

School of Computer Science and Artificial Intelligence

Lab Assignment # 1.2

Program : B. Tech (CSE)

Specialization :AIML

Course Title : AI Assisted Coding

Course Code : 23CS002PC304

Semester : VI

Academic Session : 2025-2026

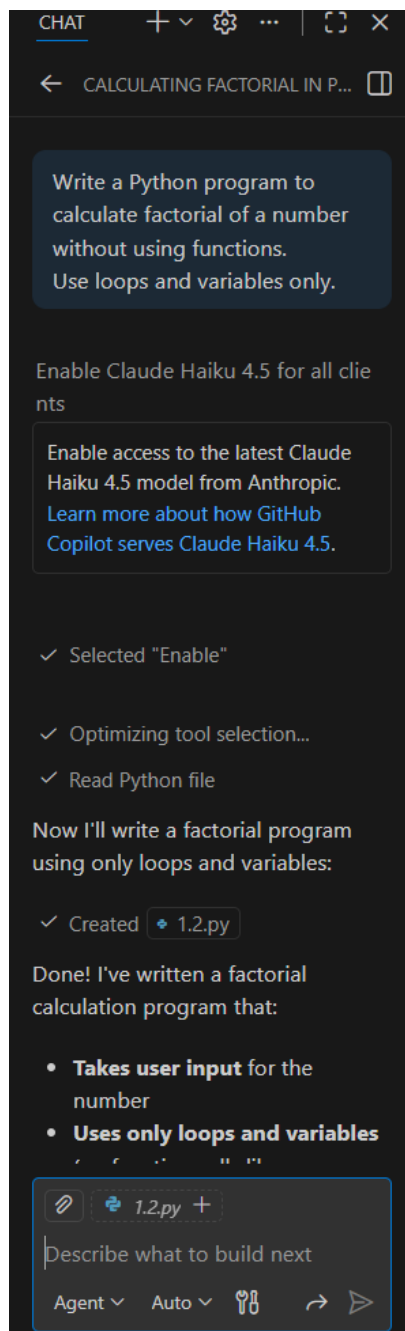
Name of Student : D.SRIRAM

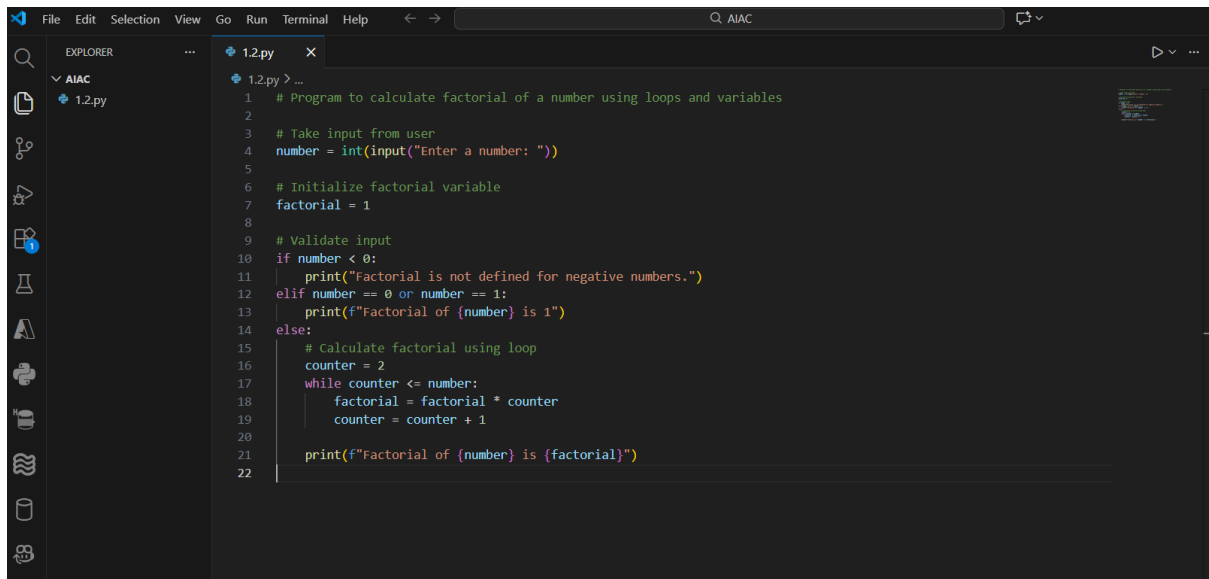
Enrollment No. : 2303A52106

Batch No. : 33

Date :09/01/26

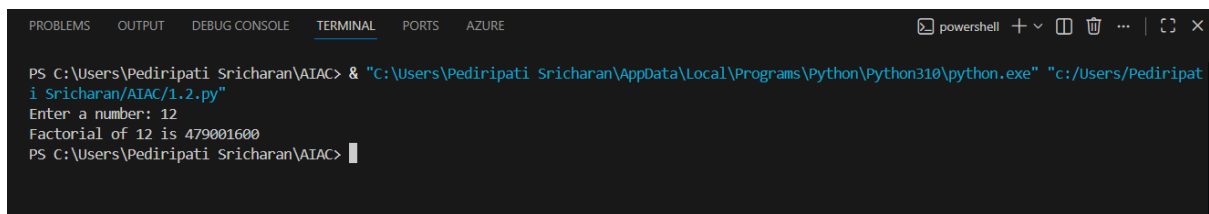
TASK _01





The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor in the center. The file explorer shows a folder named 'AIAC' containing a file '1.2.py'. The code editor displays the following Python code:

```
1 # Program to calculate factorial of a number using loops and variables
2
3 # Take input from user
4 number = int(input("Enter a number: "))
5
6 # Initialize factorial variable
7 factorial = 1
8
9 # Validate input
10 if number < 0:
11     print("Factorial is not defined for negative numbers.")
12 elif number == 0 or number == 1:
13     print(f"Factorial of {number} is 1")
14 else:
15     # Calculate factorial using loop
16     counter = 2
17     while counter <= number:
18         factorial = factorial * counter
19         counter = counter + 1
20
21     print(f"Factorial of {number} is {factorial}")
22
```



The screenshot shows a PowerShell terminal window with the following output:

```
PS C:\Users\Pediripati Sricharan\AIAC> & "C:\Users\Pediripati Sricharan\AppData\Local\Programs\Python\Python310\python.exe" "c:\Users\Pediripati Sricharan\AIAC\1.2.py"
Enter a number: 12
Factorial of 12 is 479001600
PS C:\Users\Pediripati Sricharan\AIAC>
```

EXPLANATION:

GitHub Copilot helped me quickly generate a working factorial program.

