

AI ASSISTED CODING

Lab assignment-1.1

Prompt 1: Factorial without Functions

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)

Code(screenshot):

The screenshot shows a terminal window with the following content:

```
ass-1.1.py > ...
1  # Calculate the factorial of a number using a loop (no functions)
2
3  n = int(input("Enter a number: "))
4  factorial = 1
5
6  for i in range(1, n + 1):
7      factorial *= i
8
9  print(f"Factorial of {n} is {factorial}")

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    +  ...  |  []  X
PS C:\Users\nered\OneDrive\Desktop\wtml> & C:/Users/nered/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nered/OneDrive/Desktop/wtm1/ass-1.1.py
Enter a number: 12
Factorial of 12 is 479001600
PS C:\Users\nered\OneDrive\Desktop\wtml>
```

The terminal shows the execution of the Python script `ass-1.1.py`. It prompts the user to enter a number (12), calculates the factorial (479001600), and prints the result.

Code explanation:

This code calculates the factorial of a user-provided number using a loop:

- It prompts the user to enter a number and stores it in `n`.

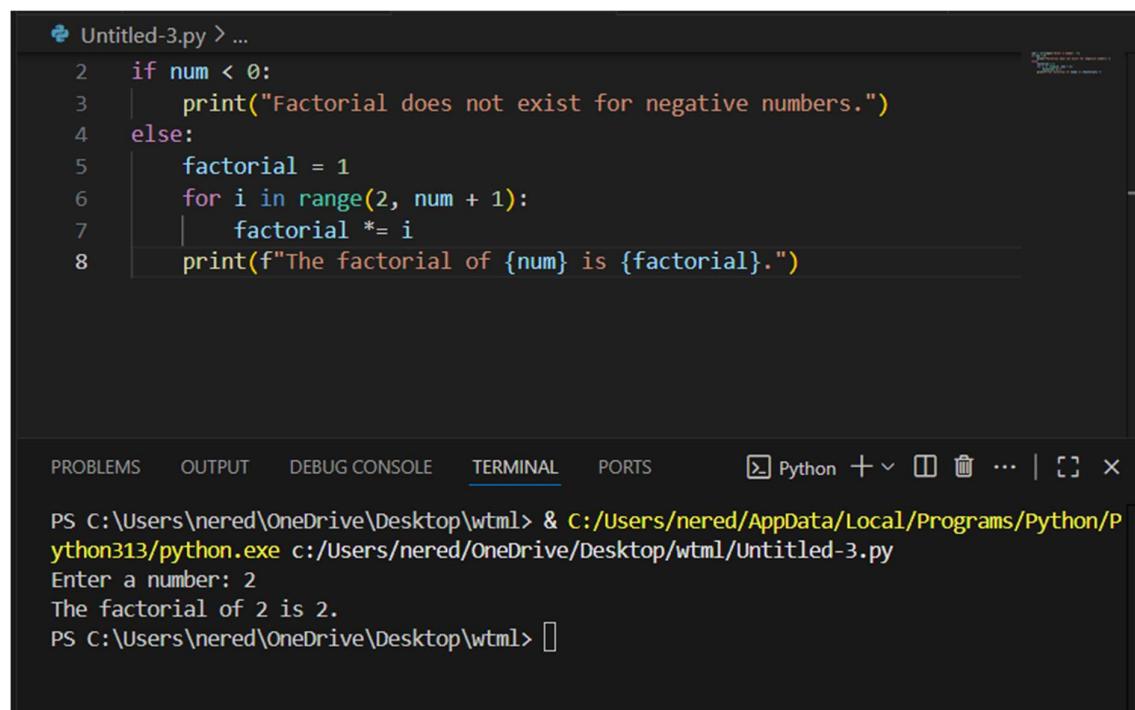
- It initializes factorial to 1.
- It uses a for loop from 1 to n, multiplying factorial by each number in the range.
- After the loop, it prints the result, which is the factorial of the input number.

Prompt 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).

Code(screen shot):



```

    Untitled-3.py > ...
2  if num < 0:
3      print("Factorial does not exist for negative numbers.")
4  else:
5      factorial = 1
6      for i in range(2, num + 1):
7          factorial *= i
8      print(f"The factorial of {num} is {factorial}.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + |

PS C:\Users\nered\OneDrive\Desktop\wtml> & C:/Users/nered/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nered/OneDrive/Desktop/wtml/Untitled-3.py
Enter a number: 2
The factorial of 2 is 2.
PS C:\Users\nered\OneDrive\Desktop\wtml> []

Code explanation:

This code calculates the factorial of a number entered by the user:

- `num = int(input("Enter a number: "))`
Prompts the user to enter a number and converts the input to an integer.
- `if num < 0:`
Checks if the number is negative.

- `print("Factorial does not exist for negative numbers.")`
If negative, prints a message since factorials are only defined for non-negative integers.
- `else:`
If the number is zero or positive:
 - `factorial = 1`
Initializes the factorial result to 1.
 - `for i in range(2, num + 1):`
Loops from 2 up to and including `num`.
 - `factorial *= i`
Multiplies `factorial` by each value of `i` in the loop.
 - `print(f"The factorial of {num} is {factorial}.")`
Prints the final factorial value

prompt 3: Factorial with Functions

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

Code(screen shot):

Code explanation:

This program calculates the factorial of a number using a user-defined function:

- The `factorial(num)` function checks if the input is negative. If so, it returns None.
 - If the input is zero or positive, it initializes `result` to 1 and multiplies it by each integer from 2 up to `num`.
 - The main code gets a number from the user, calls the `factorial` function, and stores the result.
 - If the result is None, it prints a message for negative numbers. Otherwise, it prints the factorial value.

Prompt 4: Comparative Analysis – With vs Without Functions

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

Code(screen shot):

The screenshot shows a code editor interface with multiple tabs at the top: NEWS BLOG.html, Untitled-3.py (selected), # RECIPE COLLECTION.css, and calculator. The Untitled-3.py tab contains the following Python code:

