Lab Assignment 1.2 – AI Assisted Coding

GIRUGULA VARSHINI

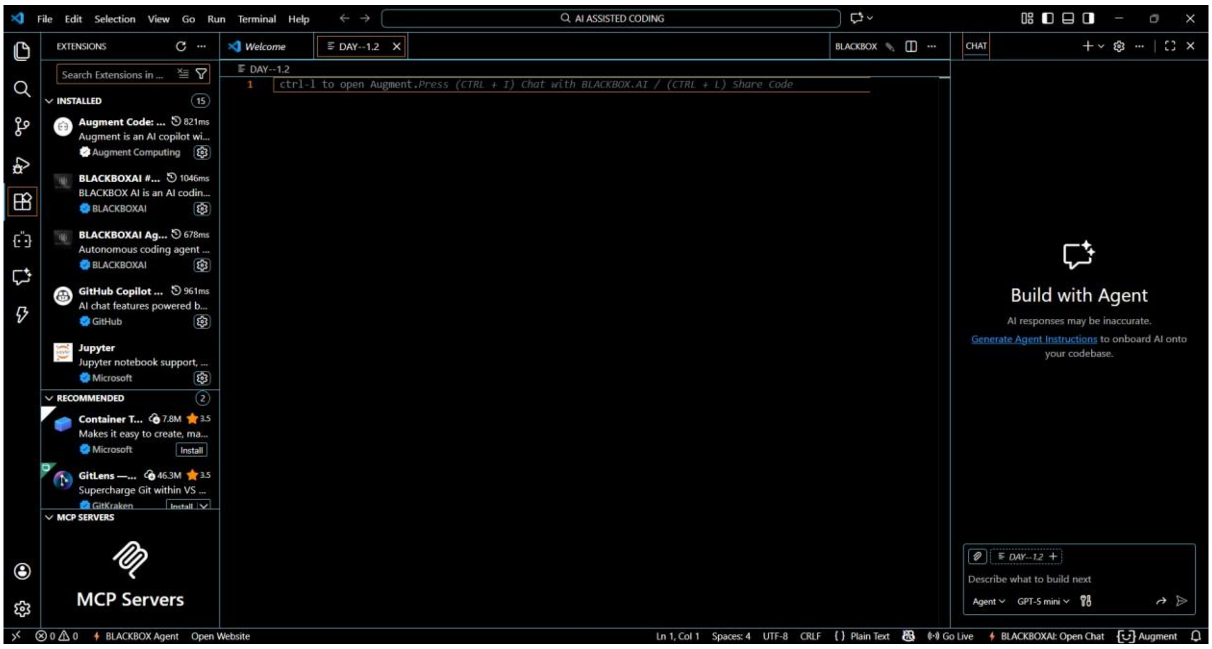
2403A51L14

Batch:51

Task 0: GitHub Copilot Installation & Configuration

Steps Followed:

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

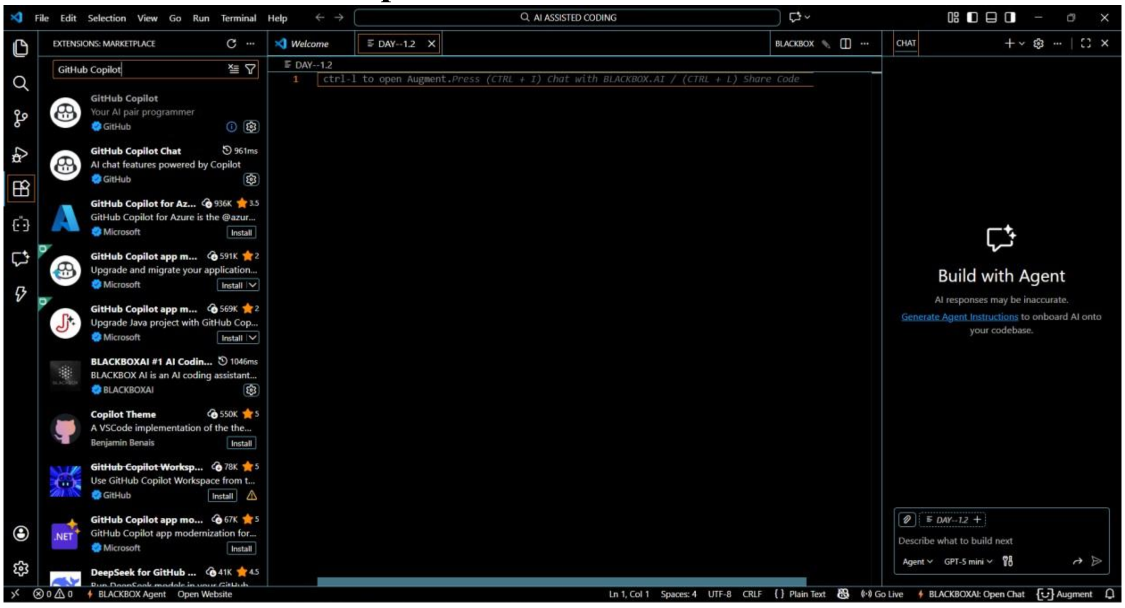


```
ctrl-l to open Augment.Press (CTRL + I) Chat with BLACKBOX.AI / (CTRL + L) Share Code
```

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

3. Searched for **GitHub Copilot**

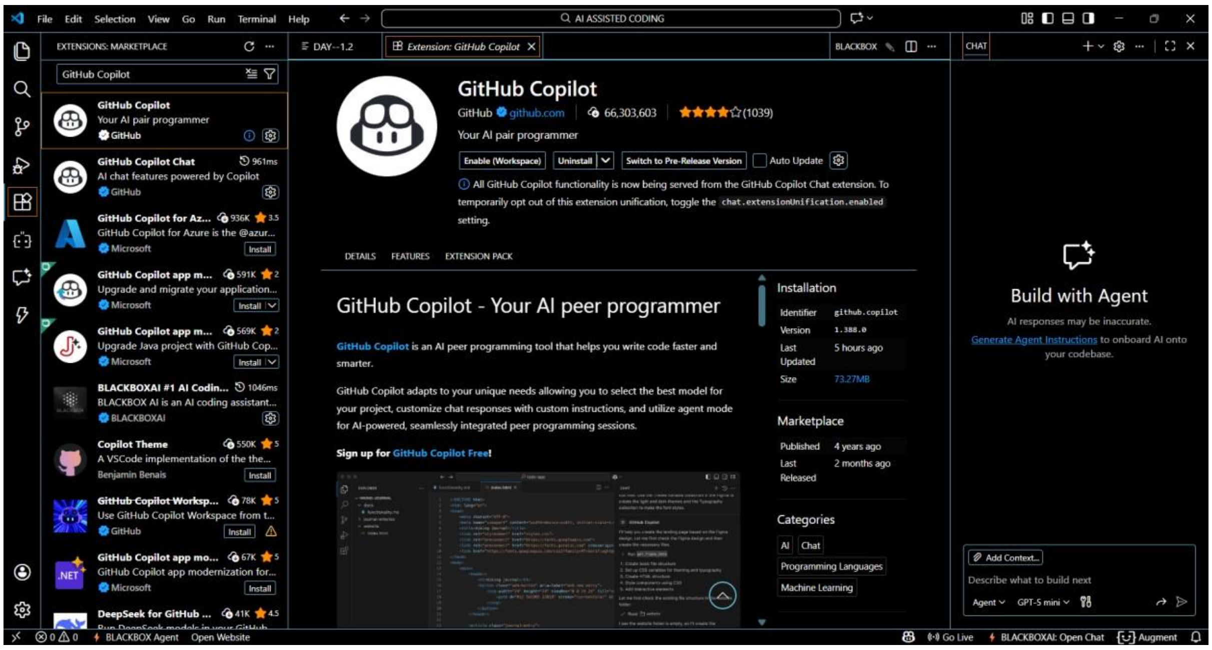


```
ctrl-l to open Augment.Press (CTRL + I) Chat with BLACKBOX.AI / (CTRL + L) Share Code
```

