Lab Assignment 1.2 – AI Assisted Coding

Harika Jupaka

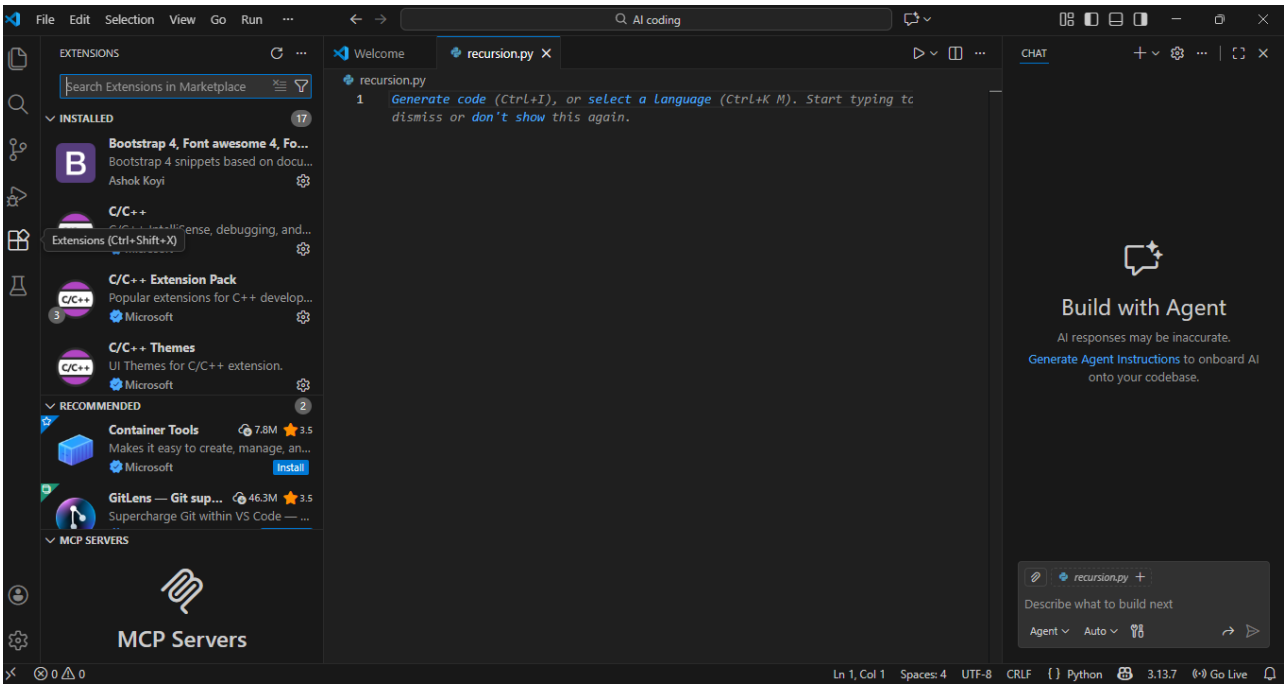
2403A51L31

B:52

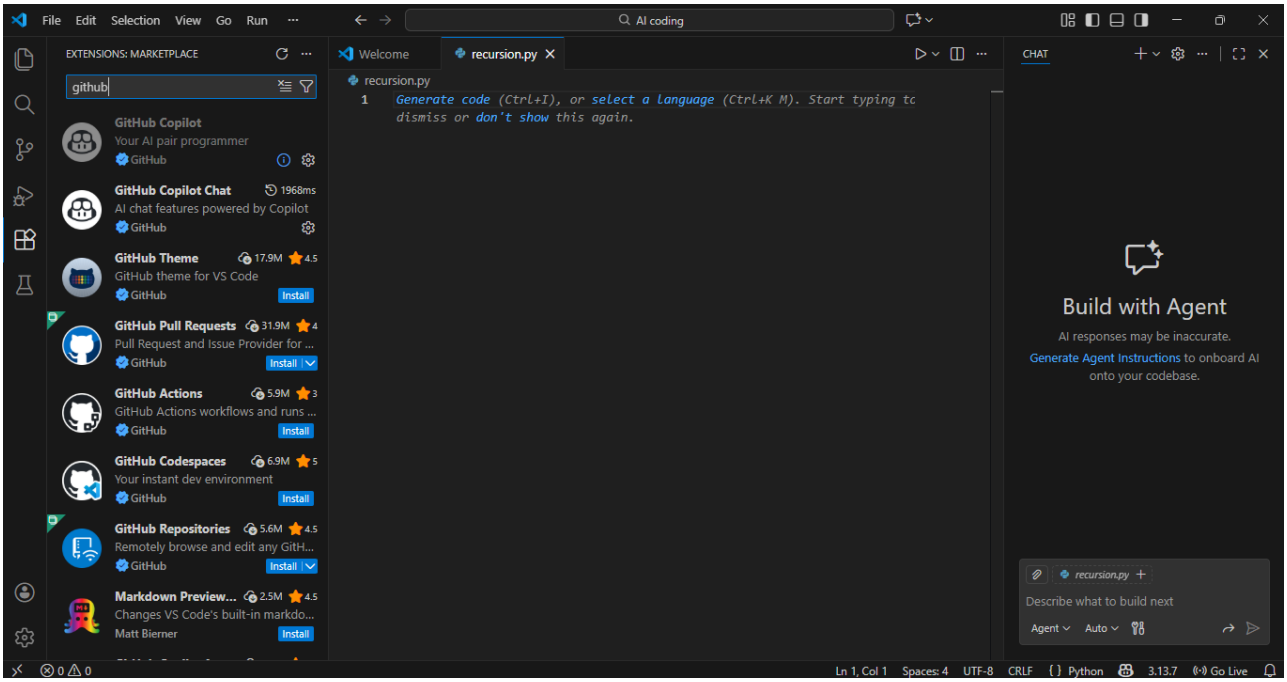
Task 0: GitHub Copilot Installation & Configuration

Steps Followed:

