

# Lab Assignment 1.2 – AI Assisted Coding

Vyshnavi Parisha

2403A51L34

B:52

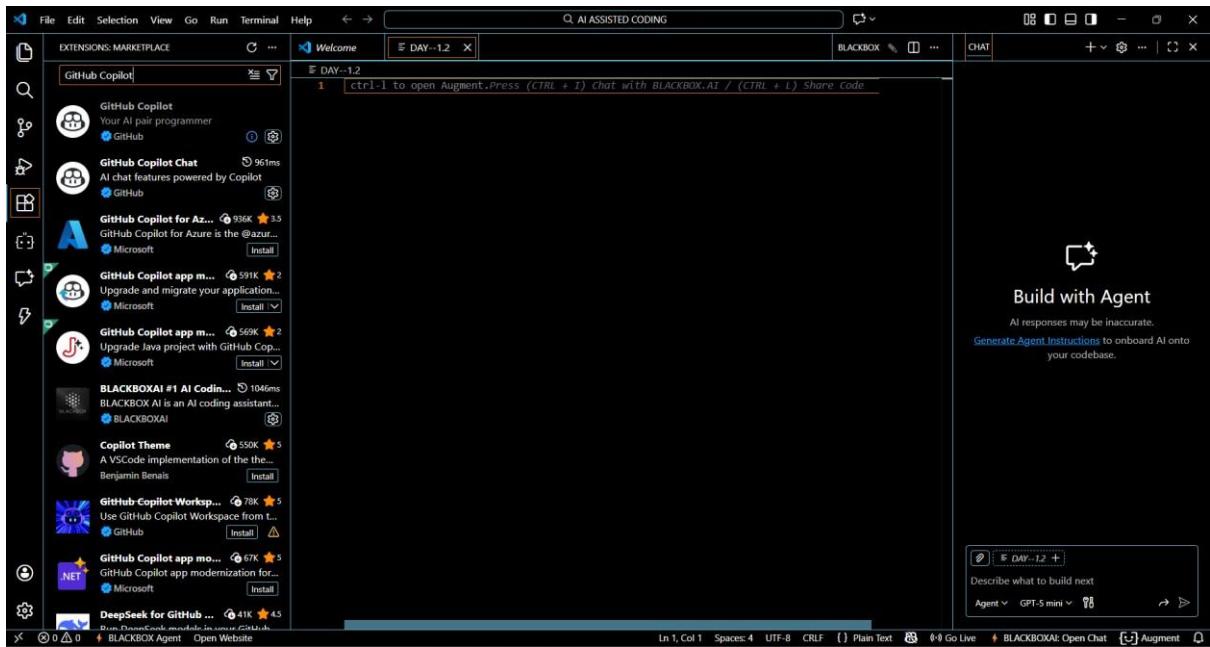
## Task 0: GitHub Copilot Installation & Configuration

### Steps Followed:

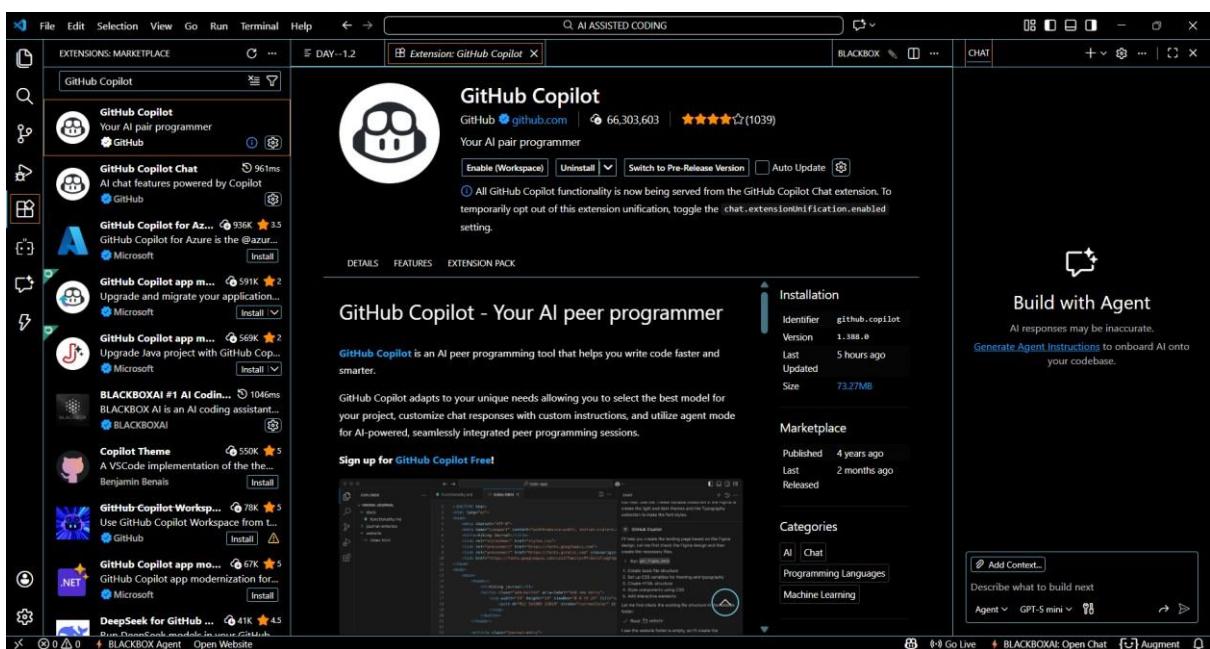
1. Installed Visual Studio Code
2. Opened Extensions Marketplace