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

4. Clicked **Install**



GitHub Copilot

GitHub github.com | 66,303,603 | ★★★★☆(1039)

Your AI pair programmer

All GitHub Copilot functionality is now being served from the GitHub Copilot Chat extension. To temporarily opt out of this extension unification, toggle the `chat.extensionUnification.enabled` setting.

GitHub Copilot - Your AI peer programmer

GitHub Copilot is an AI peer programming tool that helps you write code faster and smarter.

GitHub Copilot adapts to your unique needs allowing you to select the best model for your project, customize chat responses with custom instructions, and utilize agent mode for AI-powered, seamlessly integrated peer programming sessions.

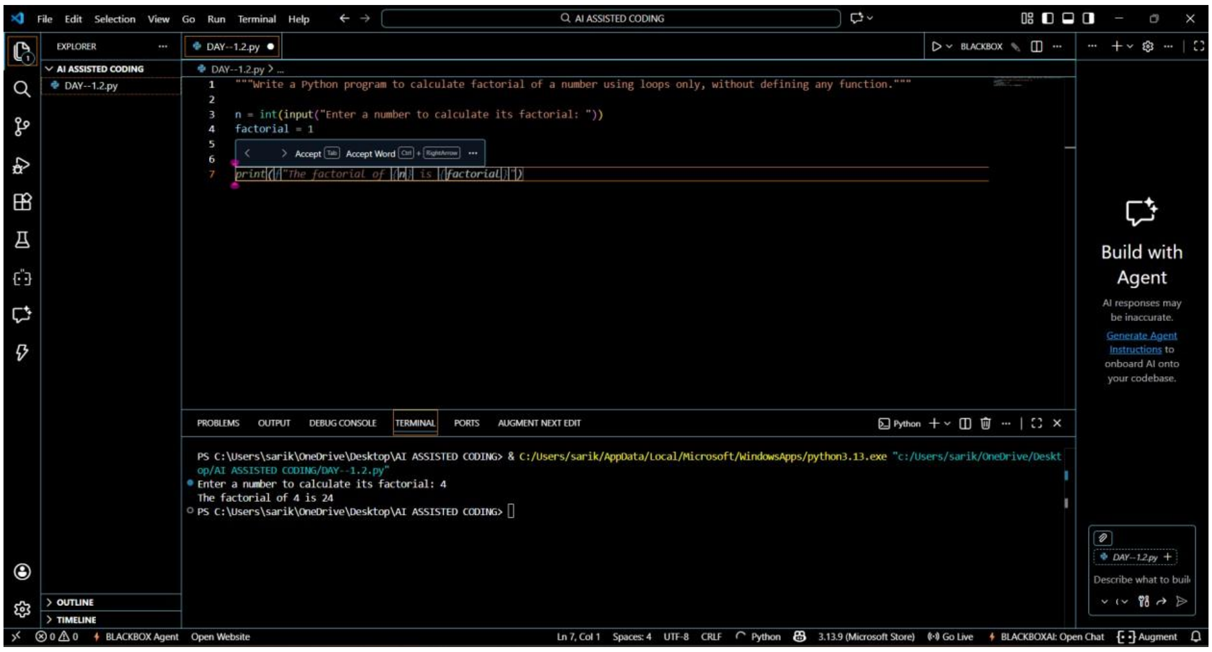
Sign up for GitHub Copilot Free!

