

Lab Assignment 1.2 – AI Assisted Coding

Harika Jupaka

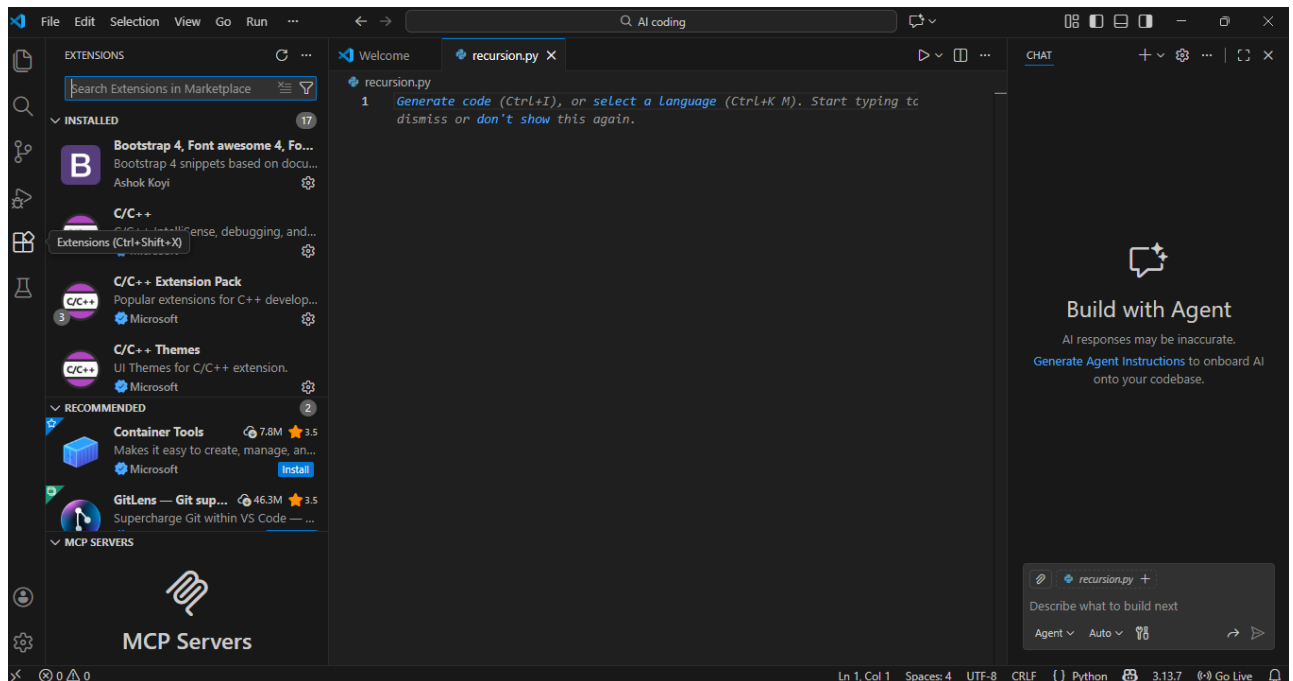
2403A51L31

B:52

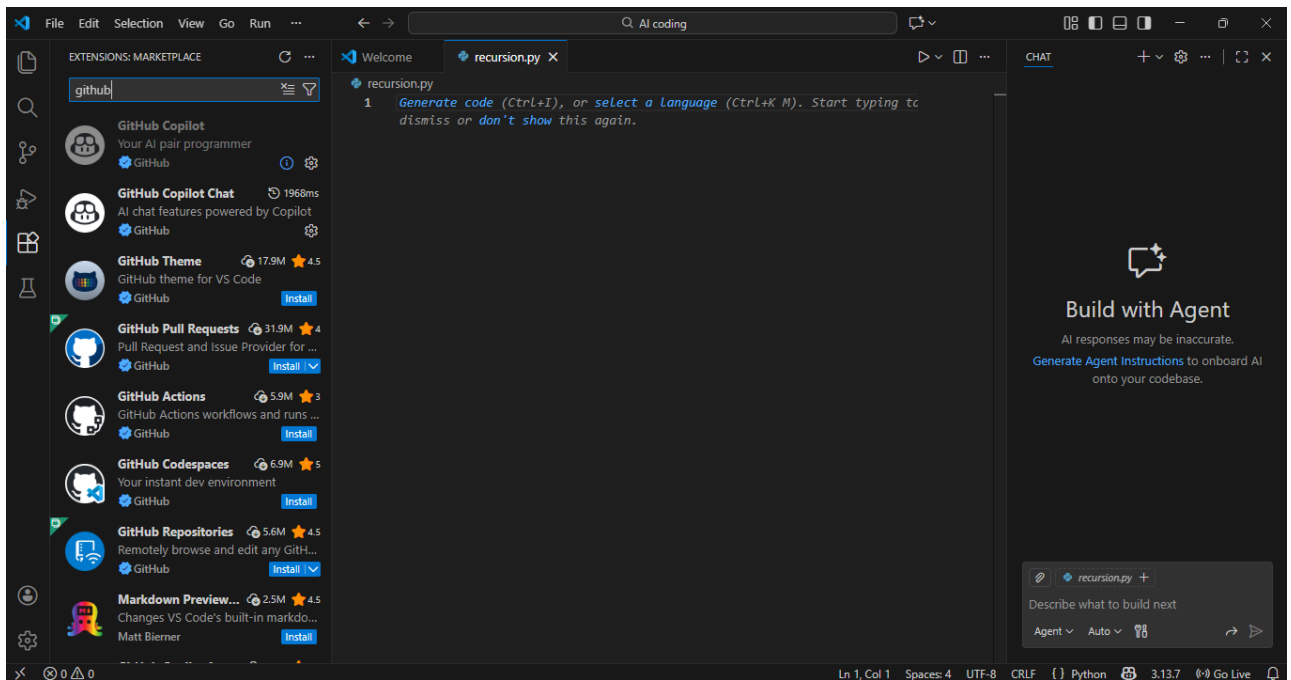
