*Assignment*

**Lab 1: Environment Setup – GitHub Copilot and VS Code Integration**

**Lab Objectives:**

* To install and configure GitHub Copilot in Visual Studio Code.
* To explore AI-assisted code generation using GitHub Copilot.
* To analyze the accuracy and effectiveness of Copilot's code suggestions.
* To understand prompt-based programming using comments and code context

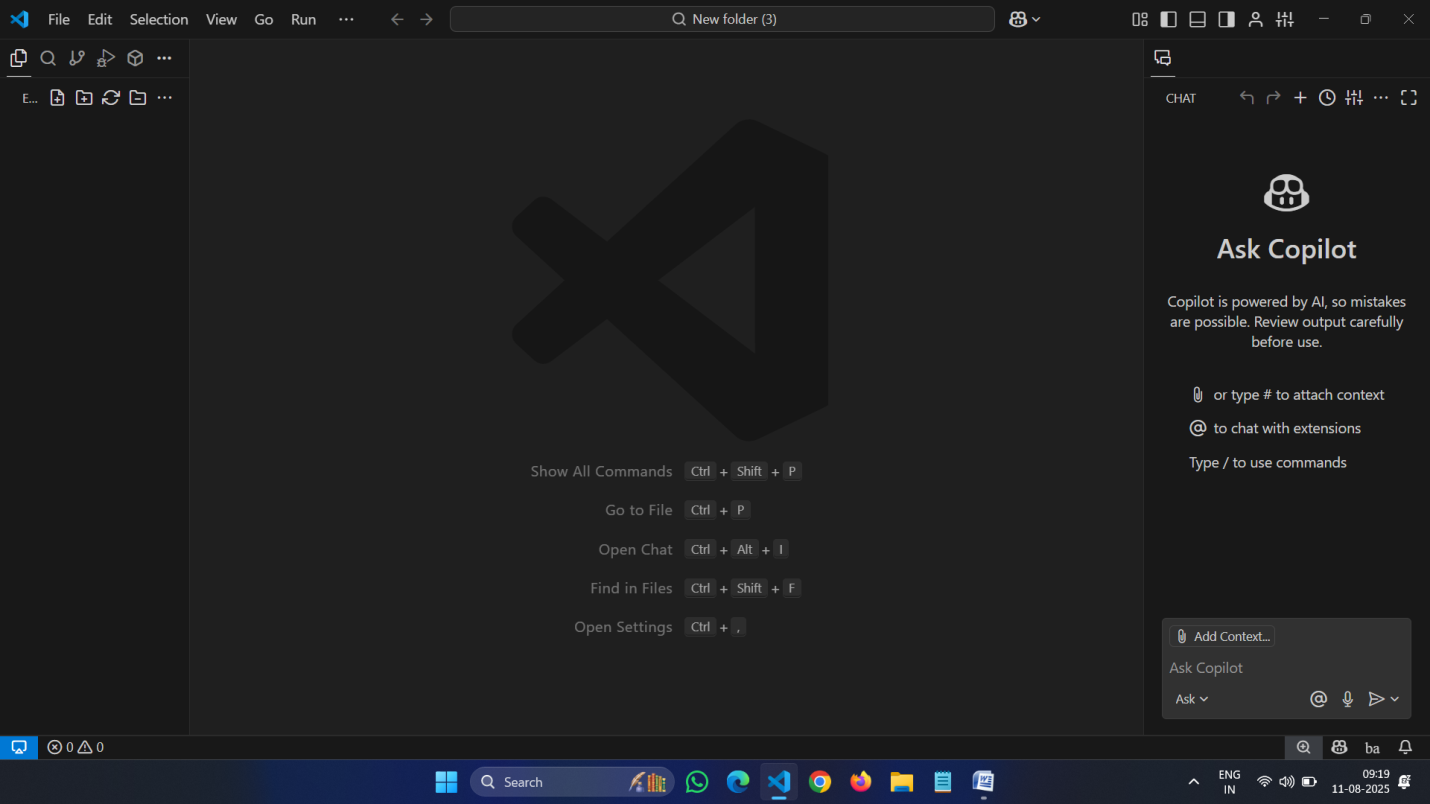
**Lab Outcomes (LOs):**

After completing this lab, students will be able to:

* Set up GitHub Copilot in VS Code successfully.
* Use inline comments and context to generate code with Copilot.
* Evaluate AI-generated code for correctness and readability.
* Compare code suggestions based on different prompts and programming styles.