Identifier	github.copilot
Version	1.388.0
Last Updated	5 hours ago
Size	73.27MB
Published	4 years ago
Last Released	2 months ago

5. Signed in with **GitHub Account**

6. Enabled Copilot suggestions

7. Verified Copilot inline suggestions in Python file

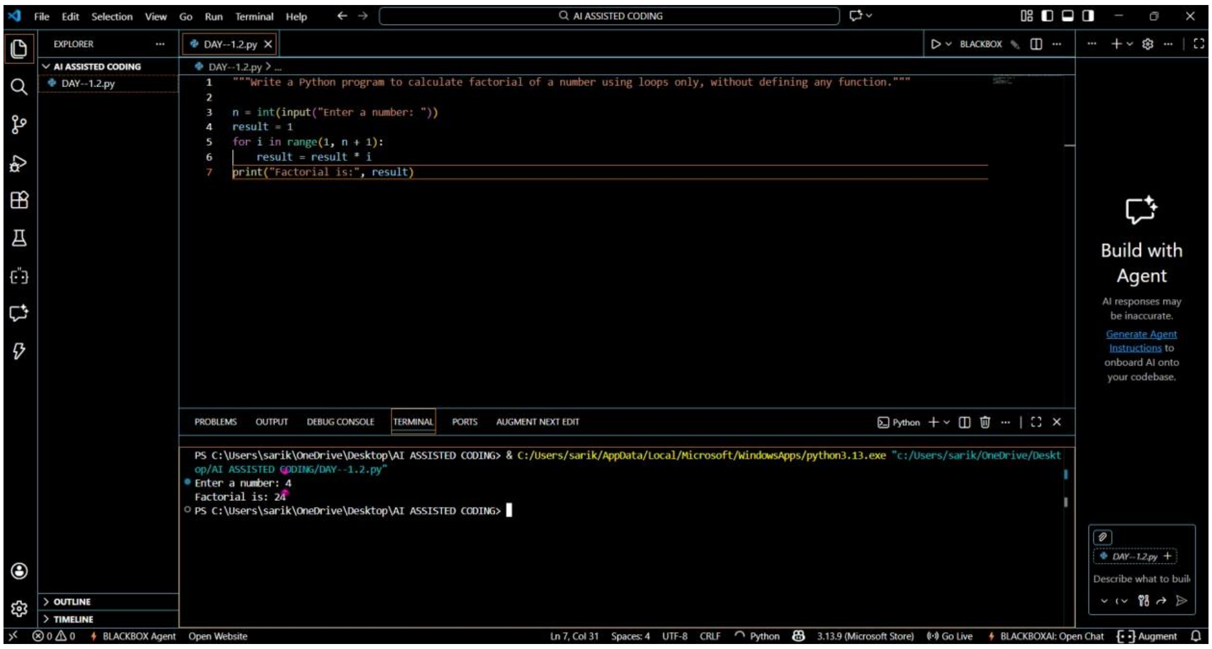


```python
"""Write a Python program to calculate factorial of a number using loops only, without defining any function."""

n = int(input("Enter a number to calculate its factorial: "))
factorial = 1


print(f"The 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 to calculate its factorial: 4
The factorial of 4 is 24
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

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.”



```python
"""Write a Python program to calculate factorial of a number using loops only, without defining any function."""

