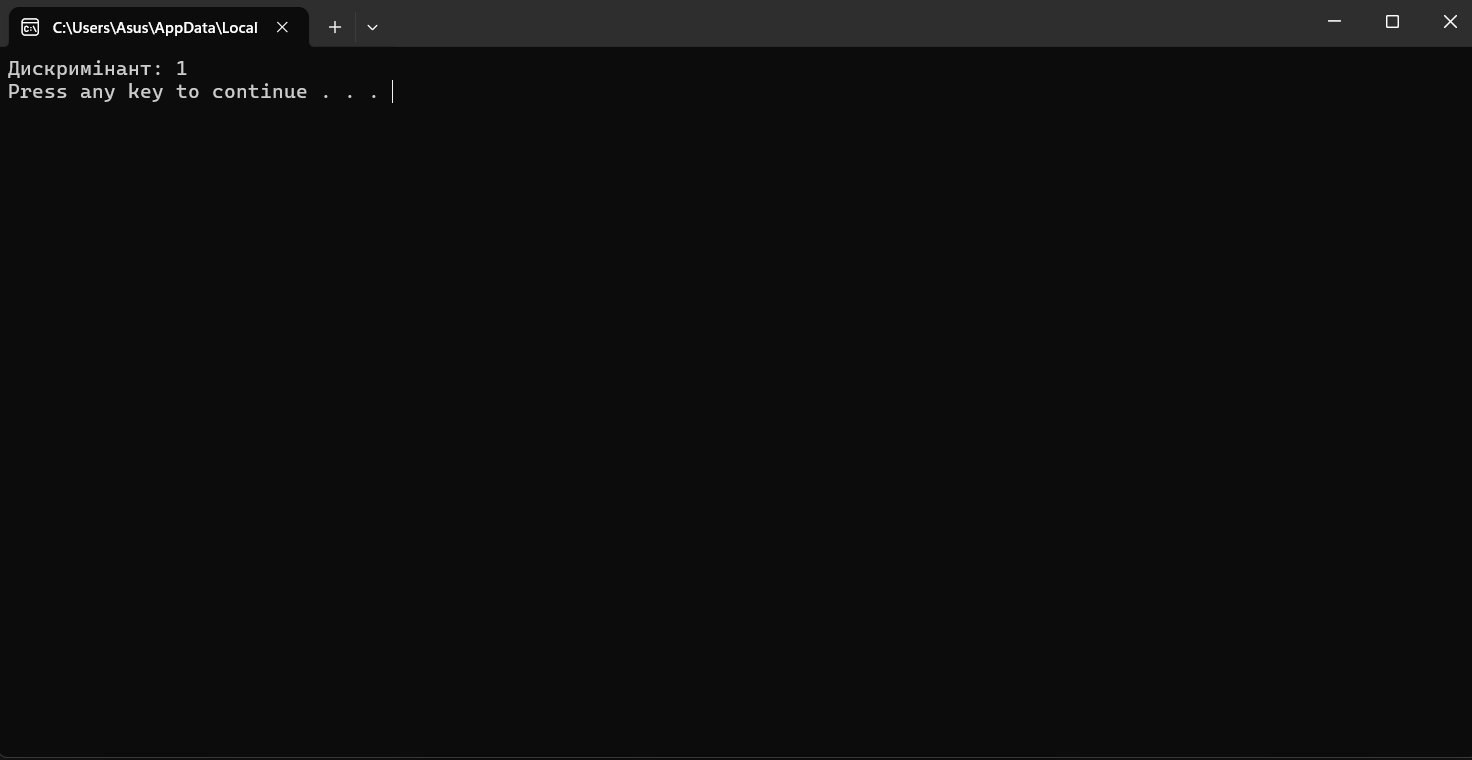
Рис. 5. Скриншот коду Task1.py

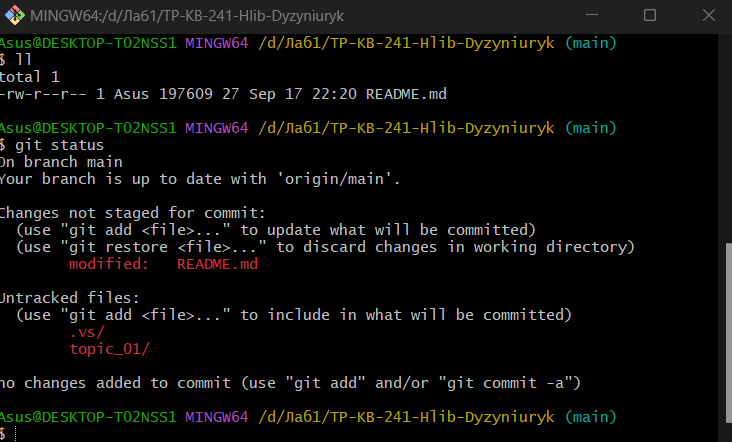
Рис. 6. Скриншот виконанного завдання

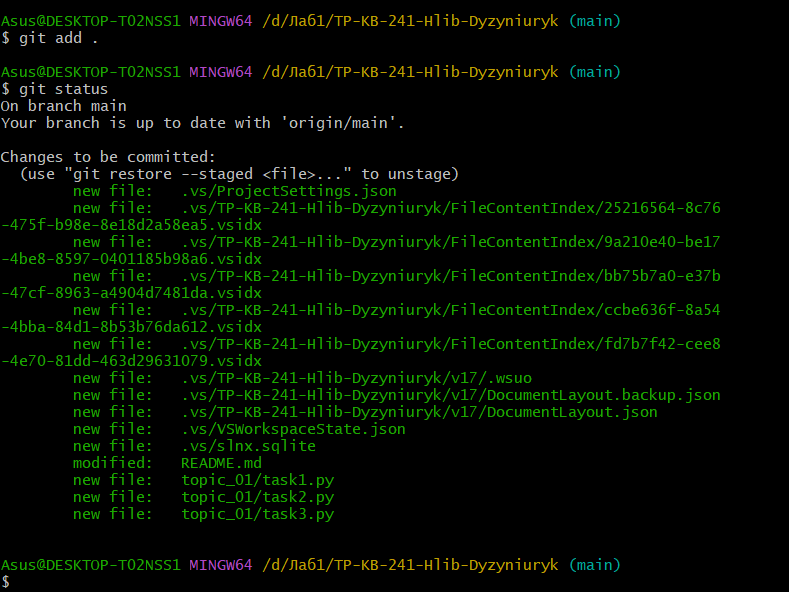
Рис. 7. Скриншот коду Task2.py

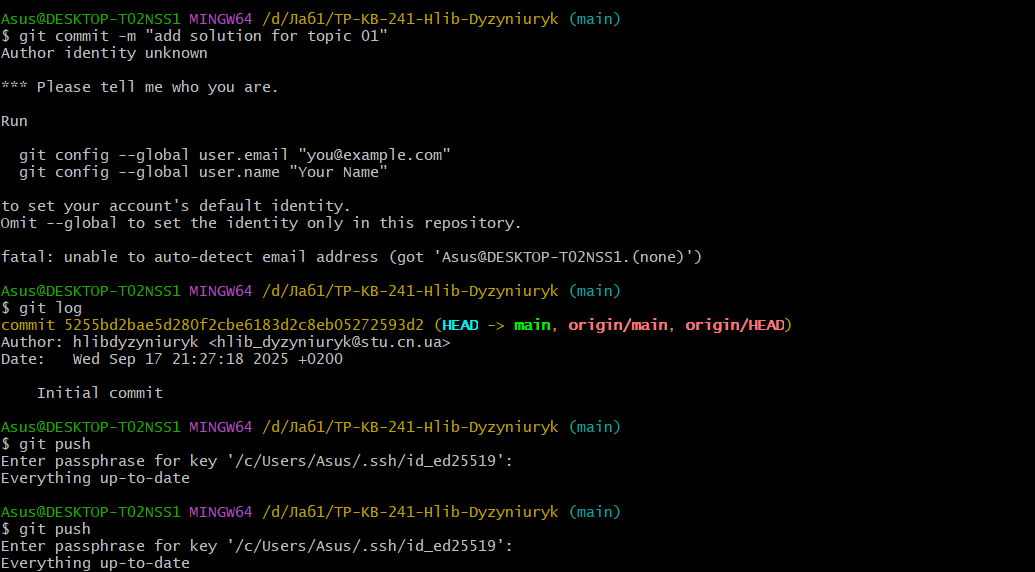
Рис. 8. Скриншот виконанного завдання

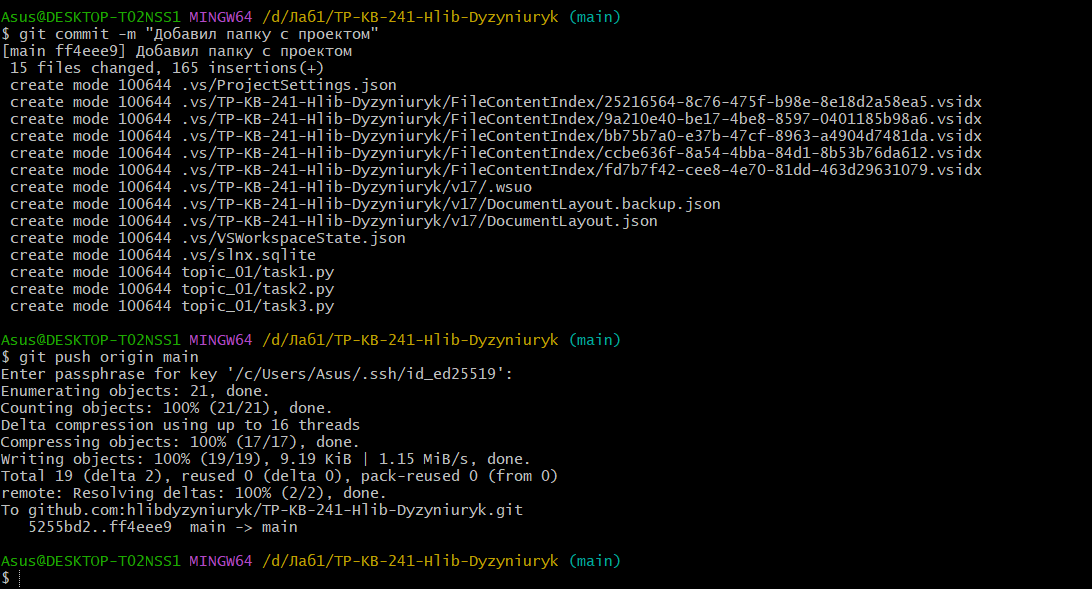
Рис. 9. Скриншот коду Task3.py

Рис. 10. Скриншот виконанного завданням

Рис. 11. Виконана команда Git Status

Рис. 12. Зміні готові для збереження

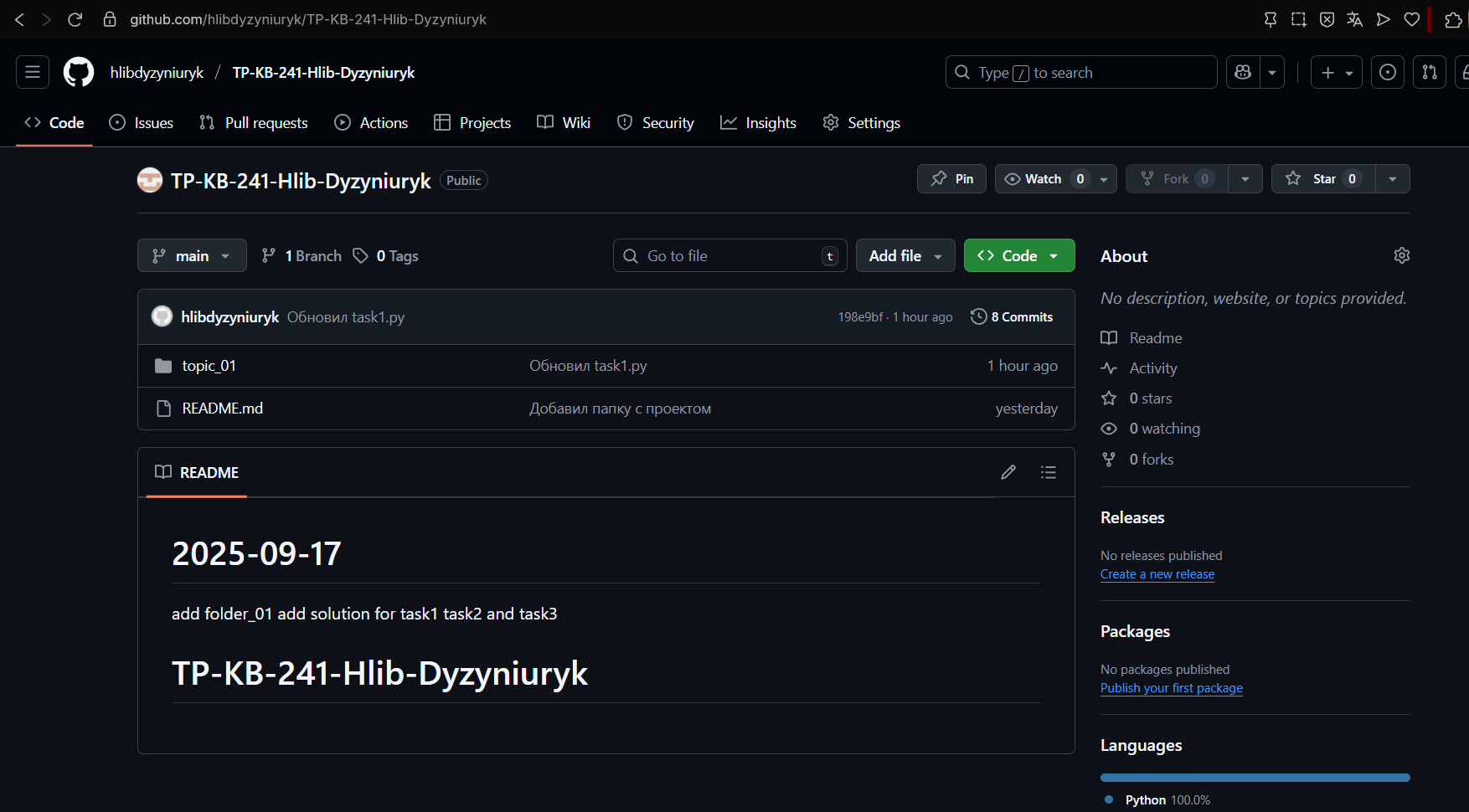
Рис. 13. Спроба збереження змін

Рис. 14. Збережено зміни

Посилання на github:

https://github.com/hlibdyzyniuryk/TP-KB-241-Hlib-Dyzyniuryk

Знімок екрану з посилання на github:



Висновок:

У процесі виконання лабораторної роботи я повторив основні дії з використання Git та GitHub: створення віддаленого репозиторію, його клонування на локальний комп’ютер, організацію структури проєкту у вигляді папок і файлів. Також закріпив навички виконання базових команд (git add, git commit, git push) та перевірки результатів у веб-інтерфейсі GitHub. Робота допомогла систематизувати знання й підготувала основу для подальших практичних завдань.

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

**Виконав: Дизинюрик Гліб Олександрович КБ-241**

Звіт до теми №2: Умовний перехід

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

**1. Програма для пошуку коренів квадратного рівняння**

Програма призначена для обчислення дискримінанта та знаходження коренів квадратного рівняння. Користувач вводить коефіцієнти a, b і c, після чого програма визначає, скільки є дійсних коренів, і виводить їх на екран.

