

Lab Assignment 1.2 – AI Assisted Coding

Harika Jupaka

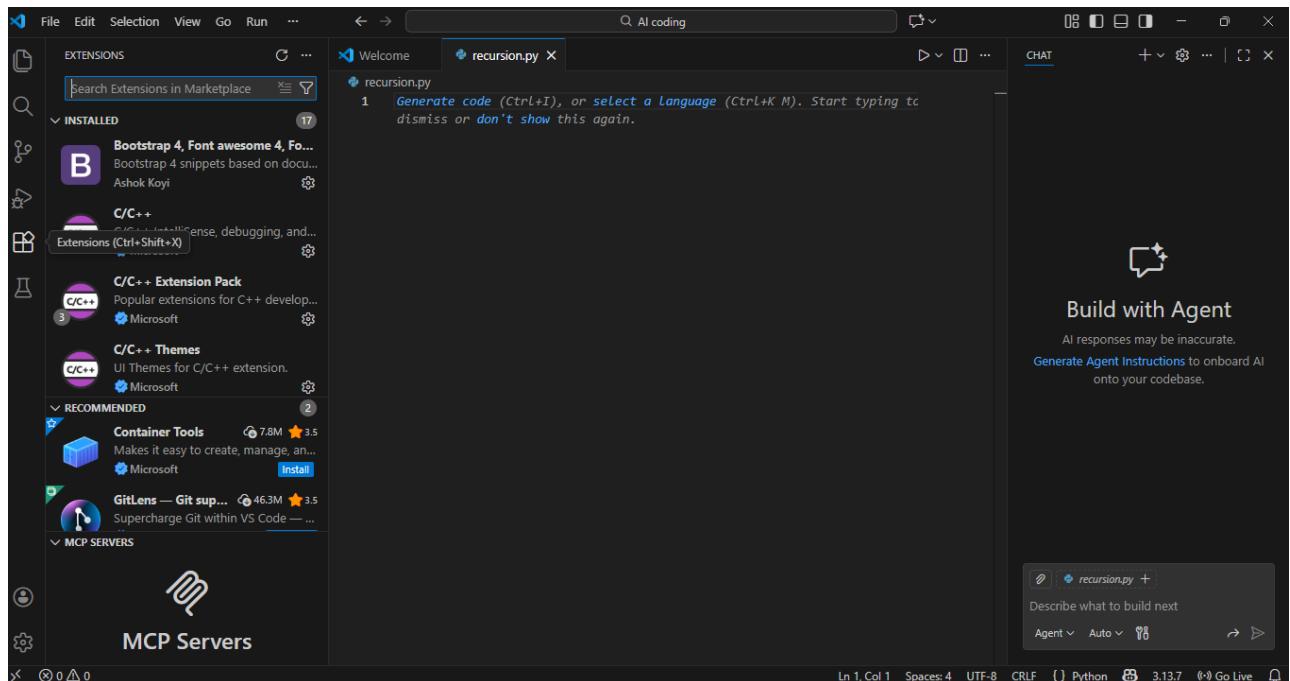
2403A51L31

B:52

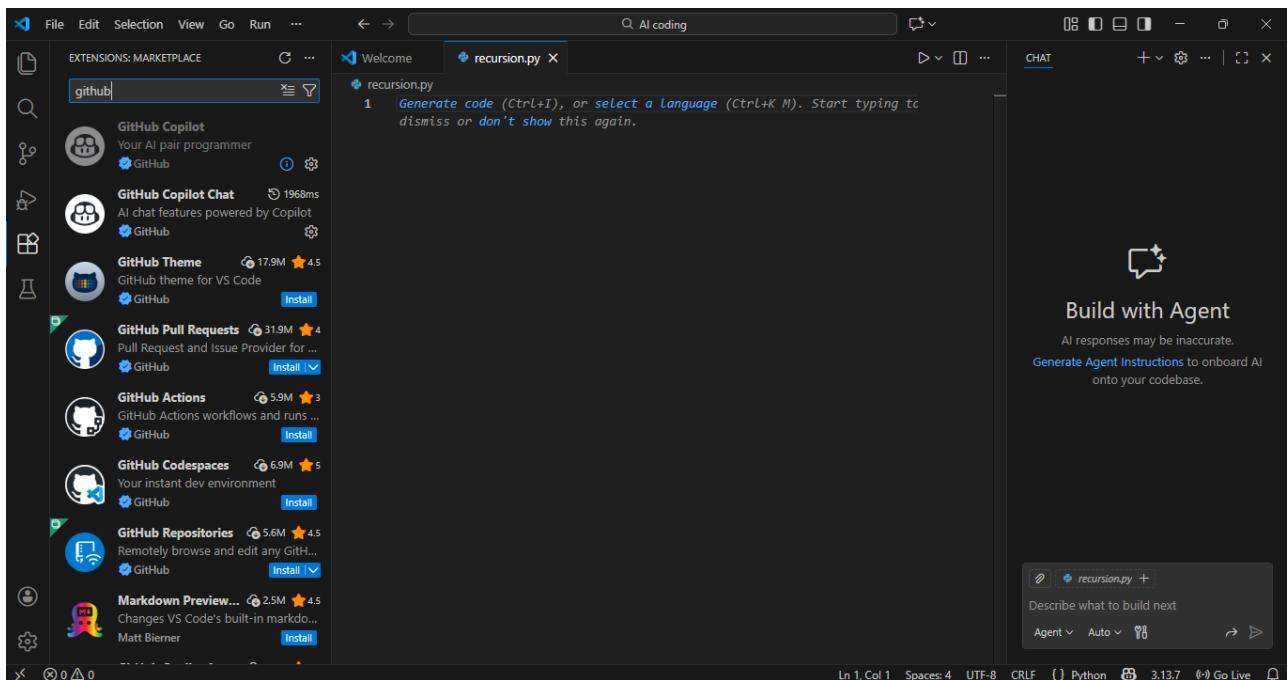
Task 0: GitHub Copilot Installation & Configuration

Steps Followed:

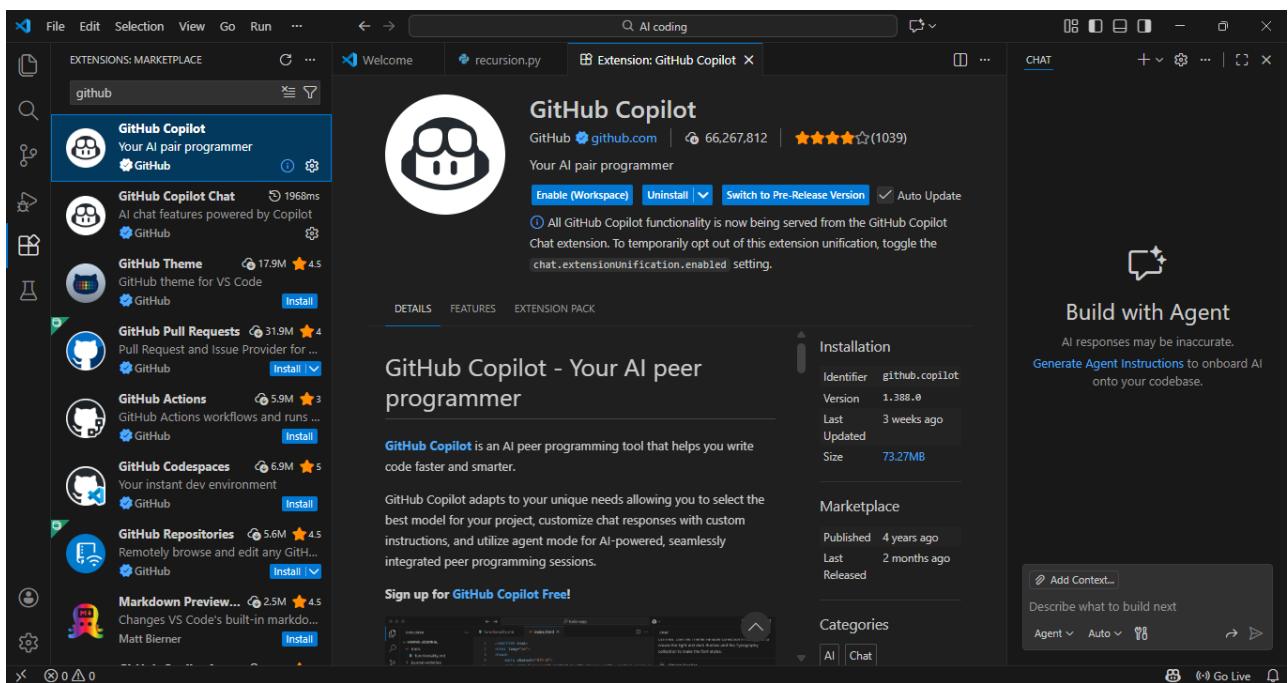
1. Installed Visual Studio Code
2. Opened Extensions Marketplace



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

The screenshot shows the VS Code interface with the GitHub Copilot extension open. The sidebar displays various GitHub extensions, with 'GitHub Copilot' selected. The main editor window shows a Python script named 'recursion.py' containing the following code:

```
n = int(input("Enter a number: "))
sum_total = 0

for i in range(1, n + 1):
    sum_total += i

print(f"Sum of numbers from 1 to {n} is: {sum_total}")
```

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.”

