

# Lab Assignment 1.2 – AI Assisted Coding

Sarika Pale

2403A51L33

B:52

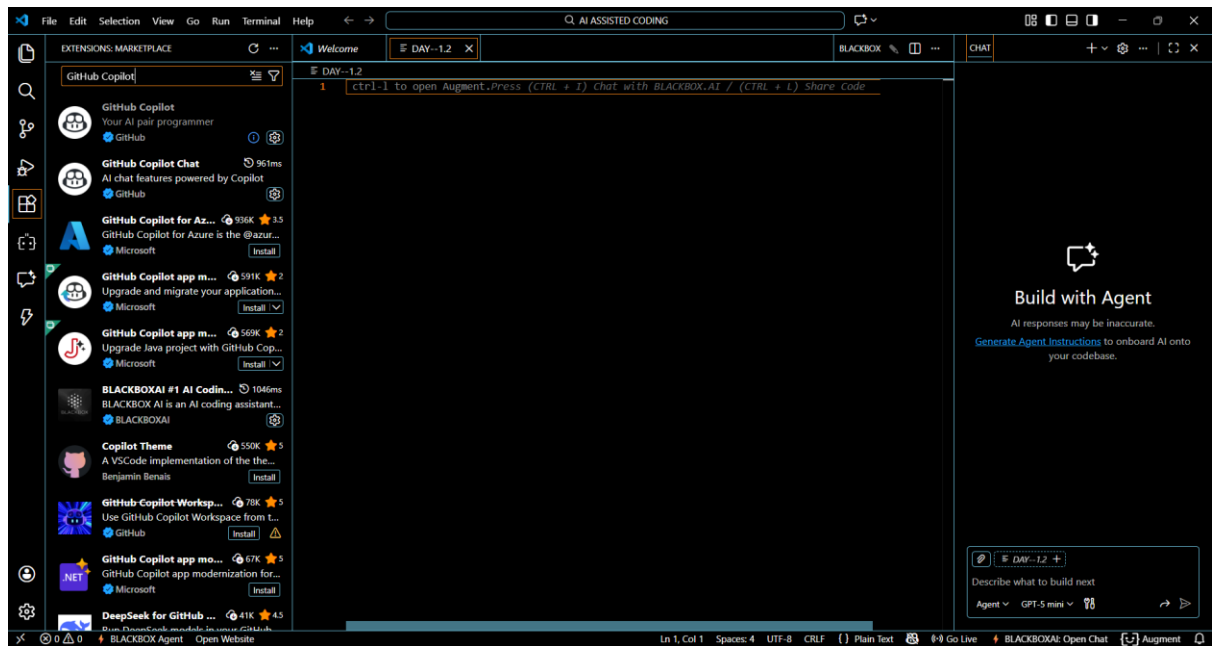
## Task 0: GitHub Copilot Installation & Configuration