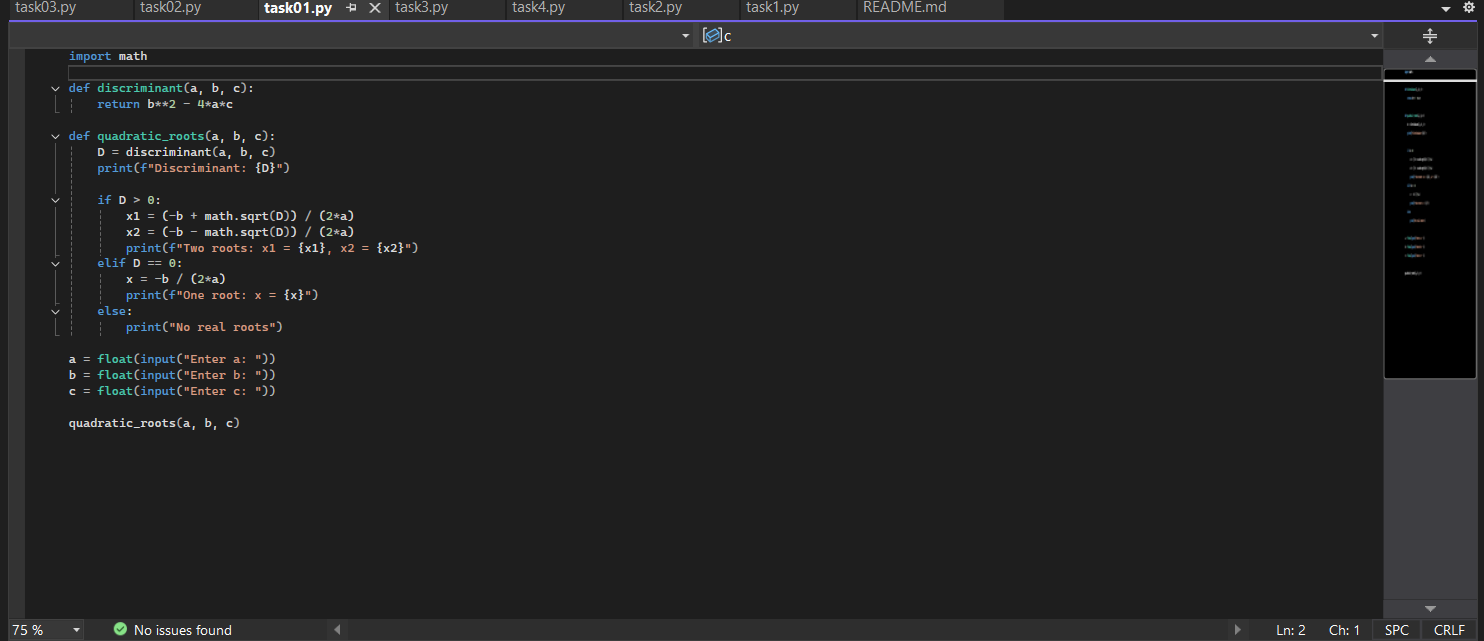
**2. Програма «Калькулятор» з використанням if-else**

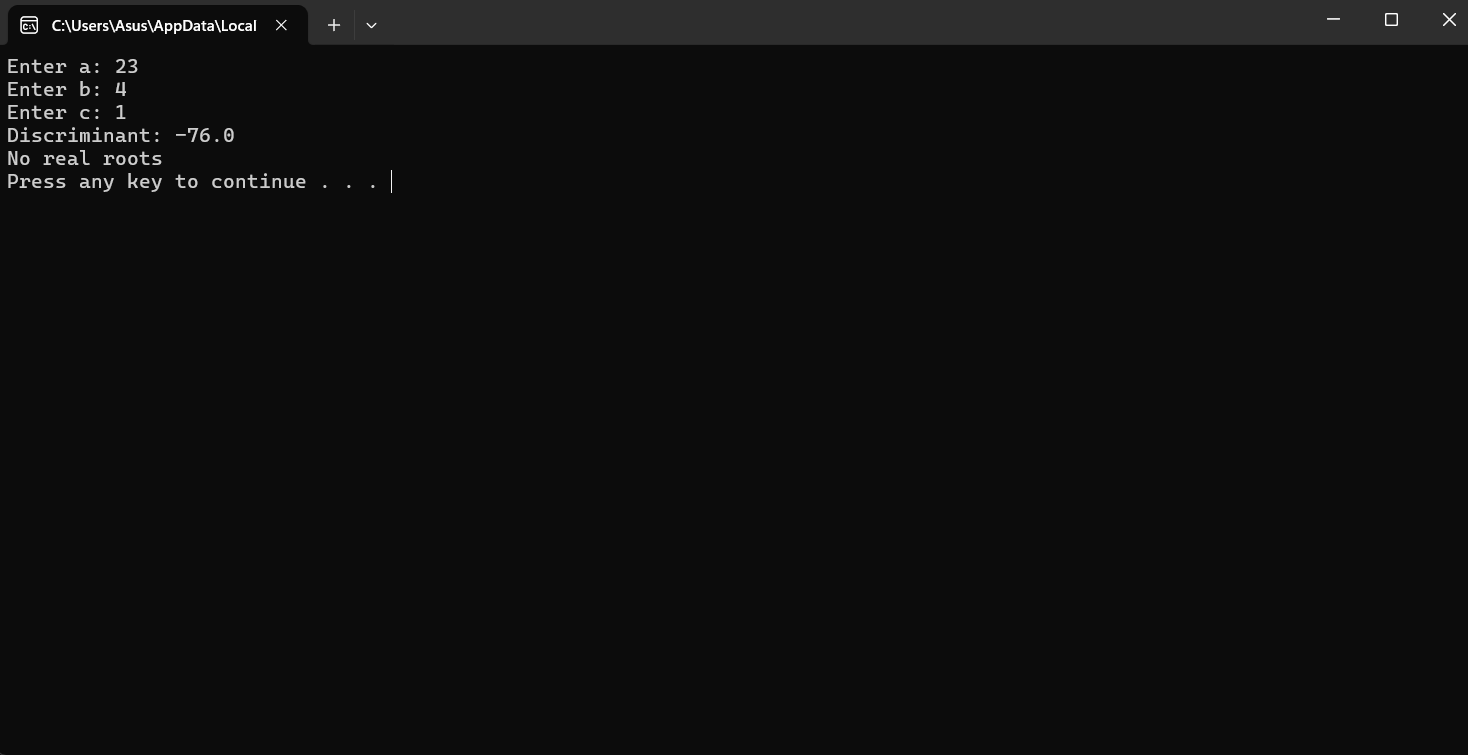
Програма виконує основні арифметичні операції — додавання, віднімання, множення і ділення. Кожна операція оформлена як окрема функція. Користувач вводить два числа та знак операції, а програма обчислює і показує результат.

**3. Програма «Калькулятор» з використанням match-case**

Програма має ту саму логіку, що й попередня, але для вибору потрібної дії використовується конструкція match-case, яка спрощує перевірку варіантів операцій і робить код зручнішим для читання.

**Хід роботи**

Рис. 1. Скриншот коду до першого завдання.

Рис. 2. Скриншот результату виведеного у терміналі.

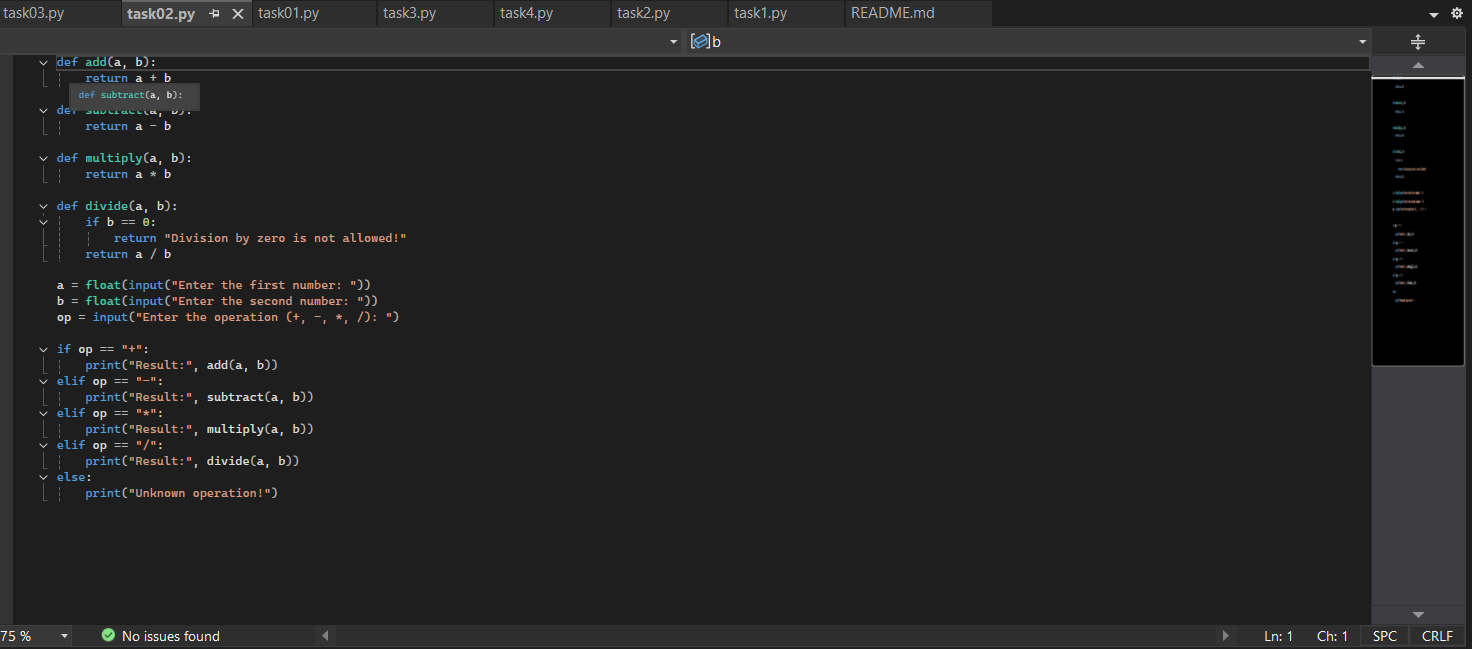
Рис. 3. Скриншот коду до другого завдання.

Рис. 4. Скриншот результату виведеного у терміналі.

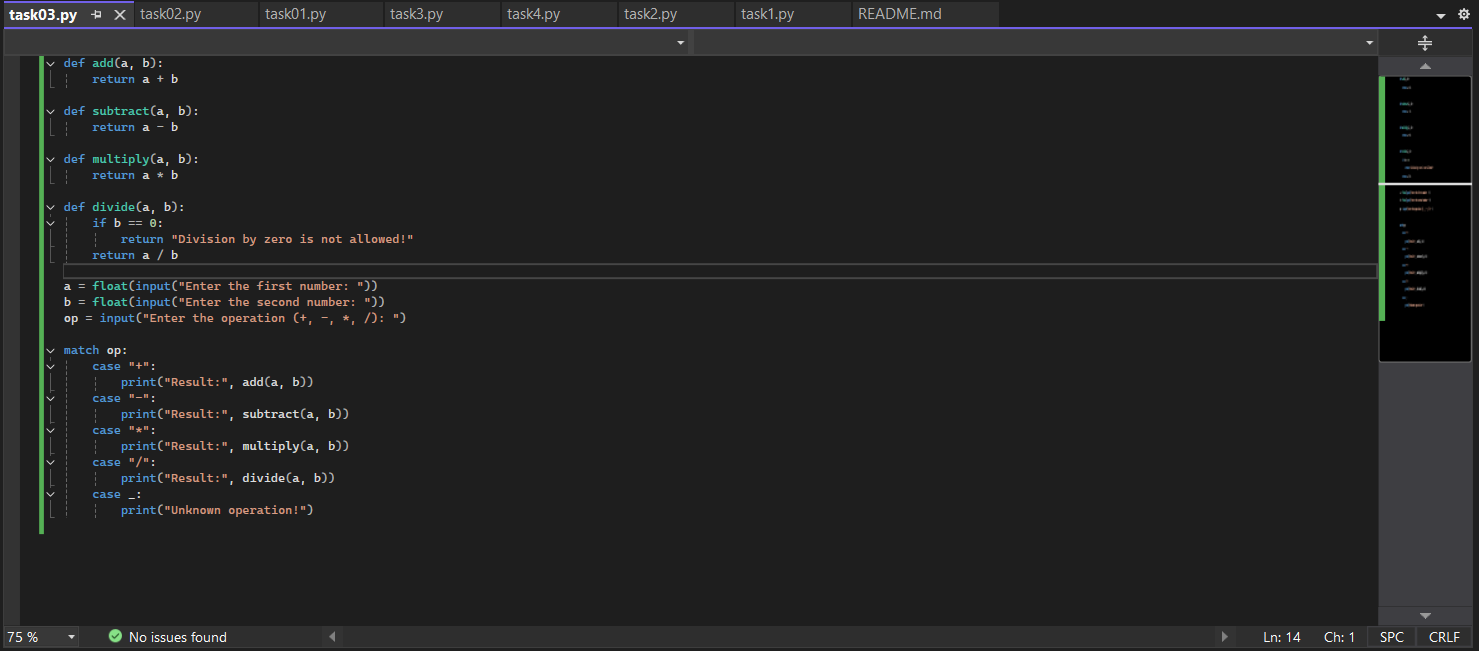
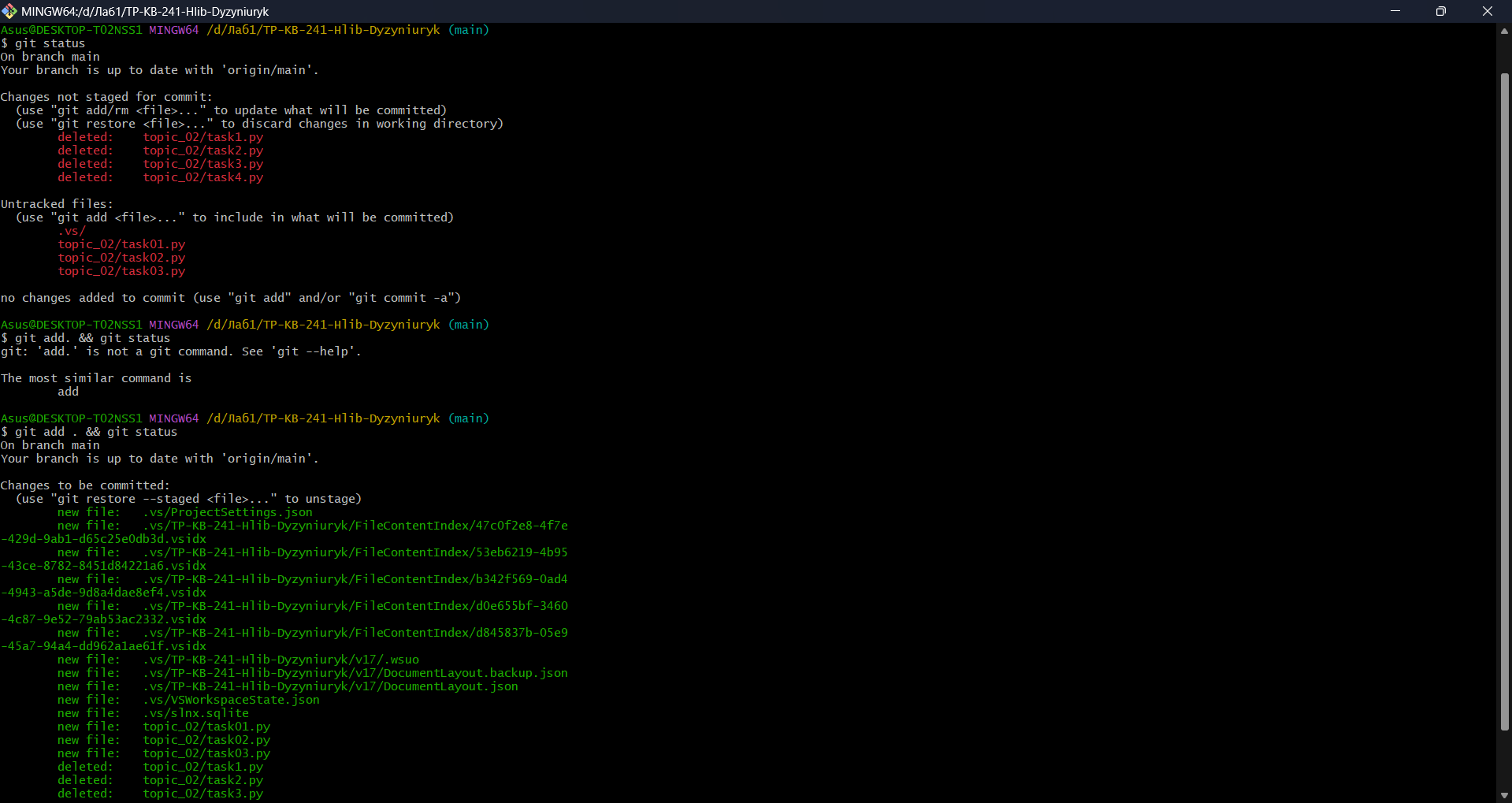
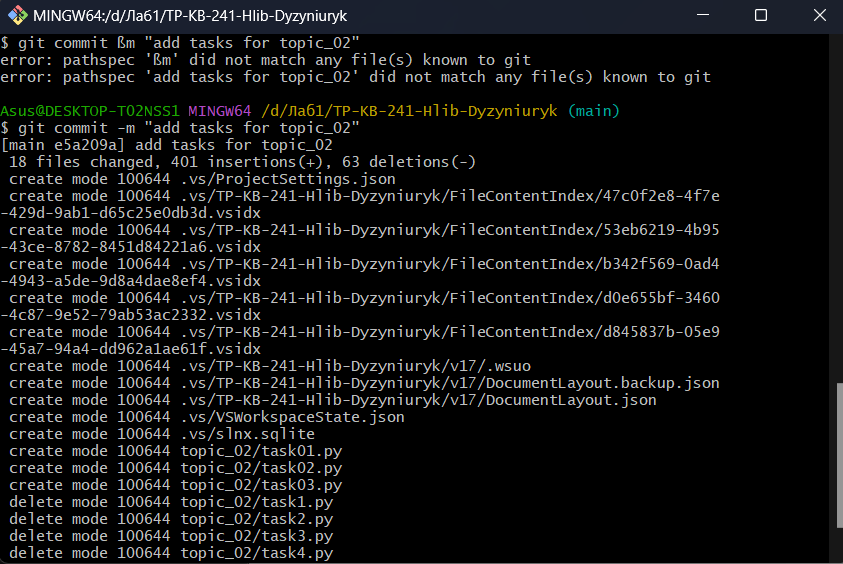
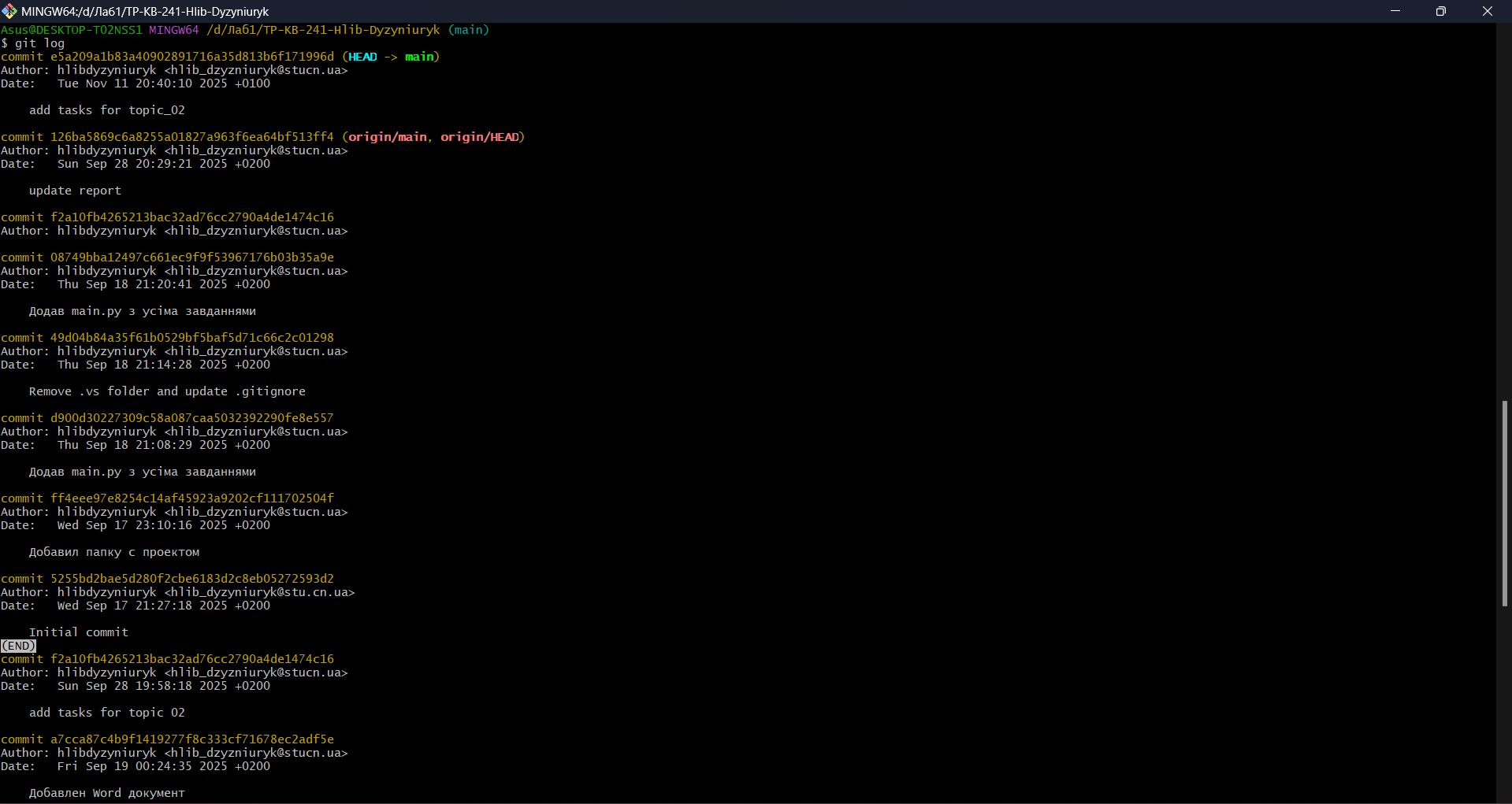
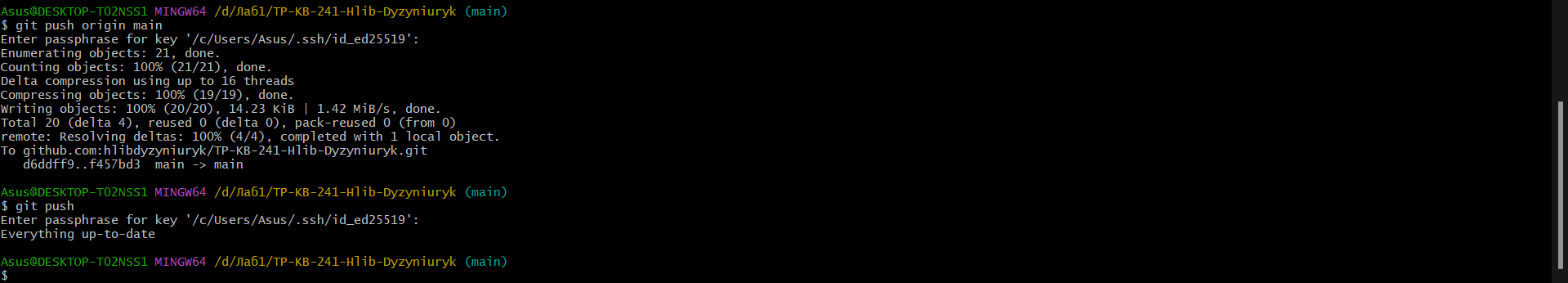
Рис. 5. Скриншот коду до третього завдання.

