

School of Computer Science and Artificial Intelligence**LAB ASSIGNMENT - 1**

Program : B. Tech (CSE)
Specialization :
Course Title : AI Assisted coding
Course Code : 23CS002PC304
Semester : II
Academic Session : 2025-2026
Name of Student : K. Deepak
Enrollment No. : 2303A51T02
Batch No. : 52
Date : 06-01-2026

Submission Instructions:

(All instructions should be followed strictly to avoid deduction of marks)

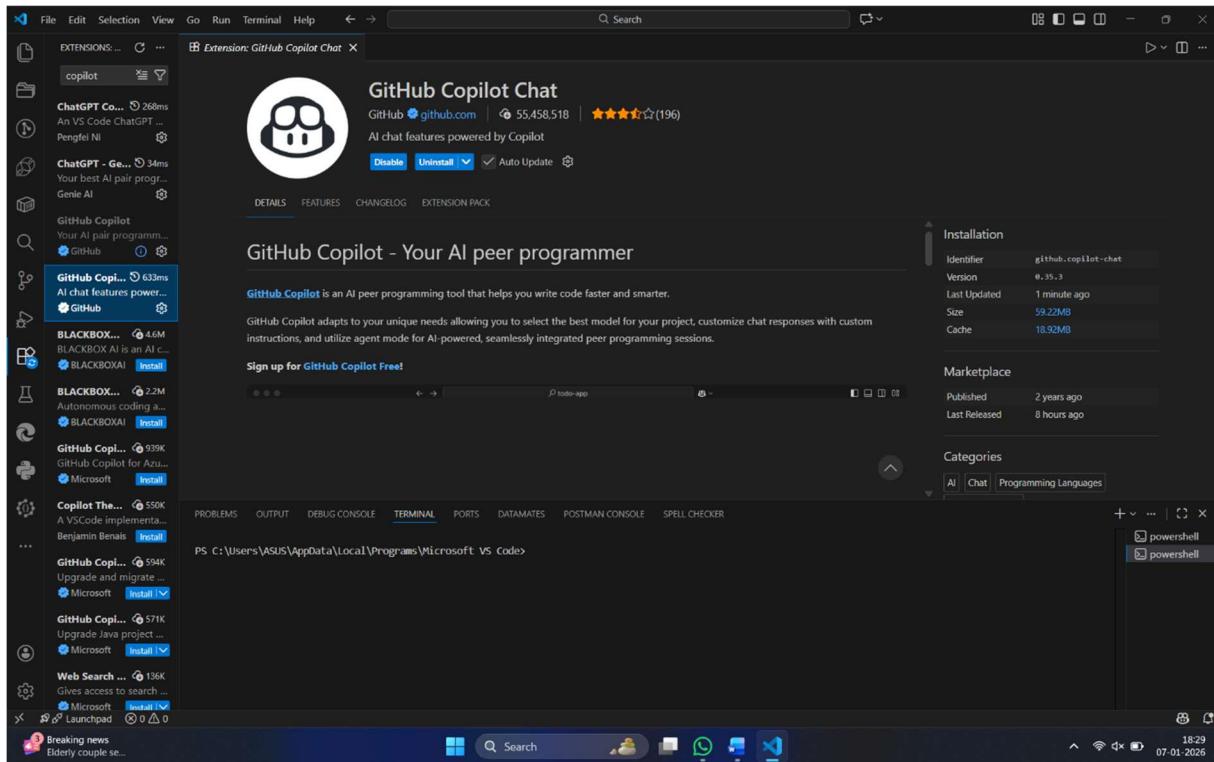
1. Use the same file to complete the assignment and don't change the settings.
2. Minimum 10 screen shots of your account should be taken to showcase your work.
3. **File Format:**