### Steps Followed:

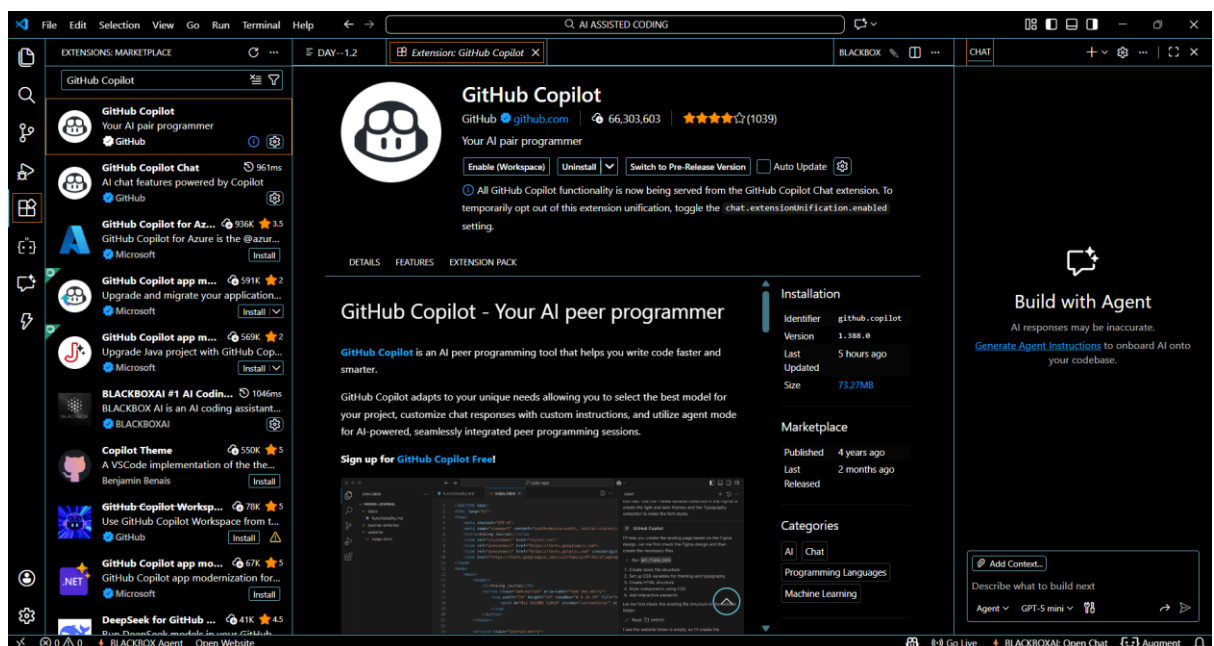
1. Installed **Visual Studio Code**
2. Opened **Extensions Marketplace**



