

Name:janagani varshitha

Enroll no :2403A54085

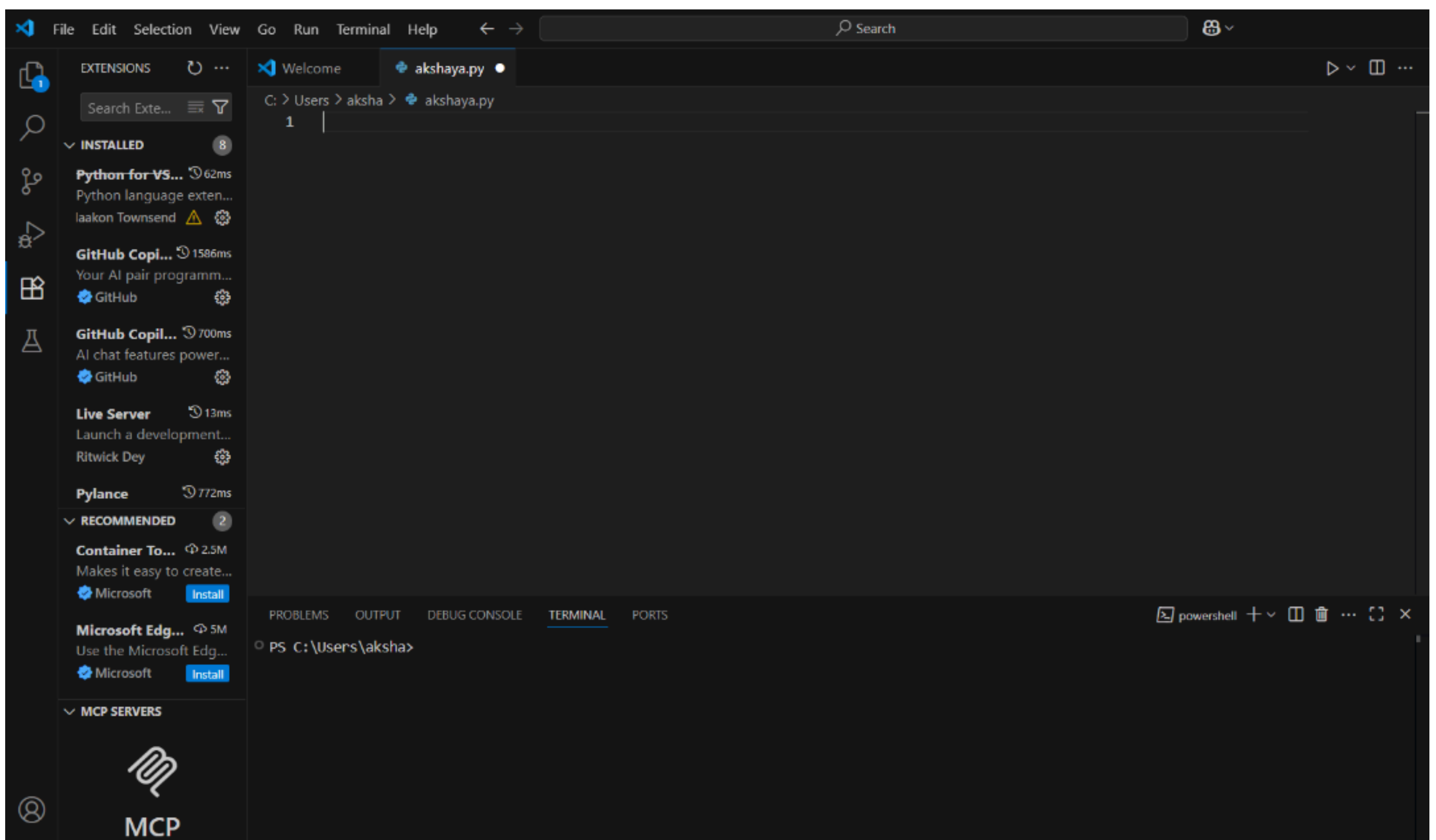
Batch : (DS)batch-3

Task 0

- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

Expected Output

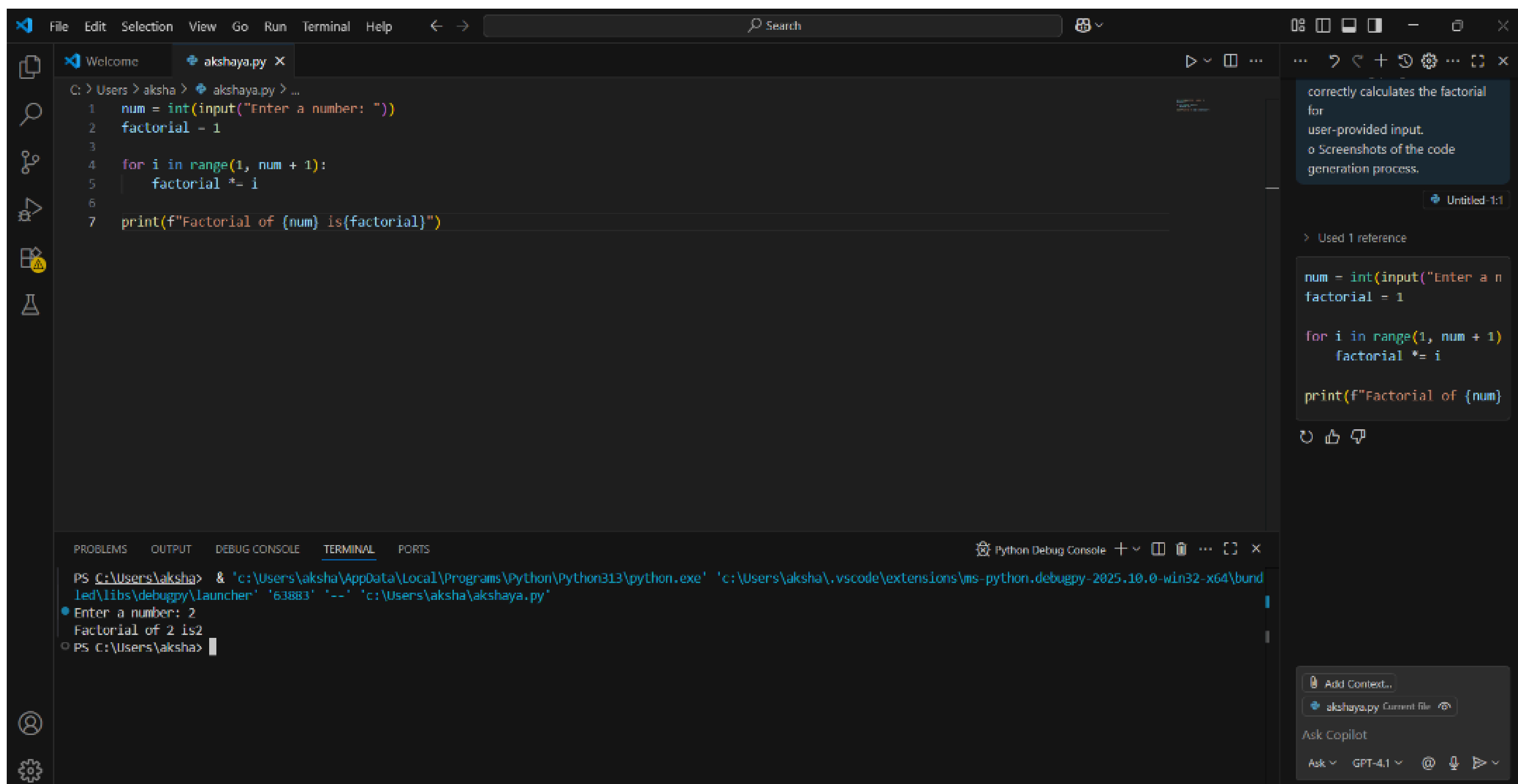
- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.



Task 1: Factorial without Functions

- Description:
Use GitHub Copilot to generate a Python program that calculates the factorial of a number without defining any functions (using loops directly in the main code).
- Expected Output:
 - A working program that correctly calculates the factorial for user-provided input.
 - Screenshots of the code generation process.