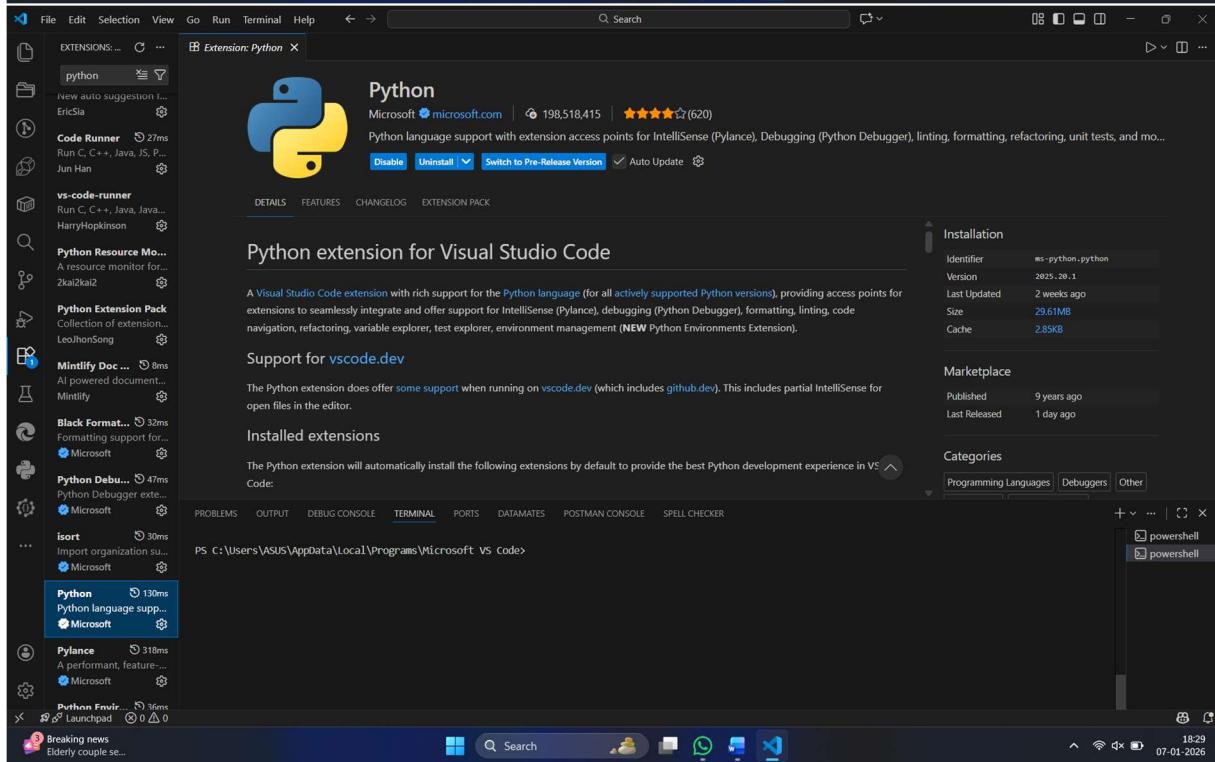
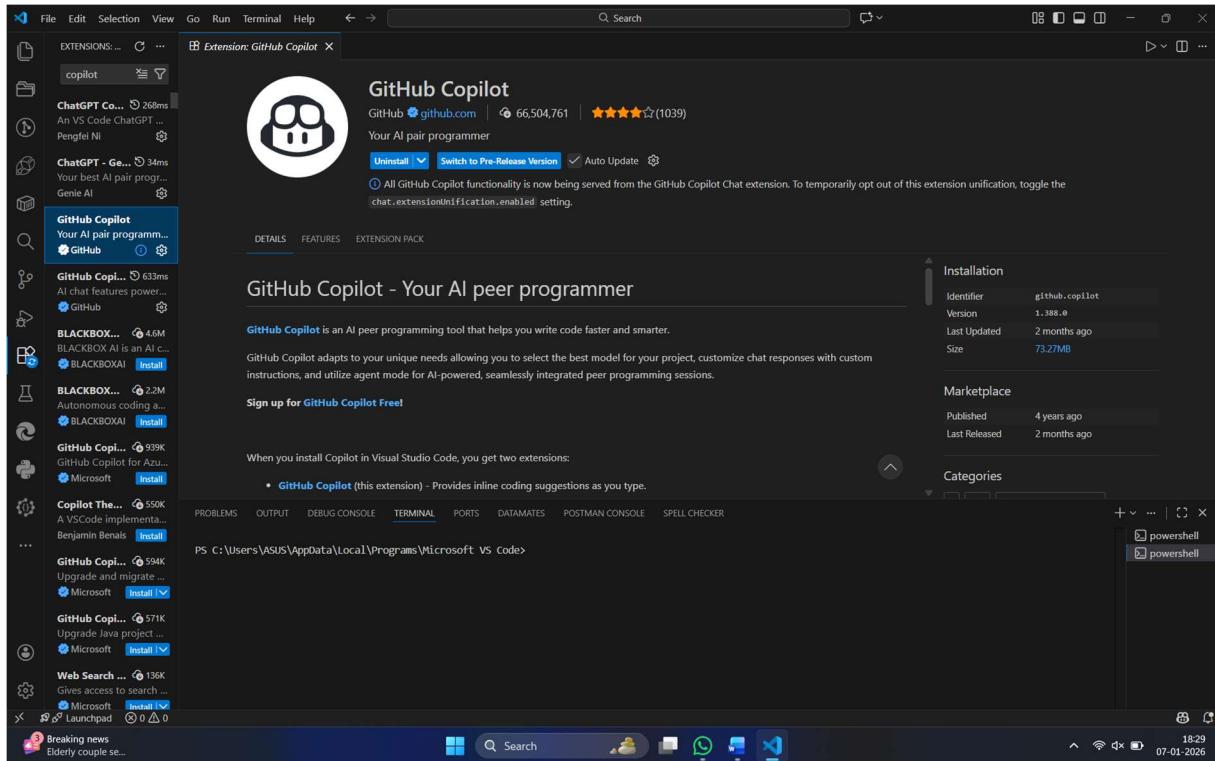
Submit your assignment as a PDF document (pdf). Ensure the file is named according to the following convention:

BNo_StudentName_SESD_A1.

Sample: B10_Rohit_22A523421_A1

4. Fill all the entries mentioned on top section.
5. Mention your AWS Academy Virtual Lab Account details as shown in the next page.
6. **Don't write on this page.**
7. All answers should be answered from next page only.

Submission Starts here**OUTPUT :****SCREENSHOTS:****Task 0: Install and configure GitHub Copilot in VS Code. Take screenshots of each step.**



Task1: Task Description

Use GitHub Copilot to generate a Python program that computes a mathematical product-based value (factorial-like logic) directly in the main execution flow, without using any user-defined functions.



The screenshot displays two instances of Microsoft Visual Studio Code running side-by-side. Both instances have a dark theme.

Top Window:

- Terminal:** Shows the command `C:\> python > aicode.py` followed by the code:

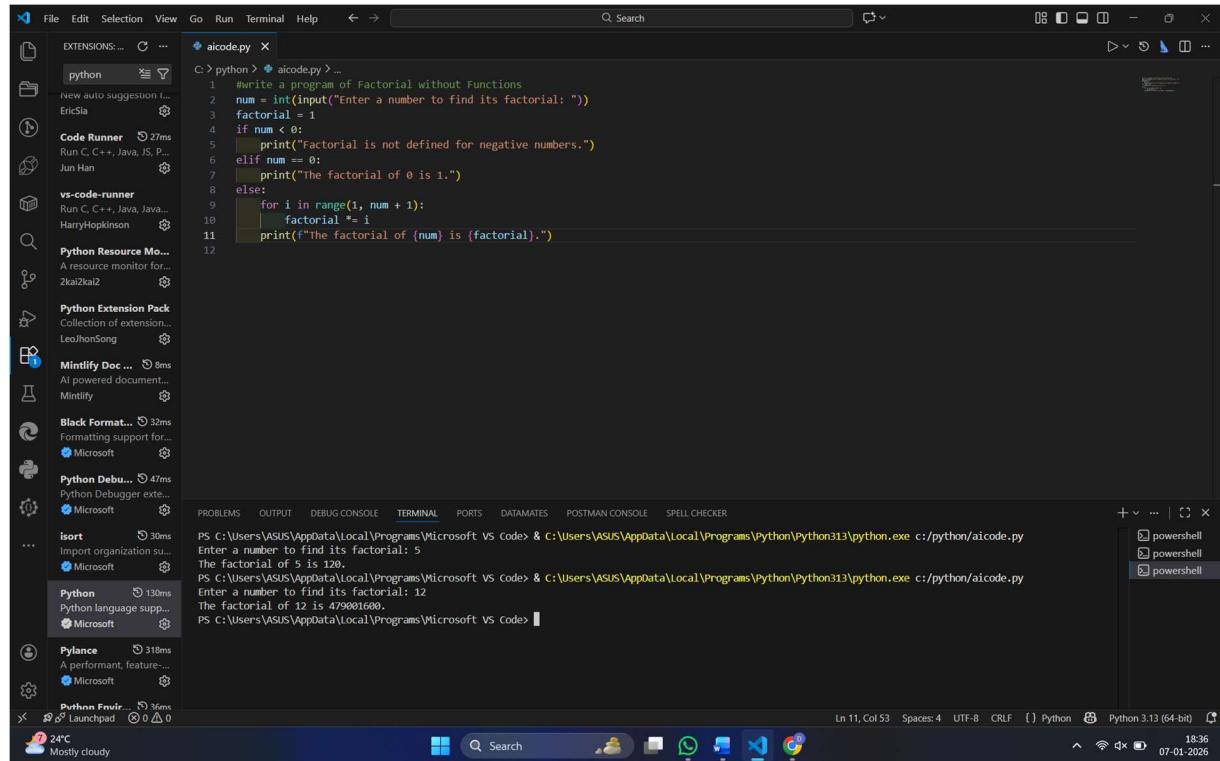
```
1 #write a program of Factorial withoutFunctions
2 num = int(input("Enter a number to find its factorial: "))
```
- Editor:** Shows the same code as the terminal.
- Sidebar (Extensions):** Lists several extensions: EXTENSIONS..., python, Code Runner, vs-code-runner, Python Resource Mo..., Python Extension Pack, Mintlify Doc ..., Black Format..., Python Debu..., isort, Pylance, Python Fruir..., and others.
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, DATAMATES, POSTMAN CONSOLE, and SPELL CHECKER. The TERMINAL tab is selected.
- Right Panel:** Shows a file tree with `powershell` and `powershell` files.

Bottom Window:

- Terminal:** Shows the command `C:\> python > aicode.py` followed by the code:

```
1 #write a program of Factorial without Functions
2 num = int(input("Enter a number to find its factorial: "))
3 factorial = 1
4 if num < 0:
5     print("Factorial is not defined for negative numbers.")