3. Searched for **GitHub Copilot**



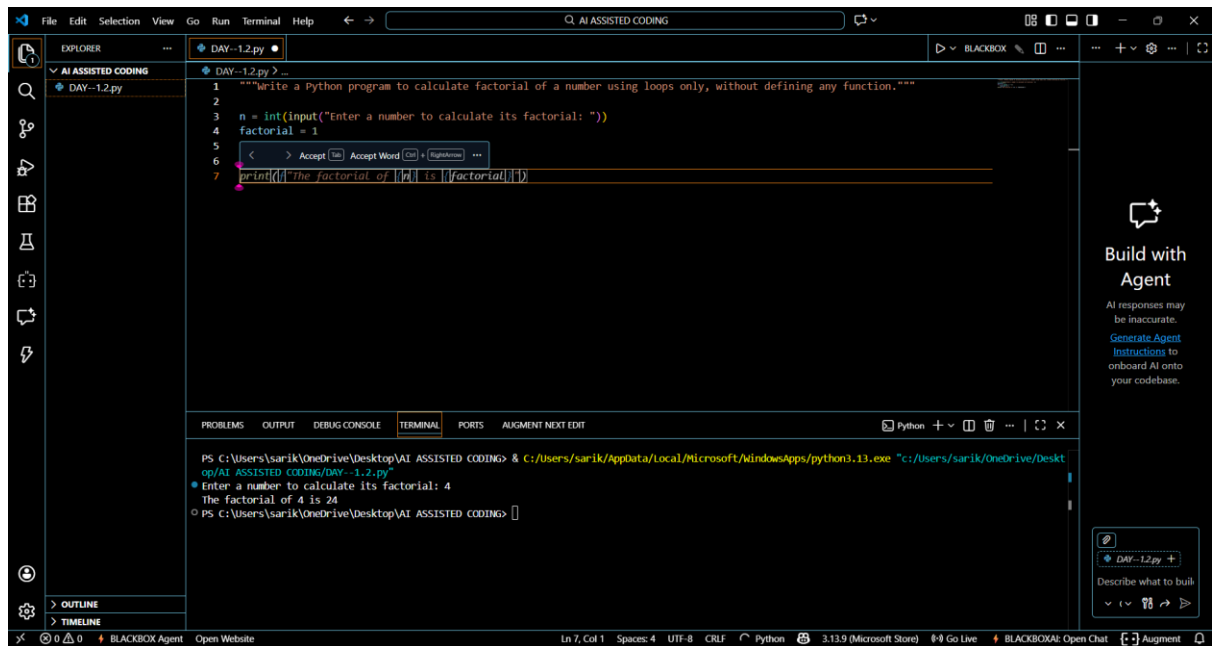
#### 4. Clicked Install



#### 5. Signed in with **GitHub Account**

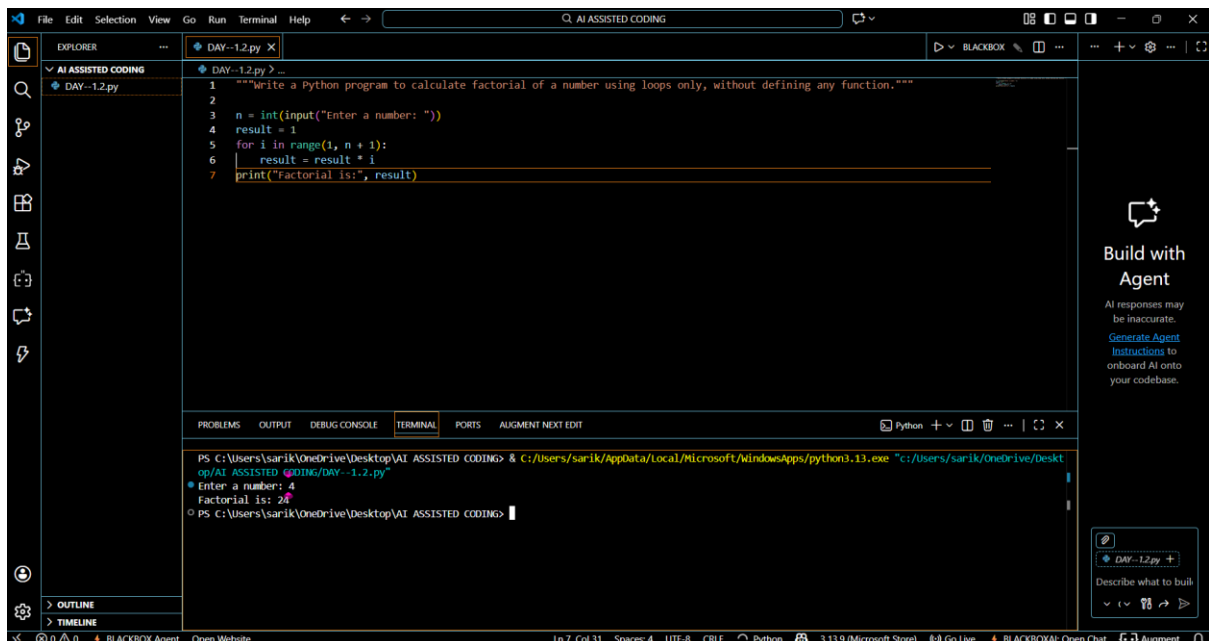
#### 6. Enabled Copilot suggestions

#### 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.”

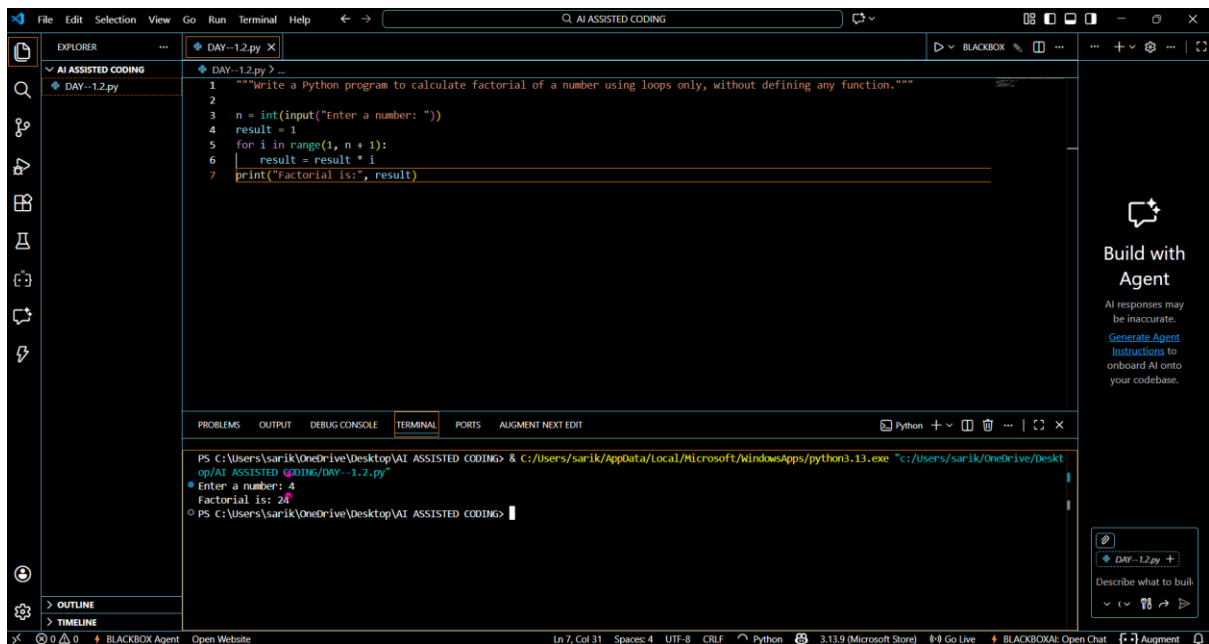


GitHub Copilot was very helpful for a beginner as it generated correct logic instantly.  
It followed basic Python syntax and loop structure accurately.

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:



```
File Edit Selection View Go Run Terminal Help
AI ASSISTED CODING
EXPLORER
DAY-1.2.py
DAY-1.2.py
1 '''Write a Python program to calculate factorial of a number using loops only, without defining any function.'''
2
3 n = int(input("Enter a number: "))
4 result = 1
5 for i in range(1, n + 1):
6     result = result * i
7 print("Factorial is:", result)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT
Python
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:\Users\sarik\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING\DAY-1.2.py"
Enter a number: 4
Factorial is: 24
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
DAY-1.2.py
Describe what to build
```

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

```
1 """Write a Python program to calculate factorial of a number using loops only, without defining any function."""
2
3 n = int(input("Enter a number: "))
4 result = 1
5 for i in range(1, n + 1):
6     result = result * i
7 print("Factorial is:", result)
8
9
10 """Optimize this code and make it more readable"""
11
12 n = int(input("Enter a number: "))
13 factorial = 1
14 for i in range(1, n + 1):
15     factorial *= i
16 print(f"Factorial of {n} is: {factorial}")
```

PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:\Users\sarik\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/DAY--1.2.py"