**Task 0 :**

* Install and configure of GitHub Copilot Take Screenshot of each one.
* Expected output :

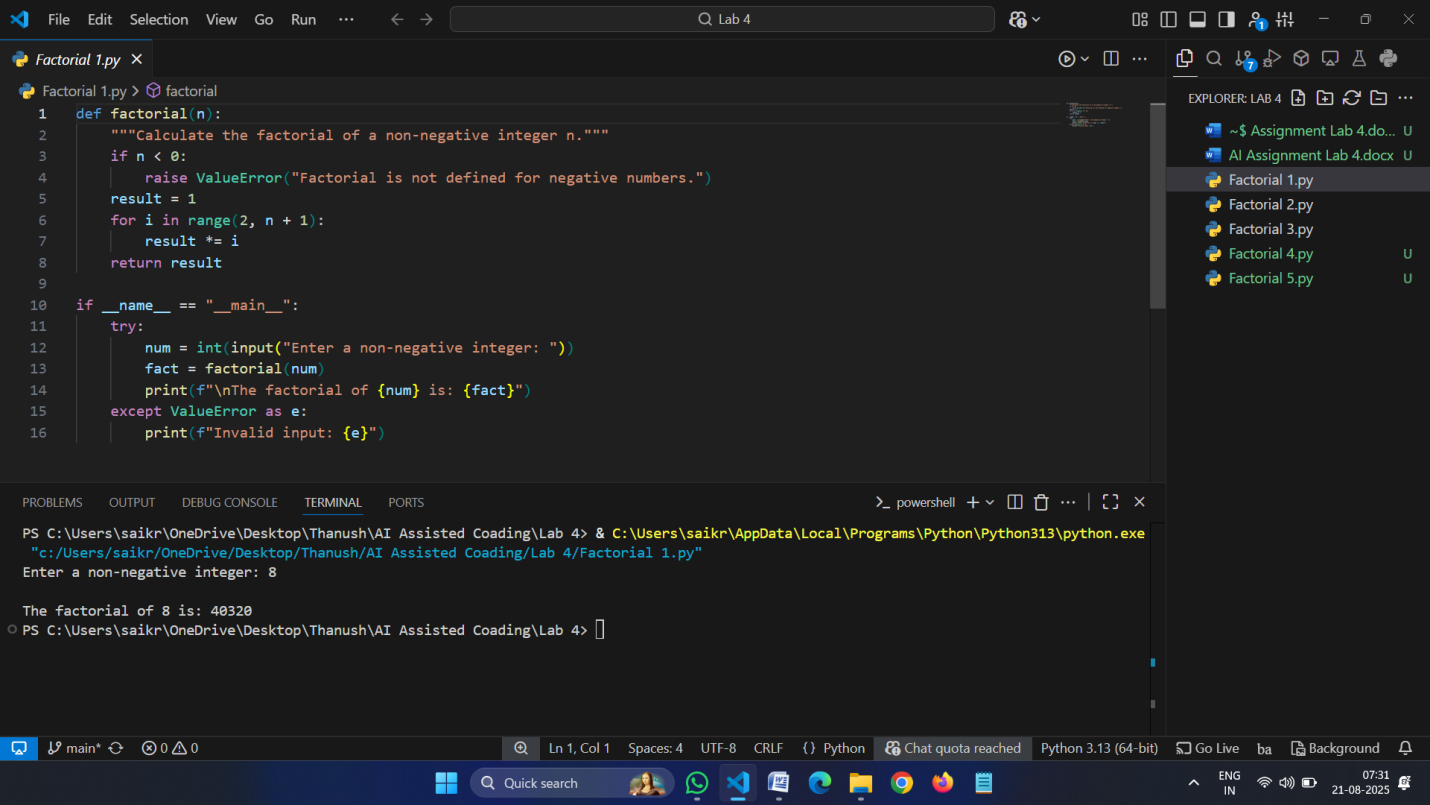


**Task 1 :** Factorial without function

* Description

Use GitHub copilot to generate a python program that calculates the factorial of number without defining any function using loops directly in the main code.