Task 0: GitHub Copilot Installation & Configuration

Steps Followed:

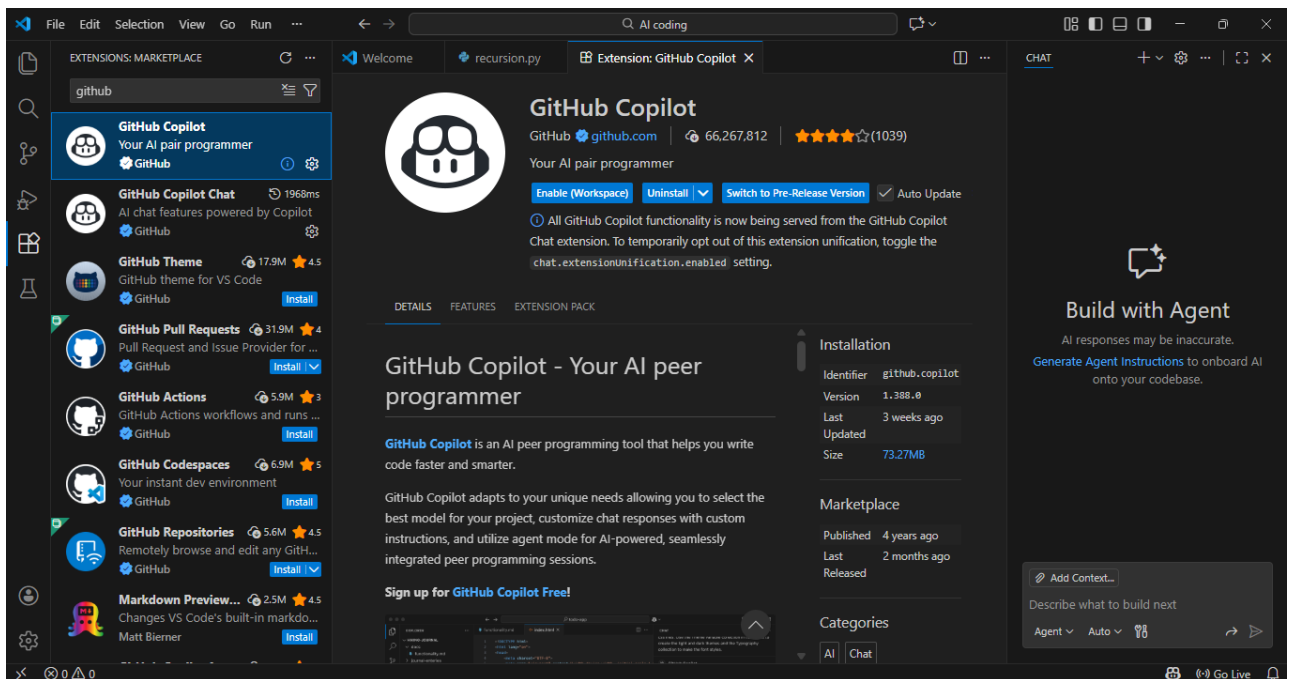
1. Installed **Visual Studio Code**
2. Opened **Extensions Marketplace**