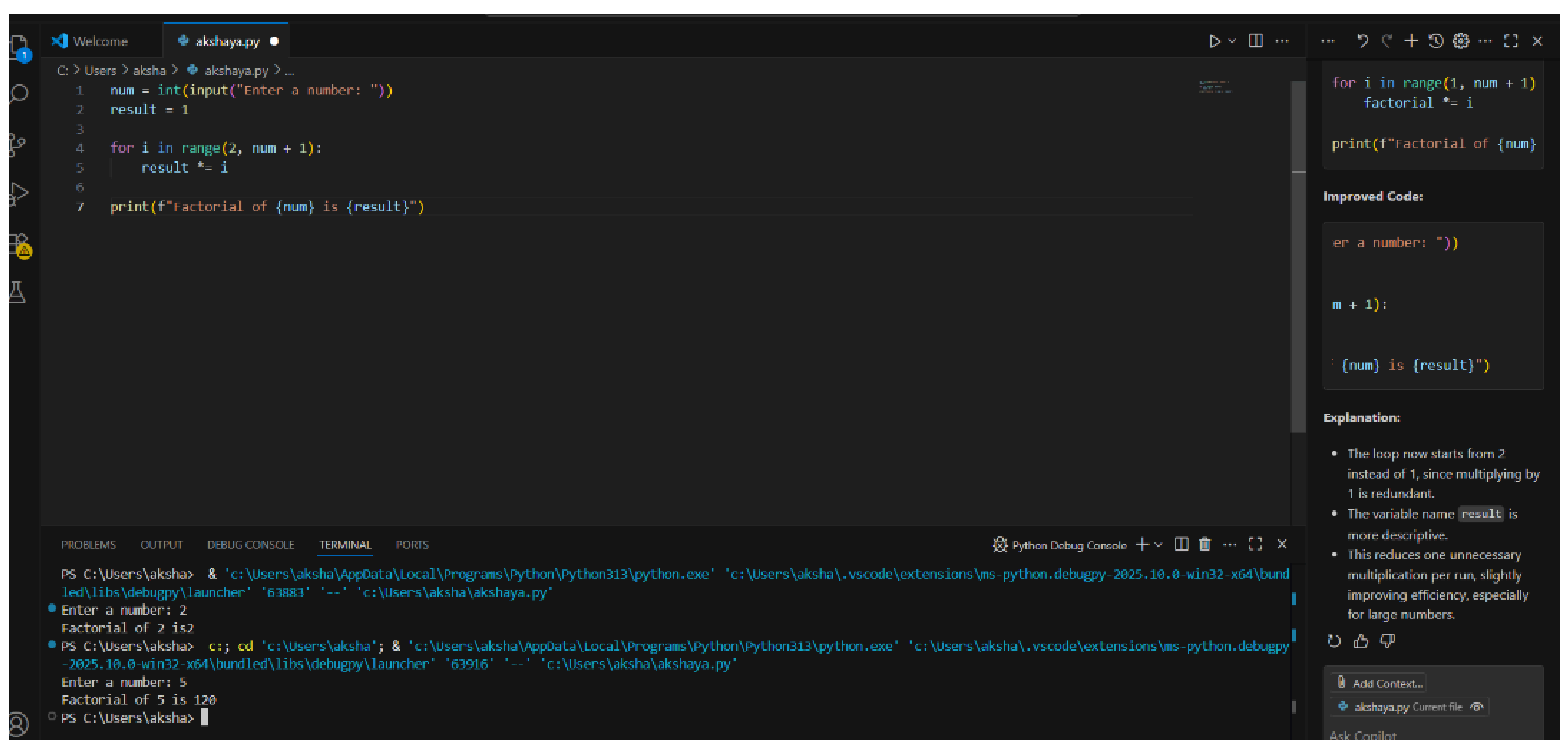
prompt : generate a python program which takes a number from the user and give the factorial as the output ,without using any functions



Task 2: Improving Efficiency

- Description:
Examine the Copilot-generated code from Task 1 and demonstrate how its efficiency can be improved (e.g., removing unnecessary variables, optimizing loops).
- Expected Output:
 - Original and improved versions of the code.
 - Explanation of how the improvements enhance performance.

