

# LAB ASSIGNMENT - 1.1

NAME : P.Hansini Reddy

ROLL NO:2403A510D5

COURSE : AI ASSISTED CODING

BATCH : 01

## QUESTIONS

The screenshot shows a Microsoft Word document titled "Lab Assignment\_1.1.docx". The content is organized into three main sections: "After completing this lab, students will be able to:", "Task 0:", and "Task 1: Factorial without Functions".

**After completing this lab, students will be able to:**

- Set up GitHub Copilot in VS Code successfully.
- Use inline comments and context to generate code with Copilot.
- Evaluate AI-generated code for correctness and readability.
- Compare code suggestions based on different prompts and programming styles.

**Task 0:**

- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.
- Expected Output:

**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).
- Expected Output:
  - A working program that correctly calculates the factorial for user-provided input.
  - Screenshots of the code generation process.

**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).
- Expected Output:

The screenshot shows a Microsoft Word document titled "Lab Assignment\_1.1.docx". The content is organized into four main sections: "Task 2: Factorial with Functions", "Task 3: Comparative Analysis – With vs Without Functions", and "Task 4: Iterative vs Recursive Factorial".

**Task 2: Factorial with Functions**

- Description:  
Use GitHub Copilot to generate a Python program that calculates the factorial of a number using a user-defined function.
- Expected Output:
  - Correctly working factorial function with sample outputs.
  - Documentation of the steps Copilot followed to generate the function.

**Task 3: Comparative Analysis – With vs Without Functions**

- Description:  
Differentiate between the Copilot-generated factorial programs with functions and without functions in terms of logic, readability, and execution.
- Expected Output:
  - A comparison table or short report explaining the differences.

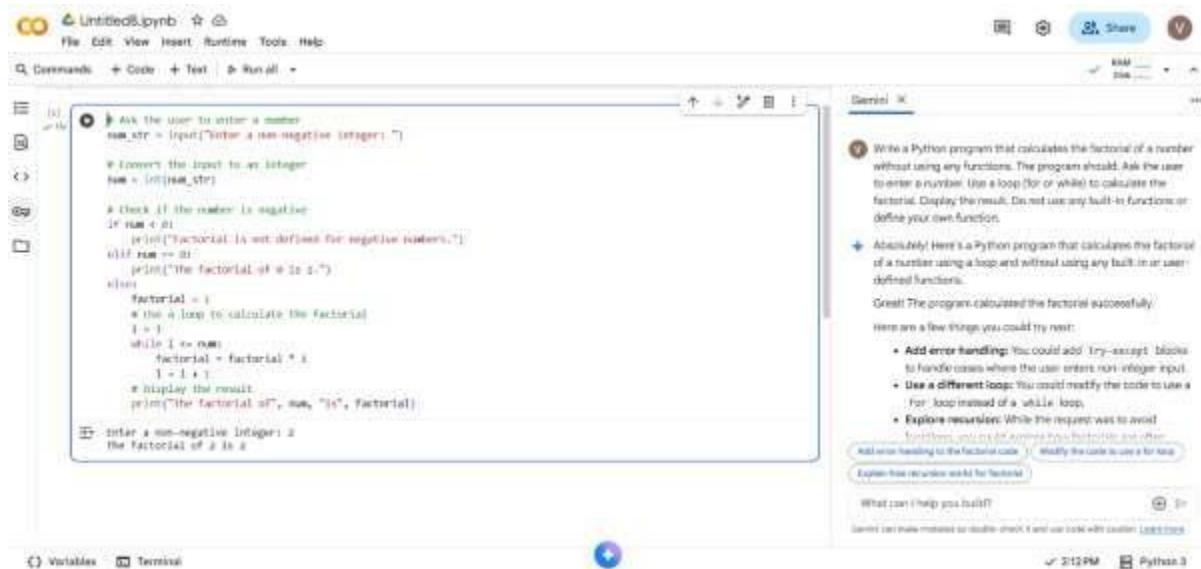
**Task 4: Iterative vs Recursive Factorial**

- Description:  
Priming 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.