3. Searched for **GitHub Copilot**



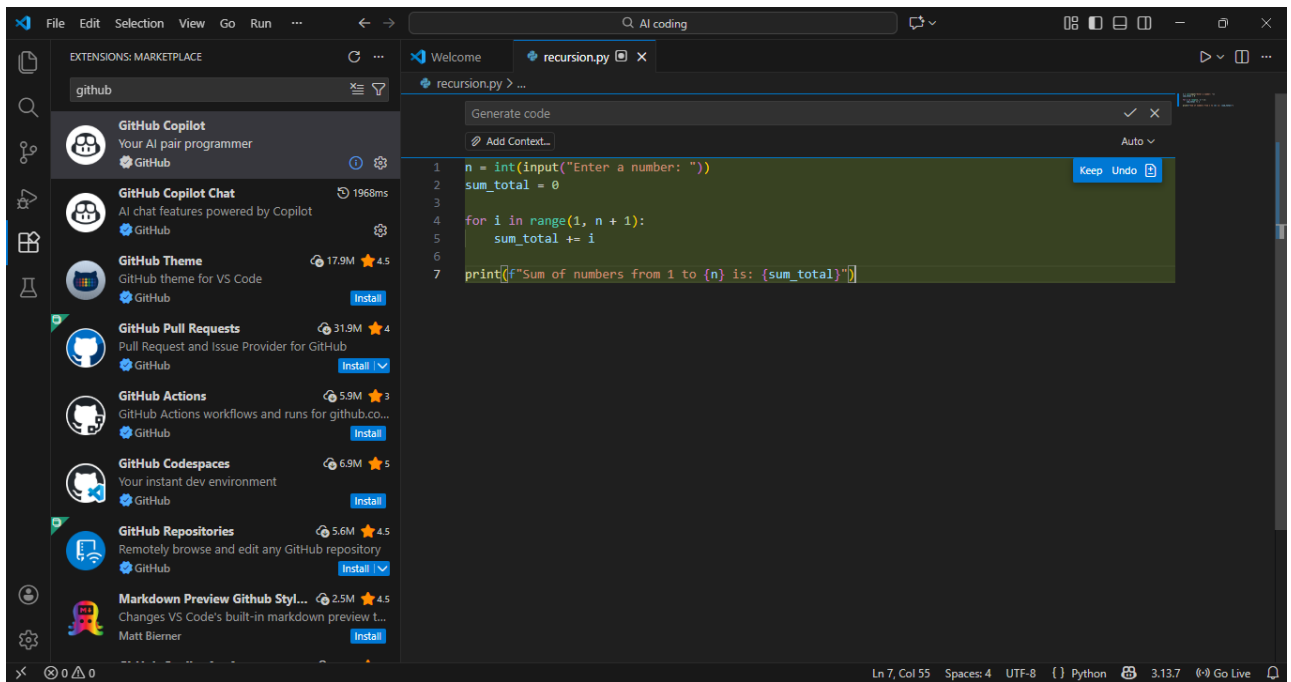
4. Clicked Install



5. Signed in with **GitHub Account**

