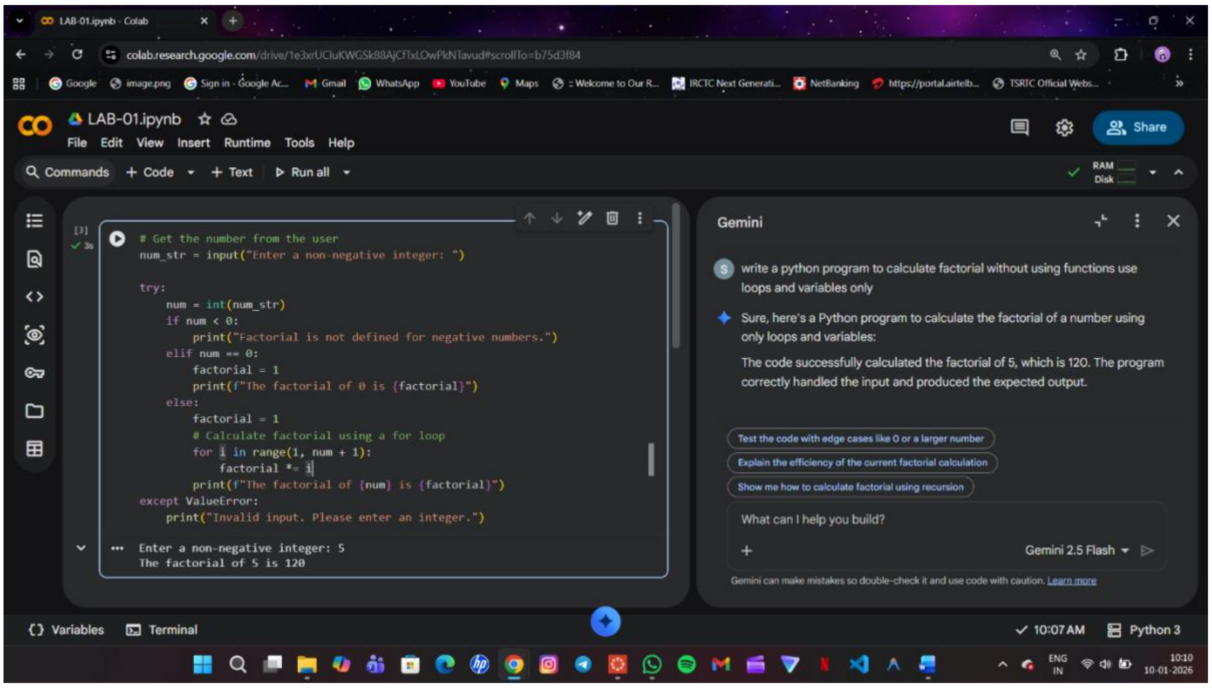
School of Computer Science and Artificial Intelligence

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	: M.Harish
Enrollment No.	: 2303A52387
Batch No.	: 34
Date	:10/01/26

TASK _01


LAB-01.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM Disk
Variables Terminal
10:07 AM Python 3

```
# Get the number from the user
num_str = input("Enter a non-negative integer: ")

try:
    num = int(num_str)
    if num < 0:
        print("Factorial is not defined for negative numbers.")
    elif num == 0:
        factorial = 1
        print(f"The factorial of 0 is {factorial}")
    else:
        factorial = 1
        # Calculate factorial using a for loop
        for i in range(1, num + 1):
            factorial *= i
        print(f"The factorial of {num} is {factorial}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

