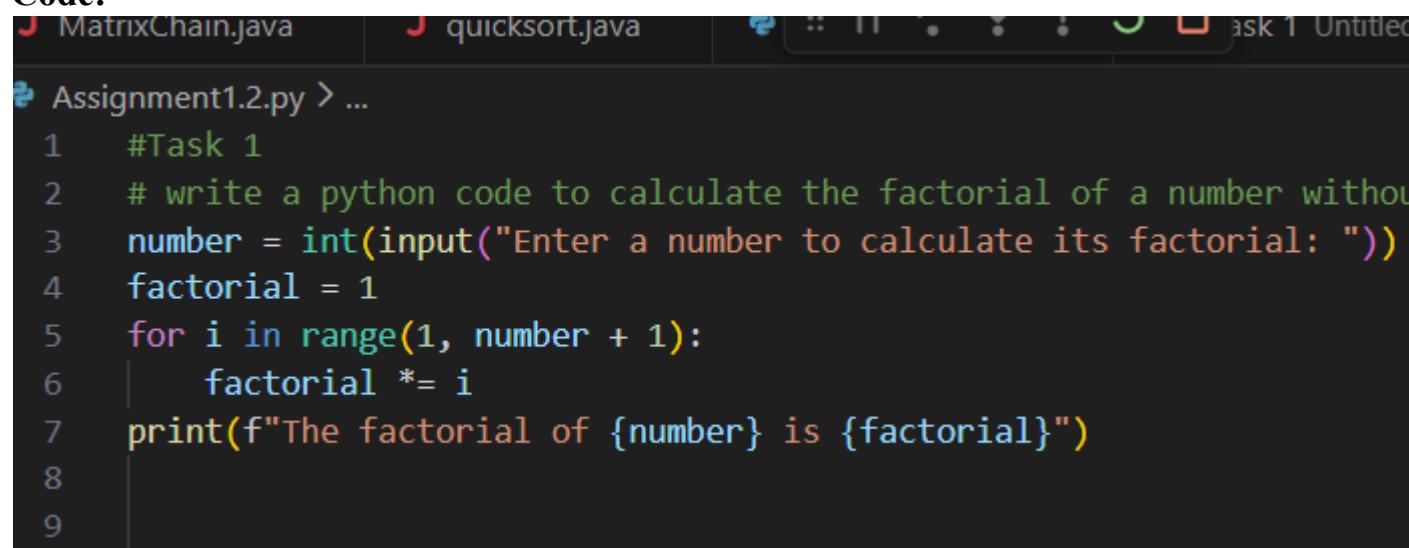


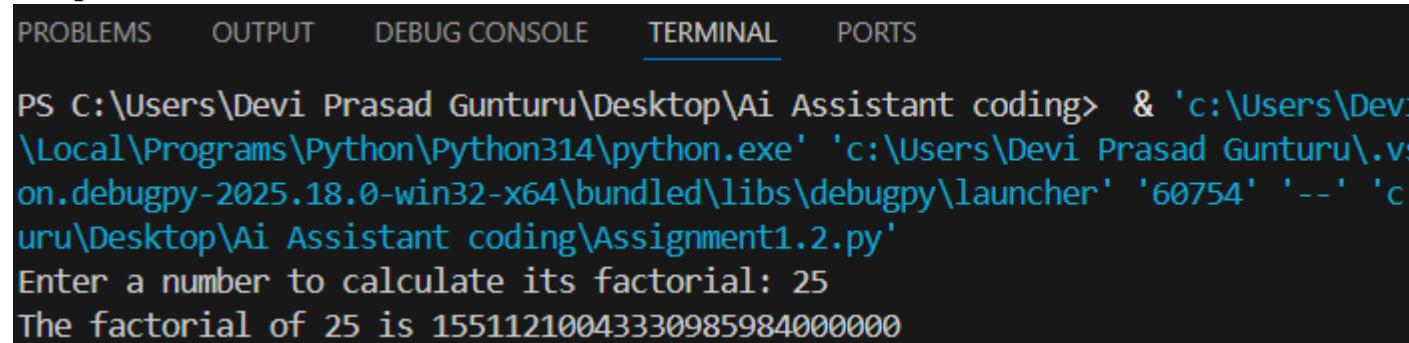
School of Computer Science and Artificial Intelligence**SLab Assignment # 1****Name of Student:** Gunturu Devi Prasad**Enrollment No. :** 2303A51538**Batch No. :** 22**Task 1: AI-Generated Logic Without Modularization (Factorial without Functions)**

Prompt: write a python code to calculate the factorial of a number without using functions.