# TASK 0:



# TASK 1 :



# TASK 2 :

The screenshot shows a Python code editor with the following code:

```
# Ask the user to enter a number
num_int = input("Enter a non-negative integer: ")

# Convert the input to an integer
num = int(num_int)

# Check if the number is negative
if num < 0:
    print("Factorial is not defined for negative numbers.")
elif num == 0:
    print("The factorial of 0 is 1.")
else:
    factorial = 1
    # Use a loop to calculate the factorial
    for i in range(1, num + 1):
        factorial *= i
    # Display the result
    print("The factorial of", num, "is", factorial)
```

Below the code, the output is shown as:

```
Enter a non-negative integer: 6
The factorial of 6 is 720
```

To the right, a Gemini AI sidebar provides feedback:

- Take the factorial program (written without functions using loops) and improve its efficiency. Remove any unnecessary variables or steps. Optimize the loop to make it cleaner and faster. Keep the logic simple and easy to understand. Display the factorial result clearly.
- Okay, I can help you with that. I will modify the existing code to improve its efficiency by removing unnecessary variables and optimizing the loop.

A "Working..." status indicator is visible below the second point.

## **TASK 3 :**

```

for i in range(1, n + 1):
    result *= i
    return result

# Ask the user to enter a number
num_str = input("Enter a non-negative integer: ")

# Convert the input to an integer
try:
    num = int(num_str)
    # Call the factorial function
    fact_result = factorial(num)

    # Display the result
    if fact_result is None:
        print("Factorial is not defined for negative numbers.")
    else:
        print(f"The factorial of {num} is {fact_result}")
except ValueError:
    print("Invalid input. Please enter an integer.")

# Enter a non-negative integer: 4
# The factorial of 4 is 24

```

**Genie X**

Write a Python program that calculates the factorial of a number using a user-defined function. The program should define a function called factorial() that uses a loop to calculate the factorial. Ask the user to enter a number. Call the function and display the result. Include clear comments or documentation to explain the code.

Context: Here's a Python program that calculates the factorial of a number using a user-defined function with a loop. Excellent! The program with the user-defined function works correctly.

Here are some potential next steps:

- Explore recursive: Write another version of the factorial function using recursion instead of a loop. This is a common way to demonstrate recursion.

[Write a recursive factorial function](#)

[Calculate factorial for larger numbers using math module](#)

[Add type validation for non-integer input](#)

What can I help you build?

Genie can make mistakes so double-check it and use code with caution. [Learn more](#)

## TASK 4 :

**Untitled2.ipynb** star @

File Edit View Insert Runtimes Tools Help

Commands + Code + Text + Run all ↗

**Genie X**

Compare the two factorial programs – one with functions and one without functions. Explain the differences between them in terms of: Logic. How the steps or structure of the code differ. Readability. How easy it is to use the code again in other programs. Execution. How the program runs or how efficient it is. Provide the comparison in a short and clear explanation or in a table format.

I can definitely help you compare the two factorial programs. Here's a breakdown of their differences:

Feature	Program without Functions (Cell #398794)	Program with User-Defined Function (Cell #5c1224)
Logic	The logic for calculating the factorial is directly in the main part of the script, executed sequentially.	The factorial calculation logic is encapsulated within a dedicated function ( <code>factorial(n)</code> ).
Readability	The code is less readable. If you needed to calculate a factorial in another part of your program or a different program, you would need to copy and paste the entire code block.	The <code>factorial(n)</code> function can be easily called multiple times within the same program or imported and used in other Python scripts.
Execution	For this specific task, the execution is similar in terms of toxic speed for typical inputs. Both use a loop to iterate and calculate the factorial.	The function introduces a slight overhead due to function call setup, but for this simple calculator, the difference is negligible. The main advantage here is code organization and readability, not execution speed.
In summary,	The program with the user-defined function is more structured, reusable, and generally considered better practice for larger or more complex programs. The program without functions is simpler for a single, straightforward task but lacks the benefits of modularity.	The program with the user-defined function is more structured, reusable, and generally considered better practice for larger or more complex programs. The program without functions is simpler for a single, straightforward task but lacks the benefits of modularity.