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 Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "AI ASSISTED CODING" containing "DAY-1.2.py".
- Code Editor:** Displays Python code to calculate factorial using loops. A tooltip from the AI feature suggests using "Accept Word" to complete the loop structure.
- Terminal:** Shows the command line output for running the script "DAY-1.2.py". The user enters "Enter a number to calculate its factorial: 4" and receives the response "The factorial of 4 is 24".
- Right-hand sidebar:** Features a "Build with Agent" section with a "Generate Agent Instructions" button and a note about AI responses being inaccurate.
- Bottom status bar:** Shows file path "BLACKBOX Agent", "Open Website", and system information like "Ln 7, Col 1", "Spaces: 4", "UTF-8", "CRLF", "Python 3.13.9 (Microsoft Store)", "Go Live", and "BLACKBOXAI: Open Chat".

## 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 a Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "EXPLORER" and a file named "DAY-1.2.py".
- Search Bar:** Displays "AI ASSISTED CODING".
- Code Editor:** Contains Python code to calculate factorial using loops only.

```
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)
```
- Bottom Status Bar:** Shows the command prompt 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"
- Terminal Output:** Displays:
  - Enter a number: 4
  - Factorial is: 24
- Right Sidebar:** Features a "Build with Agent" section with a "Generate Agent Instructions" button.
- Bottom Navigation:** Includes icons for Outline, Timeline, and Open Website, along with the BLACKBOX Agent status.

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:

The screenshot shows the AI ASSISTED CODING interface. In the Explorer panel, there is a file named 'DAY-1.2.py'. The code in the editor is:

```
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)
```

The terminal window shows the execution of the script:

```
PS C:\Users\srarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/srarik/AppData/Local/Microsoft/WindowsApps/python3.13.exe "C:/Users/srarik/OneDrive/Desktop/ai_assisted_coding/day-1.2.py"
● Enter a number: 4
Factorial is: 24
○ PS C:\Users\srarik\OneDrive\Desktop\AI ASSISTED CODING>
```

A sidebar on the right is titled 'Build with Agent' with the sub-instruction 'AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase.'

**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 *= 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}")

```

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”

```

1  """Create a Python function to calculate factorial and call it from main block"""
2
3  def calculate_factorial(num):
4      """Returns factorial of a number"""
5      result = 1
6      for i in range(1, num + 1):
7          result *= i
8      return result
9
10 number = int(input("Enter a number: "))
11 print("Factorial is:", calculate_factorial(number))

```

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”

The screenshot shows a software interface titled "AI ASSISTED CODING". On the left is an "EXPLORER" sidebar with a tree view showing "AI ASSISTED CODING" and "DAY-1.2.py". The main area displays Python code for calculating factorials:

```

30     """Generate iterative and recursive factorial programs in Python"""
31
32     """Iterative Version"""
33
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

```

Below the code editor is a terminal window showing the execution of the script:

```

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>

```

At the bottom right, there is a "Build with Agent" panel with 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