1. Installed **Visual Studio Code**
2. Opened **Extensions Marketplace**



3. Searched for **GitHub Copilot**

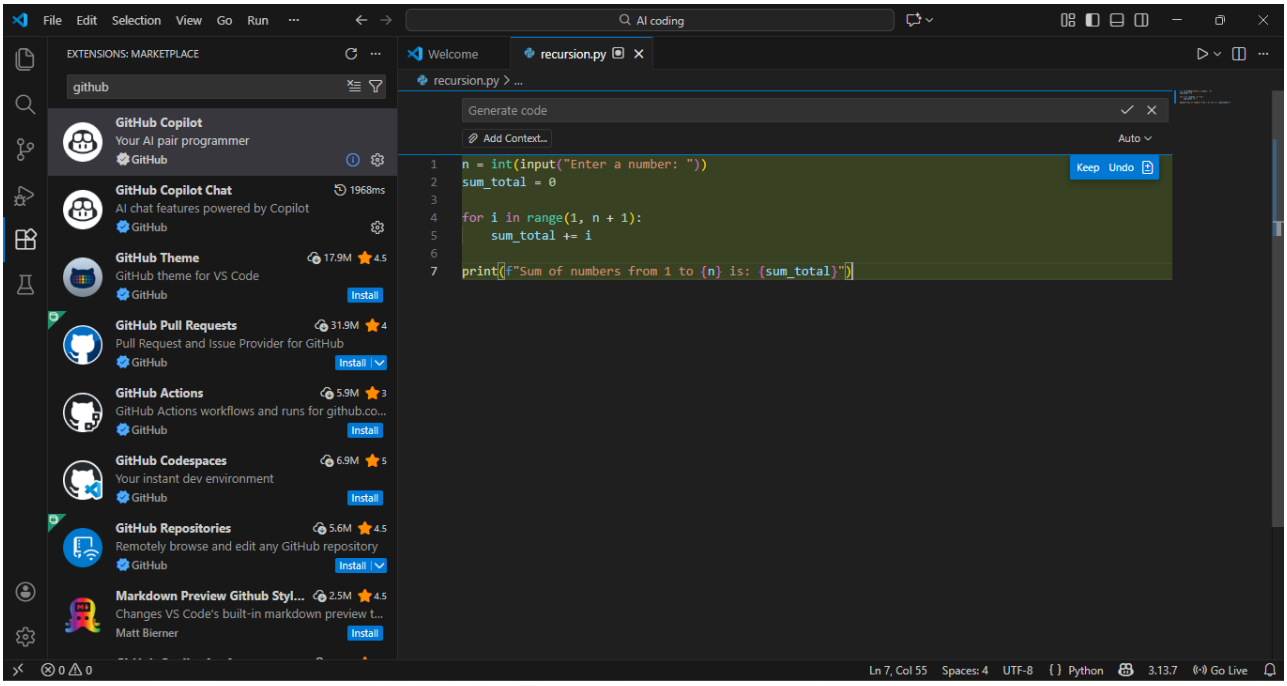


4. Clicked **Install**

5. Signed in with **GitHub Account**

6. Enabled Copilot suggestions

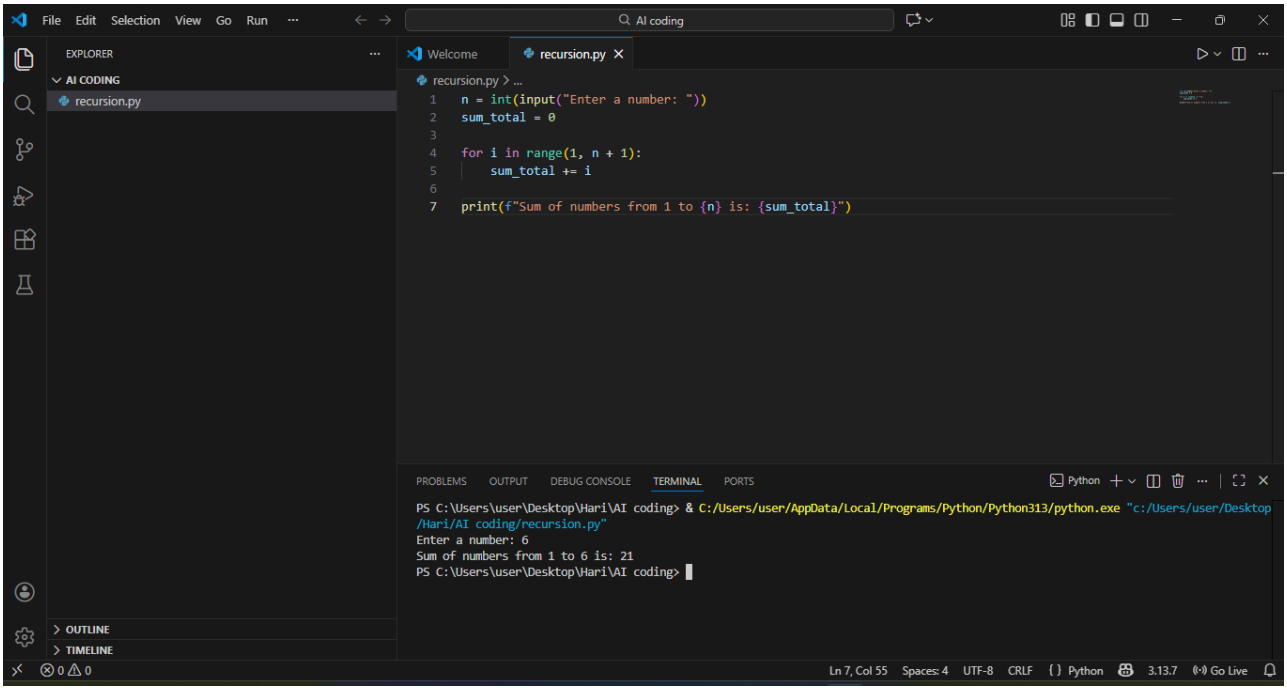
7. Verified Copilot inline suggestions in Python file



Task 1: AI-Generated Logic Without Modularization

(Factorial without Functions)

Prompt Used: "Write a Python program to calculate factorial of a number using loops only, without defining any function."