Code:-

A screenshot of a code editor window. At the top, there are tabs for 'MatrixChain.java', 'quicksort.java', and 'Assignment1.2.py'. Below the tabs, the code for 'Assignment1.2.py' is displayed in a dark-themed code editor. The code is as follows:

```
1  #Task 1
2  # write a python code to calculate the factorial of a number without
3  number = int(input("Enter a number to calculate its factorial: "))
4  factorial = 1
5  for i in range(1, number + 1):
6      factorial *= i
7  print(f"The factorial of {number} is {factorial}")
```

Output:-

A screenshot of a terminal window. The tabs at the top are 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS', with 'TERMINAL' being the active tab. The terminal output shows the command to run the Python file, the user's input of '25' for the number, and the resulting factorial value '15511210043330985984000000'.

```
PS C:\Users\Devi Prasad Gunturu\Desktop\Ai Assistant coding> & 'c:\Users\Devi Prasad Gunturu\Local\Programs\Python\Python314\python.exe' 'c:\Users\Devi Prasad Gunturu\Assignment.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60754' '--' 'c:\Users\Devi Prasad Gunturu\Desktop\Ai Assistant coding\Assignment1.2.py'
Enter a number to calculate its factorial: 25
The factorial of 25 is 15511210043330985984000000
```

Justification:-

Think of this code as a straight-line solution it solves the problem quickly and directly without any extra packaging. It's perfect for a one-off task where you just need the answer right away, though it isn't built to be shared or reused easily in bigger projects.

Task 2: AI Code Optimization & Cleanup (Improving Efficiency)**Prompt:-**

Optimize the above code by removing unnecessary variables, simplifying the loop, and improving readability.

Code:-

```
10 #Task 2
11 # write a code Optimize the above code by removing unnecessary var
12 number = int(input("Enter a number to calculate its factorial: "))
13 factorial = 1
14 for i in range(1, number + 1):
15     factorial *= i
16 print(f"The factorial of {number} is {factorial}")
17
```

Output:-

```
Enter a number to calculate its factorial: 8
The factorial of 8 is 40320
The factorial of 5 is 120
Enter a number to calculate its factorial: 
```

Justification:-

This version streamlines the logic by stripping away clutter and focusing on the most direct way to get the result. It's essentially a cleaner, more grown-up version of the previous code that's easier for a teammate to read and maintain.

Task 3: Modular Design Using AI Assistance (Factorial with Functions)

Prompt: write a python code on factorial with function

Code:-

```
19 #Task 3
20 # write a python code on factorial with function
21 def factorial(n):
22     """Calculate the factorial of a number n."""
23     if n < 0:
24         return "Factorial is not defined for negative numbers."
25     elif n == 0 or n == 1:
26         return 1
27     else:
28         result = 1
29         for i in range(2, n + 1):
30             result *= i
31         return result
32 # Example usage
33 number = 5
34 print(f"The factorial of {number} is {factorial(number)}")
35
```

Output:-

```
Enter a number to calculate its factorial: 6
The factorial of 6 is 720
```

Justification:-

This version wraps the logic into a tidy, reusable "tool" called a function, making the code much more professional and organized. By separating the calculation from the display, you can now use this same logic anywhere else in your project without having to rewrite it from scratch.

Task 4: Comparative Analysis – Procedural vs Modular AI Code (With vs Without Functions)**Prompt:-**

Write a program that compares an AI-written script that's all in one block and a version that uses clear, well-named functions, explaining which is better for understanding, reuse, debugging, big projects, and avoiding over-reliance on AI, and show it in a small table.

Code:-

```
7 #Task 4
8 #Write a program that compares an AI-written script that's all in one
9 def calculate_factorial(number):
10     if number == 0:
11         return 1
12     else:
13         return number * calculate_factorial(number - 1)
14
15 number = int(input("Enter a number to calculate its factorial: "))
16 result = calculate_factorial(number)
17 print(f"The factorial of {number} is {result}")
18
19
```

Output:-

```
Enter a number to calculate its factorial: 6
The factorial of 6 is 720
```

Justification:-

This version uses a recursive function to elegantly handle the logic, making the code much easier to read and test compared to a single long script. It's built for modern teamwork, where clear, reusable "blocks" of code are essential for building and debugging large, professional projects.

Task 5: AI-Generated Iterative vs Recursive Thinking

Prompt: write a program on create iterative version logic vs recursive version logic of factorial and explain their flow and compare readability, stack usage, performance, and when recursion should be avoided.

Code:-

```
Assignment1.2.py > ...
49
50 #Task 5
51 # write a program on create iterative version logic vs recursive v
52
53 def factorial_iterative(n):
54     result = 1
55     for i in range(1, n + 1):
56         result *= i
57     return result
58
59 def factorial_recursive(n):
60     if n == 0:
61         return 1
62     else:
63         return n * factorial_recursive(n - 1)
64
65 print("Iterative version:")
66 print(factorial_iterative(5))
67
68 print("Recursive version:")
69 print(factorial_recursive(5))
```

Output:-

```
Iterative version:
120
Recursive version:
120
PS C:\Users\Devi Prasad Gunturu\Desktop\Ai Assistant coding>
```

Justification:-

This comparison highlights the two main ways computers "think": the iterative version uses a simple, efficient loop for direct calculation, while the recursive version breaks the problem down into smaller, repeated steps. While recursion looks more elegant and mathematical, the iterative approach is often safer for

very large numbers because it doesn't risk overloading the computer's memory stack.