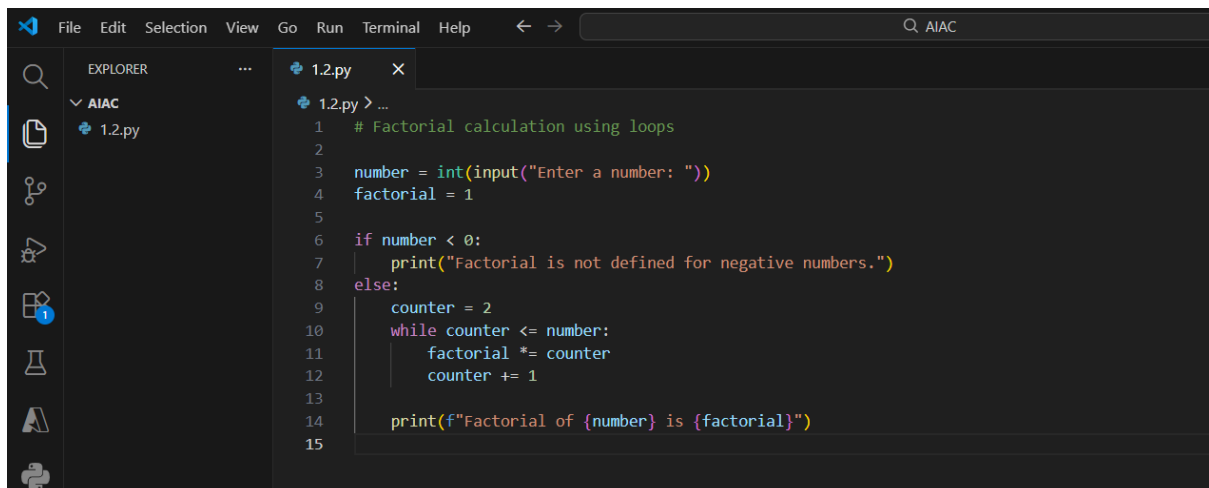
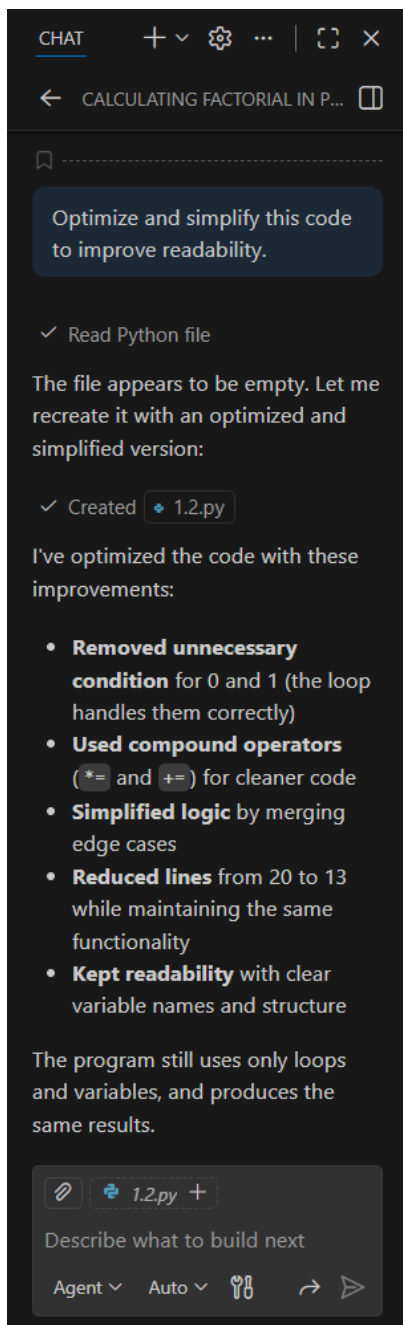
It followed basic Python syntax and logic correctly.

For beginners, it saves time and reduces syntax errors.

However, understanding the logic is still important.

Copilot gives suggestions but human review is needed

TASK_02:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
powershell + - [ ] [ ] ... | [ ] [ ] X

PS C:\Users\Pediripati Sricharan\AIAC> & "C:\Users\Pediripati Sricharan\AppData\Local\Programs\Python\Python310\python.exe" "c:/Users/Pediripati Sricharan/AIAC/1.2.py"
Enter a number: 12
Factorial of 12 is 479001600
PS C:\Users\Pediripati Sricharan\AIAC> |
```

Explanation

- Better variable names
- Cleaner output
- More readable

TASK_03:

CHAT + - [] [] ... | [] [] X

← CALCULATING FACTORIAL IN P... []

Write a Python program to calculate factorial using a function with proper comments.


✓ Read and created Python file

Done! I've written a factorial program using a function with comprehensive comments:

Key features:

- `calculate_factorial()` **function** with proper parameter and return documentation
- **Docstring** explaining the function's purpose, arguments, and return value
- **Inline comments** describing each step of the logic
- **Error handling** for negative numbers
- **Clean main code** that calls the function and displays results
- **Reusable** - the function can be called multiple times

🔄 ↶ 🏠 📄

 1.2.py +

Describe what to build next

Agent ▾ Auto ▾ 🛠️ ➡ ➤

