

# **School of Computer Science and Artificial Intelligence**

---

## **Lab Assignment # 1.2**

---

Program : B. Tech (CSE)

Specialization : AIML

Course Title : AI Assisted Coding

Course Code : 23CS002PC304

Semester : VI

Academic Session : 2025-2026

Name of Student : T.INDRANEEL

Enrollment No. : 2303A52121

Batch No. : 33

**TASK \_01**

A screenshot of the Visual Studio Code (VS Code) interface. The workspace shows a file named `lab1.py` open in the editor. The code calculates the factorial of a number using loops and variables only. A Chat panel on the right provides AI-generated feedback and suggestions.

```
1 # calculation without using functions and variables only
2
3 print("Enter a number to find factorial")
4
5 n = int(input())
6 factorial = 1
7 i = 1
8
9 while i <= n:
10     factorial = factorial * i
11     i = i + 1
12
13 print("Factorial of", n, "is", factorial)
```

CHAT

CALCULATING FACTORIAL IN PYTHON WITHOUT FUNCTIONS

write a python program to calculate factorial without using functions . use loops and variables only.

✓ Created new Python file

Done! I've written a Python program that calculates factorial using only loops and variables. The program:

- Takes a number as input
- Uses a `while` loop to multiply all numbers from 1 to n
- Stores the result in the `factorial` variable
- Prints the result

You can run it and enter any number to calculate its factorial.

PS C:\Users\chirr\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chirr/Ram/ai lab/lab1.py"
Enter a number to find factorial: 5
Factorial of 5 is 120
PS C:\Users\chirr\Ram\ai lab>

Describe what to build next

Agent Auto

A screenshot of the Visual Studio Code (VS Code) interface, showing the same workspace as the first screenshot. The code has been refactored to use a function for calculating the factorial.

```
1 # Factorial calculation without using functions
2 # Using loops and variables only
3
4 n = int(input("Enter a number to find factorial: "))
5
6 factorial = 1
7 i = 1
8
9 while i <= n:
10     factorial = factorial * i
11     i = i + 1
12
13 print("Factorial of", n, "is", factorial)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\chirr\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chirr/Ram/ai lab/lab1.py"
Enter a number to find factorial: 5
Factorial of 5 is 120
PS C:\Users\chirr\Ram\ai lab>

## EXPLANATION:

GitHub Copilot helped me quickly generate a working factorial program.

It followed basic Python syntax and logic correctly.

For beginners, it saves time and reduces syntax errors.

However, understanding the logic is still important.

Copilot gives suggestions but human review is needed

## TASK\_02:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The code editor displays a Python script named `lab1.py`. The script contains a function to calculate factorial without using functions, handling invalid input with a try-except block, and using a while loop. A tooltip from GitHub Copilot suggests optimizing the code for readability and efficiency.

```
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

try:
    num = int(input("Enter a number to find factorial: "))
    print(f"Factorial of {num} is {factorial(num)}")
except ValueError as e:
    print(e)
```

The AI Chat pane shows the following interaction:

← CALCULATING FACTORIAL IN PYTHON WITHOUT FUNCTIONS

optimize this code to improve readability and efficiency

✓ Optimized factorial code for readability and efficiency  
✓ Edited lab1.py +20 -10

Done! I've optimized the code with these improvements:

**Efficiency:**

- Replaced `while` loop with `for` loop (more efficient in Python)
- Uses `range()` for cleaner iteration
- Uses `*=` operator instead of repeated multiplication

**Readability:**

- Added input validation to catch non-integer inputs
- Added error handling with `try-except`
- Added check for negative numbers
- Improved comments explaining each section
- Used f-string formatting for better output readability

Describe what to build next

Agent Auto 100% ↗ ➔

17°C Partly sunny

Search, Taskbar, System tray, Status bar: Ln 24, Col 1 Spaces: 4 UTF-8 Python 3.12.0 ENG IN 10:03 AM 10-01-2026

```
# Factorial calculation without using functions
# Using loops and variables only

# Input validation
try:
    n = int(input("Enter a non-negative number to find factorial: "))

    # Validate input
    if n < 0:
        print("Factorial is not defined for negative numbers!")
    else:
        # Initialize result
        factorial = 1