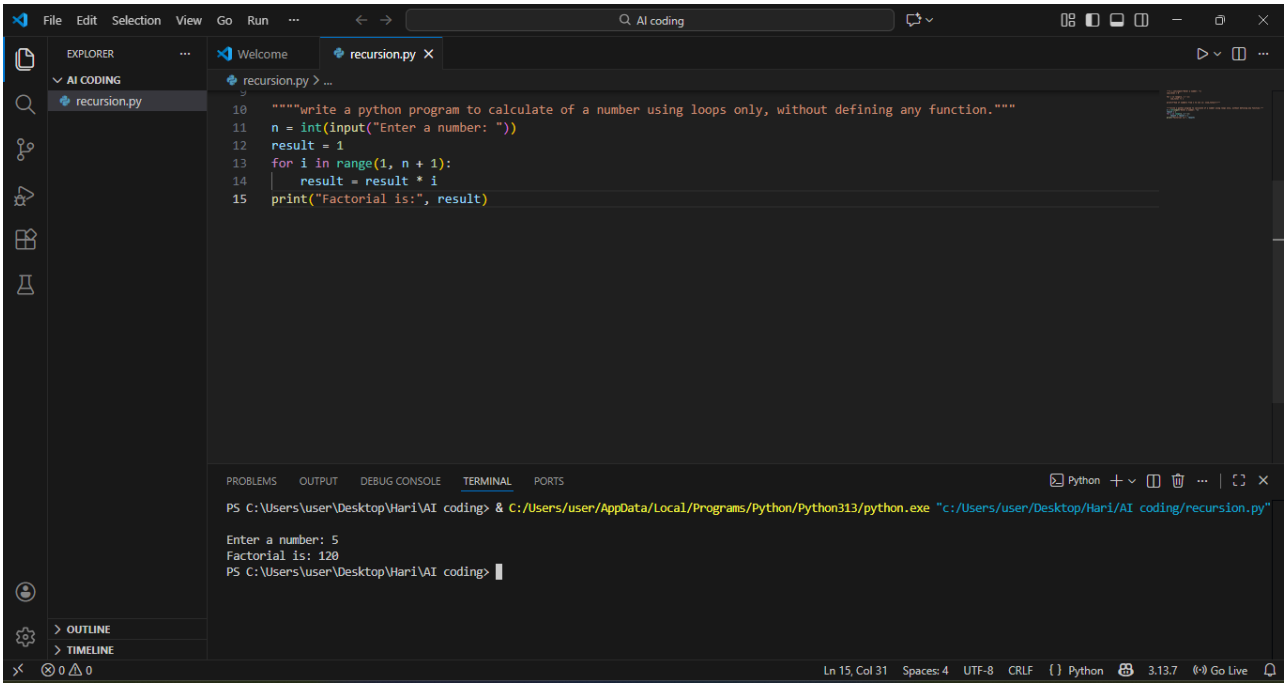
GitHub Copilot was very helpful for a beginner as it generated correct logic instantly.

It followed basic Python syntax and loop structure accurately.