Рис. 6. Скриншот результату виведеного у терміналі.

Рис. 7. Скриншот виконаних команд у терміналі Git Bash.

Рис. 8. Скриншот виконаних команд у терміналі Git Bash.

Рис. 9. Скриншот виконаних команд у терміналі Git Bash.

Рис. 10. Скриншот виконаних команд у терміналі Git Bash.

**Висновок:**

У ході виконання трьох завдань були створені програми, що демонструють використання основних конструкцій мови Python. Перша програма обчислює дискримінант і знаходить корені квадратного рівняння. Друга реалізує калькулятор із застосуванням умовних операторів if-else, а третя — з конструкцією match-case. У результаті було закріплено навички роботи з функціями, умовами та введенням-виведенням даних у Python.

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

**Виконав: Дизинюрик Гліб Олександрович КБ-241**

Звіт до теми №3: Цикли

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

**Завдання 1. Калькулятор із постійними запитами на введення**

Це калькулятор із безперервним введенням даних, який дозволяє користувачу багаторазово виконувати обчислення, доки не буде введена команда для завершення роботи. Ця програма демонструє практичне застосування циклу **while**.

**Завдання 2. Тестування функцій списків**

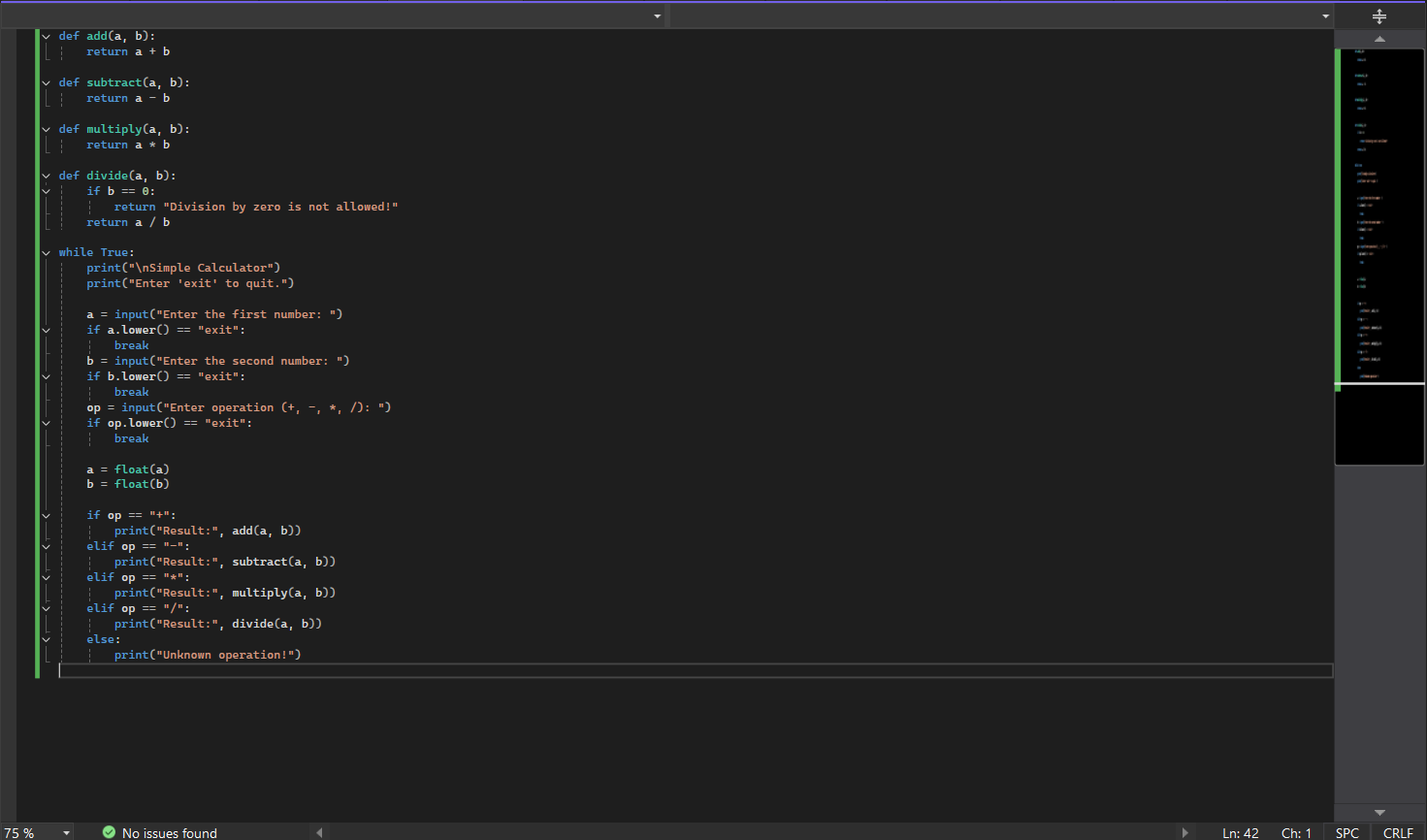
У програмі протестовано основні методи списків: **append()**, **extend()**, **insert()**, **remove()**, **clear()**, **sort()**, **reverse()**, **copy()**. Вона послідовно демонструє, як змінюється список після виконання кожної з операцій.

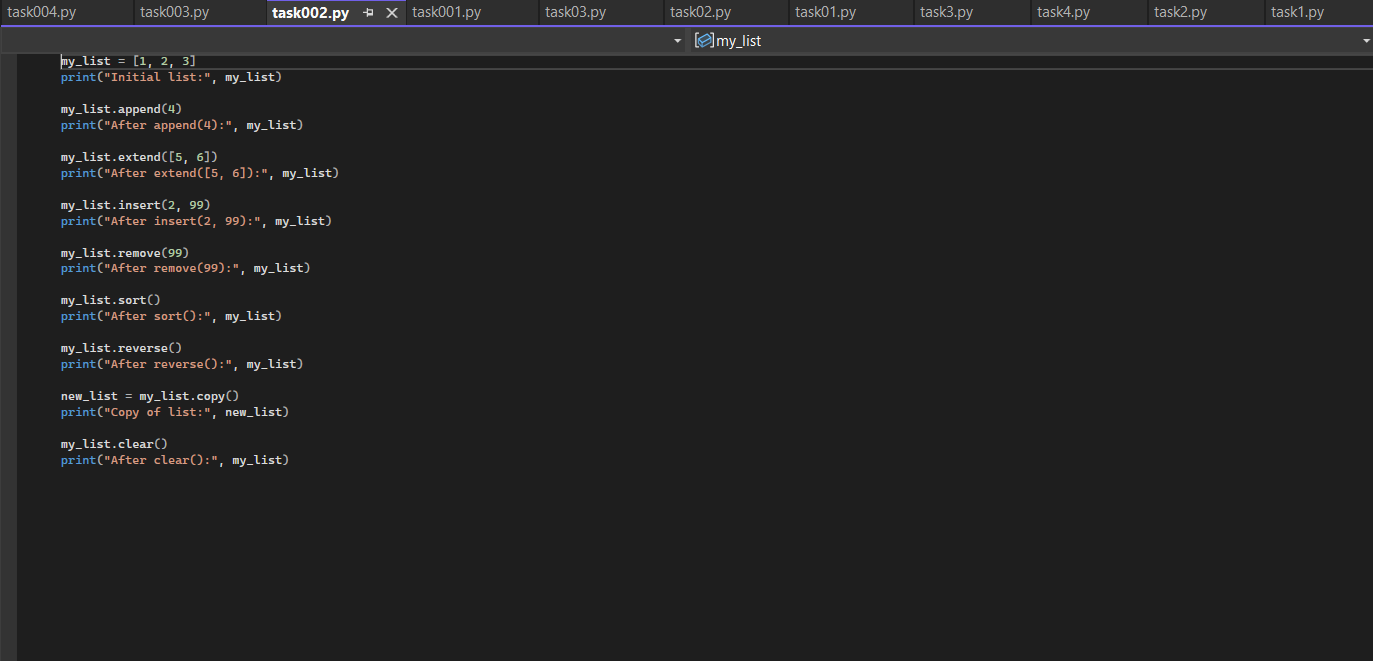
**Завдання 3. Тестування функцій словників**

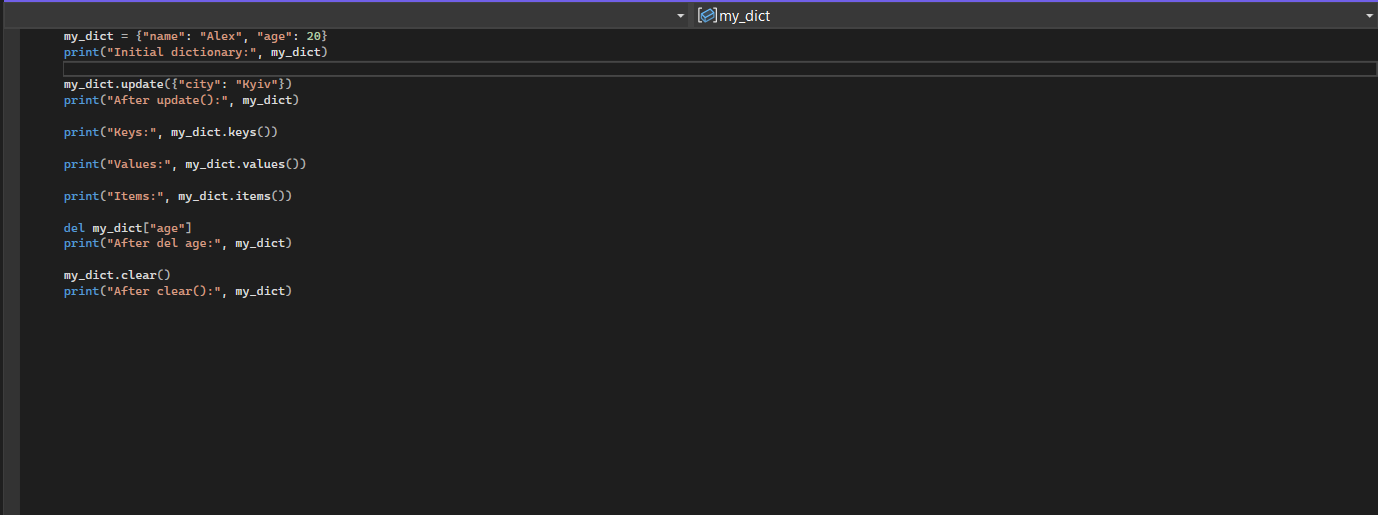
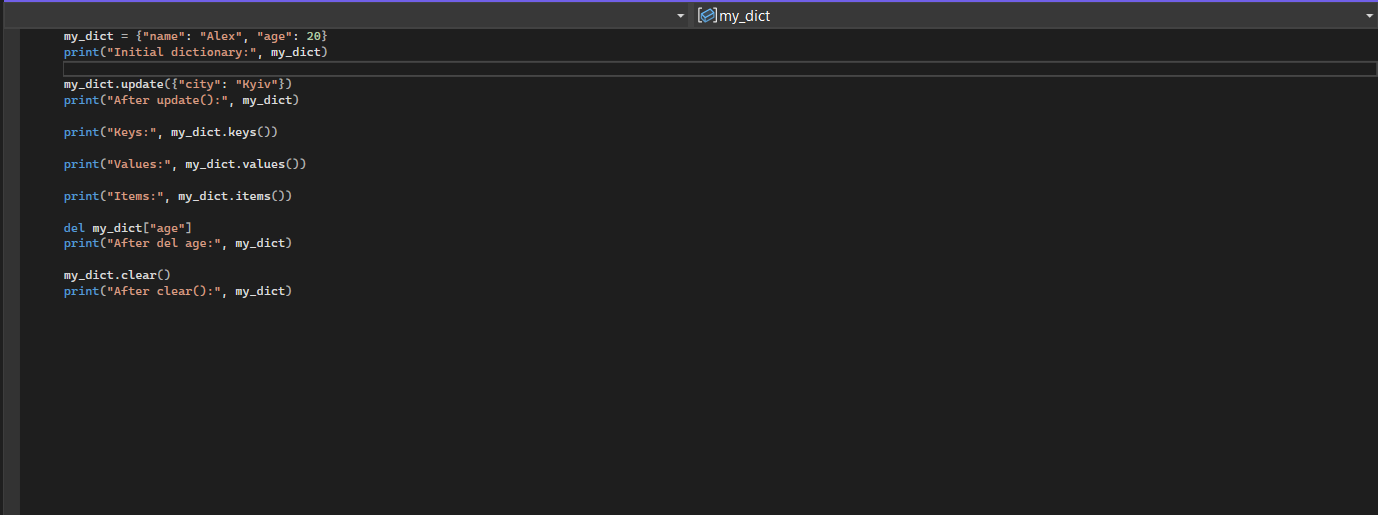
Програма перевіряє роботу методів словників — **update()**, **del**, **clear()**, **keys()**, **values()**, **items()**. Під час виконання демонструється додавання, оновлення та видалення даних, а також перегляд ключів і значень словника.

