

Lab Assignment 1.2 – AI Assisted Coding

Harika Jupaka

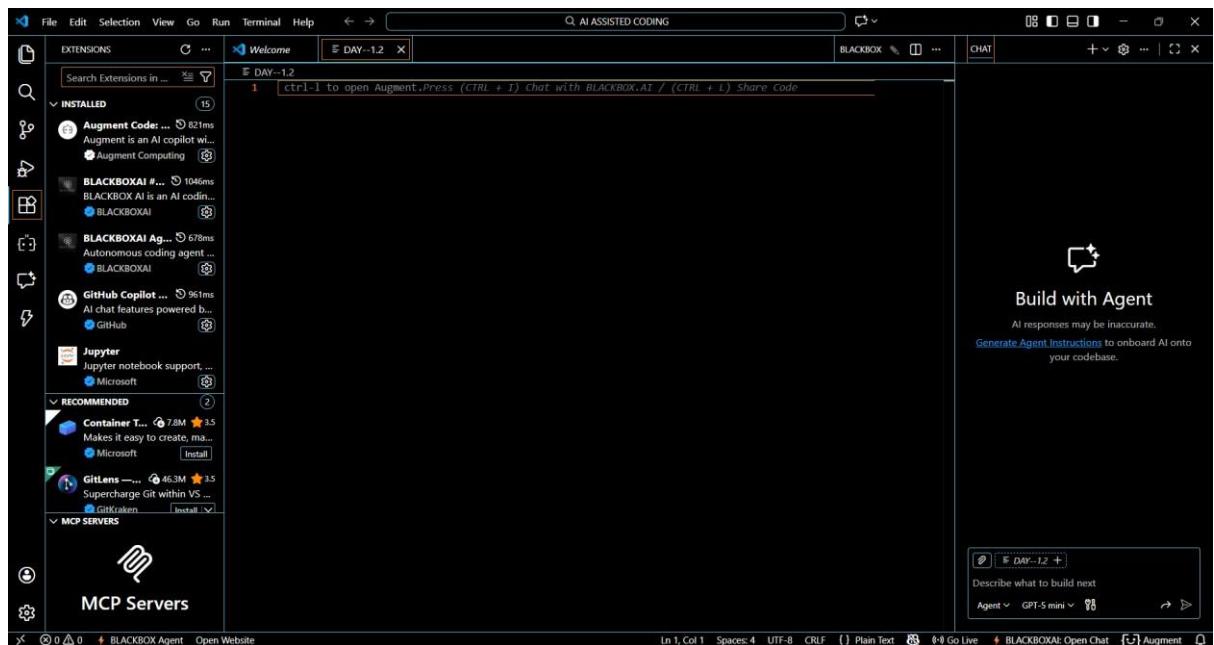
2403A51L31

B:52

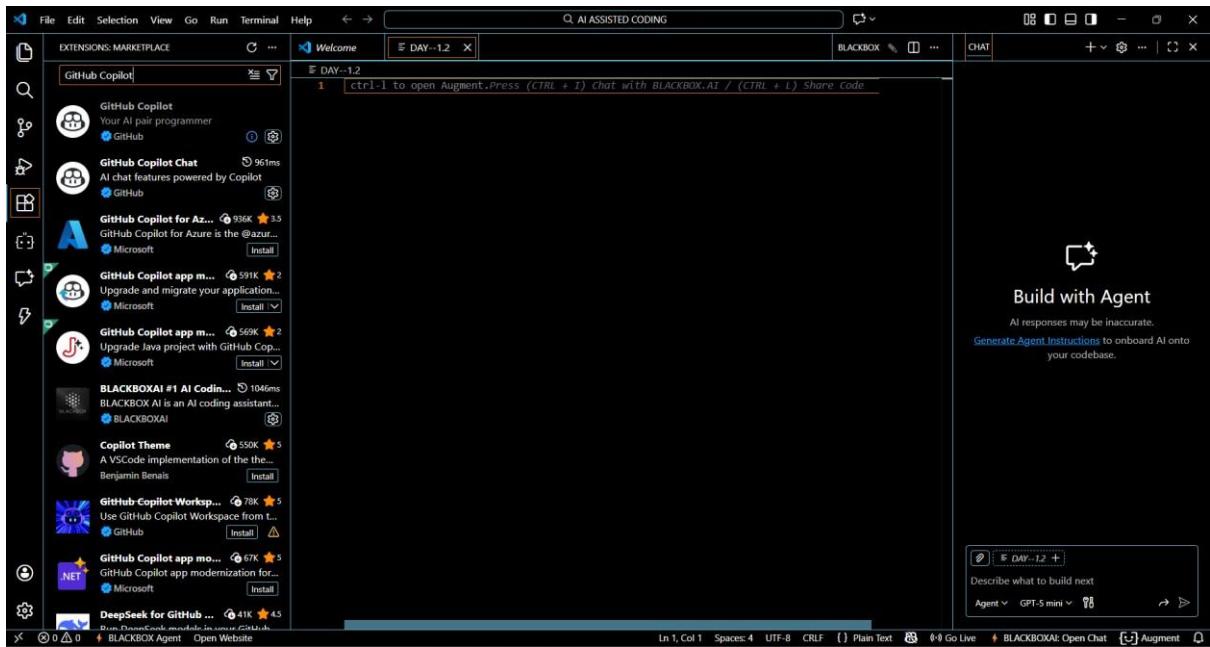
