

Lab Assignment 1.2 – AI Assisted Coding

Sarika Palle

2403A51L33

B:52

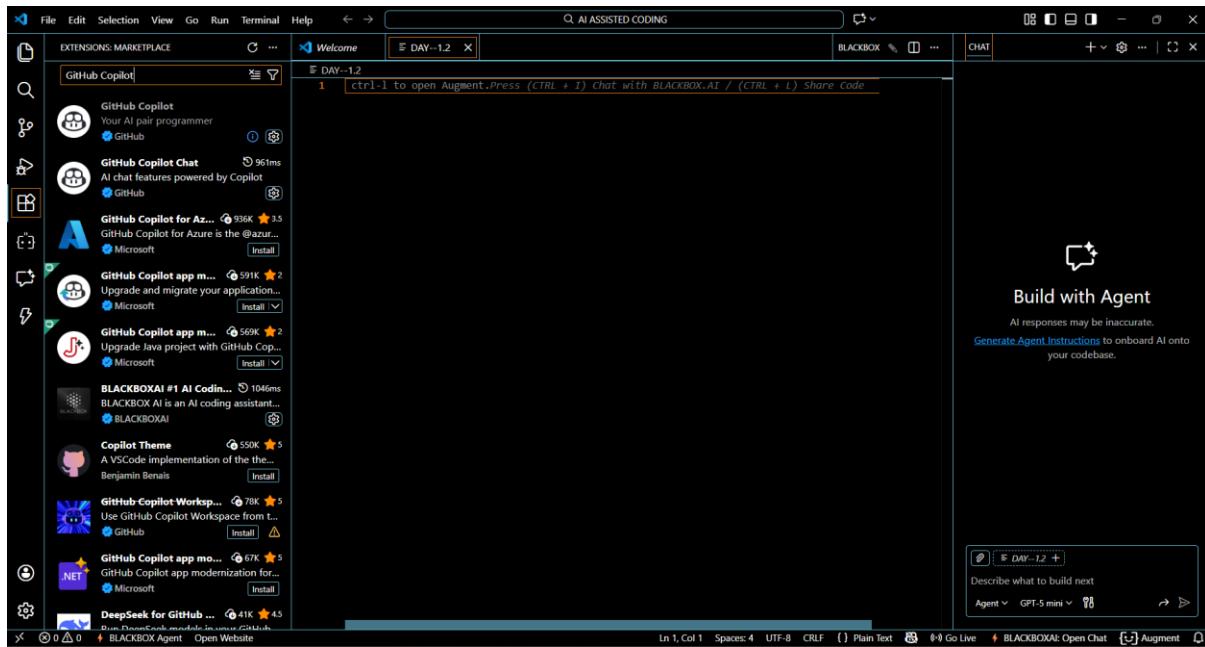
Task 0: GitHub Copilot Installation & Configuration

Steps Followed:

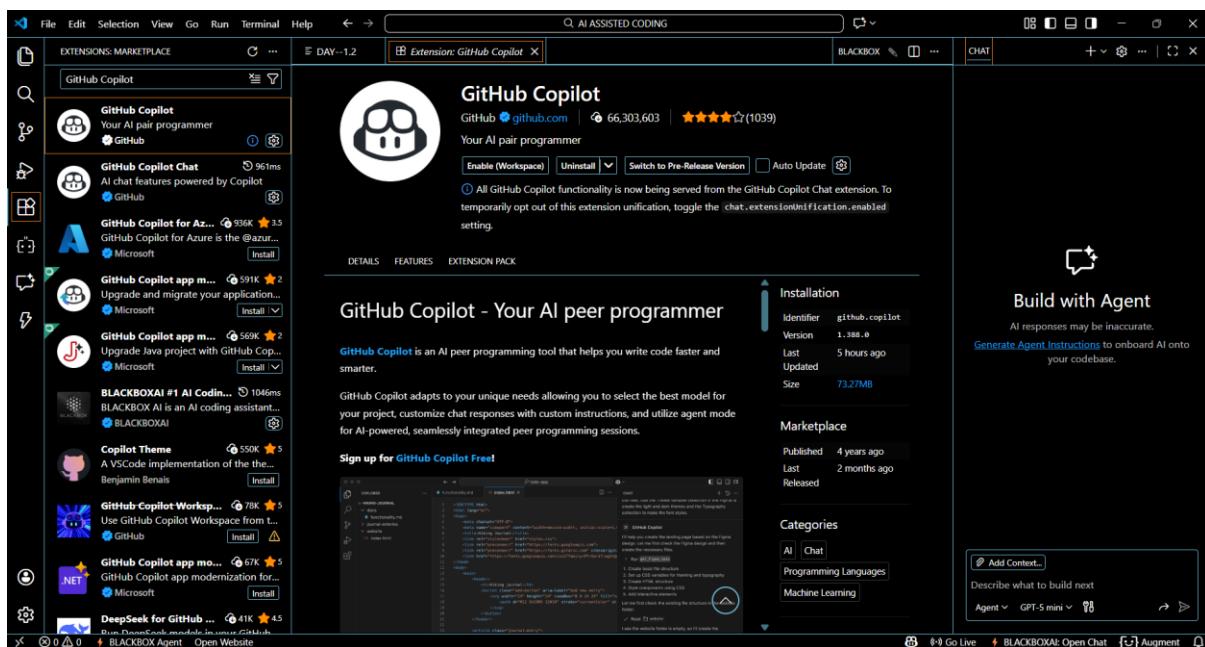
1. Installed Visual Studio Code
2. Opened Extensions Marketplace