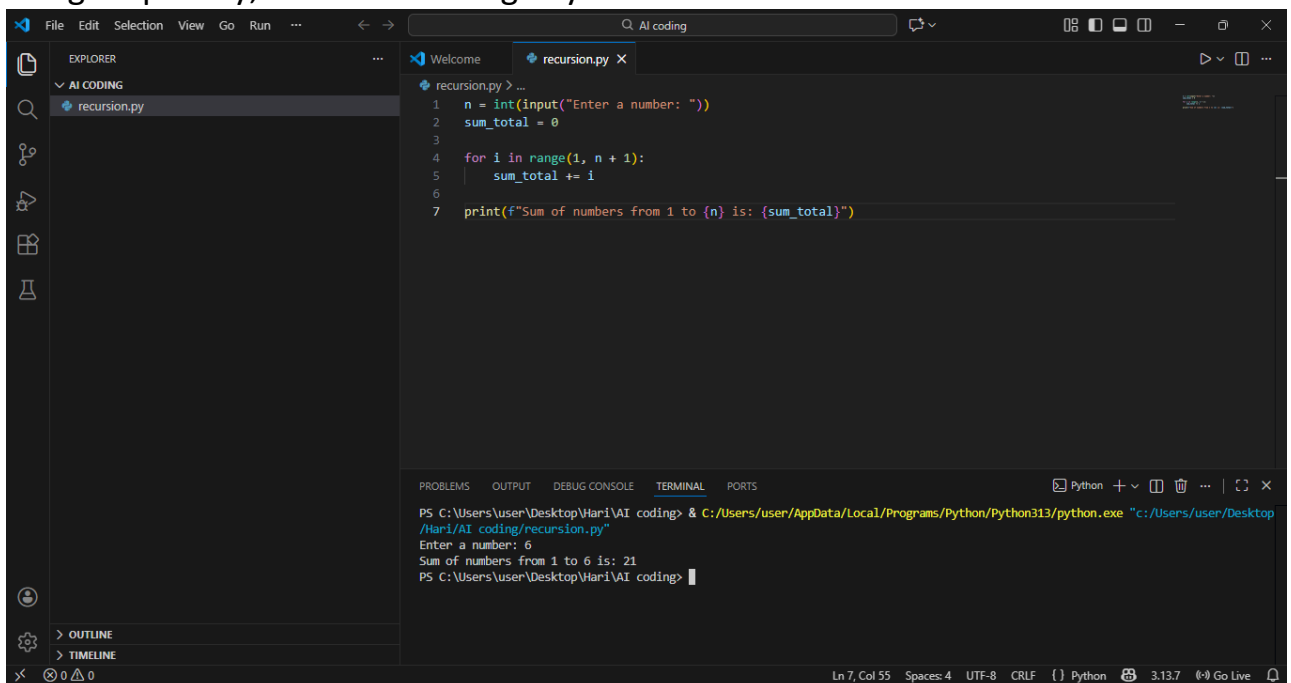
6. Enabled Copilot suggestions

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.”

