

Lab Assignment 1.5– AI Assisted Coding

D. Divya Sri

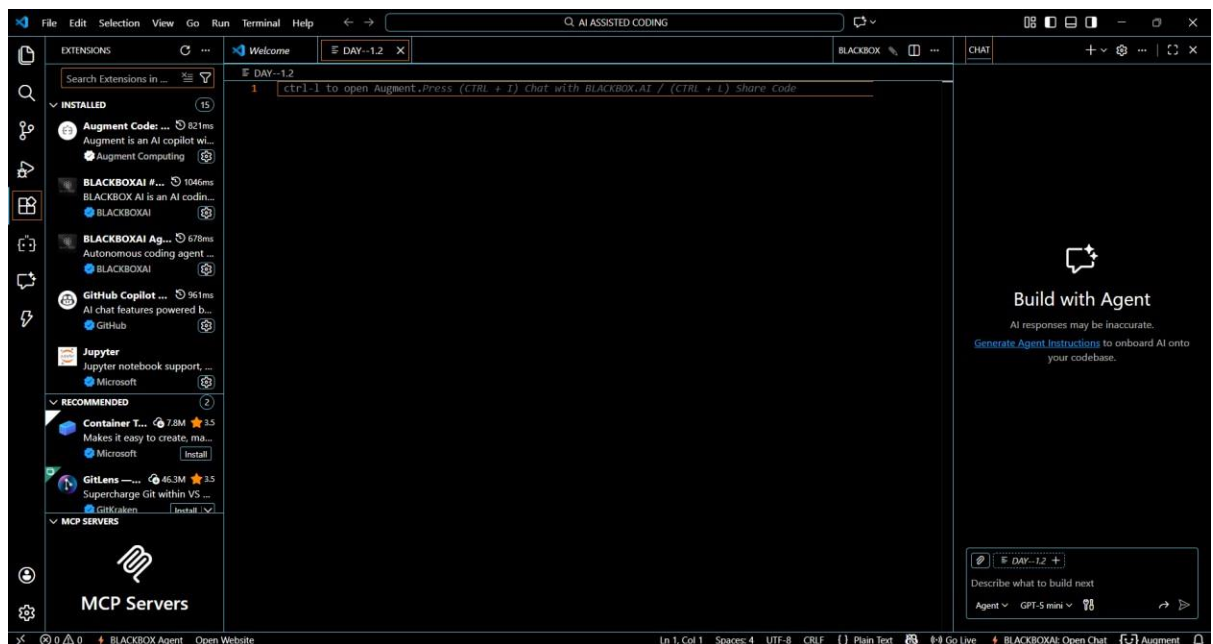
2303A51323

Batch:19

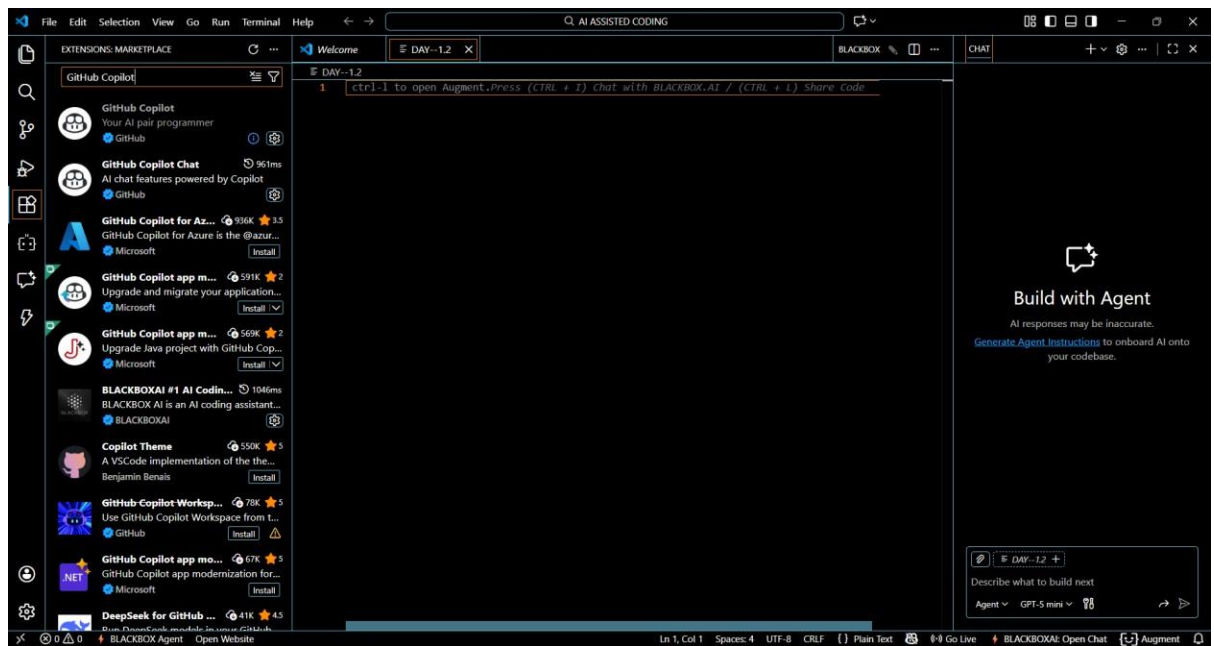