Enter a number: 4  
Factorial is: 24  
Enter a number: 2  
Factorial of 2 is: 2

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”

```
19 """Create a Python function to calculate factorial and call it from main block"""
20
21 def calculate_factorial(num):
22     """Returns factorial of a number"""
23     result = 1
24     for i in range(1, num + 1):
25         result *= i
26     return result
27
28 number = int(input("Enter a number: "))
29 print("Factorial is:", calculate_factorial(number))
```

PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:\Users\sarik\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/DAY--1.2.py"

Enter a number: 12  
Factorial is: 479001600

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”

```
30
31 """Generate iterative and recursive factorial programs in Python"""
32
33 """Iterative Version"""
34 def factorial_iterative(n):
35     result = 1
36     for i in range(1, n + 1):
37         result *= i
38     return result
39
40 """Recursive Version"""
41 def factorial_recursive(n):
42     if n == 0 or n == 1:
43         return 1
44     return n * factorial_recursive(n - 1)
45
46 number = int(input("Enter a number: "))
47 print("Iterative Factorial is:", factorial_iterative(number))
48 print("Recursive Factorial is:", factorial_recursive(number))
49
```

PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:\Users\sarik\AppData\Local\Microsoft\WindowsApps\python3.13.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/DAY--1.2.py"

Enter a number: 4  
Iterative Factorial is: 24  
Recursive Factorial is: 24

PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>

Build with Agent  
AI responses may be inaccurate.  
[Generate Agent](#)  
[Instructions](#) to onboard AI onto your codebase.

## 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