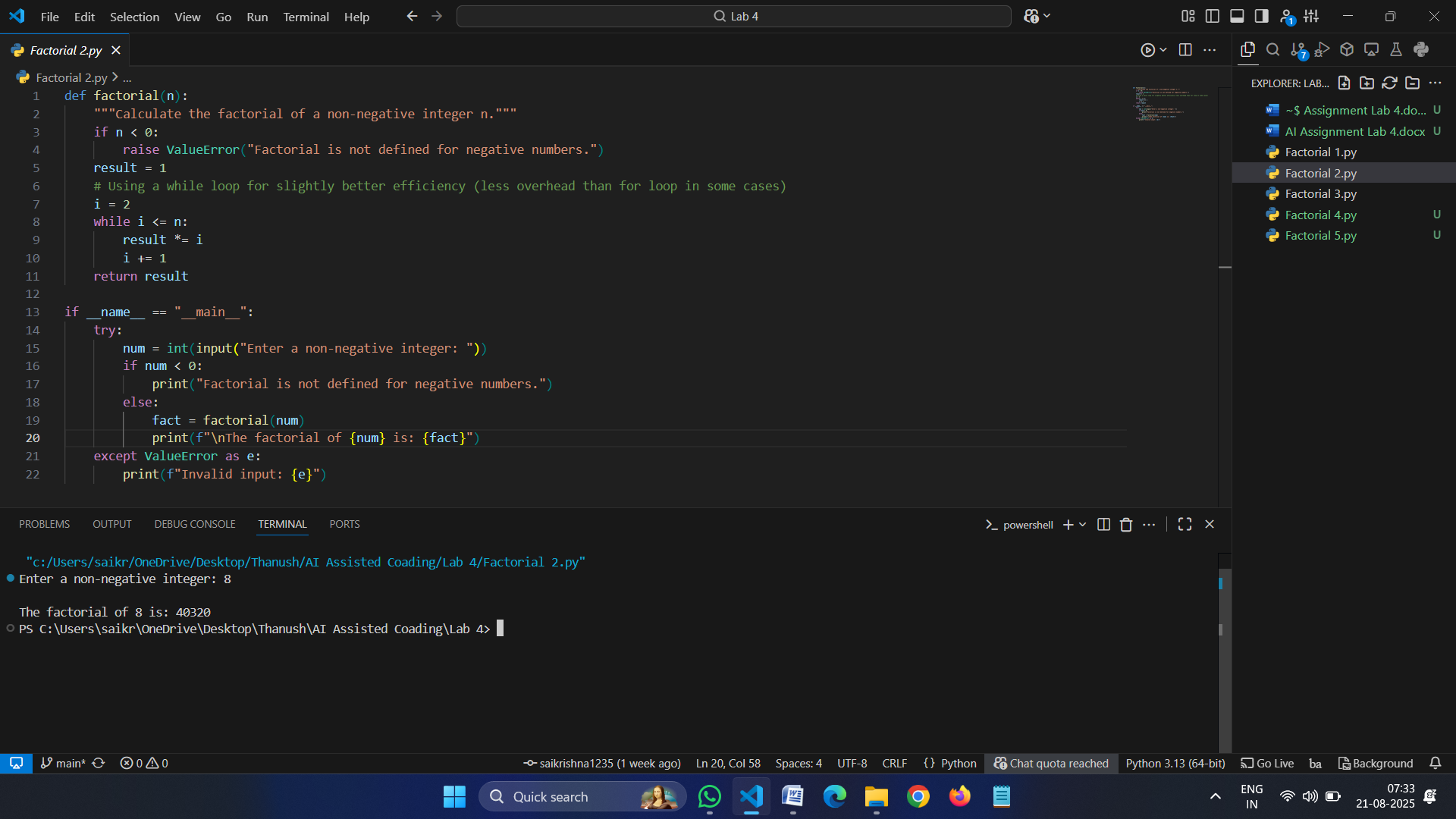
* Expected output :

