

LAB ASSIGNMENT- 1.1

TASK 1:

Factorial without functions

```
assignment1 task 1.py > -
1 #write a program in python that print factorial of a number without using function
2
3 num = input("Enter a non-negative integer: ")
4
5 # Input validation
6 while not num.isdigit():
7     num = input("Invalid input. Please enter a non-negative integer: ")
8
9 num = int(num)
10 factorial = 1
11
12 if num == 0 or num == 1:
13     factorial = 1
14 else:
15     for i in range(2, num + 1):
16         factorial *= i
17
18 print(f"The factorial of {num} is {factorial}")
```

for i in range(2, num + 1):
 factorial *= i
print(f"The factorial of {num} is {factorial}")

Workspace
Used 8 references

This code calculates the factorial of a non-negative integer entered by the user, with input validation to ensure the input is appropriate. It starts by prompting the user to enter a value. Since the `input()` function returns a string, the code uses the `isdigit()` method in a `while` loop to check if the input consists only of digits (i.e., it is a non-negative integer). If the input is invalid, the user is repeatedly prompted until a valid number is entered.

Once a valid input is received, it is converted from a string to an integer using the `int()` constructor. The variable `factorial` is initialized to 1. The code then checks if the number is 0 or 1, in which case the factorial is set to 1 (since 0! and 1! are both 1 by definition). For numbers greater than 1, a `for` loop multiplies `factorial` by each integer from 2 up to and including the entered number, effectively computing the factorial.

Finally, the result is displayed using an f-string in the `print()` function, which outputs both the original number and its factorial. This approach ensures that only valid, non-negative integers are processed and provides a clear, user-friendly output.

Add context... assignment1 task 1.py +
Add context (#), extensions (@), commands (/)
Ask GPT-4.1

TASK 2:

Improving Efficiency

```
1 #Improved program to print factorial of a number without using a function
2
3 num = int(input("Enter a number: "))
4 factorial = 1
5 for i in range(1, num + 1):
6     factorial *= i
7 print("The factorial of", num, "is", factorial)
8
```

factorial *= i
print("The factorial of", num, "is", factorial)

Workspace
Used 7 references

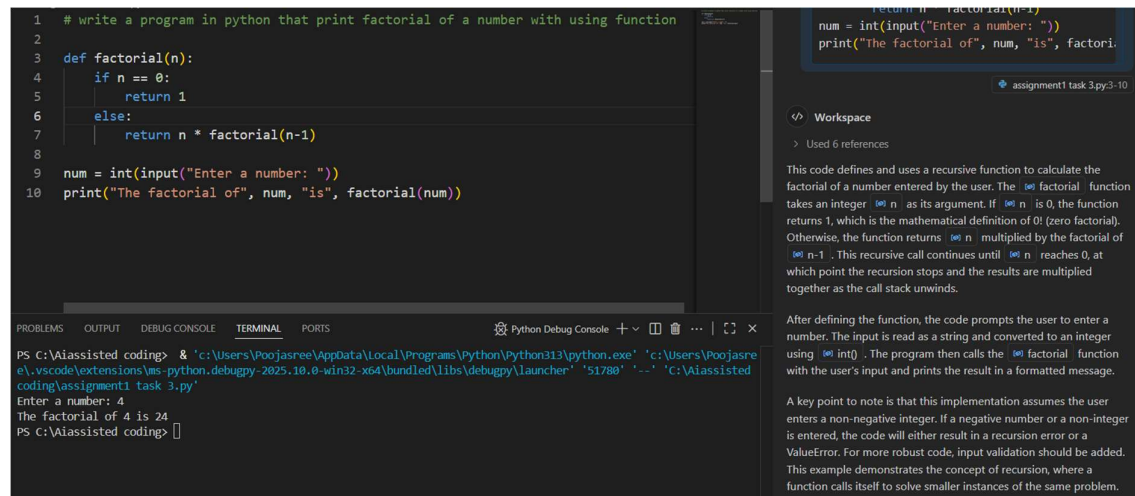
This code calculates the factorial of a number entered by the user. It starts by prompting the user for input with `input("Enter a number: ")`. The input, which is a string, is converted to an integer using `int()`, ensuring that mathematical operations can be performed on it.

The variable `factorial` is initialized to 1, which serves as the starting value for the multiplication process. The `for` loop iterates from 1 up to and including the number entered by the user (`num + 1` because the stop value in `range` is exclusive). On each iteration, the current value of `i` is multiplied with `factorial`, updating it to the running product. This loop effectively computes the factorial by multiplying all integers from 1 to `num`.

Finally, the result is displayed using the `print()` function, which outputs both the original number and its factorial. Note that this code does not include input validation, so if the user enters a negative number or a non-integer value, it may result in an error or incorrect output. For more robust programs, input validation is recommended.

TASK3:

FACTORIAL WITH FUNCTIONS



```
1 # write a program in python that print factorial of a number with using function
2
3 def factorial(n):
4     if n == 0:
5         return 1
6     else:
7         return n * factorial(n-1)
8
9 num = int(input("Enter a number: "))
10 print("The factorial of", num, "is", factorial(num))
```

assignment1 task 3.py:10

Workspace

Used 6 references

This code defines and uses a recursive function to calculate the factorial of a number entered by the user. The `factorial` function takes an integer `n` as its argument. If `n` is 0, the function returns 1, which is the mathematical definition of 0! (zero factorial). Otherwise, the function returns `n` multiplied by the factorial of `n-1`. This recursive call continues until `n` reaches 0, at which point the recursion stops and the results are multiplied together as the call stack unwinds.

After defining the function, the code prompts the user to enter a number. The input is read as a string and converted to an integer using `int()`. The program then calls the `factorial` function with the user's input and prints the result in a formatted message.

A key point to note is that this implementation assumes the user enters a non-negative integer. If a negative number or a non-integer is entered, the code will either result in a recursion error or a `ValueError`. For more robust code, input validation should be added. This example demonstrates the concept of recursion, where a function calls itself to solve smaller instances of the same problem.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python Debug Console

```
PS C:\aiassisted coding> & 'c:\Users\Poojasree\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Poojasree\
e\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundle\libs\debugpy\launcher' '51780' '--' 'c:\aiassisted
coding\assignment1 task 3.py'
Enter a number: 4
The factorial of 4 is 24
PS C:\aiassisted coding>
```

TASK 4

COMPARING FACTORIAL WITH FUNCTION AND WITHOUT FUNCTION

The difference between calculating factorial **with functions** and **without functions** is:

With Functions

- The logic for factorial is placed inside a function (e.g., `factorial(n)`).
- You can reuse the function for different inputs.
- The code is modular, easier to read, and maintain.

Without Functions

- The logic is written directly in the main code, not inside a function.
- You cannot reuse the code easily for different inputs.
- The code is less organized.

TASK 5

Iterative vs Recursive Factorial

```
assignment 1 task 5.py > factorial_recursive
1  def factorial_iterative(n):
2      result = 1
3      for i in range(2, n + 1):
4          result *= i
5      print(result)
6
7  # 2
8  def factorial_recursive(n):
9      if n == 0:
10         result = 1
11     else:
12         result = n * factorial_recursive(n - 1)
13     print(result)
```

Iterative Factorial

- Uses a loop (like for or while) to calculate the factorial.
- Updates a result variable step by step.
- Generally uses less memory (no function call stack).

Recursive Factorial

- The function calls itself to solve smaller subproblems.
- Has a base case (`n == 0`).
- Uses more memory due to function call stack.