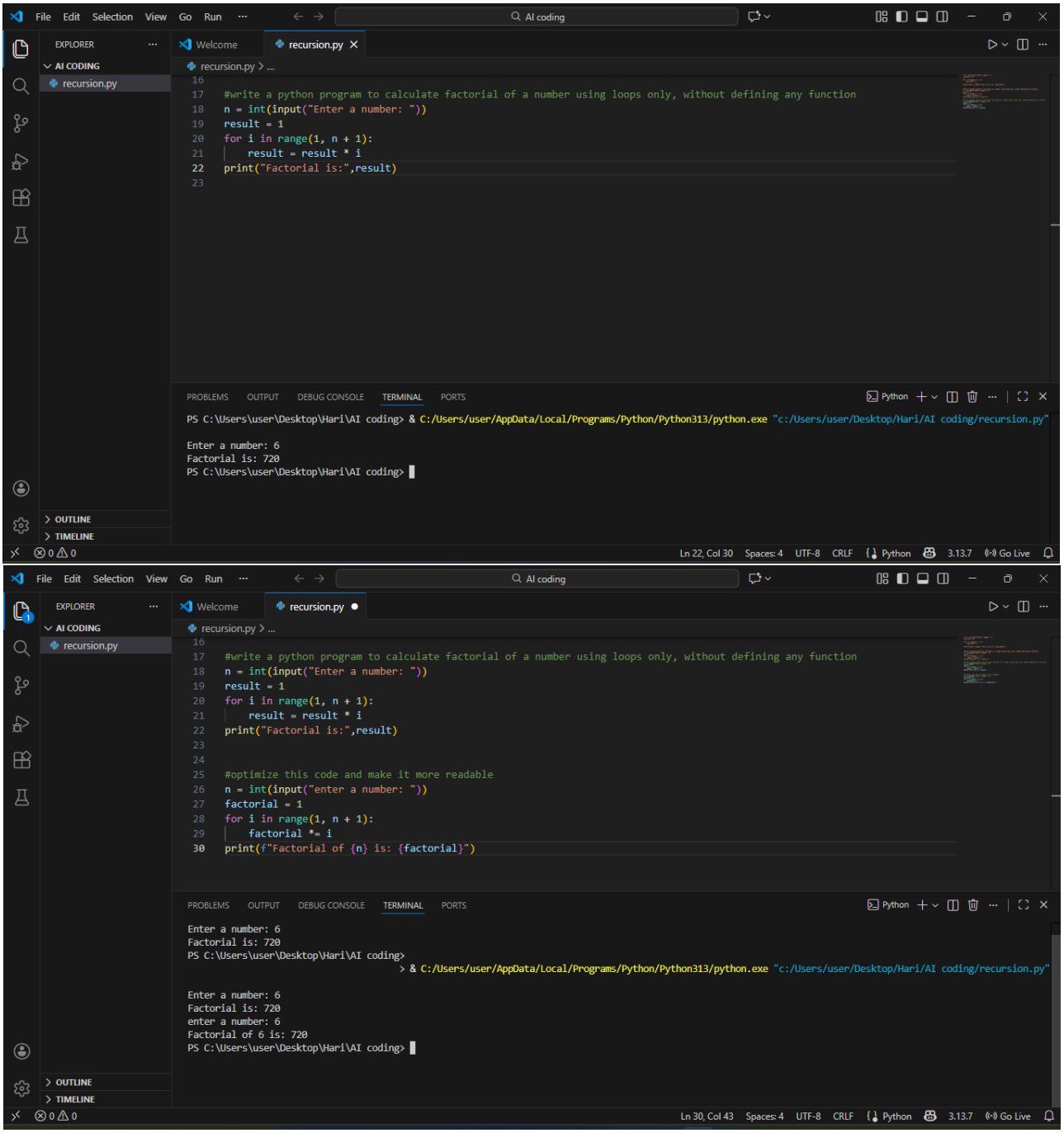
The code was readable and easy to understand.
However, it did not include input validation automatically.
Best practices like modular design were not applied unless explicitly prompted.