Task 0: GitHub Copilot Installation & Configuration

Steps Followed:

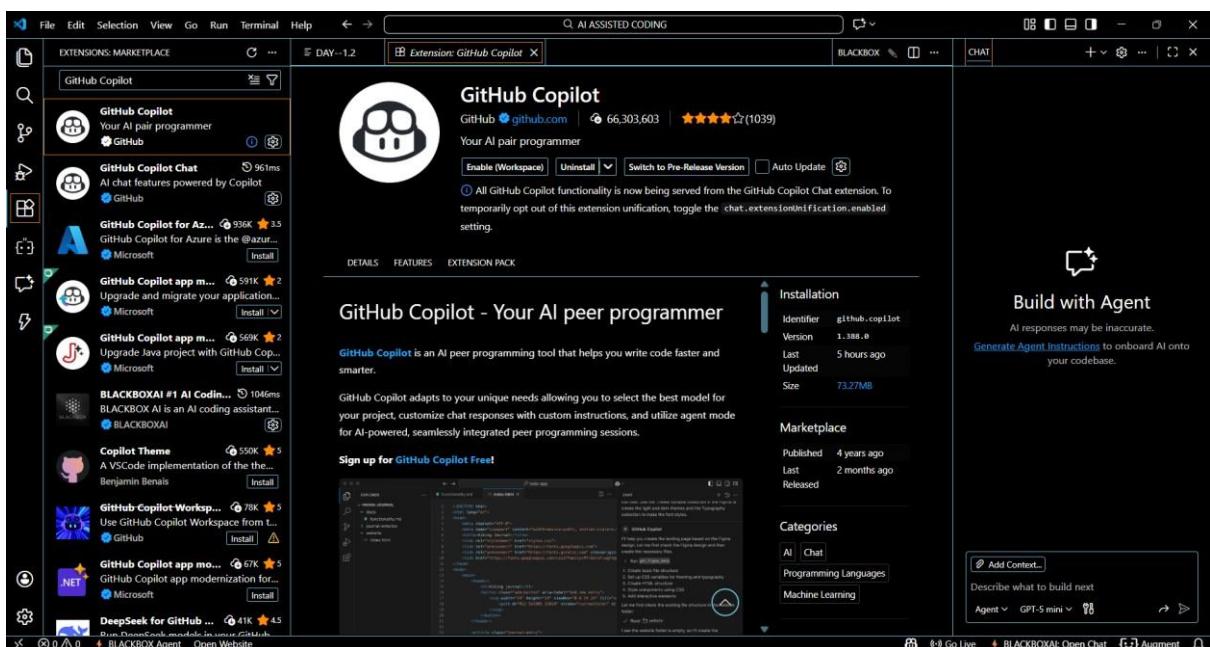
1. Installed Visual Studio Code
2. Opened Extensions Marketplace