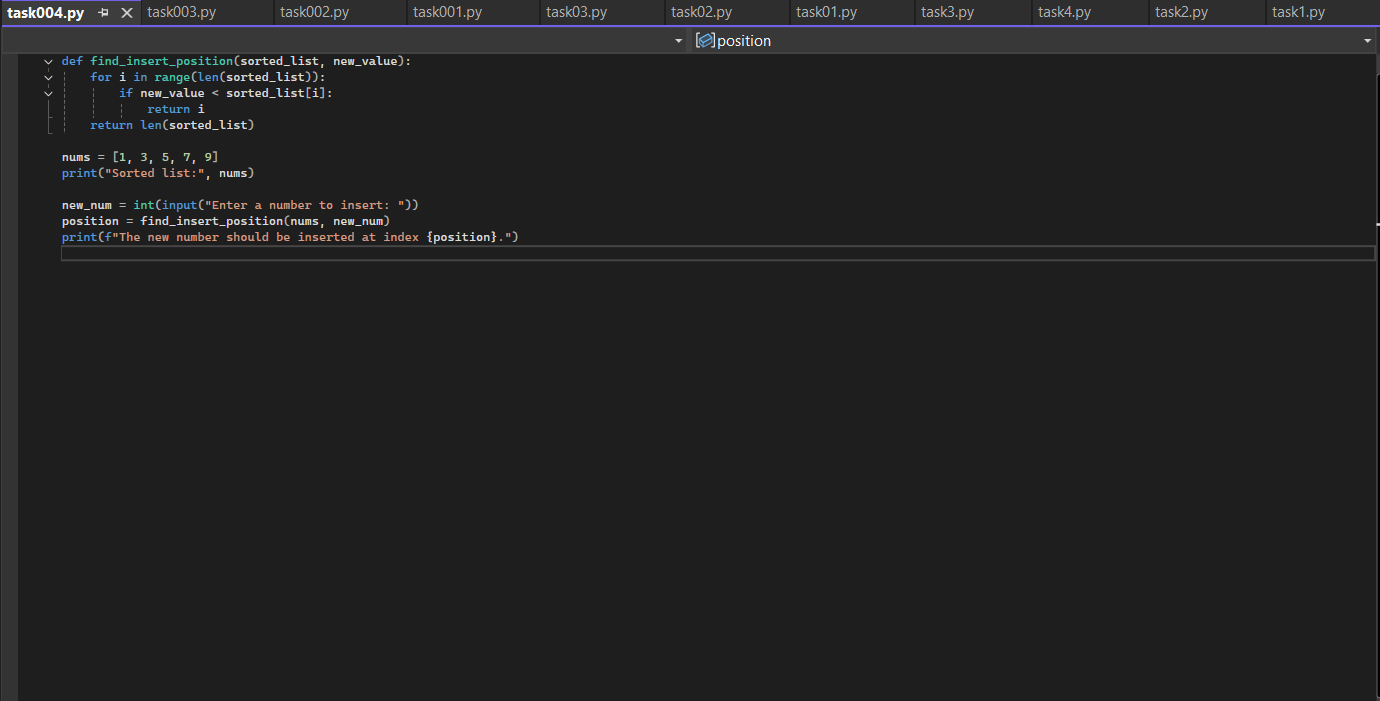
**Завдання 4. Пошук позиції для вставки нового елемента**  
Програма містить функцію, яка отримує впорядкований список і визначає позицію, куди потрібно вставити новий елемент, щоб зберегти порядок сортування. Реалізація ґрунтується на використанні циклу **for** та умовних перевірок.

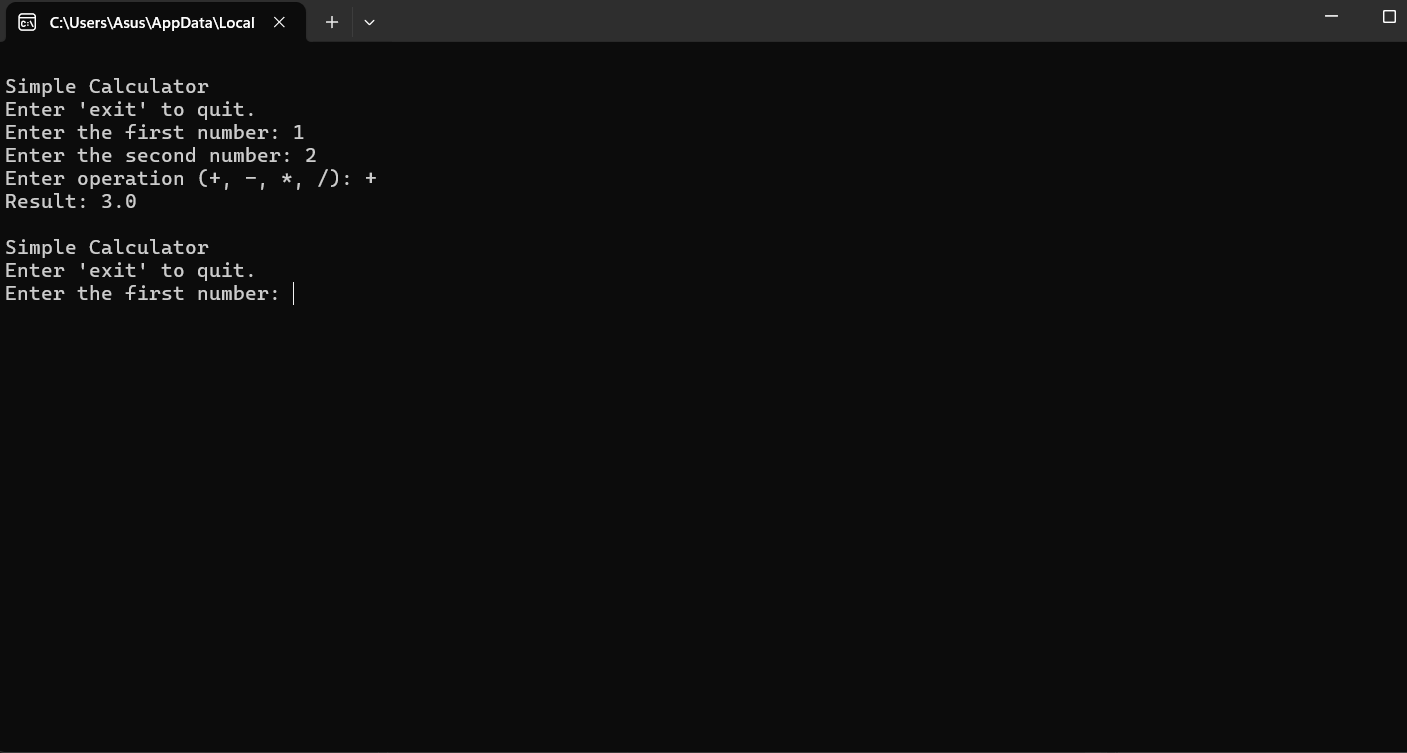
**Хід роботи**

Рис. 1. Скриншот написаного коду до першого завдання.

Рис. 2. Скриншот написаного коду до другого завдання.

Рис. 3. Скриншот написаного коду до третього завдання.

Рис. 4. Скриншот написаного коду до четвертого завдання.

Рис. 5. Скриншот виведеного результату.

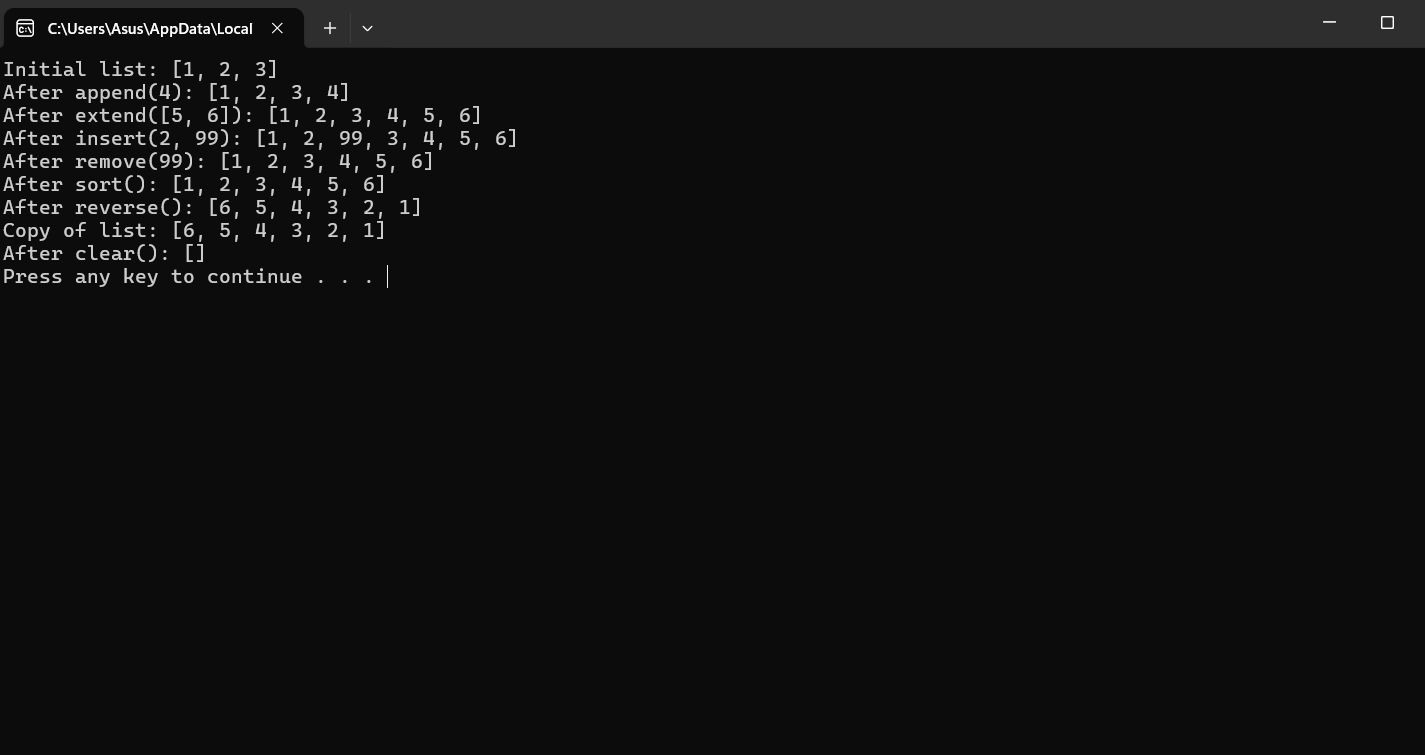
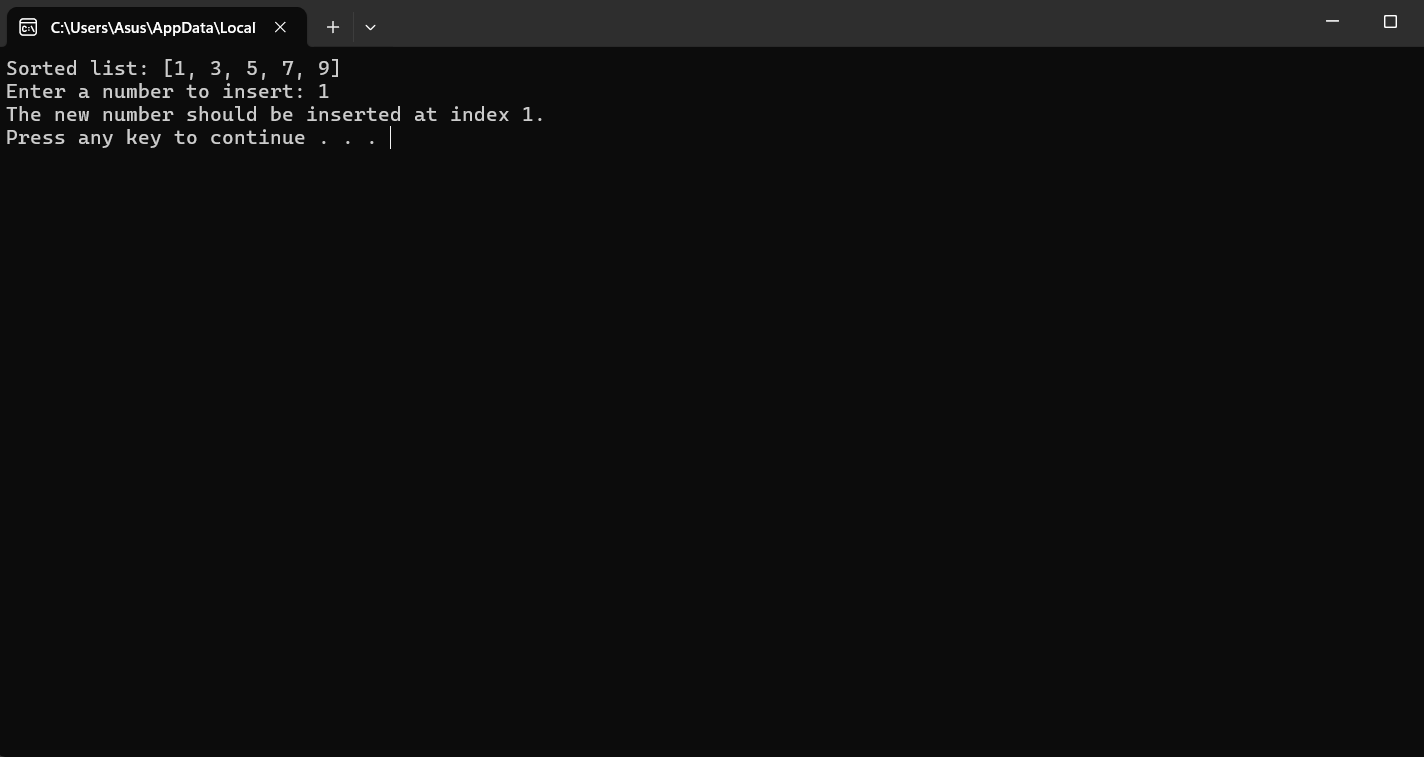
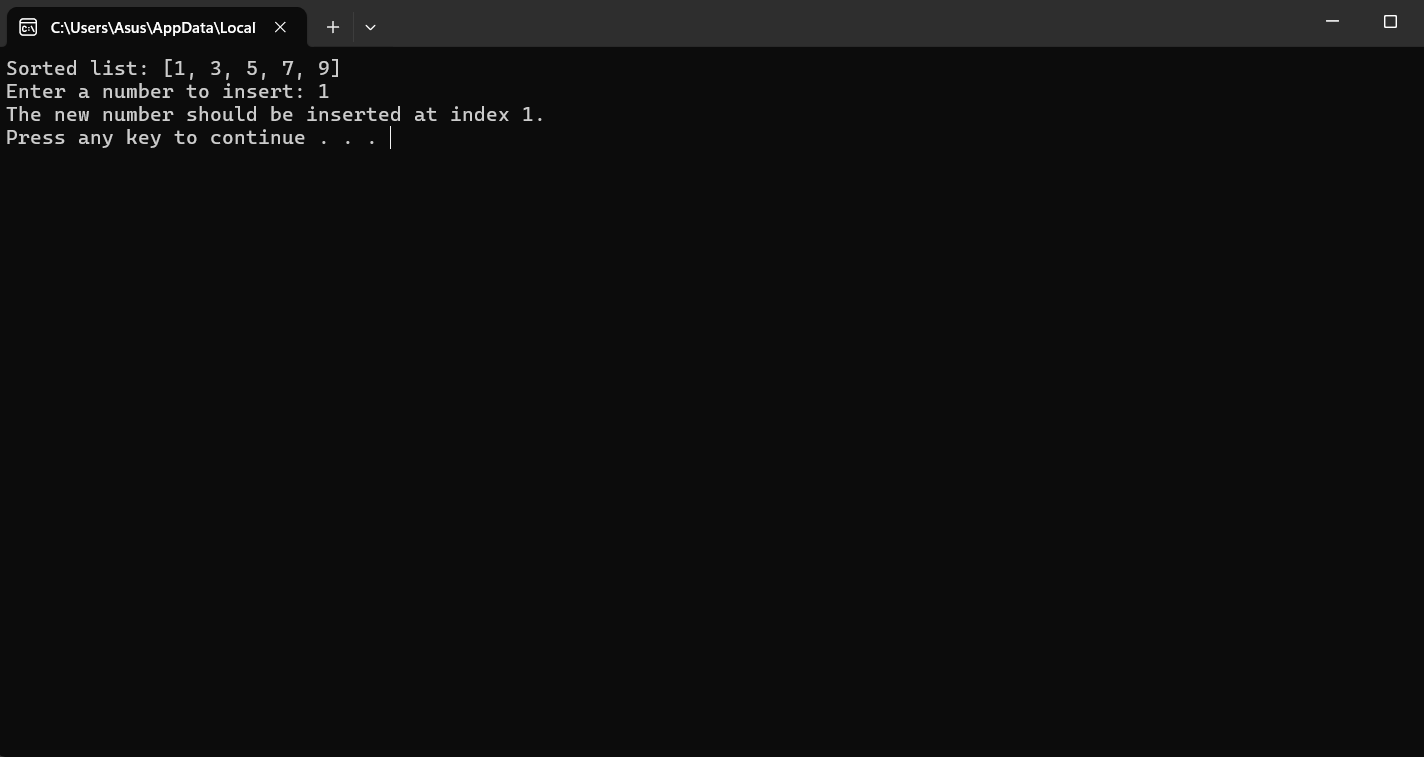
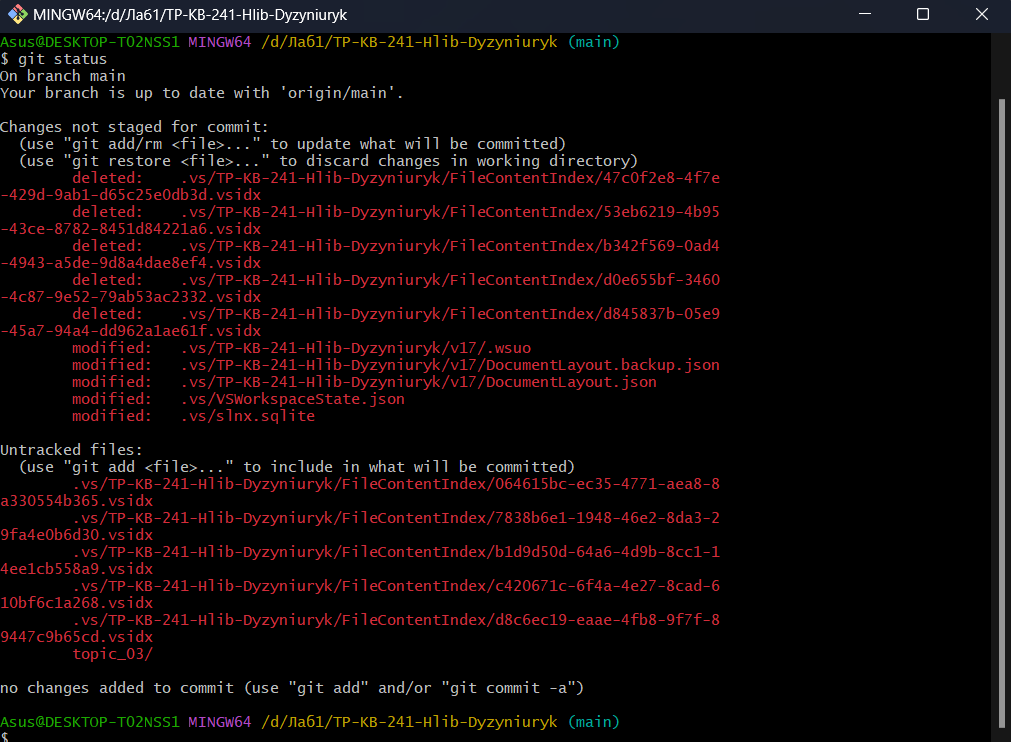
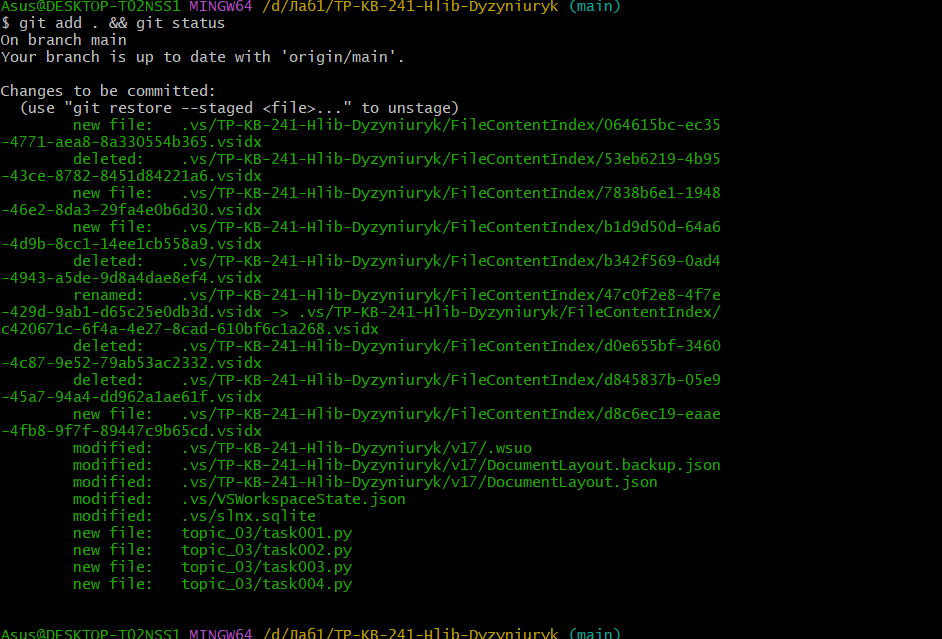
Рис. 6. Скриншот виведеного результату.