Task 2: AI Code Optimization & Cleanup

Original Code:



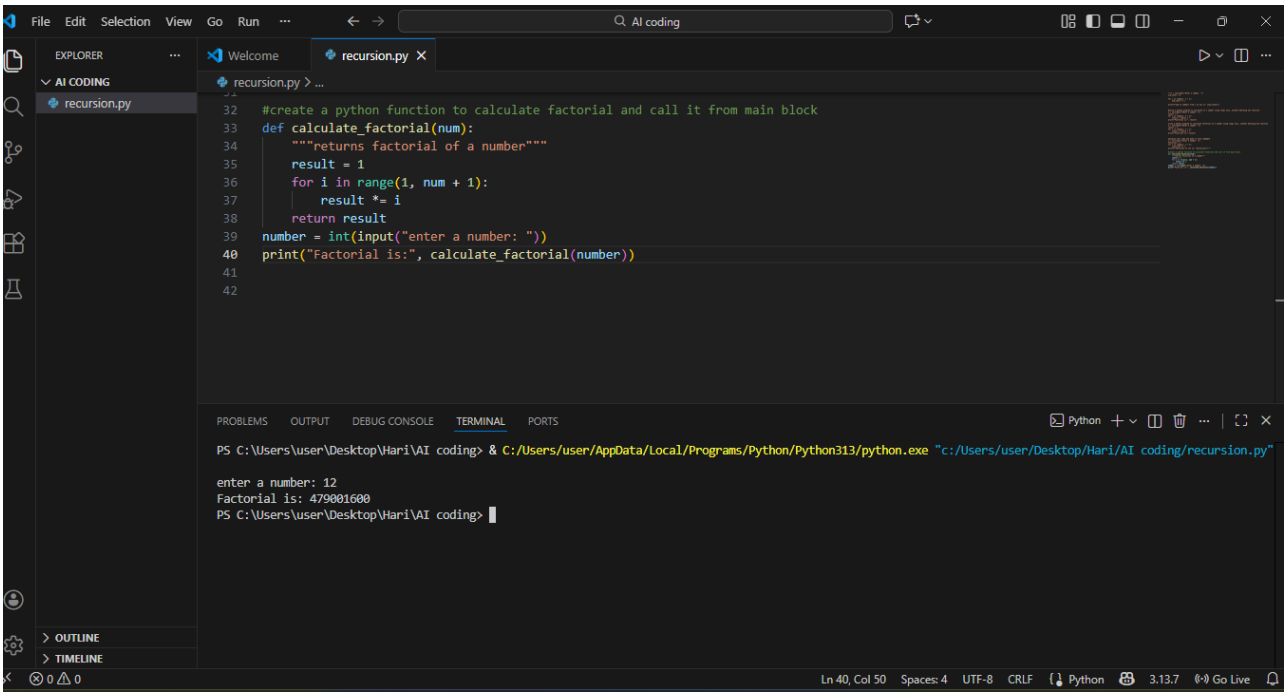
Prompt Used: “Optimize this code and make it more readable”



The optimized version improves clarity, maintainability, and readability without affecting performance.

