LAB ASSIGNMENT- 1.1

TASK 1:

Factorial without functions

assignment 1 task 1.py > ...

```python
#write a program in python that print factorial of a number without using function

num = input("Enter a non-negative integer: ")

# Input validation
while not num.isdigit():
    num = input("Invalid input. Please enter a non-negative integer: ")

num = int(num)
factorial = 1

if num == 0 or num == 1:
    factorial = 1
else:
    for i in range(2, num + 1):
        factorial *= i

print(f"The factorial of {num} is {factorial}")
```

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

assignment 1 task 1.py:3-18

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... assignment 1 task 1.py +

Add context (#), extensions (@), commands (/)

Ask GPT-4.1

TASK 2:

Improving Efficiency

```python
#Improved program to print factorial of a number without using a function

num = int(input("Enter a number: "))
factorial = 1
for i in range(1, num + 1):
    factorial *= i
print("The factorial of", num, "is", factorial)
```

```python
    factorial *= i
print("The factorial of", num, "is", factori
```

assignment1 task 2.py:3-7

Workspace

Used 7 references

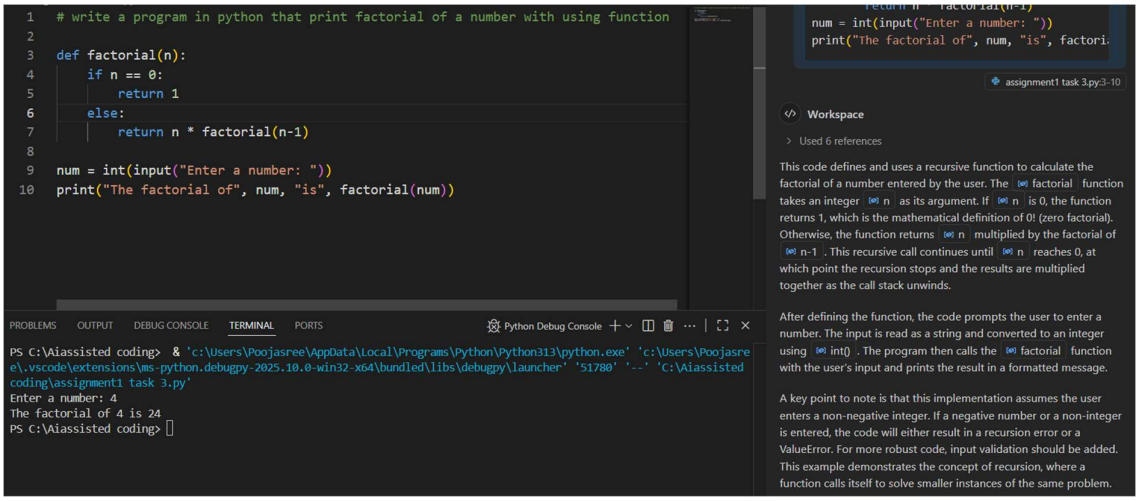
This code calculates the factorial of a number entered by the user. It starts by prompting the user for input with `input("Enter a number: ")`. The input, which is a string, is converted to an integer using `int()`, ensuring that mathematical operations can be performed on it.

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



TASK 4

COMPARING FACTORIAL WITH FUNCTION AND WITH OUT 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

```python
def factorial_iterative(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    print(result)

# 2
def factorial_recursive(n):
    if n == 0:
        result = 1
    else:
        result = n * factorial_recursive(n - 1)
    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.