Рис. 7. Скриншот виведеного результату.

Рис. 8. Скриншот виведеного результату.

Рис. 9. Скриншот виконання команд.

Рис. 10. Скриншот виконання команд.

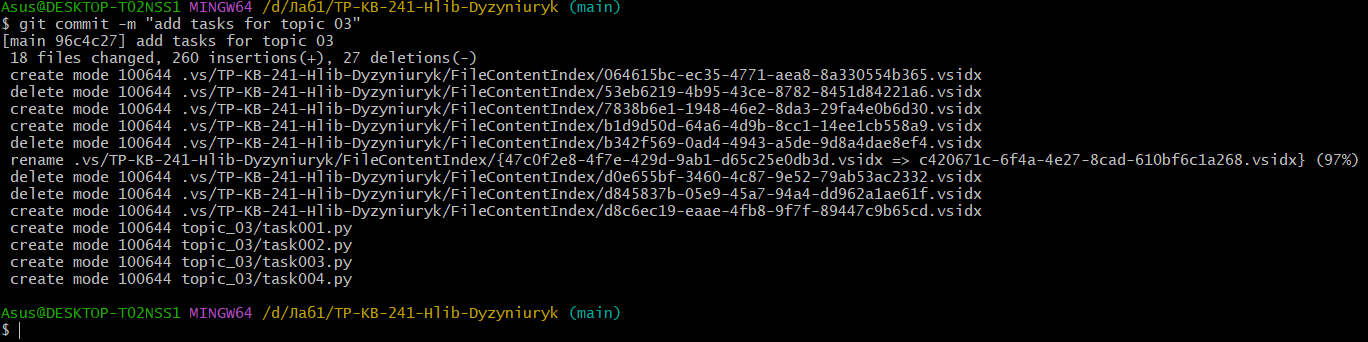
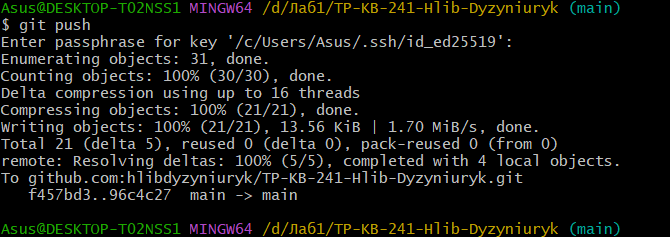
Рис. 11. Скриншот виконання команд.

Рис. 12. Скриншот виконання команд.

Рис. 13. Скриншот виконання команд.

**Висновок:**

У ході виконання практичних завдань до теми «Цикли» було розглянуто приклади використання різних видів циклів у мові Python. Під час реалізації програм вдалося закріпити навички роботи з повторюваними процесами, обробкою списків та словників, а також з умовами всередині циклів. Розроблені програми продемонстрували, як за допомогою циклів можна автоматизувати обчислення, керувати даними у структурах та реалізовувати логіку повторюваних дій у програмі.

**Звіт про виконання практичних завдань до лекцій з курсу «Технології програмування на мові Python»**  
**Виконав:** Дизинюрик Гліб Олександрович, група КБ-241  
**Тема №4:** Обробка виняткових ситуацій (Exceptions)

Під час виконання практичного завдання до теми №4 було розроблено програму калькулятор, у якій реалізовано механізм обробки виняткових ситуацій. Це дозволило зробити програму більш стійкою до помилок користувача.

**Завдання 1. Розширення програми калькулятора обробкою виняткових ситуацій**

У цьому завданні калькулятор доповнено функцією введення чисел із використанням конструкції try-except. Якщо користувач вводить нечислове значення, програма виводить повідомлення про помилку і повторює запит. Це дозволяє уникнути аварійного завершення роботи програми через помилки введення.

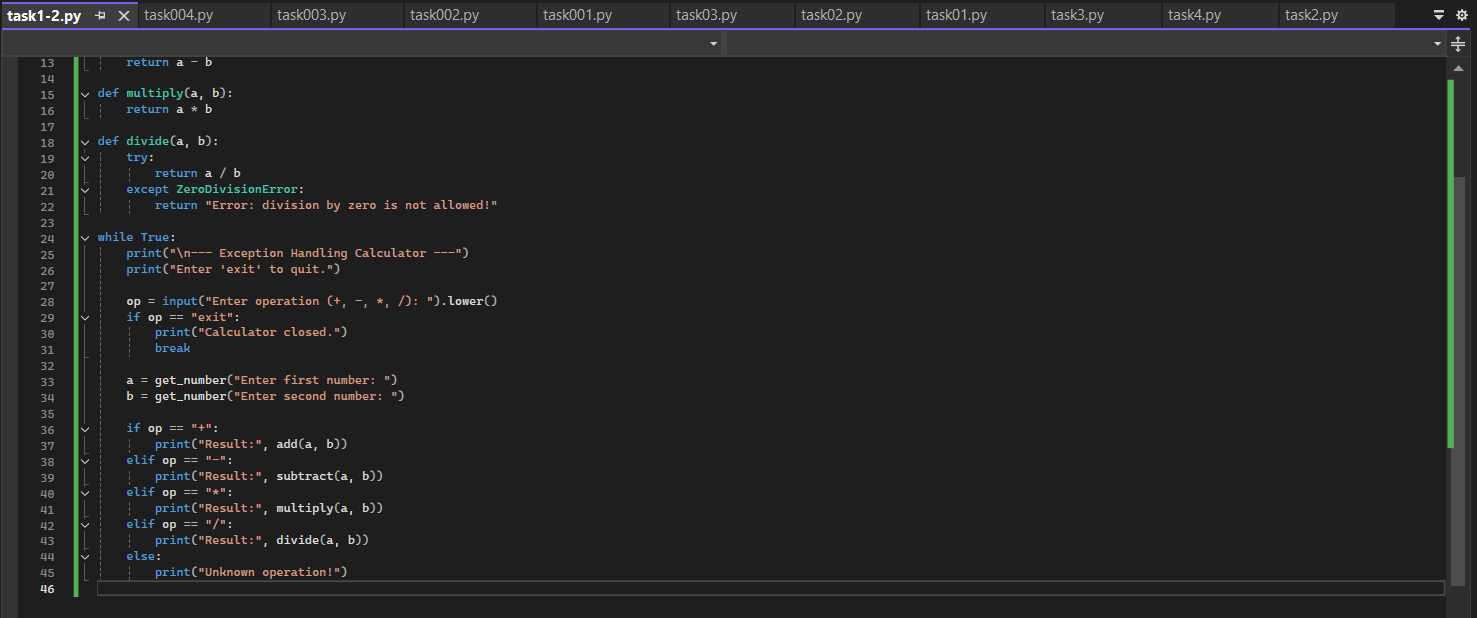
**Завдання 2. Обробка винятку ділення на нуль**

У функції ділення додано блок try-except, який перевіряє, чи друге число не дорівнює нулю. У разі спроби ділення на нуль програма не завершується з помилкою, а повідомляє користувача про неможливість виконання операції. Це демонструє практичне застосування обробки винятків у реальних задачах.

**Завдання 3. Ознайомлення зі списком виняткових ситуацій у Python**

Ознайомитись зі списком виняткових ситуацій за посиланням <https://docs.python.org/3/library/exceptions.html>

**Хід роботи**

Рис. 1. Скриншот написаного коду.

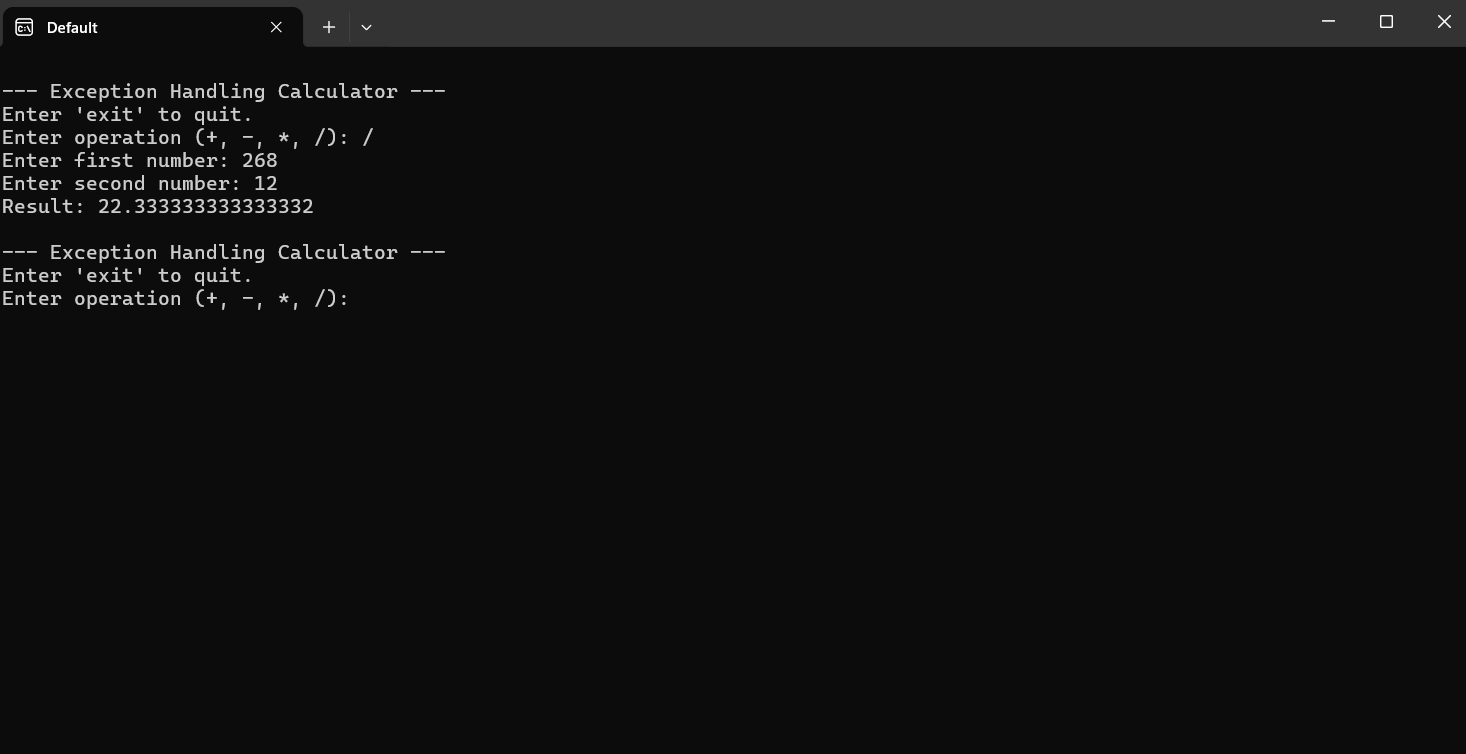
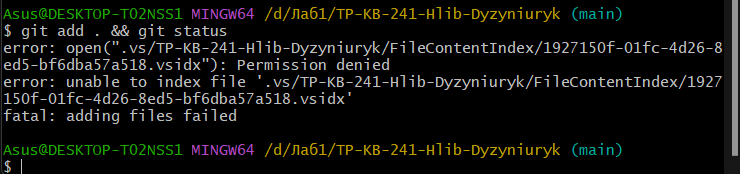
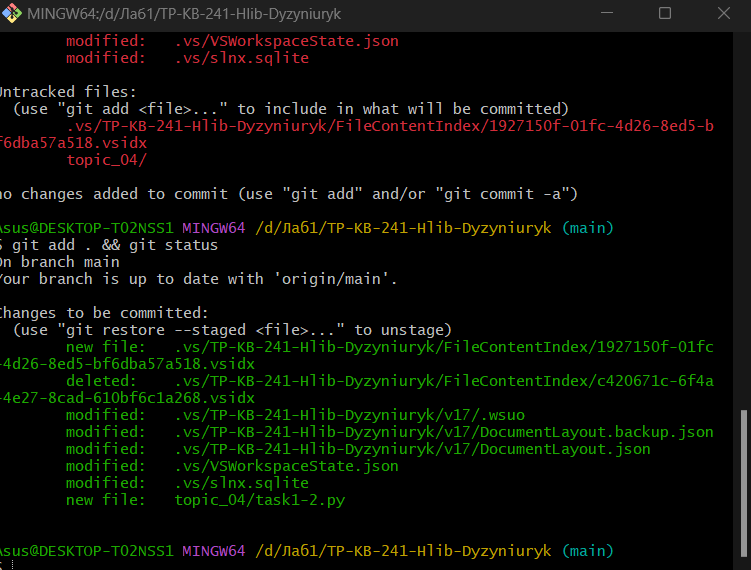
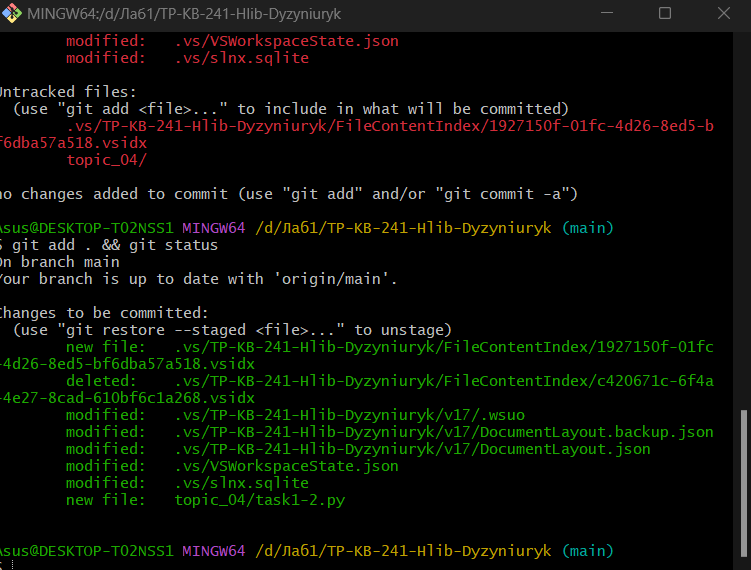
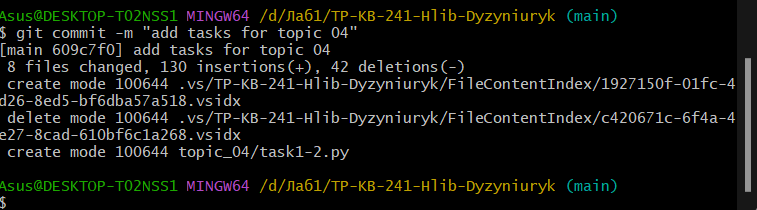
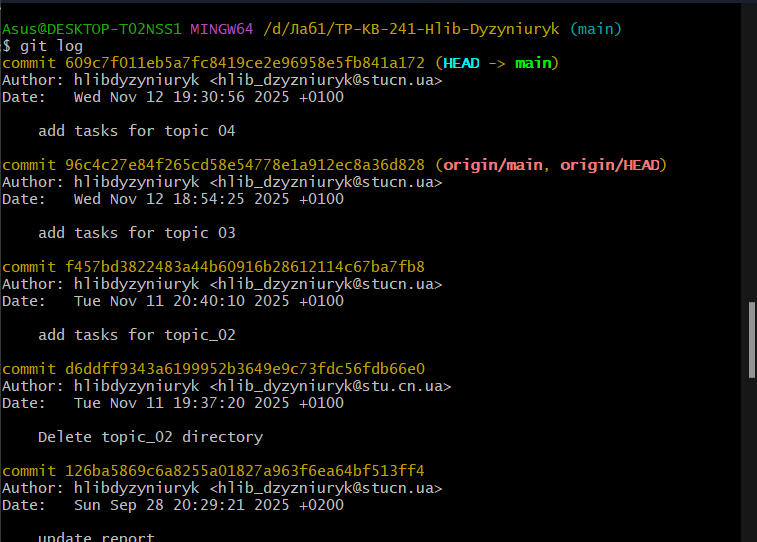
Рис. 2. Скриншот виведеного результату.