Task 0: GitHub Copilot Installation & Configuration

Steps Followed:

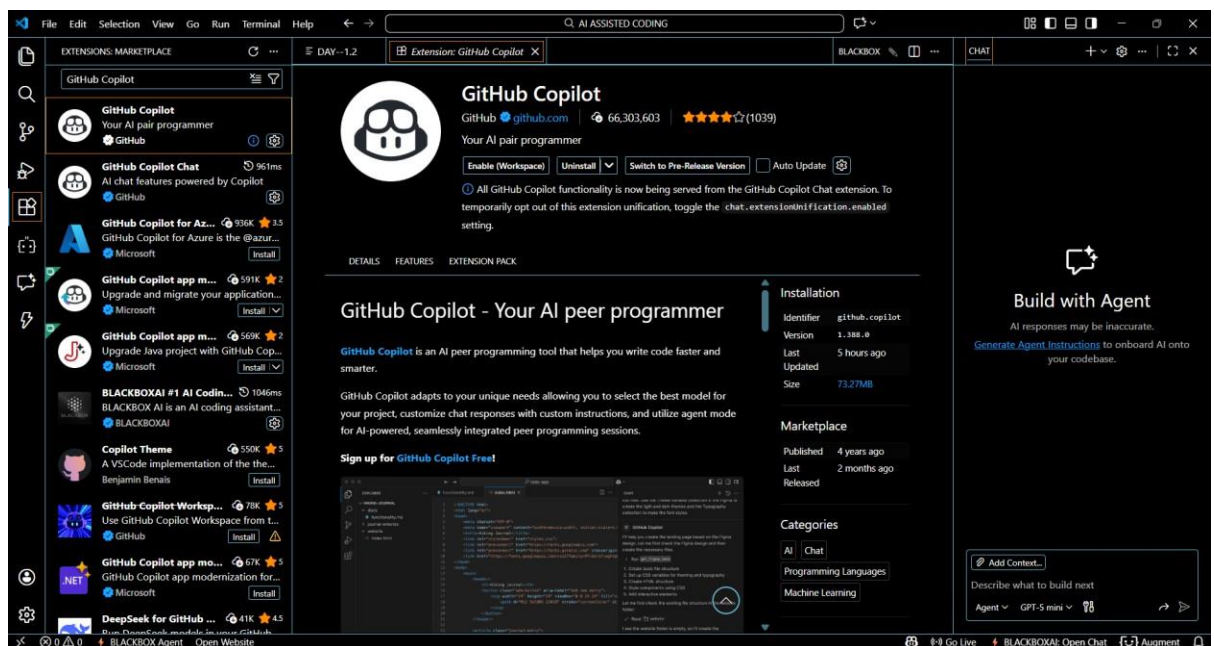
1. Installed **Visual Studio Code**
2. Opened **Extensions Marketplace**



