**ASSIGNMENT - 1.1**

---------------------------------------

**NAME: PRADEEP GUPTHA**

**HALL-TICKET: 2403A510C7**

**BATCH NO: 05**

**Task 1:** Factorial without Functions  
● Description:  
Use GitHub Copilot to generate a Python program that calculates the factorial of a number without defining any functions (using loops directly in the main code).

Prompt:

Give a Python program that calculates the factorial of a number without defining any functions which takes user input.

CODE SCREENSHOT:

A screen shot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A black background with white text

AI-generated content may be incorrect.

EXPLANATION:

A factorial code calculates the product of all positive integers up to a given number n. The factorial of n is written as n! and defined as:

n! = n × (n-1) × (n-2) × ... × 1

For example, 5! = 5 × 4 × 3 × 2 × 1 = 120.

A typical factorial function (in Python) looks like this:

* If n is 0 or 1, the function returns 1 (by definition).
* Otherwise, it multiplies n by the factorial of n-1 (recursion).
* This continues until n reaches 1.

--------------------------------------------------------------------------------------------------------------

Task 2: Improving Efficiency  
● Description:  
Examine the Copilot-generated code from Task 1 and demonstrate  
how its efficiency can be improved (e.g., removing unnecessary variables, optimizing loops).

Prompt: give improved version code of task 1

CODE SCREENSHOT:

(original with functions)

A screenshot of a computer program

AI-generated content may be incorrect.

( Improved version without any extra variables, optimizing loops):

A screenshot of a computer program

AI-generated content may be incorrect.

EXPLANATION:

* The improved code reduces unnecessary variable assignments and uses a for loop, which is generally faster and more readable than a while loop for this case.
* Starting from 2 skips a redundant multiplication.
* The code is shorter, easier to maintain, and may run slightly faster due to fewer operations and better loop optimization by Python.

----------------------------------------------------------------------------------------------------------

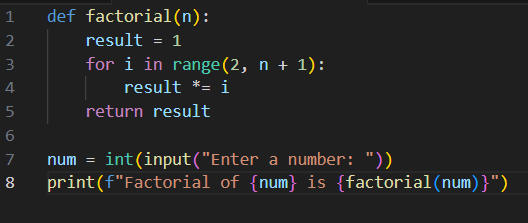
Task 3: Factorial with Functions

=>Description:  
Use GitHub Copilot to generate a Python program that calculates the  
factorial of a number using a user-defined function.

PROMPT:

Generate a Python program that calculates the factorial of a number using a user-defined function.

CODE:



OUTPUT:

A black background with white text

AI-generated content may be incorrect.

EXPLANATION:

* The factorial function computes the factorial using a loop.
* The user is prompted to enter a number, and the result is displayed.

------------------------------------------------------------------------------------------------------------

TASK 4: Comparative Analysis – With vs Without Functions

● Description:  
Differentiate between the Copilot-generated factorial program with  
functions and without functions in terms of logic, reusability, and  
execution.

PROMPT:

Generate a python code to calculate factorial of a given number with and without functions.

CODES:

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

==>>.Both approaches will give the same result. The function version is reusable and cleaner, while the non-function version is direct and simple.

EXPLANATION:

Here’s a comparison of factorial calculation with and without a function in table form:

| **Approach** | **Code Snippet** | **Explanation** |
| --- | --- | --- |
| With Function | def factorial(n):<br> result = 1<br> for i in range(2, n + 1):<br>  result \*= i<br> return result<br>num = int(input("Enter a number: "))<br>print("Factorial (with function):", factorial(num)) | Defines a reusable function factorial(n) that calculates the factorial. User input is passed to the function and result is printed. |
| Without Function | num = int(input("Enter a number: "))<br>result = 1<br>for i in range(2, num + 1):<br> result \*= i<br>print("Factorial (without function):", result) | Calculates factorial directly in the main program using a loop, then prints the result. No function is used. |

This table shows the code and explanation for both methods side by side for easy comparison.

--------------------------------------------------------------------------------------------------------------

TASK 5: Iterative vs Recursive Factorial

● Description:  
Prompt GitHub Copilot to generate both iterative and recursive  
versions of the factorial function.

Prompt:

Generate both iterative and recursive versions of the factorial function.

CODE 1:

A computer screen with colorful text

AI-generated content may be incorrect.

Output:



CODE 2:

A computer screen with white text

AI-generated content may be incorrect.

Output:



Explanation(comparing):

| **Aspect** | **Iterative** | **Recursive** |
| --- | --- | --- |
| **Readability** | Simple and clear | Elegant and closer to mathematical definition |
| **Space Usage** | **O(1)** (constant space) | **O(n)** (stack frames for each call) |
| **Time Complexity** | **O(n)** | **O(n)** |
| **Overhead** | Low (just loop) | Higher (function call overhead) |
| **Risk of Errors** | Minimal | Possible stack overflow for large n |
| **Performance** | Slightly faster due to no recursion | Slightly slower due to call overhead |
| **Ease of Debugging** | Easier (loop structure visible) | Slightly harder (nested calls) |