        # Calculate factorial using for loop
        for i in range(1, n + 1):
            factorial *= i

        # Display result
        print(f"Factorial of {n} is {factorial}")

except ValueError:
    print("Invalid input! Please enter a valid integer.")
```

```
# Input validation
try:
    n = int(input("Enter a non-negative number to find factorial: "))

    # Validate input
    if n < 0:
        print("Factorial is not defined for negative numbers!")
    else:
        # Initialize result
        factorial = 1

        # Calculate factorial using for loop
        for i in range(1, n + 1):
```

PS C:\Users\chirr\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chirr/Ram/ai lab/lab1.py"  
Enter a non-negative number to find factorial: 5  
Factorial of 5 is 120  
PS C:\Users\chirr\Ram\ai lab>

## Explanation

- Better variable names
- Cleaner output
- More readable

## TASK\_03:

The screenshot shows the Visual Studio Code interface. The Explorer sidebar has 'AI LAB' selected, showing 'lab1.py'. The main editor tab is also 'lab1.py'. The code is as follows:

```
24
25
26 # Main program
27 try:
28     n = int(input("Enter a non-negative number to find factorial: "))
29     result = calculate_factorial(n)
30     print(f"Factorial of {n} is {result}")
31
32 except ValueError as e:
33     print(f"Error: {e}")
34 except Exception as e:
35     print(f"Invalid input! Please enter a valid integer.")
```

Output in the terminal:

```
PS C:\Users\chirr\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chirr/Ram/ai lab/lab1.py"
Enter a non-negative number to find factorial: 5
Factorial of 5 is 120
PS C:\Users\chirr\Ram\ai lab>
```

CHAT panel:

- CALCULATING FACTORIAL IN PYTHON WITHOUT FUNCTIONS
- write a python program to calculate factorial using a function
- ✓ Read Python file
- ✓ Edited lab1.py +33 -21

Message: Done! I've rewritten the program to use a function. The improvements include:

Function-based approach:

- Created calculate\_factorial() function with proper documentation
- Handles edge cases (0 and 1 return 1 immediately)
- Raises ValueError for negative numbers

Better structure:

- Separates logic from user interaction
- Includes docstring explaining the function
- Cleaner error handling with specific exception messages
- The function can now be reused in other parts of a program

Bottom status bar: Agent Auto ENG IN 10:07 AM 10-01-2026

The screenshot shows the Visual Studio Code interface. The Explorer sidebar has 'AI LAB' selected, showing 'lab1.py'. The main editor tab is also 'lab1.py'. The code is as follows:

```
1 # Factorial calculation using a function
2
3 def calculate_factorial(num):
4     """
5         Calculate factorial of a number using a loop.
6
7     Args:
8         num: An integer for which factorial is to be calculated
9
10    Returns:
11        The factorial of the input number
12    """
13    if num < 0:
14        raise ValueError("Factorial is not defined for negative numbers!")
15
16    if num == 0 or num == 1:
17        return 1
18
19    factorial = 1
20    for i in range(2, num + 1):
21        factorial *= i
22
23    return factorial
24
25
26 # Main program
27 try:
28     n = int(input("Enter a non-negative number to find factorial: "))
29     result = calculate_factorial(n)
30     print(f"Factorial of {n} is {result}")
31
32 except ValueError as e:
33     print(f"Error: {e}")
34 except Exception as e:
35     print(f"Invalid input! Please enter a valid integer.")
```

Bottom status bar: 18°C Partly sunny Ln 36, Col 1 Spaces: 4 UTF-8 Python 3.12.0 10:08 AM ENG IN 10-01-2026

The screenshot shows a Python code editor interface with a dark theme. On the left is the Explorer sidebar, which has a tree view with 'AI LAB' expanded and 'lab1.py' selected. The main area is titled 'Welcome' and shows the content of 'lab1.py'. The code defines a function 'calculate\_factorial' that calculates the factorial of a number using a loop. It includes docstrings for arguments and returns. The terminal tab at the bottom shows the output of running the script, which prompts for a non-negative number, enters '5', and displays the result 'Factorial of 5 is 120'. The status bar at the bottom right shows the file is 3.12.0 and the date is 10-01-2026.

```
# Factorial calculation using a function
def calculate_factorial(num):
    """
    Calculate factorial of a number using a loop.