The screenshot shows the VS Code interface with the terminal tab active. The terminal output shows the execution of the 'recursion.py' script and its results:

```
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
Sum of numbers from 1 to 6 is: 21
PS C:\Users\user\Desktop\Hari\AI coding>
```

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

It followed basic Python syntax and loop structure accurately.

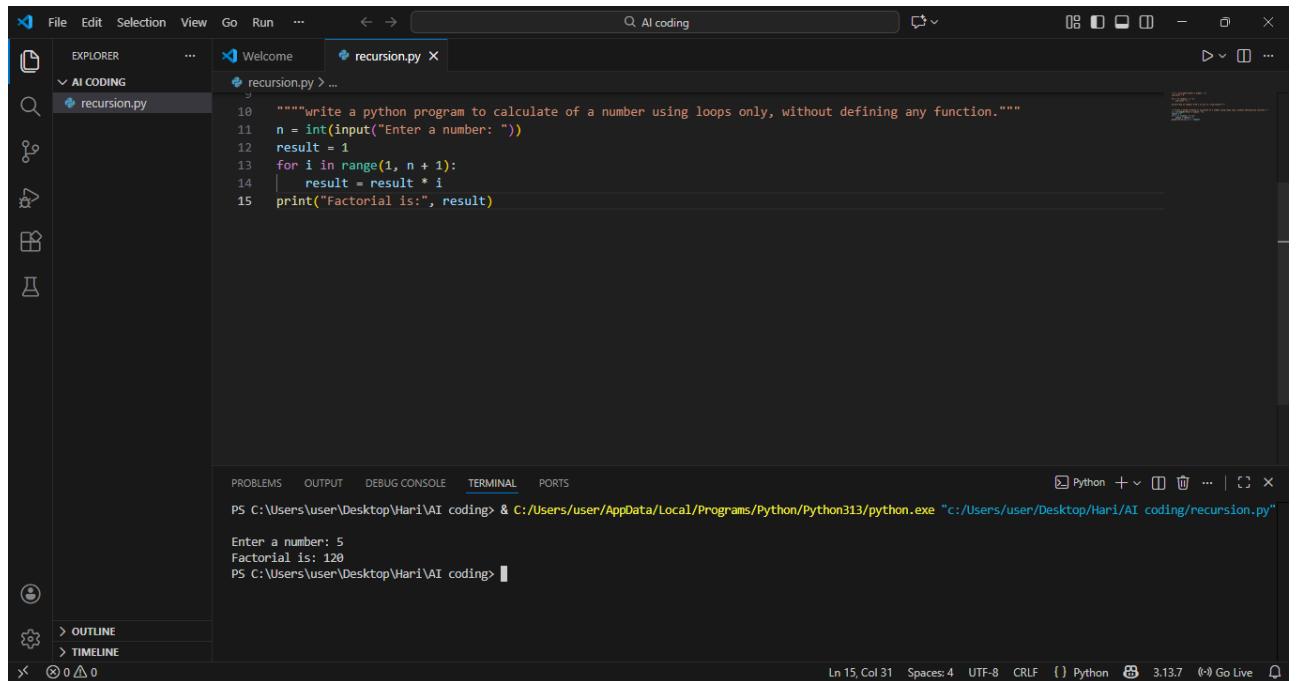
The code was readable and easy to understand.

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:



A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows the Explorer panel with a folder named 'AI CODING' containing a file named 'recursion.py'. The main editor area displays the following Python code:

```
10     """write a python program to calculate of a number using loops only, without defining any function."""
11 n = int(input("Enter a number: "))
12 result = 1
13 for i in range(1, n + 1):
14     result = result * i
15 print("Factorial is:", result)
```

The terminal at the bottom shows the output of running the script:

```
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: 5
Factorial is: 120
PS C:\Users\user\Desktop\Hari\AI coding>
```

The status bar at the bottom right indicates the code is 3.13.7 and the file is Python.

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

The image shows two side-by-side screenshots of the Visual Studio Code (VS Code) interface, demonstrating the optimization of a Python script named `recursion.py`.