****

**Task 2 :**Improving Efficiency

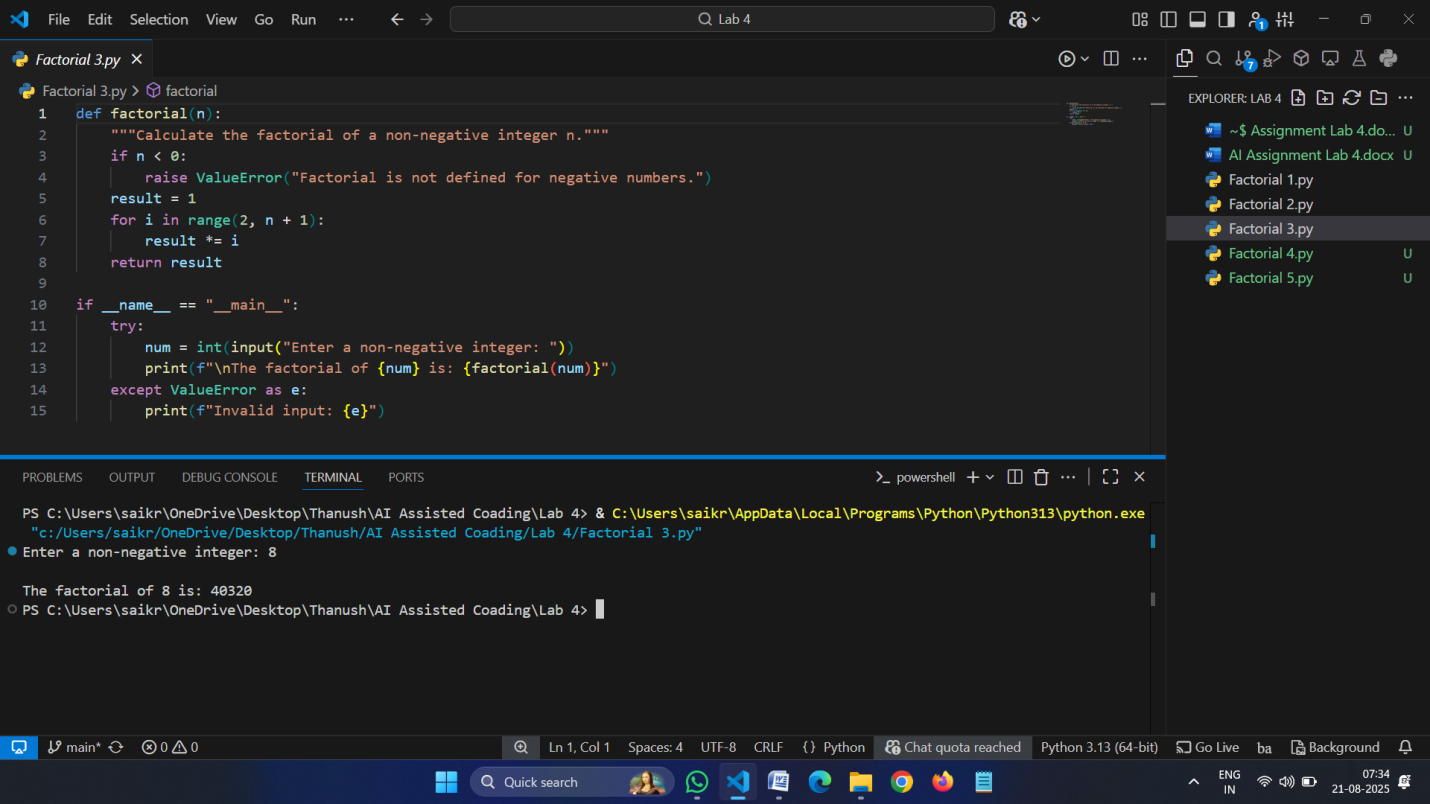
* Description
* Examine the Copilot generated code .

The code is improved by removing the function definition and calculating the factorial directly in the main block using a loop. This reduces function call overhead and simplifies the logic. The while loop starts from 2 and multiplies up to the input number, which is efficient for factorial calculation. Error handling for invalid input is also included.



**Task 3 :Factorial with functions**

* Use GitHub copilot to generate a python program that calculates the factorial of number Using user defining any function
* Expected output :
* Defined a function factorial(n) to calculate the factorial of a non-negative integer.
* Checked if the input n is negative; if so, raised a ValueError.
* Initialized result to 1.
* Used a for loop to multiply result by each integer from 2 to n.
* Returned the computed factorial value.
* In the main block, prompted the user to enter a non-negative integer.
* Called the factorial function with the user input and printed the result.
* Handled invalid input using a try-except block.