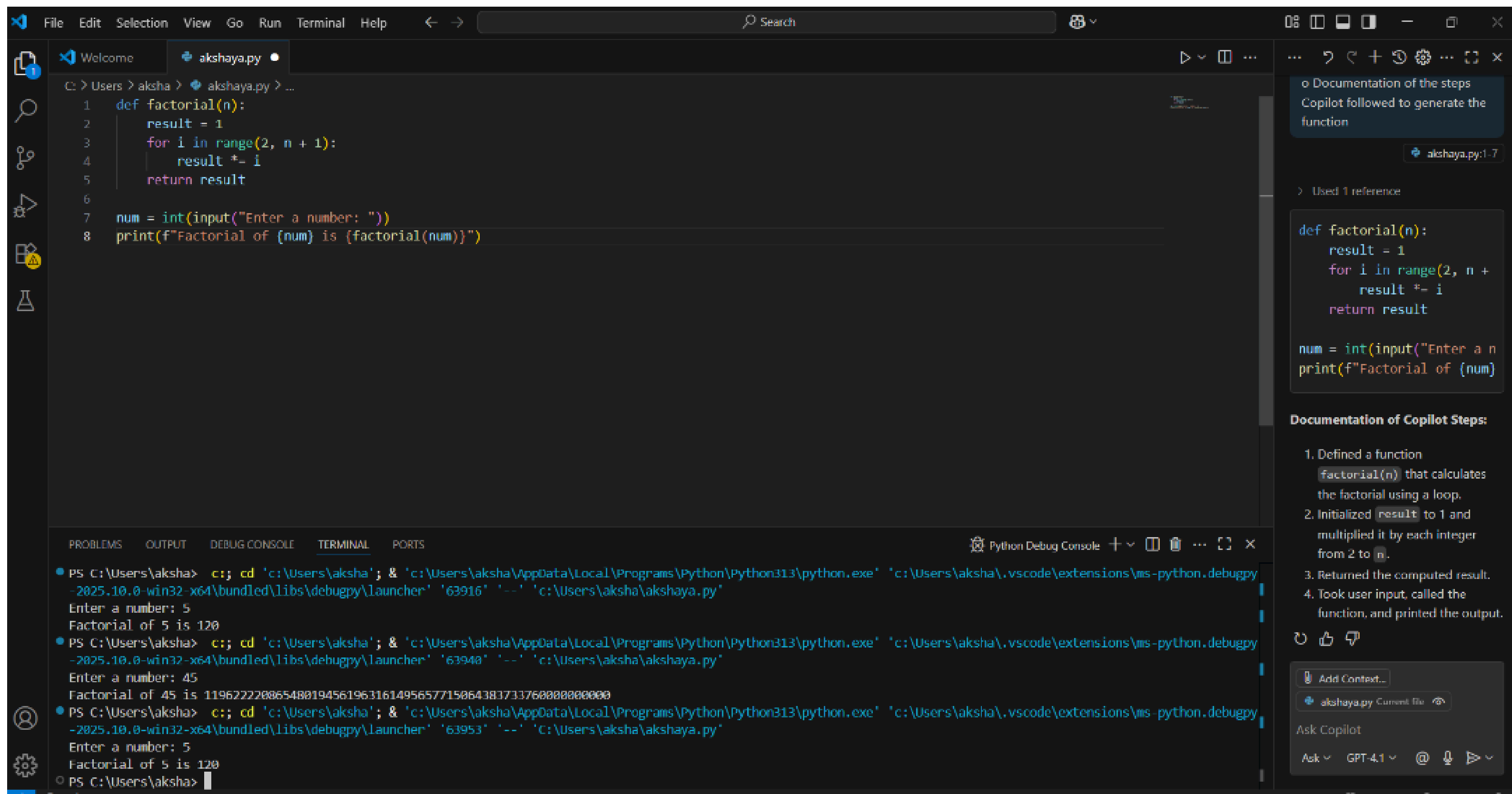
prompt : improve the efficiency from the previous code by removing unnecessary variables,optimizing loops etc...



Task 3:Factorial with Functions

- Description:
Use GitHub Copilot to generate a Python program that calculates the factorial of a number using a user-defined function.
- Expected Output:
 - Correctly working factorial function with sample outputs.
 - Documentation of the steps Copilot followed to generate the function.

Prompt :generate a python program which take the input from the user and give factorial of it by taking an user defined function