Top Screenshot:

- Code:**

```
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
```
- 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: 6
Factorial is: 720
PS C:\Users\user\Desktop\Hari\AI coding>
```

Bottom Screenshot:

- Code:**

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

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”

```
#create a python function to calculate factorial and call it from main block
def calculate_factorial(num):
    """returns factorial of a number"""
    result = 1
    for i in range(1, num + 1):
        result *= i
    return result
number = int(input("enter a number: "))
print("Factorial is:", calculate_factorial(number))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

Without

Criteria	Function	With Function
Logic Clarity	Moderate	High
Reusability	No	Yes

Debugging Ease Difficult Easy

Large Project Suitability Poor Excellent

AI Dependency Risk Higher Lower

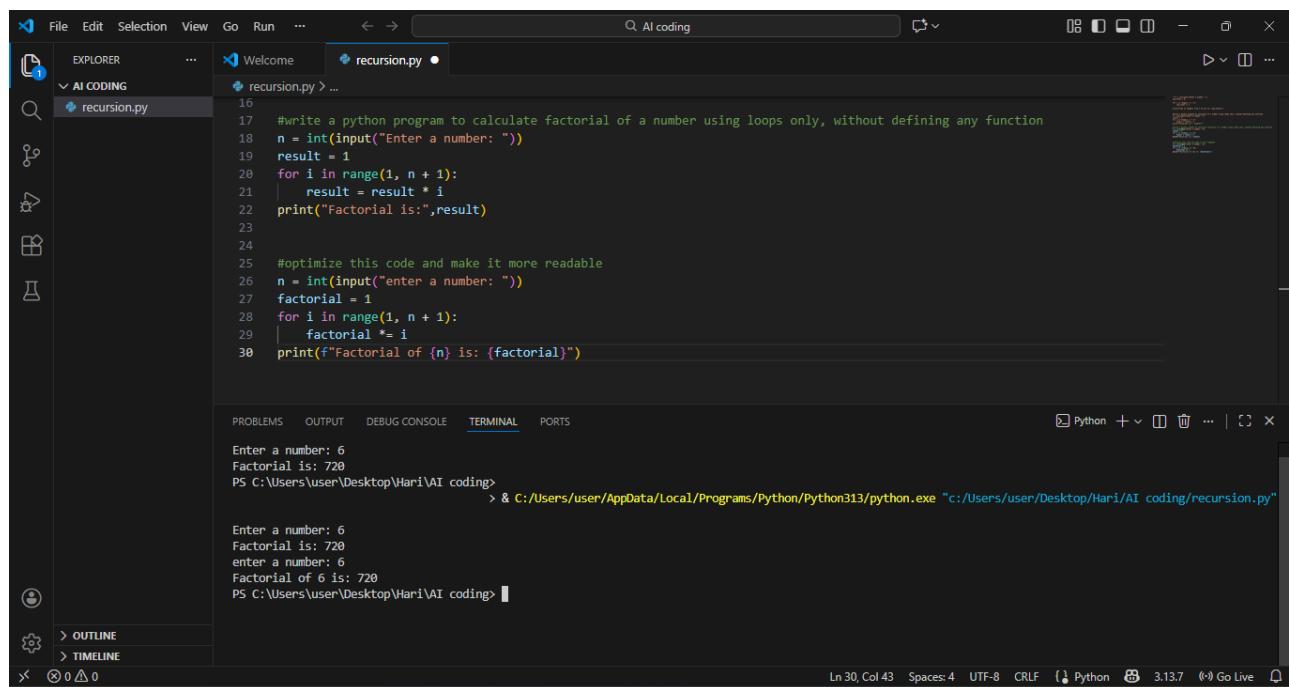
Conclusion:

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 the Visual Studio Code interface with two Python files open in the Explorer sidebar: 'recursion.py' and 'loop_factorial.py'. The 'recursion.py' file contains a recursive factorial function. The 'loop_factorial.py' file contains an iterative factorial function. Both files have code comments explaining their purpose. The terminal at the bottom shows the execution of the iterative program, where the user enters '6' and the program outputs 'Factorial is: 720'. The status bar at the bottom right indicates the code is saved in Python 3.13.7.

```
#write a python program to calculate factorial of a number using loops only, without defining any function
n = int(input("Enter a number: "))
result = 1
for i in range(1, n + 1):
    result = result * i
print("Factorial is:",result)

#optimize this code and make it more readable
n = int(input("enter a number: "))
factorial = 1
for i in range(1, n + 1):
    factorial *= i
print(f"Factorial of {n} is: {factorial}")
```

- 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