

AI-AC Assignment: 1.2

Name: Syed Zaid Sami Khizar

HT No: 2303A52016

Batch: 31

Task 1: AI-Generated Logic Without Modularization (Factorial without Functions)

Prompt typed:

Write a Python program to calculate factorial of a number using loops only and dont use any user-defined function. take input from the user and print the result.

Code and Output Generated:

The screenshot shows a code editor with a dark theme. A file named 'Assignment1_2.py' is open, containing the following Python code:

```
Assignment1_2.py > ...
1 #Task-1
2
3 num = int(input("Enter a number: "))
4
5 if num < 0:
6     print("Factorial is not defined for negative numbers.")
7 else:
8     factorial = 1
9     for i in range(1, num + 1):
10         factorial *= i
11     print(f"The factorial of {num} is {factorial}")
```

Below the code editor is a terminal window showing the execution of the script:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\zaid_\OneDrive\Desktop\AI AC> python .\Assignment1_2.py
Enter a number: 10
The factorial of 10 is 3628800
PS C:\Users\zaid_\OneDrive\Desktop\AI AC>
```

Verdict:

GitHub Copilot was very helpful in generating the basic logic quickly, especially for a beginner. It correctly followed the instruction to avoid using functions and used simple loops and variables. The code was readable and logically structured without much correction needed. Copilot also handled input and output cleanly.

Task 2: AI Code Optimization & Cleanup

Prompt Used:

Optimize this code, Reduce unnecessary variables, Make the logic more readable and clean

Code and Output Generated:

The screenshot shows a code editor with a Python file named `Assignment1_2.py`. The code calculates the factorial of a user-specified number. It includes error handling for negative numbers and uses an f-string for the final output. Below the code editor is a terminal window showing two executions of the program, both resulting in the factorial of 10 being printed as 3628800.

```
Assignment1_2.py X
Assignment1_2.py > ...
1  # Get input from user
2  num = int(input("Enter a number: "))
3
4  # Check if negative
5  if num < 0:
6      print("Factorial is not defined for negative numbers.")
7  else:
8      # Initialize factorial
9      factorial = 1
10     # Calculate factorial using loop
11     for i in range(1, num + 1):
12         factorial *= i
13     # Print result
14     print(f"The factorial of {num} is {factorial}")

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    powershell + ⌂ ...    powershell
● PS C:\Users\zaid_\OneDrive\Desktop\AI AC> python .\Assignment1_2.py
Enter a number: 10
The factorial of 10 is 3628800
● PS C:\Users\zaid_\OneDrive\Desktop\AI AC> python .\Assignment1_2.py
Enter a number: 10
The factorial of 10 is 3628800
❖ PS C:\Users\zaid_\OneDrive\Desktop\AI AC>
```

Verdict:

The optimized version improves readability by using a clearer variable name (`result` instead of `fact`).

Task 3: Modular Design Using AI Assistance

Prompt Used:

Rewrite this program using a user-defined function
Use meaningful function and variable names
Add inline comments
Take input from the user and print the result

Code and Output Generated:

The screenshot shows a code editor with Python code for calculating factorial. The code includes a function definition and a main program that prompts the user for input and prints the result. Below the code editor is a terminal window showing three executions of the script, each asking for a number and printing the factorial result.

```

Assignment1_2.py > ...
1  #Task - 3
2  def calculate_factorial(number): # Function to calculate factorial of a number
3      if number < 0: # Factorial is not defined for negative numbers
4          return None
5      factorial = 1 # Initialize factorial to 1
6      for i in range(1, number + 1): # Loop from 1 to number
7          factorial *= i # Multiply factorial by current number
8      return factorial # Return the calculated factorial
9
10 # Main program
11 number = int(input("Enter a number: ")) # Get input from user
12 result = calculate_factorial(number) # Call the function
13 if result is None: # If invalid input
14     print("Factorial is not defined for negative numbers.")
15 else: # Print the result
16     print(f"The factorial of {number} is {result}")

```

TERMINAL

```

PS C:\Users\zaid_\OneDrive\Desktop\AI AC> python .\Assignment1_2.py
Enter a number: 10
The factorial of 10 is 3628800
PS C:\Users\zaid_\OneDrive\Desktop\AI AC> python .\Assignment1_2.py
Enter a number: 10
The factorial of 10 is 3628800
PS C:\Users\zaid_\OneDrive\Desktop\AI AC> python .\Assignment1_2.py
Enter a number: 10
The factorial of 10 is 3628800

```

Verdict:

Copilot does a good job in defining the factorial in a clean function and adding in-line comments making sure the user knows what is what, this will also helps in re-usability of code as we don't have to write the code over and over about it.

Task 4: Comparison Table

Criteria	Without Functions (Task 1 & 2)	With Functions (Task 3)
Logic Clarity	Logic is directly written in the main flow, which can become confusing as the program grows.	Logic is clearly separated into a function, making the code easier to understand.
Reusability	Code cannot be reused easily; it must be rewritten if needed again.	The function can be reused in multiple programs by simply calling it.
Debugging Ease	Debugging is harder because everything is in one block.	Easier to debug since the function can be tested independently.
Suitability for Large Projects	Not suitable for large projects because code becomes messy and difficult to manage.	Highly suitable for large projects due to clean structure and modularity.
AI Dependency Risk	Higher risk: if AI generates poor structure, the whole program quality drops.	Lower risk: even if AI helps, modular design keeps the structure organized.

Task 5:

Code Prompt:

Rewrite the program

Generate an iterative version of factorial program

Generate a recursive version of factorial program

Both should take input from the user and print the result

Code and Output Generated:

```
1  def factorial_iterative(n): # Iterative function to calculate factorial
2      if n < 0: # Factorial not defined for negative numbers
3          return None
4      result = 1 # Initialize result
5      for i in range(1, n + 1): # Loop to multiply numbers
6          result *= i
7      return result
8
9  def factorial_recursive(n): # Recursive function to calculate factorial
10     if n < 0: # Factorial not defined for negative numbers
11         return None
12     if n == 0 or n == 1: # Base case
13         return 1
14     return n * factorial_recursive(n - 1) # Recursive call
15
16 # Main program
```



The factorial of 10 is 3628800
PS C:\Users\zaid_\OneDrive\Desktop\AI AC> python .\Assignment1_2.py
Enter a number: 10
Iterative: The factorial of 10 is 3628800
Recursive: The factorial of 10 is 3628800
PS C:\Users\zaid_\OneDrive\Desktop\AI AC>

Verdict:

In the iterative version, the program uses a loop to multiply numbers step by step until the final result is reached. The control stays within a single function call.

In the recursive version, the function calls itself repeatedly by reducing the value of the number each time. The computation happens when the recursive calls start returning values back up the call stack.