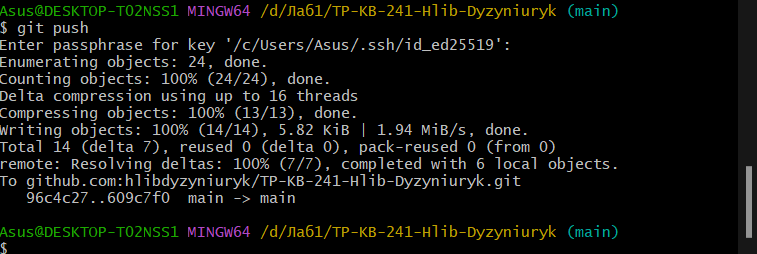
Рис. 3. Скриншот виконаних команд.

Рис. 4. Скриншот виконаних команд.

Рис. 5. Скриншот виконаних команд.

Рис. 6. Скриншот виконаних команд.

Рис. 7. Скриншот виконаних команд.

Рис. 8. Скриншот виконаних команд.

У третьому завданні потрібно було ознайомитись зі списком стандартних виняткових ситуацій, які можуть виникати під час виконання програм у Python. У мові передбачено багато типів помилок, які можна перехопити та обробити за допомогою конструкції try-except, щоб програма не завершувалася аварійно. Серед основних винятків можна виділити такі: **ValueError** — виникає, коли користувач вводить некоректне значення, наприклад текст замість числа; **ZeroDivisionError** — з’являється при спробі поділити число на нуль; **TypeError** — вказує на виконання операцій між несумісними типами даних; **IndexError** — виникає, якщо програма намагається звернутися до елемента списку, якого не існує; **KeyError** — з’являється при зверненні до неіснуючого ключа у словнику; **FileNotFoundError** — сигналізує про спробу відкрити файл, який відсутній. Ознайомлення з цими винятками допомагає краще розуміти можливі помилки під час виконання коду та вміти правильно їх обробляти для стабільної роботи програми.

Висновок:

У результаті виконання роботи було закріплено знання щодо обробки виняткових ситуацій у Python. Було створено програму калькулятор, яка здатна розпізнавати помилки введення даних і коректно реагувати на ділення на нуль, не перериваючи виконання. Ознайомлення зі списком стандартних винятків дозволило зрозуміти, які типові помилки можуть виникати під час роботи програм, та як їх можна передбачити й обробити. Отримані знання допомогли зробити програму більш надійною, стійкою до помилок користувача та зручною у використанні.

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

**Виконав: Дизинюрик Гліб Олександрович КБ-241**

Звіт до теми №5: Бібліотеки

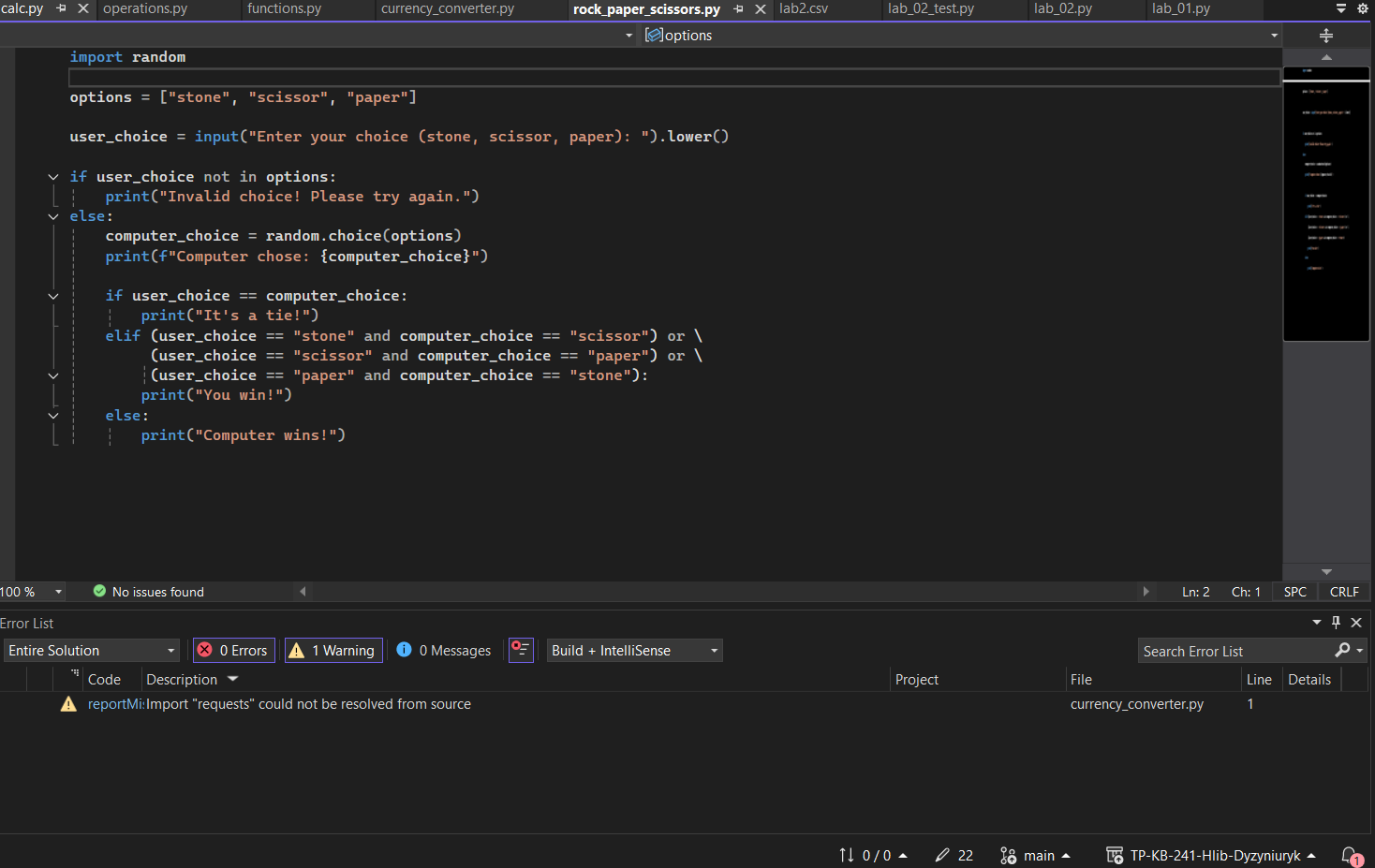
### **Мета роботи: н**авчитися використовувати умовні оператори та генерацію випадкових чисел у Python, засвоїти роботу з зовнішніми API та виконання HTTP-запитів, навчитися структурувати код за допомогою модулів та функцій.

### ****Завдання 1 — Гра “Камінь, ножиці, папір”****

Я створив програму на Python, яка дозволяє зіграти в гру **“Камінь, ножиці, папір”** з комп’ютером. Користувач вводить свій вибір — stone, scissor або paper. Комп’ютер випадково обирає свій варіант за допомогою модуля random. Далі програма порівнює вибори й визначає переможця за такими правилами:

* камінь перемагає ножиці;
* ножиці перемагають папір;
* папір перемагає камінь.

У результаті програма виводить, хто переміг — користувач, комп’ютер, чи це нічия.

Рис. 1. Скриншот написаного коду.

### **Рис. 2. Скриншот виведеного результату.**

### ****Завдання 2 — Конвертація валюти за даними НБУ****

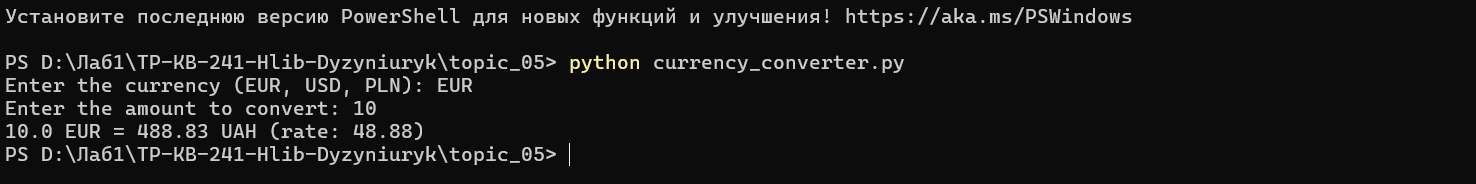
Наступна програма виконує **конвертацію іноземних валют у гривні** за актуальним курсом Національного банку України. Для цього я використав **модуль** requests, який дозволяє робити HTTP-запити. Програма звертається до офіційного API НБУ, отримує список валют та їхній курс.

Користувач вводить тип валюти (EUR, USD або PLN) і суму, яку потрібно перевести.

Програма обчислює еквівалент у гривнях і виводить результат.

Наприклад: 10 USD = 430.50 UAH (rate: 43.05)

### **Рис. 3. Скриншот написаного коду.**

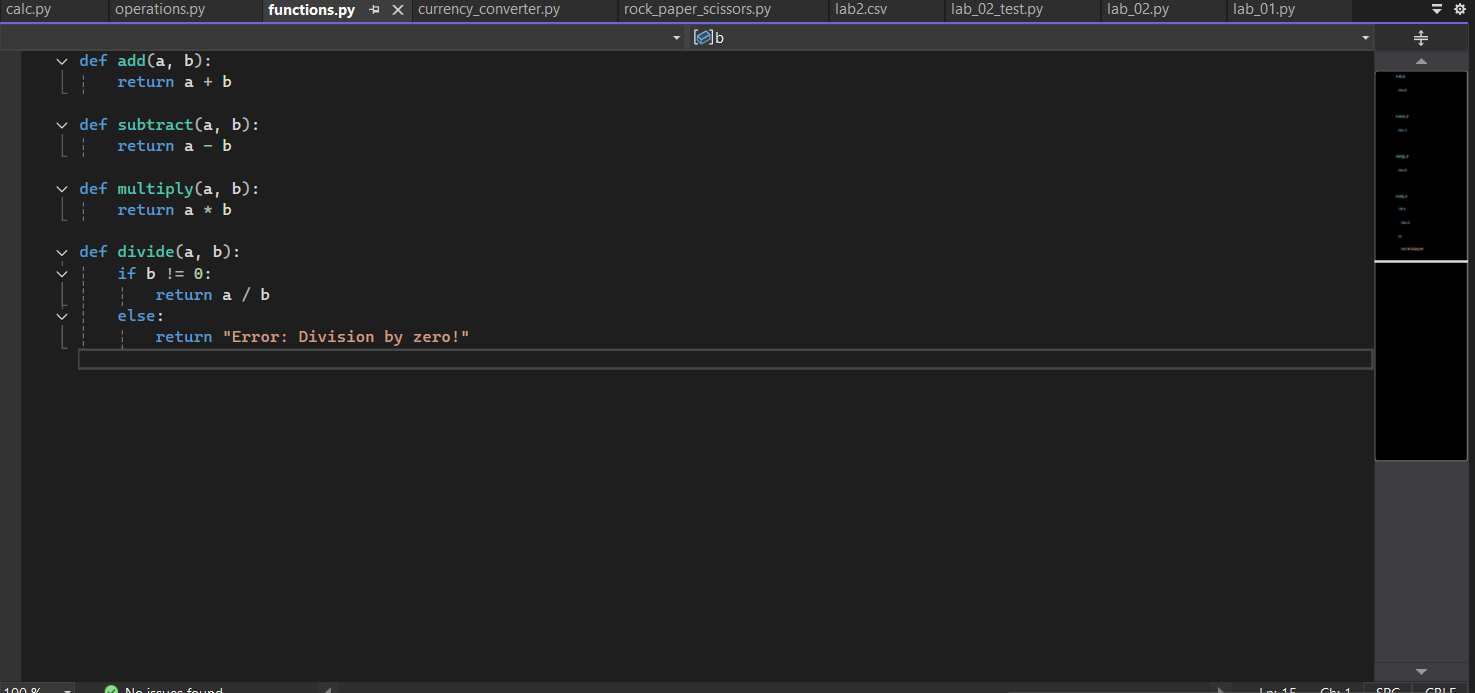
**Рис. 4. Скриншот виведеного результату.**

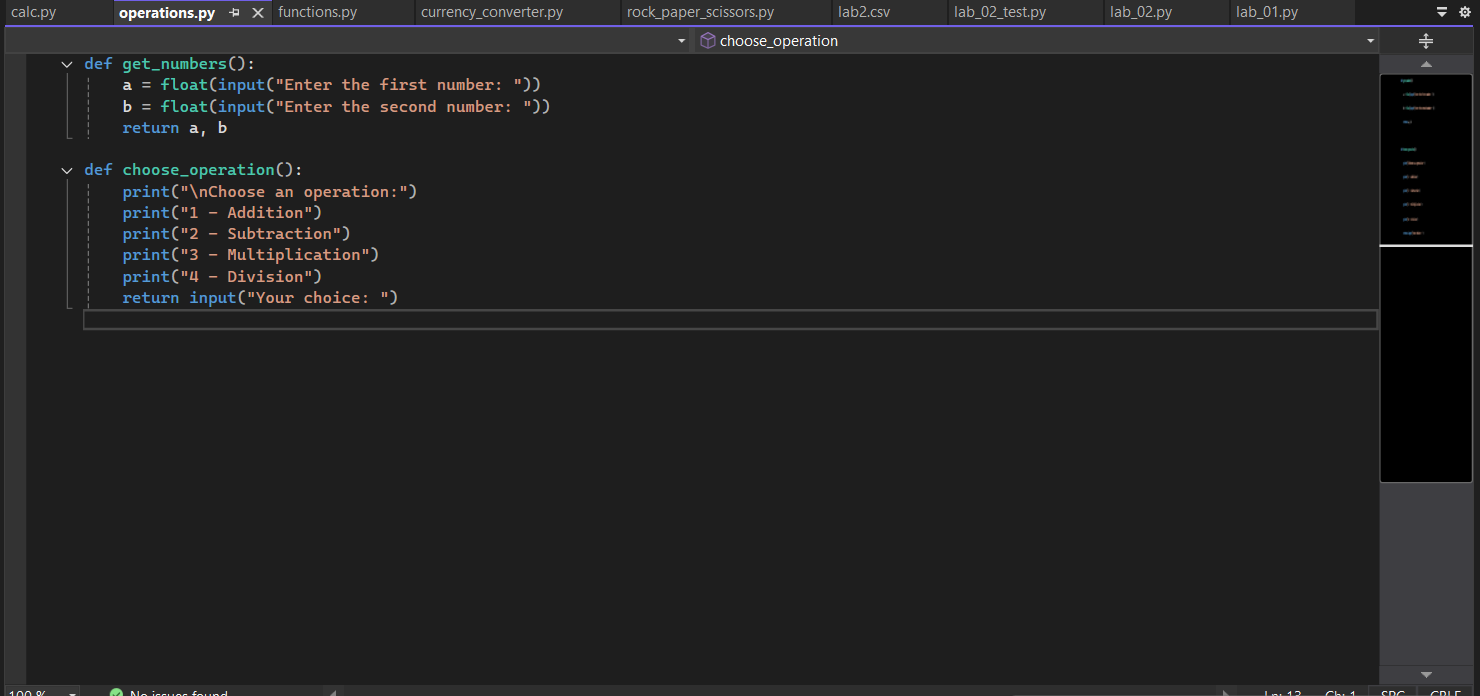
### ****Завдання 3 — Калькулятор із використанням модулів****

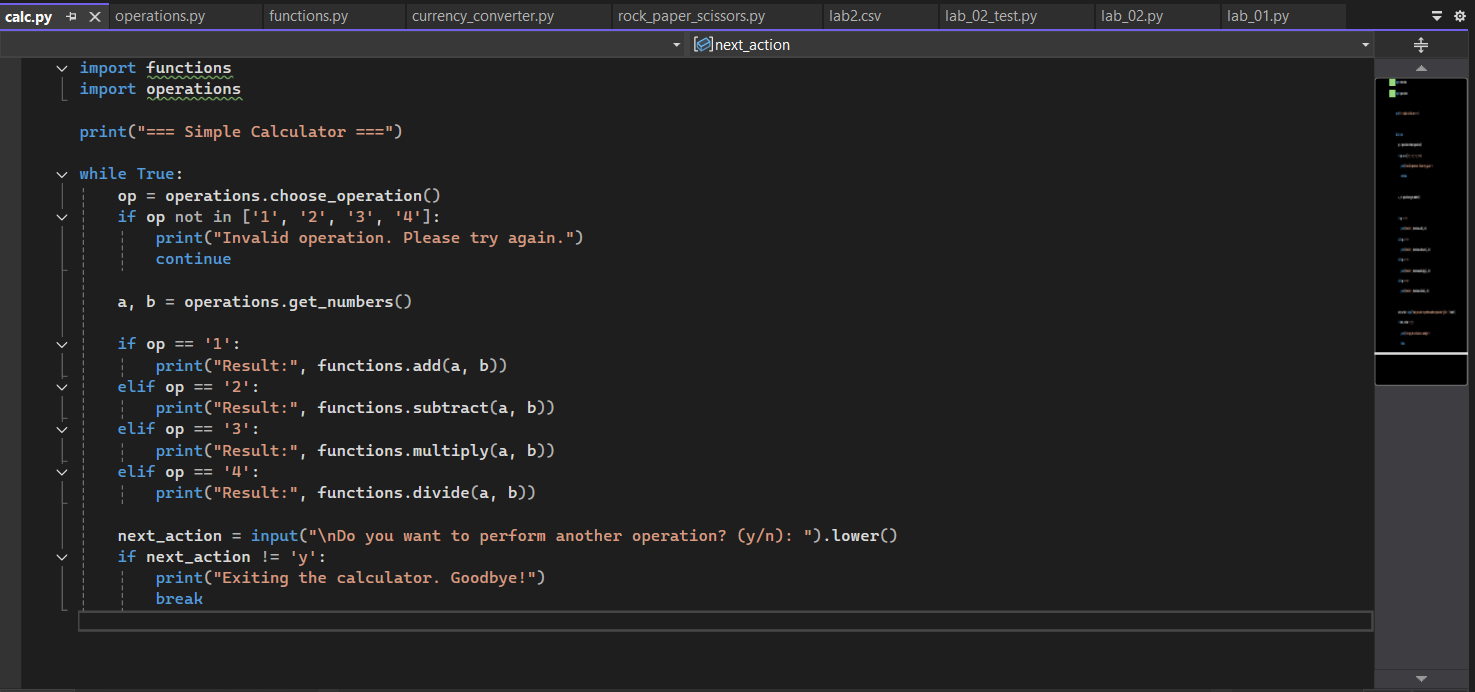
Третє завдання полягало у створенні **простого калькулятора**, розділеного на три файли:

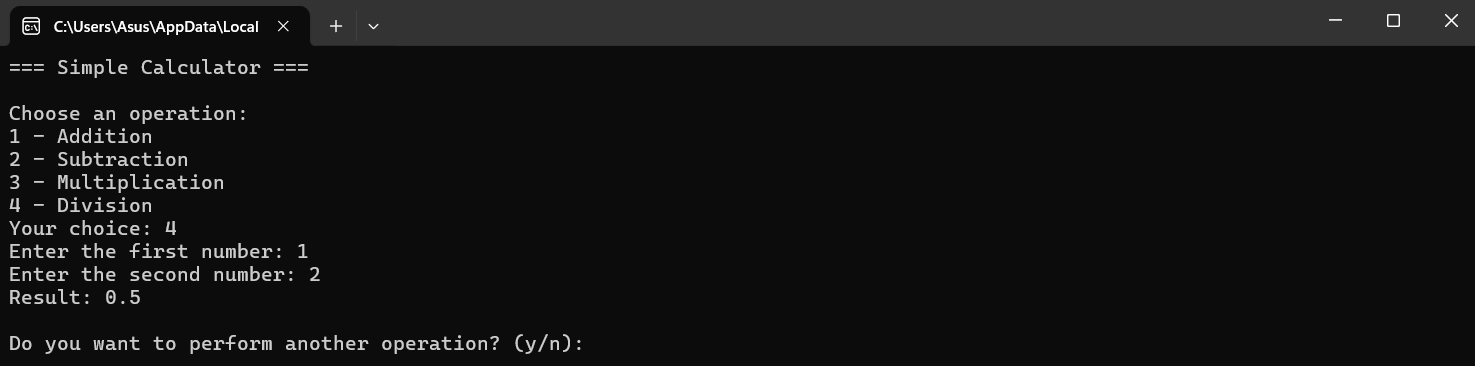
* functions.py — містить функції додавання, віднімання, множення та ділення;
* operations.py — містить функції для введення чисел та вибору операції;
* calc.py — головний файл, що імпортує два попередні модулі та реалізує логіку роботи програми.