6 elif num == 0:
7     print("The factorial of 0 is 1.")
8 else:
9     for i in range(1, num + 1):
10         factorial *= i
11     print(f"The factorial of {num} is {factorial}.")
```
- Editor:** Shows the same code as the terminal.
- Sidebar (Extensions):** Lists the same set of extensions as the top window.
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, DATAMATES, POSTMAN CONSOLE, and SPELL CHECKER. The TERMINAL tab is selected.
- Right Panel:** Shows a file tree with `powershell` and `powershell` files.

System Status: The taskbar at the bottom shows the date (07-01-2026), time (18:34), battery level (24%), and weather (Mostly cloudy).



```

aicode.py
python aicode.py ...
1 #write a program of Factorial without Functions
2 num = int(input("Enter a number to find its factorial: "))
3 factorial = 1
4 if num < 0:
5     print("Factorial is not defined for negative numbers.")
6 elif num == 0:
7     print("The factorial of 0 is 1.")
8 else:
9     for i in range(1, num + 1):
10         factorial *= i
11     print(f"The factorial of {num} is {factorial}.")
12

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS DATAMATES POSTMAN CONSOLE SPELL CHECKER
PS C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code & C:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe c:/python/aicode.py
Enter a number to find its factorial:
The factorial of 5 is 120.
PS C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code & C:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe c:/python/aicode.py
Enter a number to find its factorial:
The factorial of 12 is 479001600.
PS C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code>

```

- ❖ The Copilot is very helpful because we can generate code by just giving a prompt in Copilot Chat (ctrl + I)
- ❖ The code generated was as requested in the prompt

EXPLANATION :