What can I help you build?

Genie can make mistakes so double-check it and use code with caution. [Learn more](#)

# TASK 5 :

The screenshot shows three separate code snippets in a Python code editor, each with its own tab in the top bar. The tabs are labeled 'Untitled8.ipynb'.

**Untitled8.ipynb - Tab 1:**

```
def factorial_iterative(n):
    """Calculates the factorial of a non-negative integer using an iterative approach (loop)."""
    Args:
        n: A non-negative integer.
    Returns:
        The factorial of n. Returns None if n is negative.
    """
    if n < 0:
        return None
    elif n == 0:
        return 1
    else:
        result = 1
        for i in range(1, n + 1):
            result *= i
        return result

def factorial_recursive(n):
    """Calculates the factorial of a non-negative integer using a recursive approach."""
    Args:
        n: A non-negative integer.
    Returns:
        The factorial of n. Returns None if n is negative.
    """

```

**Untitled8.ipynb - Tab 2:**

```
def factorial(n):
    """The factorial of n. Returns None if n is negative."""
    if n < 0:
        return None
    elif n == 0:
        return 1
    else:
        return n * factorial(n - 1)

# Ask the user to enter a number
num_str = input("Enter a non-negative integer: ")

# Convert the input to an integer and handle potential errors
try:
    num = int(num_str)

    # Calculate factorial using iterative method
    iterative_result = factorial_iterative(num)
    if iterative_result is not None:
        print(f"Factorial of {num} (Iterative): {iterative_result}")
    else:
        print("Iterative method: factorial is not defined for negative numbers.")

    # Calculate factorial using recursive method
    recursive_result = factorial_recursive(num)
    if recursive_result is not None:
        print(f"Factorial of {num} (Recursive): {recursive_result}")
    else:
        print("Recursive method: factorial is not defined for negative numbers.")

except ValueError:
    print("Invalid input. Please enter an integer.")

# Enter a non-negative integer: 10
# Factorial of 10 (Iterative): 3628800
# Factorial of 10 (Recursive): 3628800
```

**Untitled8.ipynb - Tab 3:**

```
num = int(num_str)

# Calculate factorial using iterative method
iterative_result = factorial_iterative(num)
if iterative_result is not None:
    print(f"Factorial of {num} (Iterative): {iterative_result}")
else:
    print("Iterative method: factorial is not defined for negative numbers.")

# Calculate factorial using recursive method
recursive_result = factorial_recursive(num)
if recursive_result is not None:
    print(f"Factorial of {num} (Recursive): {recursive_result}")
else:
    print("Recursive method: factorial is not defined for negative numbers.")

except ValueError:
    print("Invalid input. Please enter an integer.")

# Enter a non-negative integer: 10
# Factorial of 10 (Iterative): 3628800
# Factorial of 10 (Recursive): 3628800
```

**General X:**

- Write a Python program that shows two ways to find the factorial of a number: Iterative method – using a loop (for or while). Recursive method – where the function calls itself. The program should: Define two separate functions [factorial\_iterative() and factorial\_recursive()]. Ask the user to enter a number. Call both functions and display their results clearly for comparison, include comments or docstrings explaining each method.
- Okay, I can create a Python program that shows both the iterative and recursive methods for calculating the factorial of a number.  
Great! You now have a program that shows both the iterative and recursive ways to calculate factorials.  
Here are some things you could explore next:
  - Compare execution time of iterative vs. recursive factorial
  - Explore the Recursion Depth Exceeded error
  - Show how to implement memoization for the recursive factorial functionWhat can I help you build?