Після запуску користувач може вибрати потрібну операцію, ввести два числа та отримати результат. Якщо користувач хоче продовжити — програма виконує наступну дію; якщо ні — завершує роботу.

Рис. 5. Скриншот написаного коду

Рис. 6. Скриншот написаного коду.

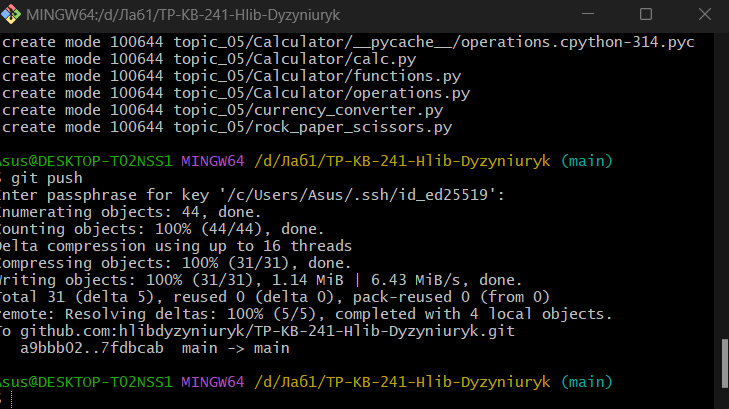
Рис. 7. Скриншот написаного коду.

Рис. 8. Скриншот виведеного результату.

### Рис. 9. Скриншот виконаних дій.

### Рис. 10. Скриншот виконаних дій.

### Рис. 11. Скриншот виконаних дій.

Рис. 12. Скриншот виконаних дій.

### ****Результати роботи****

У ході виконання завдання я:

* створив програму для гри “Камінь, ножиці, папір” з використанням модуля random;
* реалізував програму-конвертер валют з використанням API НБУ та модуля requests;
* розробив калькулятор, який працює з окремими модулями та демонструє принцип структурованості коду.

### **Висновок:**

Під час виконання лабораторної роботи я закріпив знання про використання модулів у Python, роботу з API та організацію програм у вигляді окремих файлів. Я навчився створювати програми, які взаємодіють з користувачем, обробляють введені дані та отримують інформацію з мережі.  
Також я навчився застосовувати модульний підхід для зручності розробки й повторного використання коду.

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

**Виконав: Дизинюрик Гліб Олександрович КБ-241**

## **Звіт до теми №6: Робота з файлами**

### **Мета роботи: н**авчитися використовувати модульну структуру програм у Python, підключати власні модулі, реалізовувати механізм логування дій програми, а також застосовувати стандартні бібліотечні функції для обробки структурованих даних.

## ****Завдання 1 — Калькулятор з використанням модулів та логування****

У першому завданні я реалізував простий калькулятор мовою Python із використанням модульної структури програми. Проєкт складається з трьох файлів:

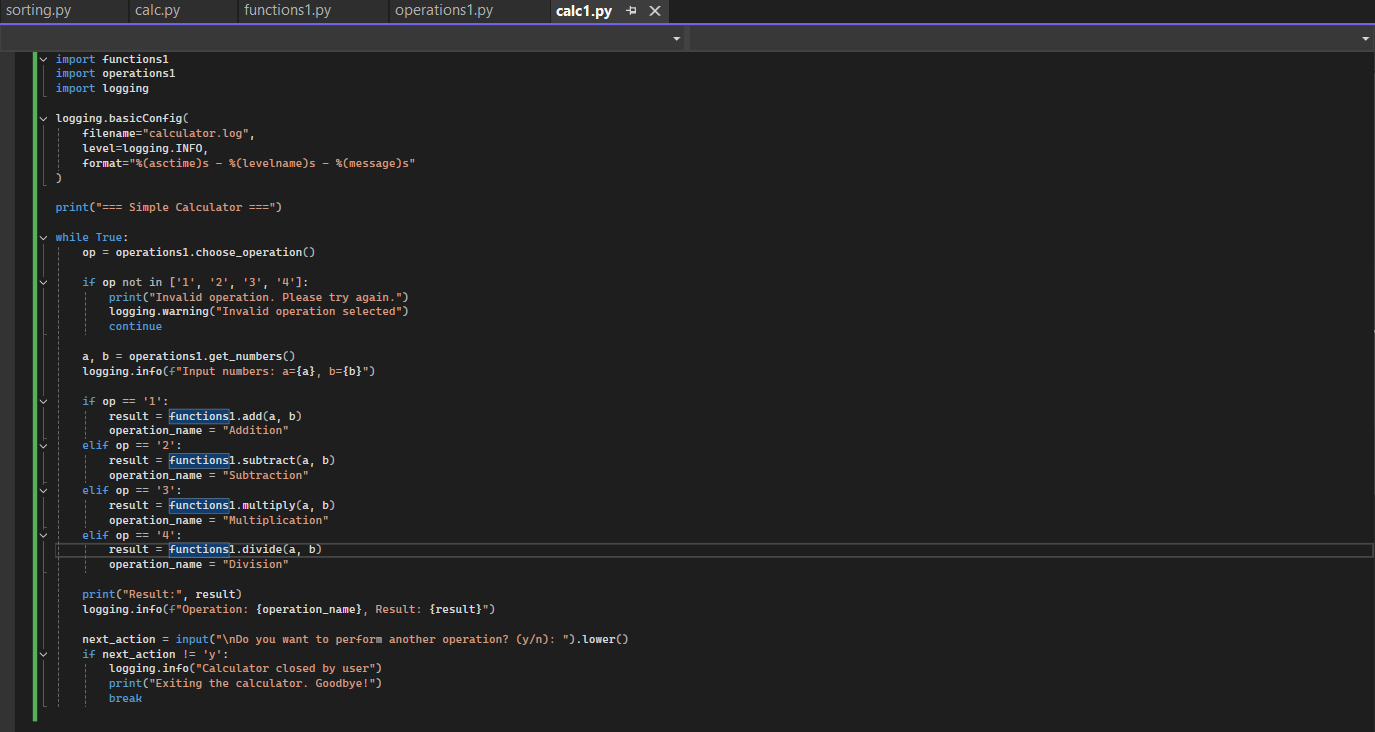
* **functions1.py** — містить функції додавання, віднімання, множення та ділення;
* **operations1.py** — відповідає за введення чисел користувачем та вибір математичної операції;
* **calc1.py** — головний файл, який імпортує інші модулі та реалізує основну логіку роботи програми.

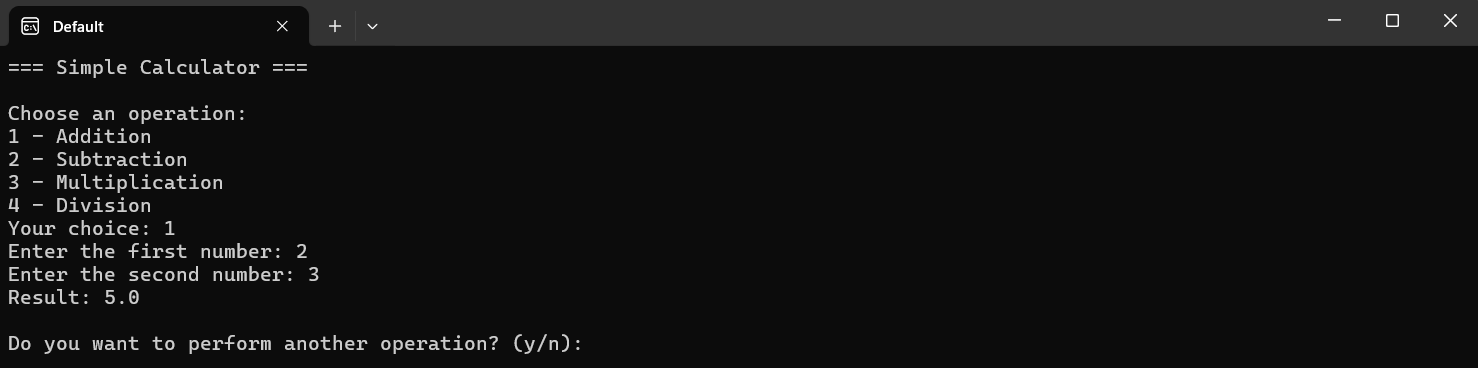
Користувач обирає операцію, вводить два числа та отримує результат обчислення. Програма працює в циклі та дозволяє виконувати декілька операцій підряд.

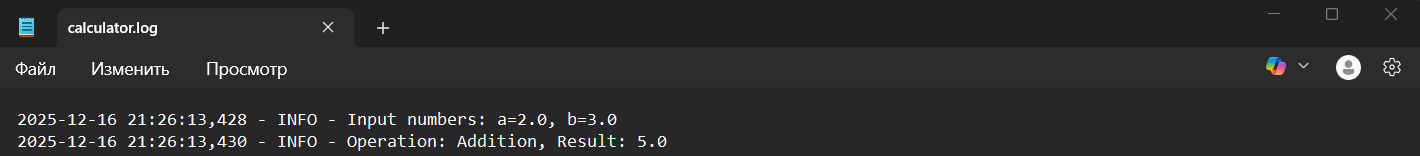
Додатково було реалізовано **механізм логування** за допомогою стандартного модуля logging. У файл calculator.log записується така інформація:

* введені користувачем числа;
* обрана математична операція;
* результат виконання операції;
* завершення роботи програми.

Це дозволяє відстежувати всі дії, виконані програмою під час роботи.

**Рис. 1.** Скриншот написаного коду калькулятора з логуванням.

**Рис. 2.** Скриншот результату роботи програми.

**Рис. 3.** Скриншот файлу журналу логування calculator.log.

## ****Завдання 2 — Сортування списку словників із використанням lambda-функцій****

У другому завданні було виконано сортування не відсортованого списку, елементами якого є словники з двома параметрами: ім’я та оцінка.

Було створено список студентів такого вигляду:

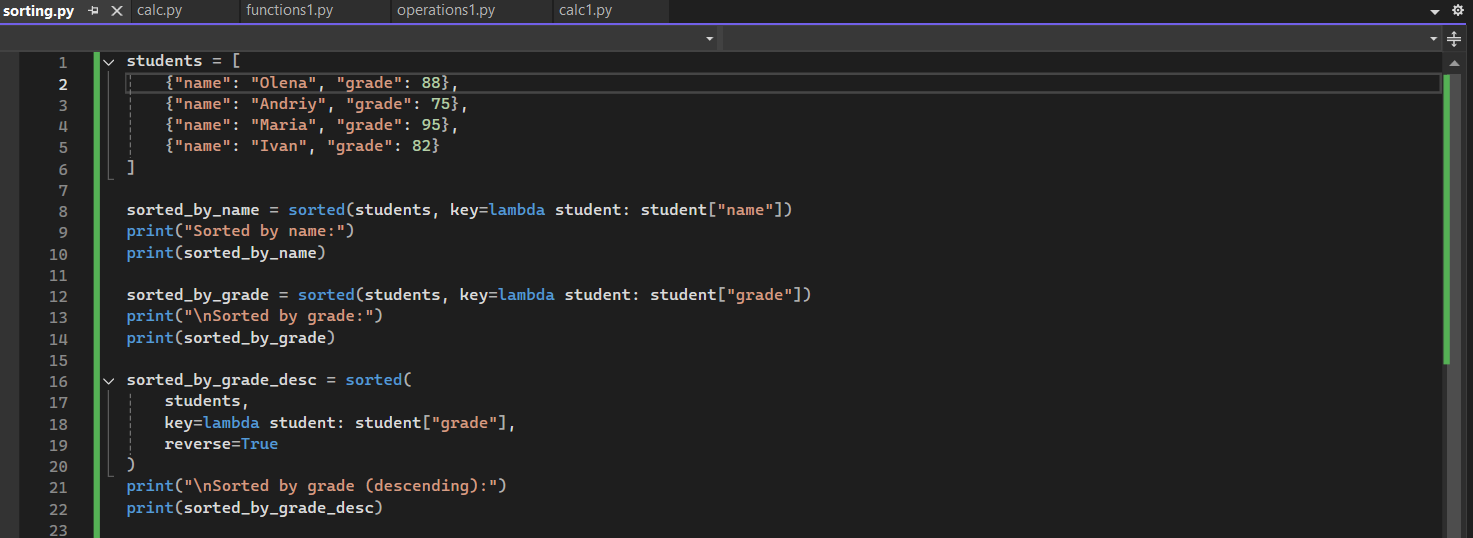
* ім’я студента;
* його оцінка.

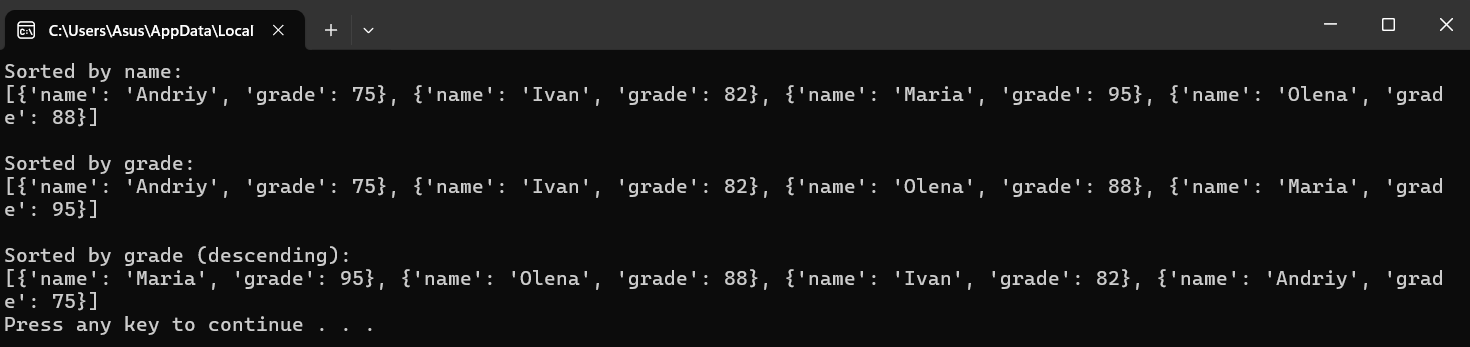
Для сортування використовувалась стандартна функція sorted(). Як другий параметр передавалась **lambda-функція**, яка повертала відповідне поле словника.

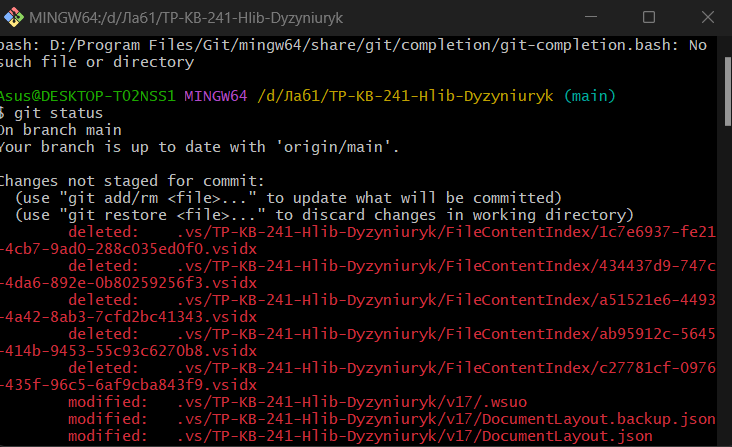
Було реалізовано:

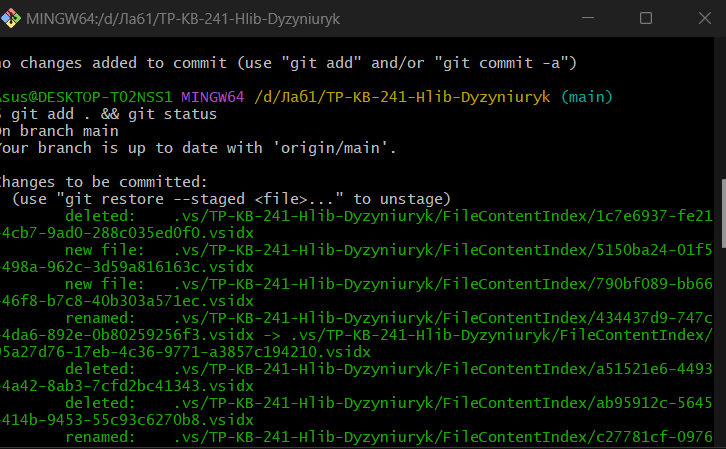
* сортування списку за ім’ям студента;
* сортування списку за оцінкою;
* сортування за оцінкою у зворотному порядку.