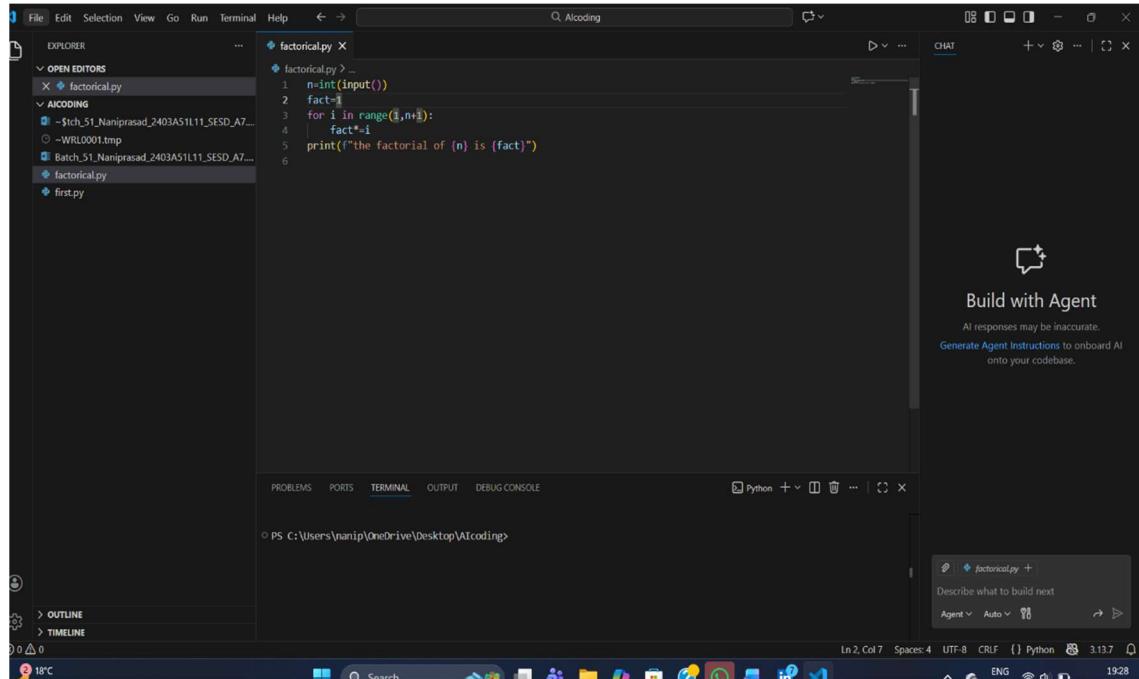
- 1 The program asks the user to enter a number.
- 2 It stores the entered value as an integer.
- 3 The code calculates the factorial by multiplying numbers from 1 to the given number.
- 4 The final multiplied result is stored as the factorial value.
- 5 The program prints the factorial of the given number as output.

TASK - 2

Task Description

Analyze the code generated in Task 1 and use Copilot again to:

- ❖ Reduce unnecessary variables
- ❖ Improve loop clarity
- ❖ Enhance readability and efficiency