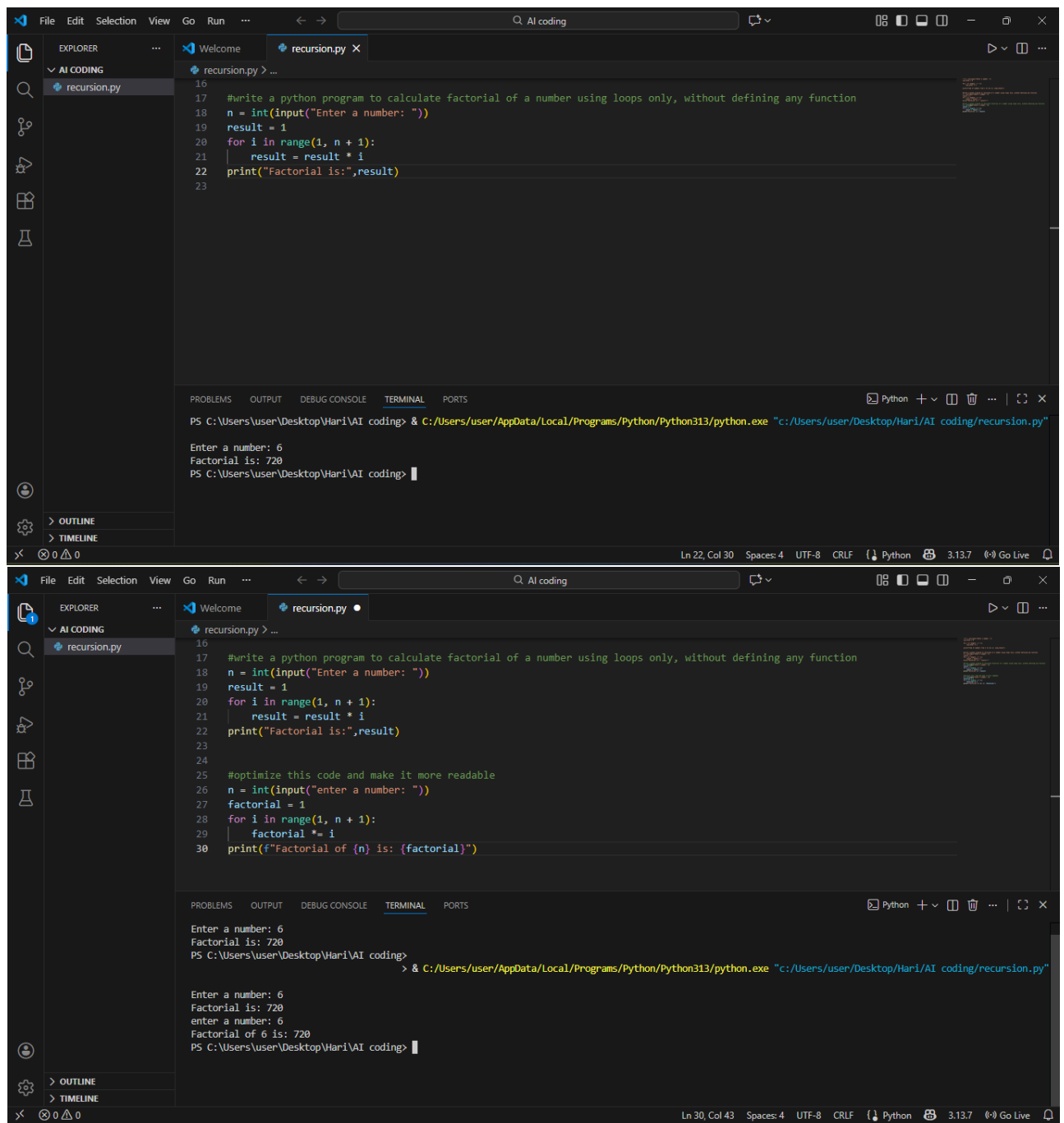


GitHub Copilot was very helpful for a beginner as it generated correct logic instantly.

Best practices like modular design were not applied unless explicitly prompted.

Original Code:

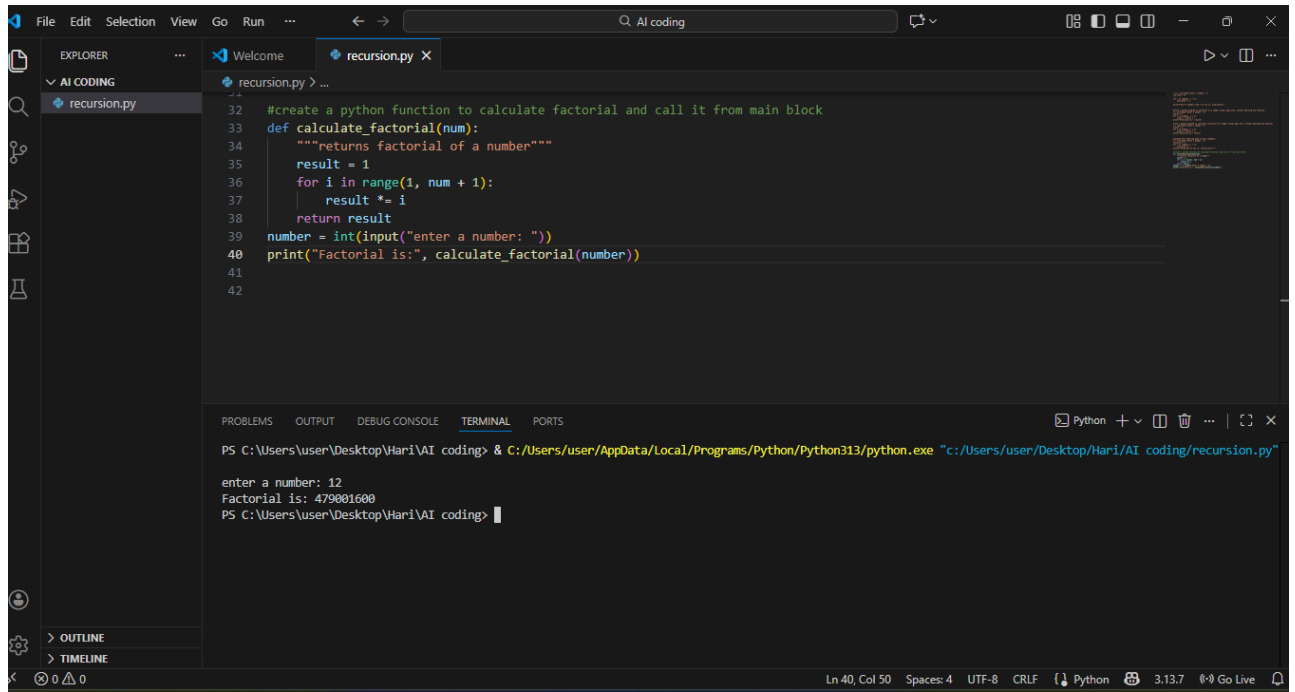




The optimized version improves clarity, maintainability, and readability without affecting performance.

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”



The screenshot shows the Visual Studio Code editor with a file named `recursion.py` open. The code defines a function `calculate_factorial(num)` that calculates the factorial of a number using a loop. It then prompts the user to enter a number and prints the result. The terminal output shows the program running successfully for the input 12.

```
32 #create a python function to calculate factorial and call it from main block
33 def calculate_factorial(num):
34     """returns factorial of a number"""
35     result = 1
36     for i in range(1, num + 1):
37         result *= i
38     return result
39 number = int(input("enter a number: "))
40 print("Factorial is:", calculate_factorial(number))
41
42
```

Terminal Output:

```
PS C:\Users\user\Desktop\Hari\AI coding> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/user/Desktop/Hari/AI coding/recursion.py"
enter a number: 12
Factorial is: 479001600
PS C:\Users\user\Desktop\Hari\AI coding>
```

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:

The screenshot shows a VS Code editor with two Python files. The first file, `recursion.py`, contains two programs. The first program is an iterative factorial calculation using a loop. The second program is a recursive factorial calculation using function calls. The terminal shows the execution of the iterative version, which prompts for a number and outputs the factorial.

```

16
17 #write a python program to calculate factorial of a number using loops only, without defining any function
18 n = int(input("Enter a number: "))
19 result = 1
20 for i in range(1, n + 1):
21     result = result * i
22 print("Factorial is:",result)
23
24
25 #optimize this code and make it more readable
26 n = int(input("enter a number: "))
27 factorial = 1
28 for i in range(1, n + 1):
29     factorial *= i
30 print(f"Factorial of {n} is: {factorial}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

Enter a number: 6
Factorial is: 720
PS C:\Users\user\Desktop\Hari\AI coding>
> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/user/Desktop/Hari/AI coding/recursion.py"

Enter a number: 6
Factorial is: 720
enter a number: 6
Factorial of 6 is: 720
PS C:\Users\user\Desktop\Hari\AI coding>

```

Ln 30, Col 43 Spaces: 4 UTF-8 CRLF Python 3.13.7 Go Live

- 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