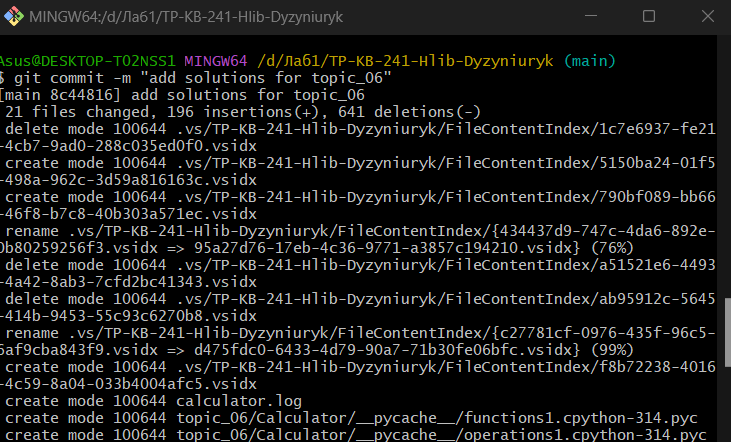
Використання lambda дозволило виконати сортування без створення окремих функцій, що зробило код компактним та зрозумілим.

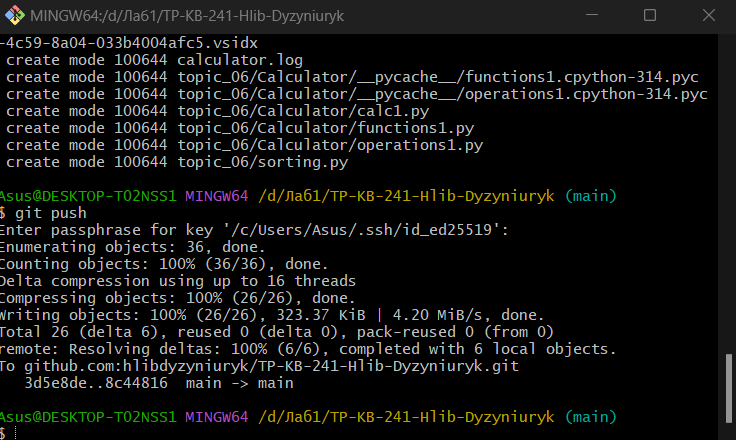
**Рис. 4.** Скриншот коду сортування списку словників.

**Рис. 5.** Скриншот результату виконання програми сортування.

Рис. 6. Скриншот виконання дій.

Рис. 7. Скриншот виконання дій.

**Рис. 8. Скриншот виконання дій.**

Рис. 9. Скриншот виконання дій.

### **Висновок:**

Під час виконання лабораторної роботи я навчився структурувати Python-програми з використанням модулів, реалізовувати логування дій програми, а також використовувати стандартні функції мови Python разом із lambda-функціями для обробки та сортування даних.

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

**Виконав: Дизинюрик Гліб Олександрович КБ-241**

# ****Звіт до Теми №7: Об’єктно-орієнтоване програмування****

## **Мета роботи:** ознайомитимся з принципами об’єктно-орієнтованого програмування в мові Python, навчитися створювати класи та об’єкти, використовувати стандартні методи класів, виконувати сортування об’єктів за допомогою sorted() і lambda, а також реалізовувати програму з використанням ООП та модульного підходу.

## ****Завдання 1 — Ознайомлення з класами в Python****

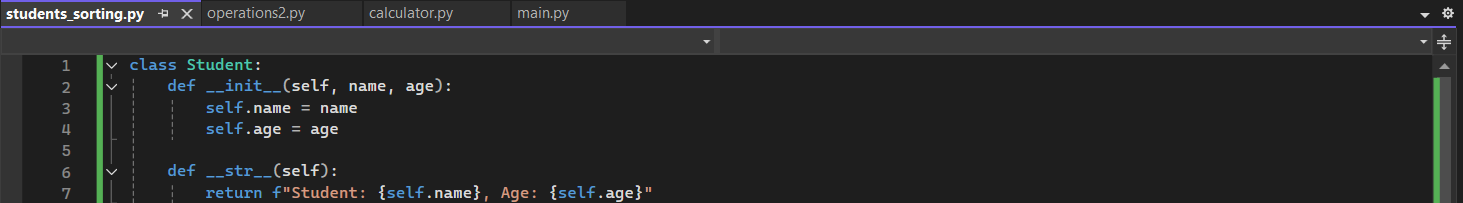
У процесі виконання роботи було опрацьовано документацію Python, що описує використання класів. Клас у Python є шаблоном для створення об’єктів і може містити атрибути та методи. Об’єкти класу створюються на основі описаного класу та використовуються для зберігання і обробки даних.

## ****Завдання 2 — Стандартні методи класу**** \_\_init\_\_ ****та**** \_\_str\_\_

Метод \_\_init\_\_(self) є конструктором класу та викликається автоматично під час створення об’єкта. Він використовується для ініціалізації атрибутів об’єкта.

Метод \_\_str\_\_(self) визначає текстове представлення об’єкта та використовується під час виведення об’єкта на екран за допомогою функції print().

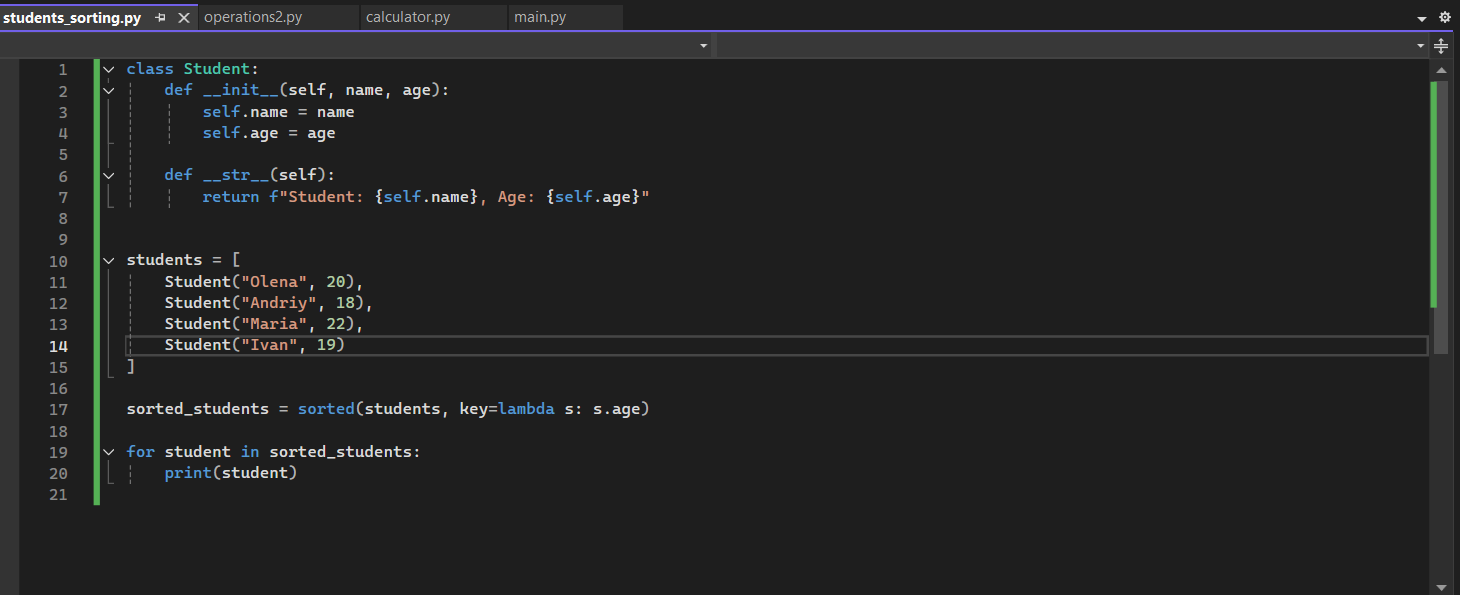
Приклад використання цих методів реалізовано у класах Student та Calculator.

**Рис. 1.** Приклад використання методів \_\_init\_\_ та \_\_str\_\_.

## ****Завдання 3 — Клас Student та сортування об’єктів****

У цьому завданні було створено клас Student з атрибутами name та age. Далі створено список, елементами якого є об’єкти класу Student.

Для сортування списку використано стандартну функцію sorted(). Як ключ сортування застосовано lambda-функцію, яка повертає значення віку студента. Після сортування елементи списку виводяться на екран у впорядкованому вигляді.

**Рис. 2.** Скриншот коду класу Student та сортування списку.

**Рис. 3.** Скриншот результату виконання програми.

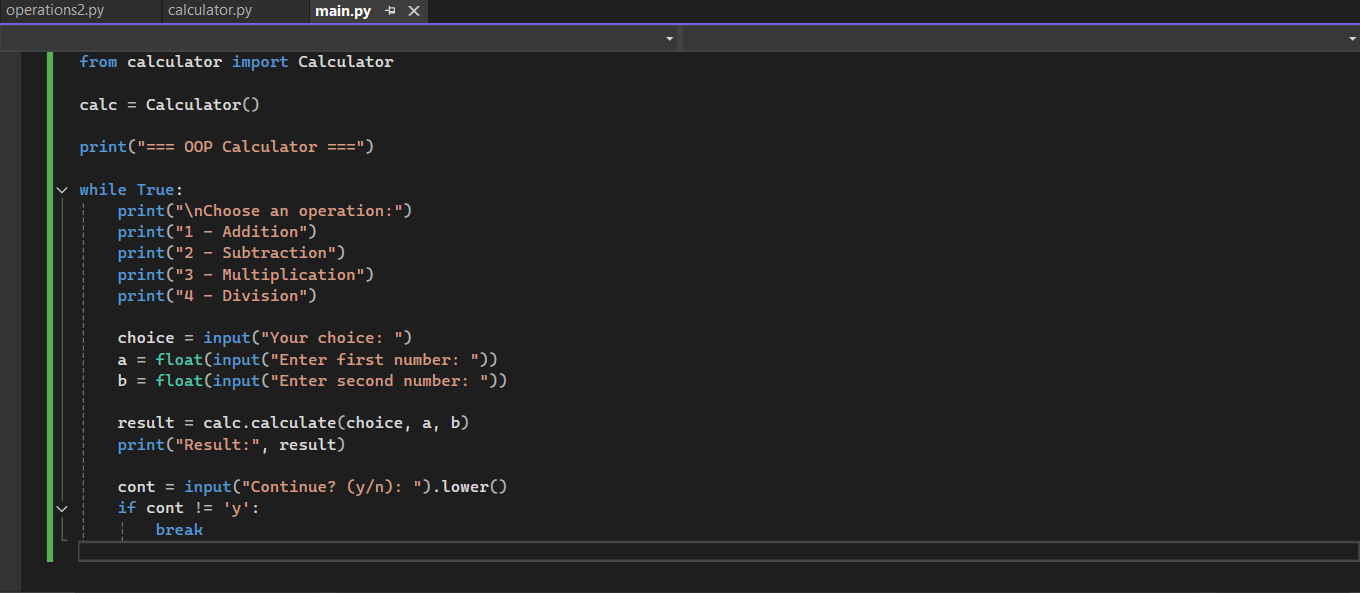
## ****Завдання 4 — Калькулятор з використанням ООП****

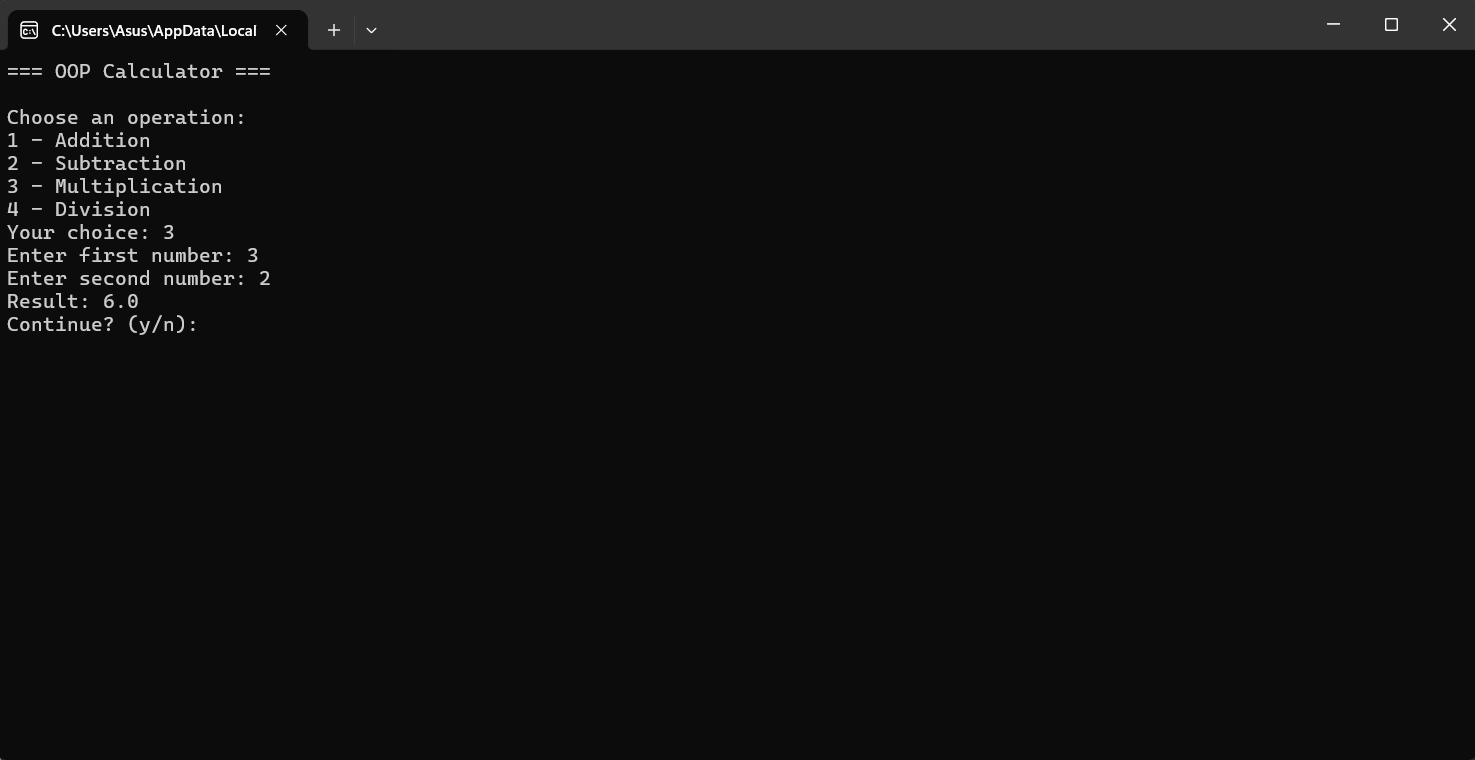
У цьому завданні програму «Калькулятор» було переписано з використанням принципів об’єктно-орієнтованого програмування та модульного підходу.

Проєкт складається з трьох файлів:

* **operations.py** — містить клас Operations з методами додавання, віднімання, множення та ділення;
* **calculator.py** — містить клас Calculator, який використовує об’єкт класу Operations;
* **main.py** — головний файл програми, який реалізує взаємодію з користувачем.

Користувач обирає математичну операцію, вводить два числа та отримує результат обчислення. Програма працює в циклі та дозволяє виконувати декілька операцій поспіль.

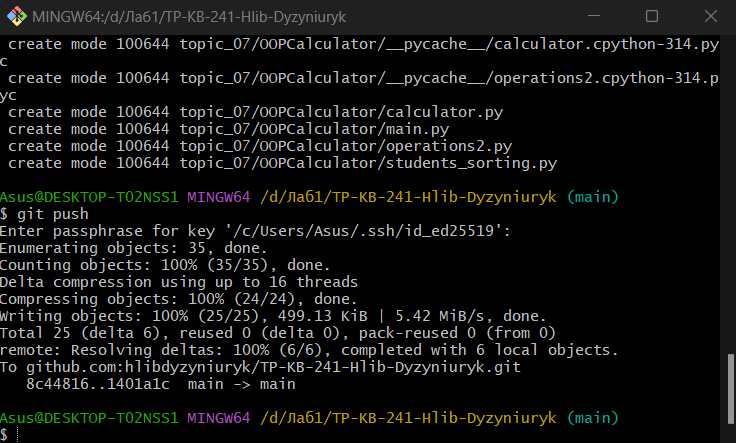
**Рис. 4.** Скриншот коду калькулятора з використанням ООП.

**Рис. 5.** Скриншот результату роботи програми.

## **Рис. 6. Скриншот виконання дій.**

## **Рис. 7. Скриншот виконання дій.**

## **Рис. 8. Скриншот виконання дій.**

**Рис. 9. Скриншот виконання дій.**

## **Висновок**

Під час виконання лабораторної роботи я ознайомився з основами об’єктно-орієнтованого програмування в Python, навчився створювати класи та об’єкти, використовувати стандартні методи \_\_init\_\_ і \_\_str\_\_, виконувати сортування об’єктів за допомогою sorted() та lambda, а також реалізовувати програму з використанням ООП і модульної структури.