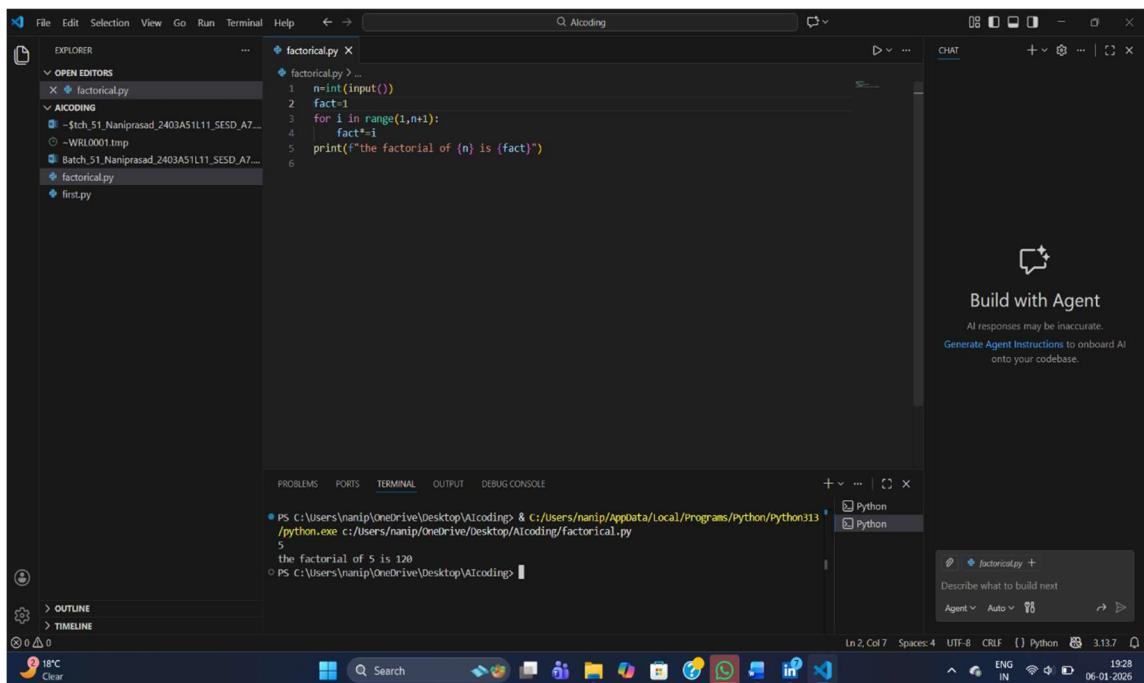
```

File Edit Selection View Go Run Terminal Help <- > Q Alcoding
EXPLORER OPEN EDITORS factorial.py ...
AICODING -$tch_51_Naniprasad_2403A51L11_SESD_A7...
-WRL0001.tmp Batch_51_Naniprasad_2403A51L11_SESD_A7...
factorial.py first.py
factorial.py > ...
1 n=int(input())
2 fact=1
3 for i in range(1,n+1):
4     fact*=i
5 print("the factorial of {} is {}".format(n,fact))
6

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE
PS C:\Users\Naniprasad\OneDrive\Desktop\AIcoding>

```

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.



```

File Edit Selection View Go Run Terminal Help <- > Q Alcoding
EXPLORER OPEN EDITORS factorial.py ...
AICODING -$tch_51_Naniprasad_2403A51L11_SESD_A7...
-WRL0001.tmp Batch_51_Naniprasad_2403A51L11_SESD_A7...
factorial.py first.py
factorial.py > ...
1 n=int(input())
2 fact=1
3 for i in range(1,n+1):
4     fact*=i
5 print("the factorial of {} is {}".format(n,fact))
6

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE
PS C:\Users\Naniprasad\OneDrive\Desktop\AIcoding> & c:/users/nanip/AppData/Local/Programs/Python/Python313/python.exe c:/users/nanip/OneDrive/Desktop/AIcoding/factorial.py
the factorial of 5 is 120
PS C:\Users\Naniprasad\OneDrive\Desktop\AIcoding>

```

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

What was improved?

- Shorter multiplication statement
- **factorial = factorial * i → factorial *= i**
- Removed unnecessary comment

❖ The loop logic is self-explanatory, so the comment was removed.

- ❖ **Readability**
 - Fewer lines and less clutter make the code easier to read.
- ❖ **Maintainability**
 - Cleaner code is easier to modify and debug.
 - Reduced redundancy lowers the chance of mistakes.
- ❖ **Performance**
 - Performance is effectively the same.
- ❖ ***= is marginally optimized at the bytecode level, but the difference is negligible.**

Task3

Task Description

Use GitHub Copilot to generate a modular version of the program by:

- ❖ Creating a user-defined function
- ❖ Calling the function from the main block

```
aicode3.py
#write a program for factorial of a given number using function
```

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

- File Explorer:** On the left, it lists extensions and recent files. Recent files include "aicode3.py", "EricSla", "Code Runner", "vs-code-runner", "Python Resource Monitor", "Python Extension Pack", "Mintlify Doc", "Black Formatter", "Python Debugger", "isort", "Python", "Pylance", and "Python Frantic".
- Code Editor:** The main area displays the code for "aicode3.py":

```
python > aicode3.py > -
1 #write a program for factorial of a given number using function
2 def factorial(n):
3     if n == 0 or n == 1:
4         return 1
5     else:
6         return n * factorial(n - 1)
7 num = int(input("Enter a number to find its factorial: "))
8 result = factorial(num)
9 print(f"The factorial of {num} is {result}")
10
```

- Status Bar:** At the bottom, it shows "Ln 10, Col 1" and "Python 3.13 (64-bit)".
- System Tray:** On the far right, it shows the date "07-01-2026" and system icons.