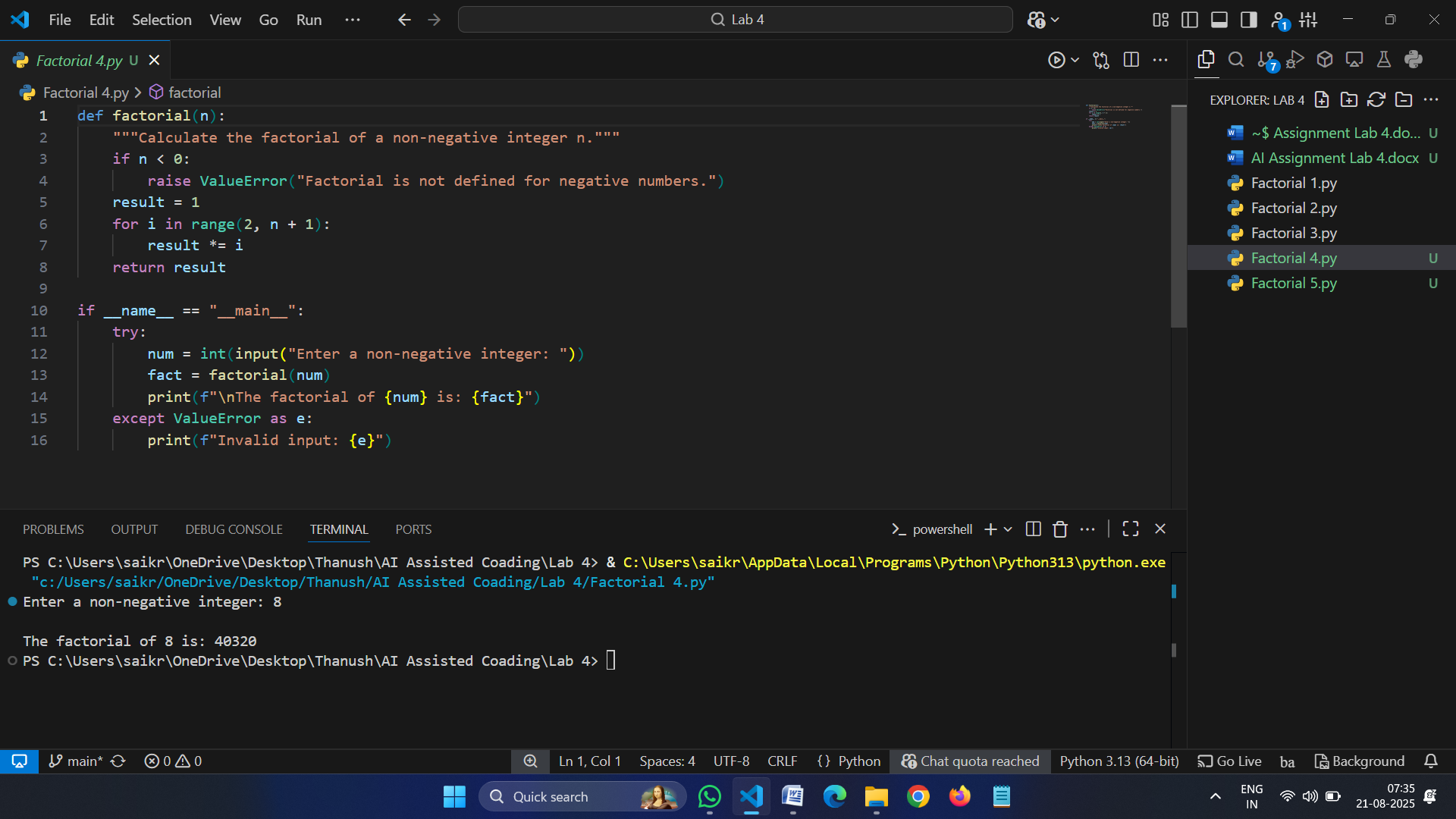


**Task 4 :** Comparative Analysis – With vs Without Functions

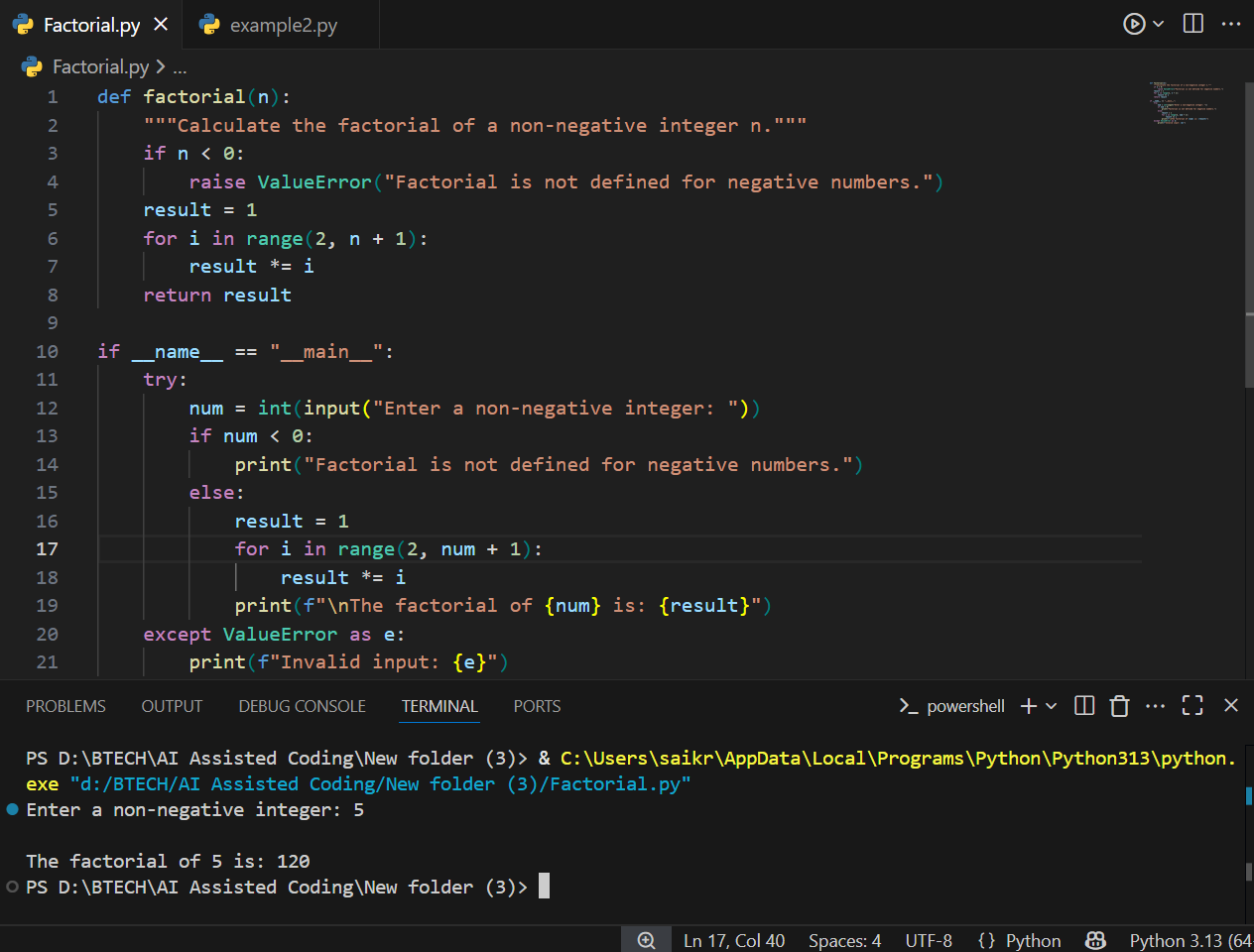
* Description

Differentiate between the copilot generated factorial program with functions and without functions in the terms of logic ,reusability,and execution.

* Expected Output :
  + With functions



* Without functions



**Logic**

* **With Function:**  
  The logic is encapsulated in a user-defined function ([factorial(n)](vscode-file://vscode-app/c:/Users/saikr/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)), which takes an argument and returns the factorial. The main code calls this function.
* **Without Function:**  
  The logic is written directly in the main code block. The loop and calculation are performed inline, without any encapsulation.

**2. Reusability**

* **With Function:**  
  Highly reusable. The [factorial](vscode-file://vscode-app/c:/Users/saikr/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) function can be called multiple times with different arguments, in other parts of the program or from other modules.
* **Without Function:**  
  Not reusable. The code can only be executed as written, and cannot be called with different inputs without rewriting or copying the logic.

**3. Execution**

* **With Function:**  
  Slight overhead due to function calls, but allows for cleaner code, easier testing, and better error handling.
* **Without Function:**  
  Executes sequentially in the main block. Slightly faster for very simple scripts due to no function call overhead, but less organized and harder to maintain for larger programs.

**Summary:**  
Using functions improves code organization, reusability, and maintainability, while direct logic in the main block is simpler but less flexible and harder to reuse.

**Task 5 :**Iterative vs Recursive Factorial

* Description :

Prompt GitHub Copilot to generate both iterative and recursive versions of the factorial function

* Expected Output :