3. Searched for **GitHub Copilot**



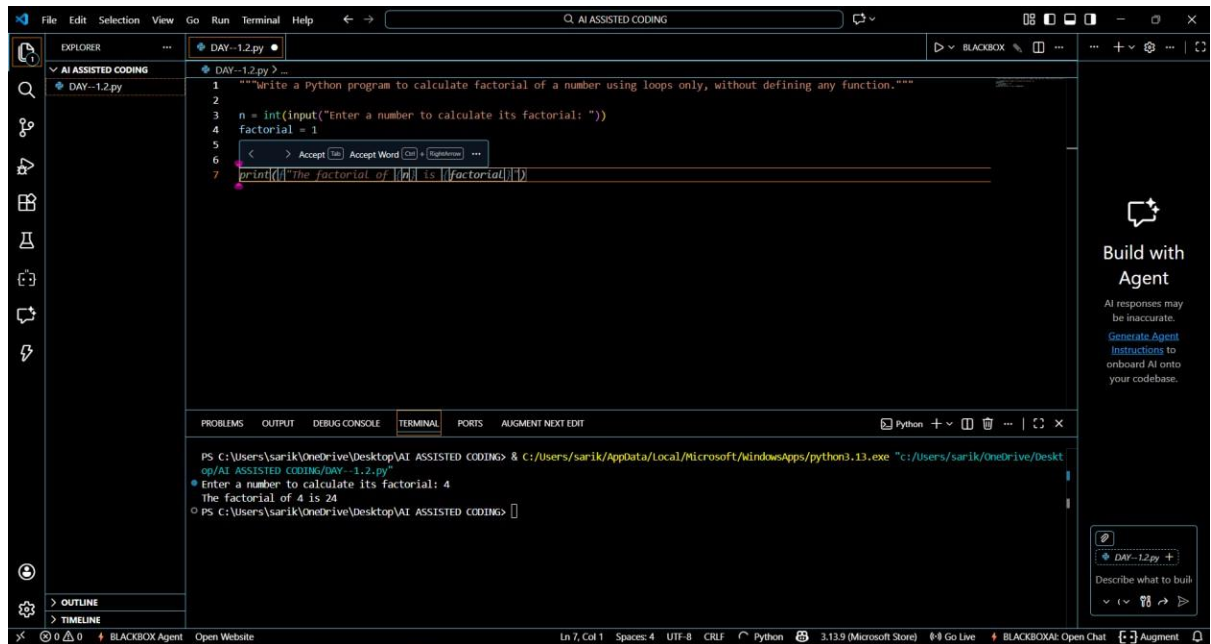
4. Clicked Install



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 (String Reversal Without Functions)

Prompt Used: "Generate a python code to reverse a given string which accepts the user input and does not use any user-defined functions."

```
19 #task1
20 #prompt: generate a python code to reverse a given string which accepts the user input
21 # and does not use any user-defined functions
22 # Reverse string without slicing
23 string=input()
24 reversed_string = ""
25 for char in string:
26     reversed_string = char + reversed_string
27 print(reversed_string)
28
29 #sample input: hello
30 #sample output : olleh
31
```

Expected Output Explanation

The program successfully reverses the input string by iterating through the characters in reverse order using a loop. Since no user-defined functions or built-in reversal methods are used, the logic remains fully procedural and easy to trace step by step.

Observations

GitHub Copilot accurately generated the required logic based on the provided comments. The generated code follows all constraints and produces correct output. This approach is suitable for small, single-use programs; however, it lacks reusability and scalability for larger applications.