    Args:
        num: An integer for which factorial is to be calculated

    Returns:
        The factorial of the input number
    """
    if num < 0:
```

```
PS C:\Users\chirr\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chirr/Ram/ai lab/lab1.py"
Enter a non-negative number to find factorial: 5
Factorial of 5 is 120
PS C:\Users\chirr\Ram\ai lab>
```

## EXPLANATION:

Using functions improves reusability.

The same function can be used in many programs.

Code becomes cleaner and easier to maintain.

## TASK 04:

### Comparative Analysis – Procedural vs Modular AI Code

#### ***Procedural (Without Functions) vs Modular (With Functions)***

In Task 1, the factorial program was written using a procedural approach, where all the logic was implemented directly in the main execution flow without using any user-defined functions. In Task 3, the same logic was rewritten using a modular approach by creating a separate function to calculate the factorial. Both approaches produce

the same output, but they differ significantly in terms of design quality and usability.

### **Logic Clarity:**

The procedural version is simple and easy to understand for small programs. However, as the program grows, the logic becomes harder to follow because everything is written in one place. In contrast, the modular version separates the factorial logic into a function, making the code more organized and easier to read.

### **Reusability:**

The procedural code cannot be reused easily because the logic is tied to a single script. The modular version allows the factorial function to be reused in multiple programs without rewriting the same code, which saves time and effort.

### **Debugging Ease:**

Debugging procedural code is more difficult because errors can affect the entire program. In modular code, each function can be tested separately, making it easier to find and fix errors.

### **Suitability for Large Projects:**

Procedural code is suitable only for small, simple programs. For large projects, modular code is preferred because it supports better structure, teamwork, and maintenance.

### **AI Dependency Risk:**

When using AI tools like GitHub Copilot, procedural code may be generated quickly but often lacks proper structure. Modular code encourages better design practices, even when AI is used. This reduces the risk of poor-quality code.

### **Conclusion:**

While procedural programming is useful for quick tasks and learning basics, modular programming is more efficient, reusable, and suitable for real-world software development. Using functions improves clarity, maintainability, and scalability, making modular code the better choice for professional projects.

## TASK\_05:

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a file named `lab1.py` under the `AI LAB` folder.
- Code Editor:** Displays the content of `lab1.py`. The code includes two approaches for calculating factorial: an iterative approach using a loop and a recursive approach using a function. It also includes error handling for non-negative integers.
- Terminal:** Shows the command line output of running the script. It prints the path `PS C:\Users\chiru\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chiru/Ram/ai lab/lab1.py"`, prompts for a non-negative number, and then prints "Factorial of 5 is 120".
- Chat:** A sidebar titled "CALCULATING FACTORIAL IN PYTHON WITHOUT FUNCTIONS" contains a message from an AI assistant: "write factorial using iteration and recursion". Below it, the user's response is shown: "Created Python program for factorial calculations", "Edited lab1.py", and "Done! I've created a program with both approaches".
- Session Log:** Shows a session titled "Calculating factorial in Python without fun..." with a status of "Local • now". It lists "1 file +42 -15" and "Sorry, I can't assist with that. Finished in 15s. Local • 13 mins".
- Bottom Bar:** Includes standard system icons for search, taskbar, and system status.

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a file named `lab1.py` under the `AI LAB` folder.
- Code Editor:** Displays the content of `lab1.py`. This version of the script includes both iterative and recursive factorial functions. The iterative function uses a loop, while the recursive function calls itself. Both functions handle negative numbers by raising a ValueError.
- Terminal:** Shows the command line output of running the script. It prints the path `PS C:\Users\chiru\Ram\ai lab> & "C:/Program Files/Python312/python.exe" "c:/Users/chiru/Ram/ai lab/lab1.py"`, prompts for a non-negative number, and then prints "Factorial of 5 is 120".
- Bottom Bar:** Includes standard system icons for search, taskbar, and system status.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar has icons for Explorer, Search, Problems, and others. The main area shows a file named 'lab1.py' with the following code:

```
1 # Factorial calculation using both iterative and recursive approaches
2
3 def factorial_iterative(num):
4     """
5         Calculate factorial using iteration (loop).
6
7     Args:
8         num: A non-negative integer
9
10    Returns:
11        The factorial of the input number
12    """
13    if num < 0:
```

The terminal at the bottom shows the output of running the script with the command 'python lab1.py'. It prints 'Factorial of 5:' followed by 'Iterative approach: 120' and 'Recursive approach: 120'.

## Explanation

- Iterative uses loop
- Recursive calls itself
- Recursion uses more memory
- Iterative is faster