The screenshot shows the Microsoft Visual Studio Code interface. On the left is the sidebar with various extensions listed. The main area has a code editor with a Python script named `aicode3.py`. The terminal at the bottom shows the script being run and printing the factorial of 6.

```
File Edit Selection View Go Run Terminal Help ← → Q Search EXTENSIONS ... python New auto suggestion ... EricSia Code Runner vs-code-runner Python Resource Mo... Python Extension Pack Mintlify Doc ... Black Format... Python Debub... isort Python Pylance Python PEP8 Launcher EXTENSIONS ... python New auto suggestion ... EricSia Code Runner vs-code-runner Python Resource Mo... Python Extension Pack Mintlify Doc ... Black Format... Python Debub... isort Python Pylance Python PEP8 Launcher python aicode3.py ... #write a program for factorial of a given number using function def factorial(n): if n == 0 or n == 1: return 1 else: return n * factorial(n - 1) num = int(input("Enter a number to find its factorial: ")) result = factorial(num) print(f"The factorial of {num} is {result}") 10 PS C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code & C:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe c:/python/aicode3.py Enter a number to find its factorial: 6 The factorial of 6 is 728 PS C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code>
```

❖ Modularity improves reusability by:

Allowing the factorial() function to be reused in multiple programs without rewriting code.

Improving code organization, where logic is separated from input/output handling.

Supporting scalability, as the same function can be extended or integrated into larger projects.

#Task4

Task Description

Compare the non-function and function-based Copilot-generated programs on the following criteria:

- ❖ Logic clarity
- ❖ Reusability
- ❖ Debugging ease
- ❖ Suitability for large projects
- ❖ AI dependency risk

```
factorial.py
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result

# Get input from user
num = int(input("Enter a number: "))
print(f"Factorial of {num} is: {factorial(num)}")
```

Compare the non-function and function-based Copilot-generated programs on the following criteria:

- Logic clarity
- Reusability

Add Context...

```
PS C:\Users\nanip\OneDrive\Desktop\AIcoding> & C:/Users/nanip/AppData/Local/Programs/Python/python313/python.exe c:/users/nanip/oneDrive/Desktop/AIcoding/factorial.py
Enter a number: 5
Factorial of 5 is: 120
PS C:\Users\nanip\OneDrive\Desktop\AIcoding>
```

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE

Python Python

Describe what to build next

Agent Auto 1946 06-01-2026

```

22     print(f"Factorial of {num} is: {factorial(num)}")
23 
24 # Comparison Analysis
25 print("\n--- Comparison of Approaches ---")
26 print(" Non-function approach:")
27 print(" Logic clarity: Simple but mixed concerns")
28 print(" Reusability: Low - code cannot be reused")
29 print(" Debugging ease: Harder - logic embedded in main flow")
30 print(" Large projects: Poor - code duplication likely")
31 print(" AI dependency: Moderate - straightforward logic")
32 
33 print("\nFunction-based approach:")
34 print(" Logic clarity: High - isolated and documented")
35 print(" Reusability: High - can be imported and used anywhere")
36 print(" Debugging ease: Easy - logic is testable and modular")
37 print(" Large projects: Excellent - follows best practices")
38 print(" AI dependency: Lower - cleaner code requires less AI assistance")
39 
40 print("\nRecommendation: Use function-based approach for production code")

```

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE

PS C:\Users\nanip\OneDrive\Desktop\AIcoding & C:/Users/nanip/AppData/Local/Programs/Python/Python313 /python.exe c:/Users/nanip/OneDrive/Desktop/AIcoding/factorical.py