n = int(input("Enter a number: "))
result = 1
for i in range(1, n + 1):
    result = result * i
print("Factorial is:", result)
```

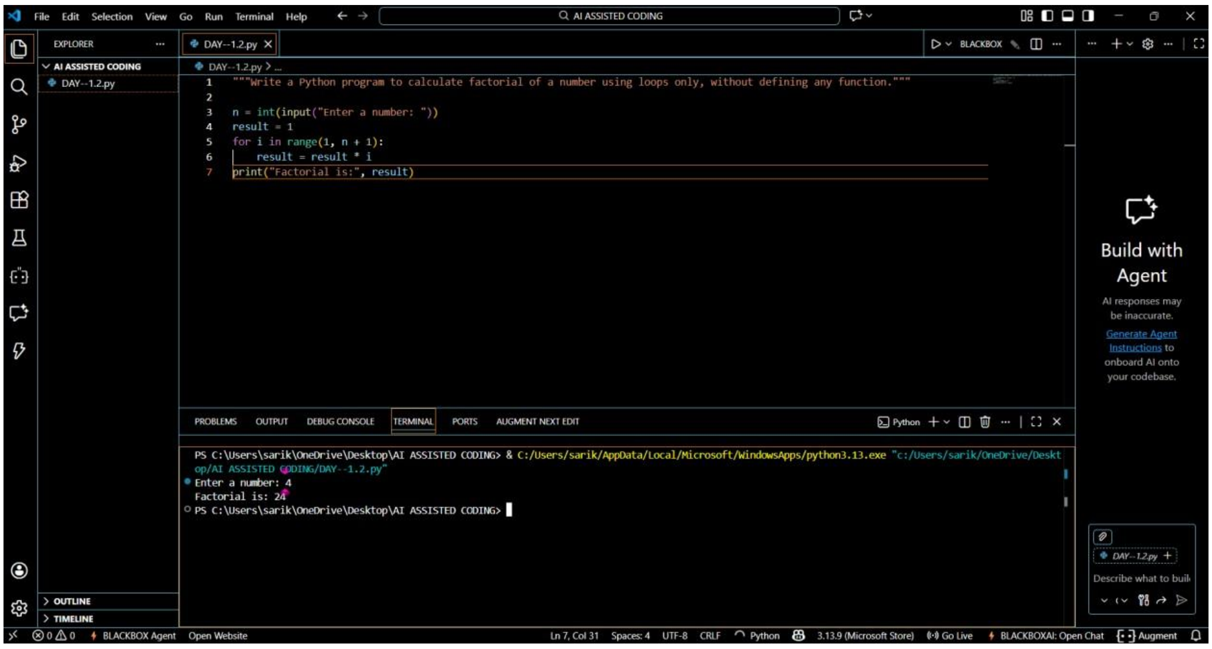
```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/sarik/OneDrive/Deskt
op/AI ASSISTED CODING/DAY--1.2.py"
Enter a number: 4
Factorial is: 24
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

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

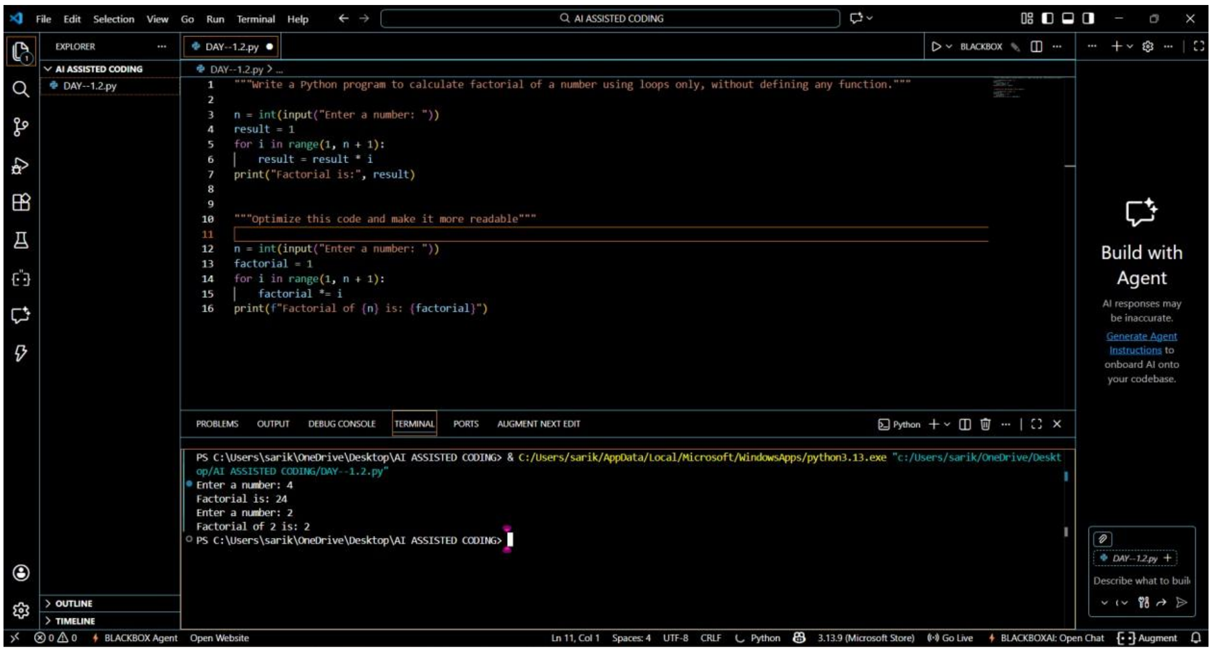
Code:



```python
"""Write a Python program to calculate factorial of a number using loops only, without defining any function."""

n = int(input("Enter a number: "))
result = 1
for i in range(1, n + 1):
    result = result * i
print("Factorial is:", result)
```