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

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows a file named "DAY-1.2.py".
- Terminal:** Displays the following Python code:

```
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 to calculate its factorial: "))
4 factorial = 1
5
6 print(f"The factorial of {n} is {factorial}")
7
```
- Status Bar:** Shows the path "C:\Users\sarik\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Microsoft/WindowsApps/python3.13.exe" and the command "c:/Users/sarik/OneDrive/Desktop/".
- Bottom Right Panel:** A "Build with Agent" panel is open, showing a tooltip from the AI Agent suggesting "Accept" or "Accept Word". It also includes instructions: "AI responses may be inaccurate." and "Generate Agent Instructions to onboard AI onto your codebase."

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.”

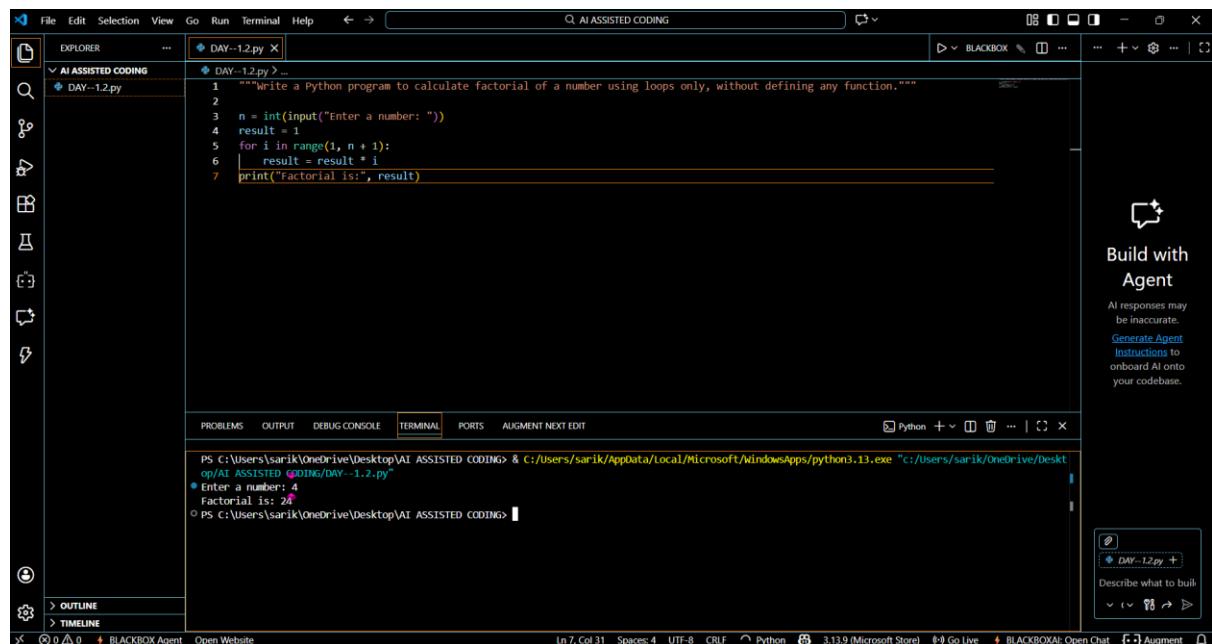
GitHub Copilot was very helpful for a beginner as it generated correct logic instantly.

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:



A screenshot of the Microsoft Visual Studio Code (VS Code) interface. The window title is "AI ASSISTED CODING". The left sidebar shows the "EXPLORER" view with a file named "DAY-1.2.py" selected. The main editor area contains the following Python code:

```
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 bottom right corner of the editor has a tooltip for "Build with Agent" which says: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase." The bottom status bar shows the command line 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"

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

The screenshot shows the AI ASSISTED CODING interface in a code editor. The left sidebar has icons for Explorer, Search, Find, Open, Timeline, and Outline. The main area shows a Python script named DAY--1.2.py. The code calculates the factorial of a number using loops. A tooltip from the AI Agent suggests optimizing the code for readability. The terminal below shows the execution of the script and its output.

```
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}")
```

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
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED 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”

The screenshot shows the AI ASSISTED CODING interface in a code editor. The left sidebar has icons for Explorer, Search, Find, Open, Timeline, and Outline. The main area shows a Python script named DAY--1.2.py. The code defines a function calculate_factorial(num) and calls it from the main block. A tooltip from the AI Agent suggests returning the factorial of a number. The terminal below shows the execution of the script and its output.

```
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 number = int(input("Enter a number: "))
28 print(f"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
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED 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”

```

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         else:
45             return n * factorial_recursive(n - 1)
46
47 number = int(input("Enter a number: "))
48 print("Iterative Factorial is:", factorial_iterative(number))
49 print("Recursive Factorial is:", factorial_recursive(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: 4
Iterative Factorial is: 24
Recursive Factorial is: 24
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>

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