```
1.2.py X
C:\Users\Pediripati Sricharan\AIAC\1.2.py
1  # Program to calculate factorial using a function with proper comments
2
3  def calculate_factorial(n):
4      """
5      Calculate the factorial of a number.
6
7      Args:
8      |   n (int): The number to calculate factorial for
9
10     Returns:
11     |   int: The factorial of n, or None if invalid input
12     """
13     # Check for invalid input (negative numbers)
14     if n < 0:
15         print("Error: Factorial is not defined for negative numbers.")
16         return None
17
18     # Base case: factorial of 0 or 1 is 1
19     if n == 0 or n == 1:
20         return 1
21
22     # Initialize result variable
23     result = 1
24
25     # Multiply all numbers from 2 to n
26     counter = 2
27     while counter <= n:
28         result *= counter
29         counter += 1
30
31     return result
32
33
34 # Main program - Get input and display result
35 number = int(input("Enter a number: "))
36 factorial_result = calculate_factorial(number)
37
38 if factorial_result is not None:
39     print(f"Factorial of {number} is {factorial_result}")
40
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
PS C:\Users\Pediripati Sricharan\AIAC> & "C:\Users\Pediripati Sricharan\AppData\Local\Programs\Python\Python310\python.exe" "c:/Users/Pediripati Sricharan/AIAC/1.2.py"
Enter a number: 12
Factorial of 12 is 479001600
PS C:\Users\Pediripati Sricharan\AIAC>
```

EXPLANATION:

Using functions improves reusability.

The same function can be used in many programs.

Code becomes cleaner and easier to maintain.

TASK 04:

Comparative Analysis – Procedural vs Modular AI Code

Procedural (Without Functions) vs Modular (With Functions)

In Task 1, the factorial program was written using a procedural approach, where all the logic was implemented directly in the main execution flow without using any user-defined functions. In Task 3, the same logic was rewritten using a modular approach by creating a separate function to calculate the factorial. Both approaches produce the same output, but they differ significantly in terms of design quality and usability.

Logic Clarity:

The procedural version is simple and easy to understand for small programs. However, as the program grows, the logic becomes harder to follow because everything is written in one place. In contrast, the modular version separates the factorial logic into a function, making the code more organized and easier to read.

Reusability:

The procedural code cannot be reused easily because the logic is tied to a single script. The modular version allows the factorial function to be reused in multiple programs without rewriting the same code, which saves time and effort.

Debugging Ease:

Debugging procedural code is more difficult because errors can affect

the entire program. In modular code, each function can be tested separately, making it easier to find and fix errors.

Suitability for Large Projects:

Procedural code is suitable only for small, simple programs. For large projects, modular code is preferred because it supports better structure, teamwork, and maintenance.

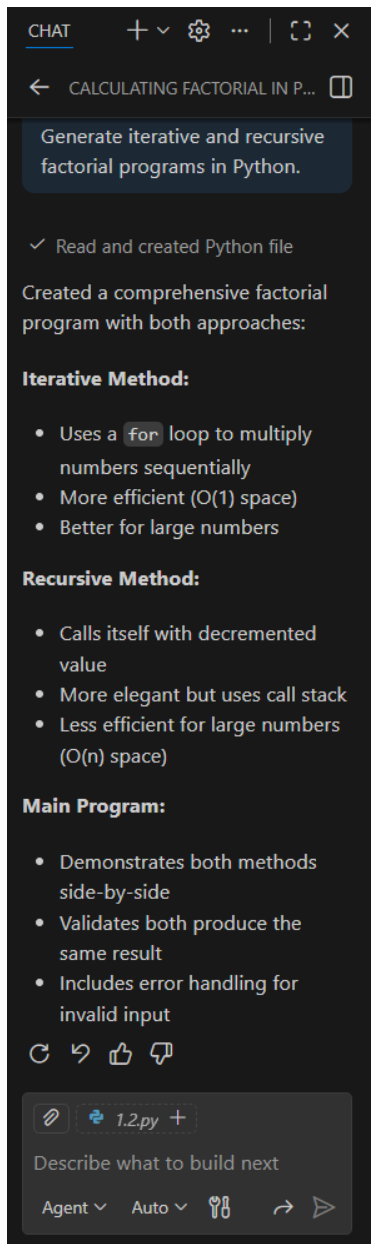
AI Dependency Risk:

When using AI tools like GitHub Copilot, procedural code may be generated quickly but often lacks proper structure. Modular code encourages better design practices, even when AI is used. This reduces the risk of poor-quality code.

Conclusion:

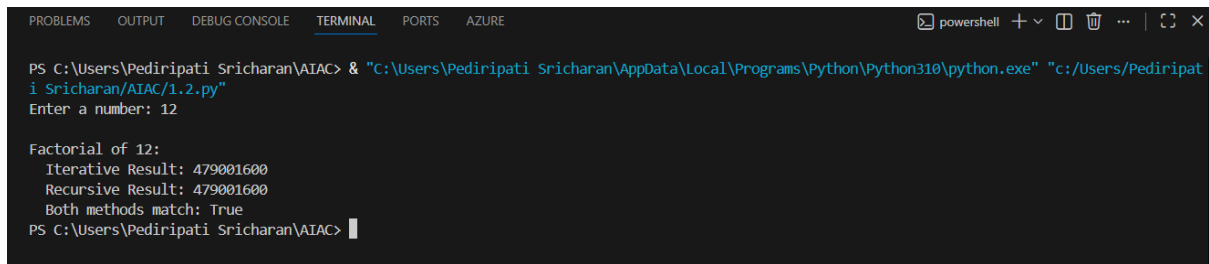
While procedural programming is useful for quick tasks and learning basics, modular programming is more efficient, reusable, and suitable for real-world software development. Using functions improves clarity, maintainability, and scalability, making modular code the better choice for professional projects.

TASK_05:



```
1.2.py X
C:\Users\Pediripati Sricharan\AIAC\1.2.py
1 # Factorial Programs: Iterative and Recursive Approaches
2
3 # ===== ITERATIVE APPROACH =====
4 def factorial_iterative(n):
5     """
6     Calculate factorial using iteration (loop).
7     Time Complexity: O(n)
8     Space Complexity: O(1)
9     """
10    # Handle invalid input
11    if n < 0:
12        return None
13
14    # Initialize result
15    result = 1
16
17    # Multiply numbers from 2 to n
18    for i in range(2, n + 1):
19        result *= i
20
21    return result
22
23
24 # ===== RECURSIVE APPROACH =====
25 def factorial_recursive(n):
26     """
27     Calculate factorial using recursion.
28     Time Complexity: O(n)
29     Space Complexity: O(n) - due to call stack
30     """
31    # Handle invalid input
32    if n < 0:
33        return None
34
35    # Base case: factorial of 0 or 1 is 1
36    if n == 0 or n == 1:
37        return 1
```

```
38
39    # Recursive case: n! = n * (n-1)!
40    return n * factorial_recursive(n - 1)
41
42
43 # ===== MAIN PROGRAM =====
44 def main():
45     """Main program to demonstrate both approaches."""
46
47     try:
48         number = int(input("Enter a number: "))
49
50         # Calculate using iterative approach
51         iter_result = factorial_iterative(number)
52
53         # Calculate using recursive approach
54         recur_result = factorial_recursive(number)
55
56         # Display results
57         if iter_result is None:
58             print("Error: Factorial is not defined for negative numbers.")
59         else:
60             print(f"\nFactorial of {number}:")
61             print(f"    Iterative Result: {iter_result}")
62             print(f"    Recursive Result: {recur_result}")
63             print(f"    Both methods match: {iter_result == recur_result}")
64
65     except ValueError:
66         print("Error: Please enter a valid integer.")
67
68
69 # Run the program
70 if __name__ == "__main__":
71     main()
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE powershell + - [ ] [X] ... | [ ] [X] X

PS C:\Users\Pediripati Sricharan\AIAC> & "C:\Users\Pediripati Sricharan\AppData\Local\Programs\Python\Python310\python.exe" "c:/Users/Pediripati Sricharan/AIAC/1.2.py"
Enter a number: 12

Factorial of 12:
  Iterative Result: 479001600
  Recursive Result: 479001600
  Both methods match: True
PS C:\Users\Pediripati Sricharan\AIAC> |
```

Explanation

- Iterative uses loop
- Recursive calls itself
- Recursion uses more memory
- Iterative is faster