Task 3: Modular Design Using AI Assistance (Factorial with Functions)

Prompt Used: “Create a Python function to calculate factorial and call it from main block”



Modularity improves reusability by allowing the same function to be used across multiple programs. It also simplifies testing and debugging.

Task 4: Comparative Analysis

Procedural vs Modular AI Code

Criteria	**Without Function**	**With Function**
Logic Clarity	Moderate	High
Reusability	No	Yes

Debugging Ease	Difficult	Easy
Large Project Suitability	Poor	Excellent
AI Dependency Risk	Higher	Lower

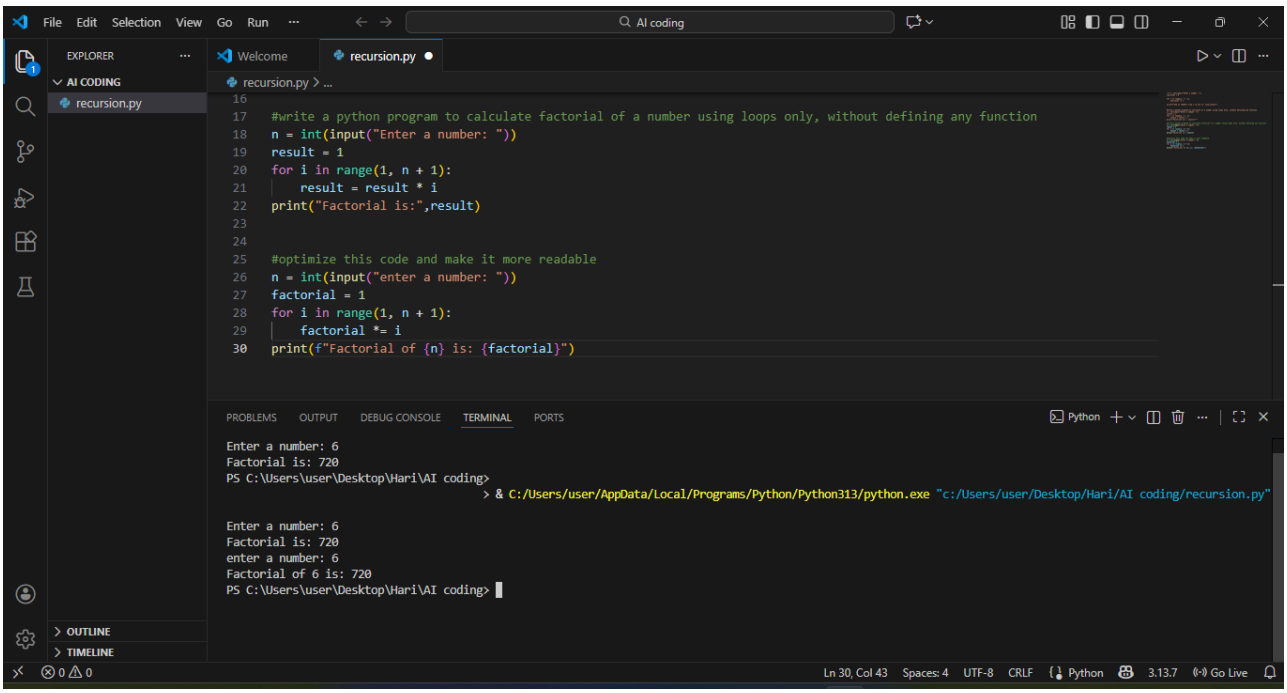
Conclusion:

Function-based design is more scalable and suitable for real-world applications.

Task 5: Iterative vs Recursive AI Code

Prompt Used: “Generate iterative and recursive factorial programs in Python”

Execution Flow Explanation:



- Iterative version uses a loop and constant memory.
- Recursive version uses function calls and stack memory.

Comparison:

Aspect	Iterative	Recursive
Readability	Simple	Elegant

Stack Usage	No	Yes
Performance	Faster	Slower
Risk	Low	Stack Overflow
Recommendation	Preferred	Avoid for large inputs