Task 2: Efficiency & Logic Optimization (Readability Improvement)

Original Code:

```
19  #task1
20  #prompt: generate a python code to reverse a given string which accepts the user input
21  # and does not use any user-defined functions
22  # Reverse string without slicing
23  string=input()
24  reversed_string = ""
25  for char in string:
26      reversed_string = char + reversed_string
27  print(reversed_string)
28
29  #sample input: hello
30  #sample output : olleh
31
32
```

Prompt Used: "Simplify this string reversal code"

```
33 # Task 2: String reversal
34 #prompt: simplify this string reversal code
35 user_input = input("Enter a string to reverse: ")
36 reversed_input = user_input[::-1]
37 print(reversed_input)
38 #sample input: hello
39 #sample output : olleh
40
```

Expected Output Explanation

The original code reverses the string using a loop and an extra variable. It works, but it is a little long and harder to read.

The optimized code uses slicing to reverse the string in one line. This makes the code shorter, cleaner, and easier to understand.

Observation

GitHub Copilot gives correct code. When we ask it to optimize, it gives a better and simpler solution. AI helps in coding, but we should always check the code.

Task 3: Modular Design Using AI Assistance (String Reversal Using Functions)

Prompt Used: "Write a Python function to reverse a string with comments"

```
42 #task3
43 #prompt:Write a Python function to reverse a string with comments
44 def reverse_string(text):
45     """
46     This function returns the reversed version of the input string
47     """
48     return text[::-1]
49
50 user_input = input("Enter a string: ")
51 result = reverse_string(user_input)
52 print("Reversed string:", result)
53
54 #output
55 #Enter a string: Copilot
56 #Reversed string: tolip oC
57
```

Expected Output Explanation

The function `reverse_string()` takes a string as input and returns the reversed string using slicing. The main program calls this function and prints the result. Comments are added to explain what the function does.

Observation

Using a function makes the code easy to reuse, easy to understand, and easy to maintain. GitHub Copilot correctly generated a clean and readable function when given a clear prompt.

Task 4: Comparative Analysis – Procedural vs Modular Approach (With vs Without Functions)

Comparison Table

Aspect	Without Functions (Procedural)	With Functions (Modular)
Code Clarity	Logic is mixed in main code	Code is well organized
Reusability	Cannot be reused easily	Can be reused anywhere
Debugging Ease	Debugging is harder	Debugging is easier
Large-Scale Use	Not suitable	Suitable

Short Analytical Report

The procedural approach without functions is simple and easy to write for small programs. However, the code becomes difficult to manage and reuse as the program grows.

The modular approach using functions improves clarity and reusability. It makes debugging easier and is more suitable for large-scale applications. Therefore, using functions is the better design choice for real-world and collaborative software development.

Conclusion

Function-based (modular) programming is preferred over procedural programming for clean, reusable, and maintainable code.

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to String Reversal)

Prompt Used: "Write a Python program to reverse a string using a loop. Take input from the user. Do not use built-in functions or slicing. Implement logic in the main program"

```
58
59 #task5
60 #prompt: Write a Python program to reverse a string using a loop
61 # Take input from the user
62 # Do not use built-in functions or slicing
63 # Implement logic in the main program
64 text = input("Enter a string: ")
65 reversed_text = ""
66
67 for i in range(len(text) - 1, -1, -1):
68     reversed_text += text[i]
69
70 print("Reversed string:", reversed_text)
71 #sample input: Enter a string: AI Assisted Coding
72 #sample output: Reversed string: gnidoC detsissA IA
73
```

Comparison

- **Execution Flow:**
Loop-based method reverses the string step by step.
Slicing method reverses the string in one line.
- **Time Complexity:**
Both methods have **O(n)** time complexity.
- **Performance for Large Inputs:**
Slicing is faster because it is internally optimized in Python.
- **When to Use:**
Loop-based method is good for learning logic.
Slicing method is better for real applications.

Observation

GitHub Copilot can generate different correct solutions, and choosing the best one depends on readability and performance.