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 Microsoft Visual Studio Code interface with the "AI ASSISTED CODING" extension active. The "EXPLORER" sidebar shows a file named "DAY-1.2.py". The main code editor window 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 to calculate its factorial: "))
4 factorial = 1
5
6   < > Accept Word + Rightbrace ...
7 print(f"The factorial of {n} is {factorial}")
```

A tooltip from the AI extension is visible over the loop structure, suggesting "Accept Word + Rightbrace ...". The terminal below shows the execution of the script and its output:

```
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 to calculate its factorial: 4
The factorial of 4 is 24
○ PS C:\Users\srarik\OneDrive\Desktop\AI ASSISTED CODING>
```

The status bar at the bottom indicates "Ln 7, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.9 (Microsoft Store) Go Live BLACKBOX: Open Chat Augment".

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 Microsoft Visual Studio Code interface with the "AI CODING" extension active. The "EXPLORER" sidebar shows a file named "recursion.py". The main code editor window contains the following Python code:

```
1 n = int(input("Enter a number: "))
2 sum_total = 0
3
4 for i in range(1, n + 1):
5     sum_total += i
6
7 print(f"Sum of numbers from 1 to {n} is: {sum_total}")
```

The terminal below shows the execution of the script and its 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
Sum of numbers from 1 to 6 is: 21
PS C:\Users\user\Desktop\Hari\AI coding>
```

The status bar at the bottom indicates "Ln 7, Col 55 Spaces: 4 UTF-8 CRLF Python 3.13.7 Go Live".

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)
```

In the Terminal tab, the command 'python DAY-1.2.py' is run, and the output is:

```
PS C:\Users\srarik\OneDrive\Desktop\AI ASSISTED CODING> python DAY-1.2.py
Enter a number: 2
Factorial is: 2
PS C:\Users\srarik\OneDrive\Desktop\AI ASSISTED CODING>
```

The interface includes a sidebar for 'Build with Agent' and a bottom status bar with various icons and information.

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

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** AI ASSISTED CODING
- Explorer:** Shows a file named DAY-1.2.py.
- Code Editor:**

```

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}")

```
- Terminal:**

```

PS C:\Users\srilk\OneDrive\Desktop\AI ASSISTED CODING & C:/Users/srik/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/srik/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\srilk\OneDrive\Desktop\AI ASSISTED CODING>

```
- Bottom Status:** Ln 11, Col 1, Spaces: 4, UTF-8, CRLF, Python 3.13.9 (Microsoft Store), Go Live, BLACKBOXAI: Open Chat, Augment.
- Right Sidebar:**
 - Build with Agent:** A button with a circular arrow icon.
 - A note: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase."
 - An input field: "Describe what to build" with a "DAY-1.2.py +" button.

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”

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”

File Edit Selection View Go Run Terminal Help ← → AI ASSISTED CODING BLACKBOX

EXPLORER ... DAY-1.2.py

AI ASSISTED CODING DAY-1.2.py

```
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 number = int(input("Enter a number:"))
46 print("Iterative Factorial is:", factorial_iterative(number))
47 print("Recursive Factorial is:", factorial_recursive(number))
48
49
```

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
- Iterative Factorial is: 24
- Recursive Factorial is: 24

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

DAY-1.2.py +

Describe what to build

< > OUTLINE < > TIMELINE

BLACKBOX Agent Open Website

In 48, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.9 (Microsoft Store) ⌂ Go Live BLACKBOX: Open Chat

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