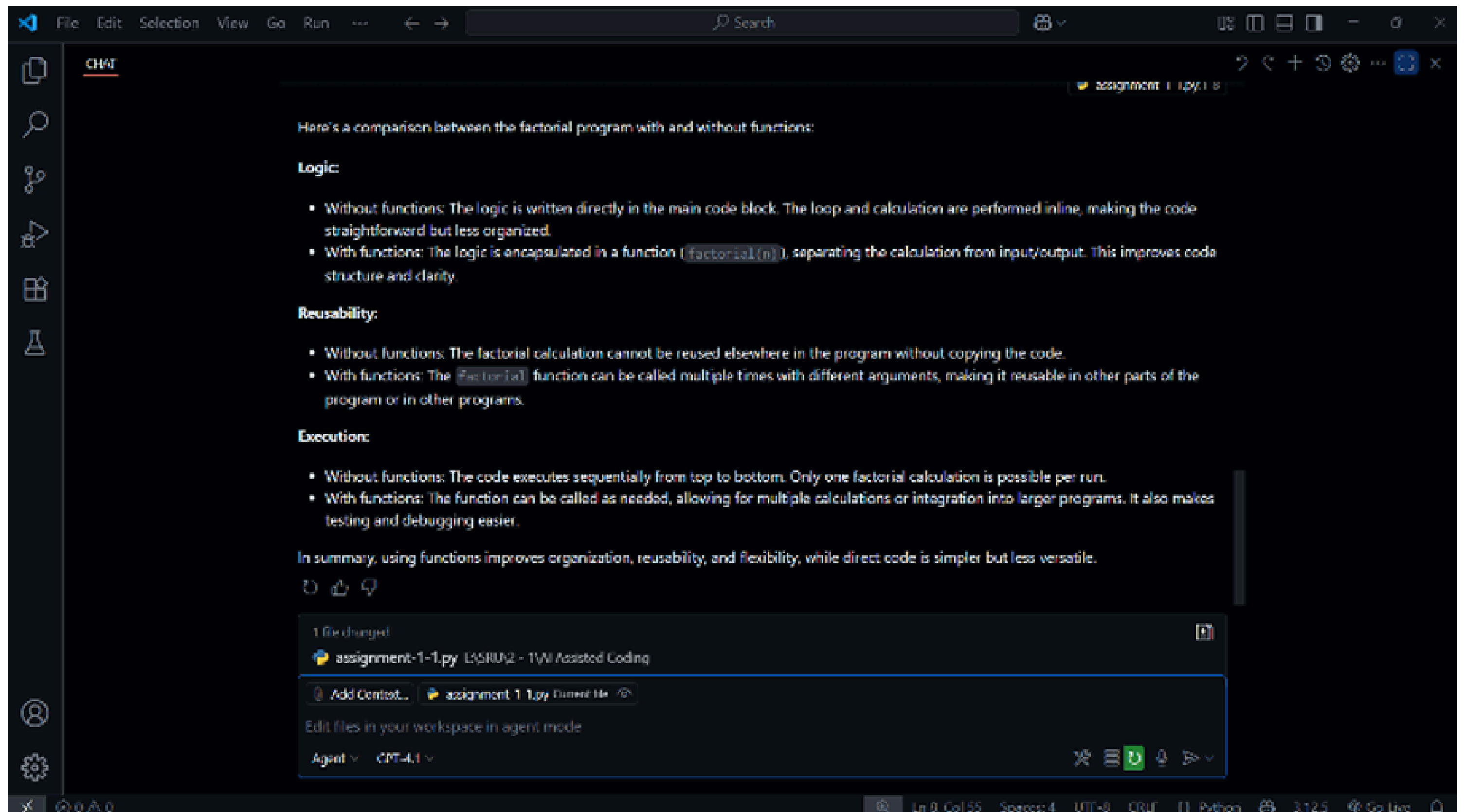


Task 4: Comparative Analysis – With vs Without Functions

- Description:
Differentiate between the Copilot-generated factorial program with functions and without functions in terms of logic, reusability, and execution.
- Expected Output:
A comparison table or short report explaining the differences

Prompt:

- Differentiate between program with functions and without functions in terms of logic, reusability, and execution.



Task 5: Iterative vs Recursive Factorial

- Description:
Prompt GitHub Copilot to generate both iterative and recursive versions of the factorial function.
- Expected Output:
 - Two correct implementations.
 - A documented comparison of logic, performance, and execution flow between iterative and recursive approaches.
- Prompt : generate both iterative and recursive versions of the factorial function

The screenshot displays the Visual Studio Code (VS Code) interface. The main editor window shows a file named `akshaya.py` with the following Python code:

```

1  # Iterative factorial function
2  def factorial_iterative(n):
3      result = 1
4      for i in range(2, n + 1):
5          result *= i
6      return result
7
8  # Recursive factorial function
9  def factorial_recursive(n):
10     if n == 0 or n == 1:
11         return 1
12     else:
13         return n * factorial_recursive(n - 1)
14
15 num = int(input("Enter a number: "))
16 print(f"Iterative: Factorial of {num} is {factorial_iterative(num)}")
17 print(f"Recursive: Factorial of {num} is {factorial_recursive(num)}")

```

The bottom panel shows the **TERMINAL** output, which includes the command prompt execution of the script for inputs 45, 5, and 4. The output correctly calculates the factorials using both iterative and recursive methods.

On the right side, a Copilot-generated document is visible, titled "akshaya.py:1". It contains a comparison of the two implementations:

```

# Iterative factorial func
def factorial_iterative(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

# Recursive factorial func
def factorial_recursive(n):
    if n == 0 or n == 1:

```

A "Snipping Tool" window is also open in the bottom right corner, displaying the message: "Screenshot copied to clipboard. Automatically saved to screenshots folder. Mark-up and share".