Factorial of 5 is: 120

PS C:\Users\nanip\OneDrive\Desktop\AIcoding> []

Ln 39, Col 75 Spaces: 4 UTF-8 {} Python 3.13.7 19:46 06-01-2026

```

20     num = int(input("Enter a number: "))
21     print(f"Factorial of {num} is: {factorial(num)}")
22 
23 # Comparison Analysis
24 print("\n--- Comparison of Approaches ---")
25 print(" Non-function approach:")
26 print(" Logic clarity: Simple but mixed concerns")
27 print(" Reusability: Low - code cannot be reused")
28 print(" Debugging ease: Harder - logic embedded in main flow")
29 print(" Large projects: Poor - code duplication likely")
30 print(" AI dependency: Moderate - straightforward logic")
31 
32 print("\nFunction-based approach:")
33 print(" Logic clarity: High - isolated and documented")
34 print(" Reusability: High - can be imported and used anywhere")
35 print(" Debugging ease: Easy - logic is testable and modular")

```

PROBLEMS PORTS TERMINAL OUTPUT DEBUG CONSOLE

PS C:\Users\nanip\OneDrive\Desktop\AIcoding & C:/Users/nanip/AppData/Local/Programs/Python/Python313 /python.exe c:/Users/nanip/OneDrive/Desktop/AIcoding/factorical.py

--- Comparison of Approaches ---

Non-function approach:

- Logic clarity: Simple but mixed concerns
- Reusability: Low - code cannot be reused
- Debugging ease: Harder - logic embedded in main flow
- Large projects: Poor - code duplication likely
- AI dependency: Moderate - straightforward logic

Function-based approach:

- Logic clarity: High - isolated and documented
- Reusability: High - can be imported and used anywhere
- Debugging ease: Easy - logic is testable and modular
- Large projects: Excellent - follows best practices
- AI dependency: Lower - cleaner code requires less AI assistance

Ln 31, Col 11 Spaces: 4 UTF-8 {} Python 3.13.7 19:46 06-01-2026

TASK - 5

Task Description

Prompt Copilot to generate:

An iterative version of the logic

A recursive version of the same logic

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar displays a list of extensions: python, Code Runner, vs-code-runner, Python Resource Monitor, Python Extension Pack, Mintlify Doc, Black Format, Python Debug, isort, Python, Pylance, and Python Envir. The main editor window contains the following Python code:

```
C:\> python > pppp.py
1 #write a python program using recursion for factorial of a number
```

The terminal at the bottom shows the command being run and the resulting output:

```
PS C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code> & C:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe c:/python/aiode3.py
Enter a number to find its factorial: 6
The factorial of 6 is 720
PS C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code>
```

This screenshot shows the same Microsoft Visual Studio Code interface as the first one, but with additional code added to the editor. The code now includes a recursive factorial function and an input prompt:

```
C:\> python > pppp.py
1 #write a python program using recursion for factorial of a number
2 def factorial(n):
3     if n == 0 or n == 1:
4         return 1
5     else:
6         return n * factorial(n - 1)
7 num = int(input("Enter a number to find its factorial: "))
8 result = factorial(num)
9 print(f"The factorial of {num} is {result}")
```

The terminal output remains the same as in the first screenshot.

A screenshot of Microsoft Visual Studio Code (VS Code) interface. The code editor shows a Python script named `pppp.py` with the following content:

```
C:\> python > pppp.py > ...
1 #write a python program using recursion for factorial of a number
2 def factorial(n):
3     if n == 0 or n == 1:
4         return 1
5     else:
6         return n * factorial(n - 1)
7 num = int(input("Enter a number to find its factorial: "))
8 result = factorial(num)
9 print(f"The factorial of {num} is {result}")


```

The terminal below shows the execution of the script and the output:

```
PS C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code> & C:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe c:/python/pppp.py
Enter a number to find its factorial: 4
The factorial of 4 is 24

PS C:\Users\ASUS\AppData\Local\Programs\Microsoft VS Code>
```

The status bar at the bottom right indicates the following information: Ln 9, Col 45, Spaces: 4, UTF-8, CRLF, Python, Python 3.13 (64-bit), 2008, 07-01-2026.