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/sarik/OneDrive/Deskt
op/AI ASSISTED CODING/DAY--1.2.py"
Enter a number: 4
Factorial is: 24
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

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



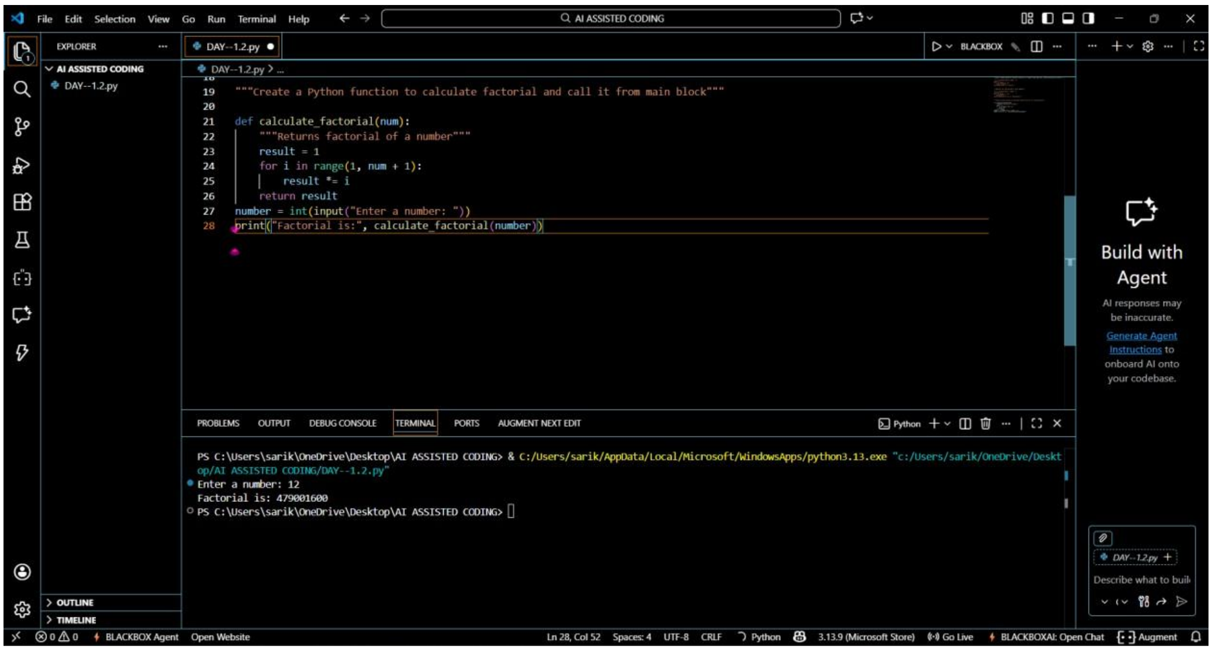
```python
"""Write a Python program to calculate factorial of a number using loops only, without defining any function."""

n = int(input("Enter a number: "))
result = 1
for i in range(1, n + 1):
    result = result * i
print("Factorial is:", result)


"""Optimize this code and make it more readable"""

n = int(input("Enter a number: "))
factorial = 1
for i in range(1, n + 1):
    factorial *= i
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/Deskt
op/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”