```
1 # Function to reverse a string
2 def reverse_string(s):
3     return s[::-1]
4 # Recursive version of factorial
5 def factorial_recursive(n):
6     if n == 0 or n == 1:
7         return 1
8     else:
9         return n * factorial_recursive(n - 1)
10 # Iterative version of factorial
11 def factorial_iterative(n):
12     result = 1
13     for i in range(2, n + 1):
14         result *= i
15     return result
16 # Example usage
17 num = int(input("Enter a number to calculate its factorial: "))
18 print(f"Recursive: Factorial of {num} is {factorial_recursive(num)}")
19 print(f"Iterative: Factorial of {num} is {factorial_iterative(num)}")
20
```

Below the code editor is a terminal window with the following text:

```
PS C:\Users\nered\OneDrive\Desktop\wtml> & C:/Users/nered/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nered/OneDrive/Desktop/wtml/Untitled-3.py
Enter a number to calculate its factorial: 5
Recursive: Factorial of 5 is 120
Iterative: Factorial of 5 is 120
PS C:\Users\nered\OneDrive\Desktop\wtml>
```

Code explanation:

- **reverse_string(s):**

This function takes a string `s` and returns its reverse using slicing (`s[::-1]`).

- **factorial_recursive(n):**

This function calculates the factorial of `n` recursively.

- If `n` is 0 or 1, it returns 1 (base case).

- Otherwise, it returns `n * factorial_recursive(n - 1)`.

- **factorial_iterative(n):**

This function calculates the factorial of `n` using a loop.

- It initializes `result` to 1.

- Then multiplies `result` by each number from 2 to `n`.
- **Example usage:**
 - The user is prompted to enter a number.
 - The program prints the factorial of that number using both the recursive and iterative functions.

Prompt 5: Iterative vs Recursive Factorial

● Description:

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

● Expected Output:

- Two correct implementations.
- A documented comparison of logic, performance, and execution flow between iterative and recursive approaches.

Code (screen shot):

```

❸ Untitled-3.py > ...
1 # Iterative version of factorial
2 def factorial_iterative(n):
3     """
4         Calculates factorial using a loop.
5         Returns 1 for n=0 or n=1.
6     """
7     result = 1
8     for i in range(2, n + 1):
9         result *= i
10    return result
11
12 # Recursive version of factorial
13 def factorial_recursive(n):
14     """
15         Calculates factorial using recursion.
16         Returns 1 for n=0 or n=1 (base case).
17     """
18     if n == 0 or n == 1:
19         return 1
20     else:
21         return n * factorial_recursive(n - 1)
22
23 # Example usage
24 num = int(input("Enter a number: "))
25 print(f"Iterative: Factorial of {num} is {factorial_iterative(num)}")
26 print(f"Recursive: Factorial of {num} is {factorial_recursive(num)}")
27
28 # Comparison:
29 # - Logic: Iterative uses a loop to multiply numbers; recursive calls itself, reducing n each time.
30 # - Performance: Iterative is generally faster and uses less memory, as recursion adds call stack overhead.
31 # - Execution flow: Iterative runs in a single loop; recursive breaks the problem into smaller subproblems until the base case is reached.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\nered\Desktop\wtm1> & C:/Users/nered/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nered/Desktop/wtm1/Untitled-3.py
Enter a number: 4
Iterative: Factorial of 4 is 24
Recursive: Factorial of 4 is 24
PS C:\Users\nered\Desktop\wtm1>

```

Code explanation:

The code provides two ways to calculate the factorial of a number:

1. Iterative Version ([factorial_iterative](#))

- Uses a loop to multiply numbers from 2 up to n .
- Returns 1 for $n = 0$ or $n = 1$.
- Efficient in terms of speed and memory.

2. Recursive Version ([factorial_recursive](#))

- Calls itself with $n - 1$ until it reaches the base case ($n = 0$ or $n = 1$).
- Returns 1 for the base case.
- Less efficient for large n due to call stack overhead.

Example usage:

- Prompts the user for a number.
- Prints the factorial using both methods.

Comparison:

- *Logic:* Iterative uses a loop; recursive breaks the problem into smaller subproblems.
- *Performance:* Iterative is faster and uses less memory.
- *Execution flow:* Iterative runs in a single loop; recursive uses multiple function calls until the base case.