```python
"""Create a Python function to calculate factorial and call it from main block"""

def calculate_factorial(num):
    """Returns factorial of a number"""
    result = 1
    for i in range(1, num + 1):
        result *= i
    return result
number = int(input("Enter a number: "))
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
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

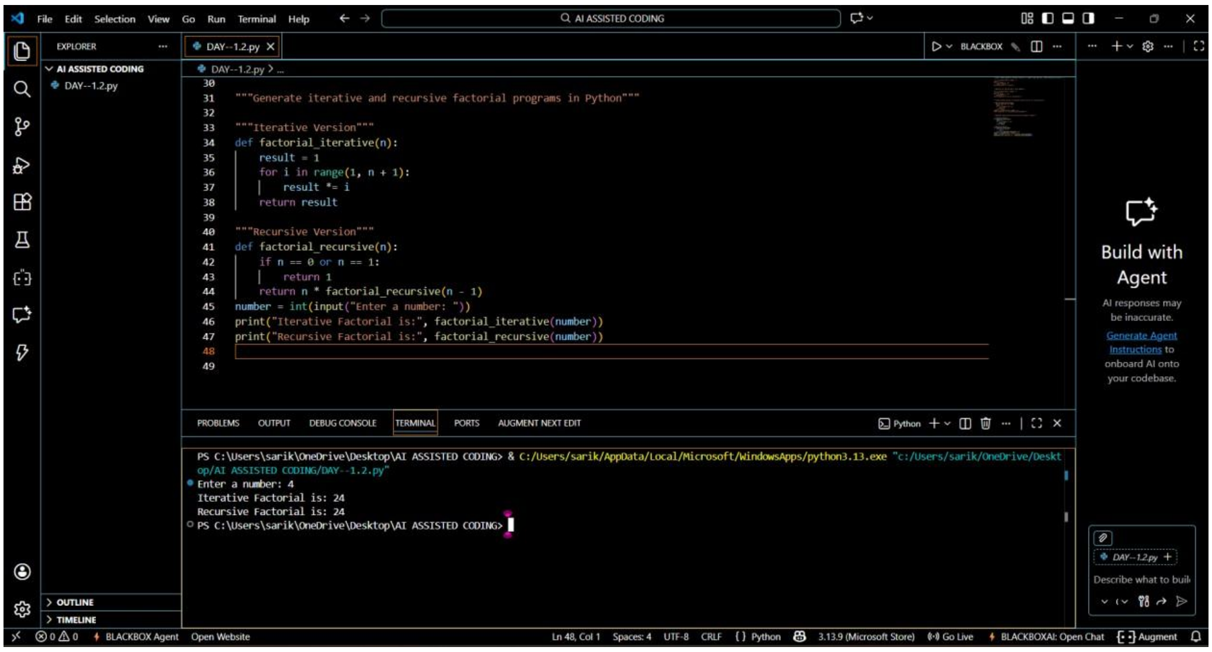
Criteria	**Without Function**	**With Function**
Logic Clarity	Moderate	High
Reusability	No	Yes
Debugging Ease	Difficult	Easy
Large Project Suitability	Poor	Excellent
AI Dependency Risk	Higher	Lower

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”



```python
"""Generate iterative and recursive factorial programs in Python"""

"""Iterative Version"""
def factorial_iterative(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

"""Recursive Version"""
def factorial_recursive(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial_recursive(n - 1)
number = int(input("Enter a number: "))
print("Iterative Factorial is:", factorial_iterative(number))
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/Deskt
op/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