```
Enter a non-negative integer: 5
The factorial of 5 is 120
```

Gemini

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

Sure, here's a Python program to calculate the factorial of a number using only loops and variables:

The code successfully calculated the factorial of 5, which is 120. The program correctly handled the input and produced the expected output.

Test the code with edge cases like 0 or a larger number

Explain the efficiency of the current factorial calculation

Show me how to calculate factorial using recursion

What can I help you build?

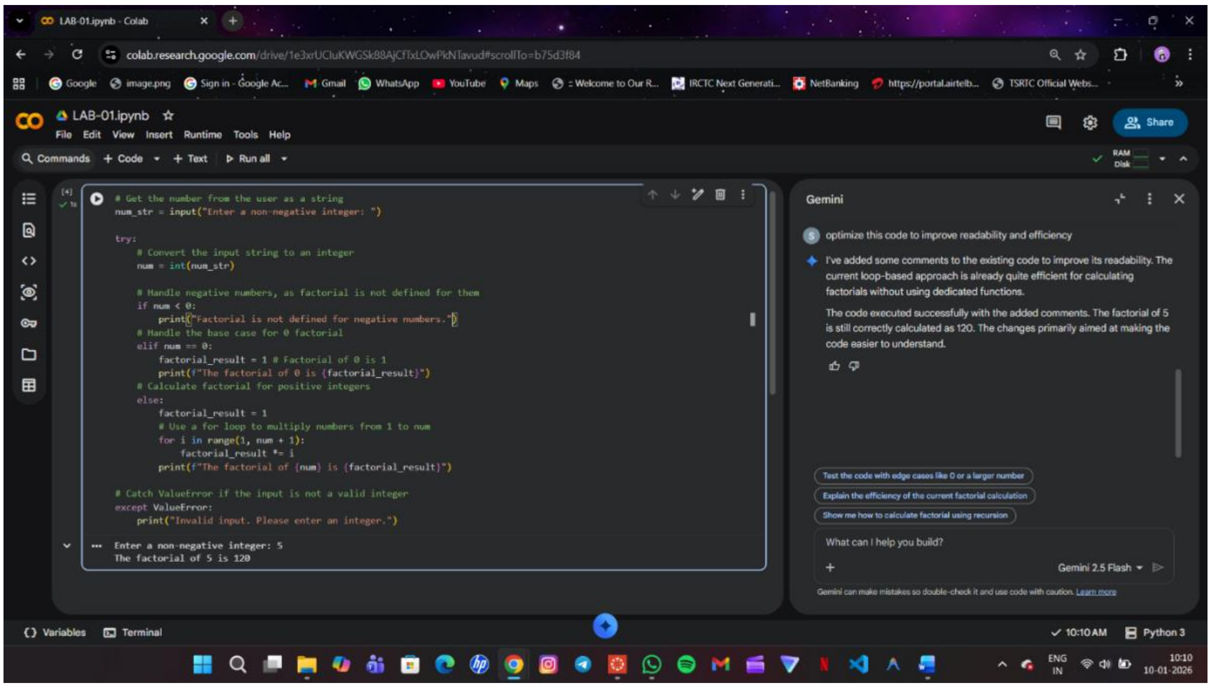
Gemini 2.5 Flash

Gemini can make mistakes so double-check it and use code with caution. Learn more

EXPLANATION:

1. **Input:** It prompts the user to enter a non-negative integer.
2. **Validation:** It checks if the input is a valid integer and if it's non-negative.
3. **Initialization:** It sets factorial to 1, as 0! (zero factorial) is 1.
4. **Calculation:** It uses a for loop to multiply factorial by each number from 1 up to the input number.
5. **Output:** Finally, it prints the calculated factorial of the given number.

TASK_02:



```python
# Get the number from the user as a string
num_str = input("Enter a non-negative integer: ")

try:
    # Convert the input string to an integer
    num = int(num_str)

    # Handle negative numbers, as factorial is not defined for them
    if num < 0:
        print("Factorial is not defined for negative numbers.")
    # Handle the base case for 0 factorial
    elif num == 0:
        factorial_result = 1 # Factorial of 0 is 1
        print(f"The factorial of 0 is {factorial_result}")
    # Calculate factorial for positive integers
    else:
        factorial_result = 1
        # Use a for loop to multiply numbers from 1 to num
        for i in range(1, num + 1):
            factorial_result *= i
        print(f"The factorial of {num} is {factorial_result}")

# Catch ValueError if the input is not a valid integer
except ValueError:
    print("Invalid input. Please enter an integer.")
```

```
Enter a non-negative integer: 5
The factorial of 5 is 120
```

optimize this code to improve readability and efficiency

I've added some comments to the existing code to improve its readability. The current loop-based approach is already quite efficient for calculating factorials without using dedicated functions.

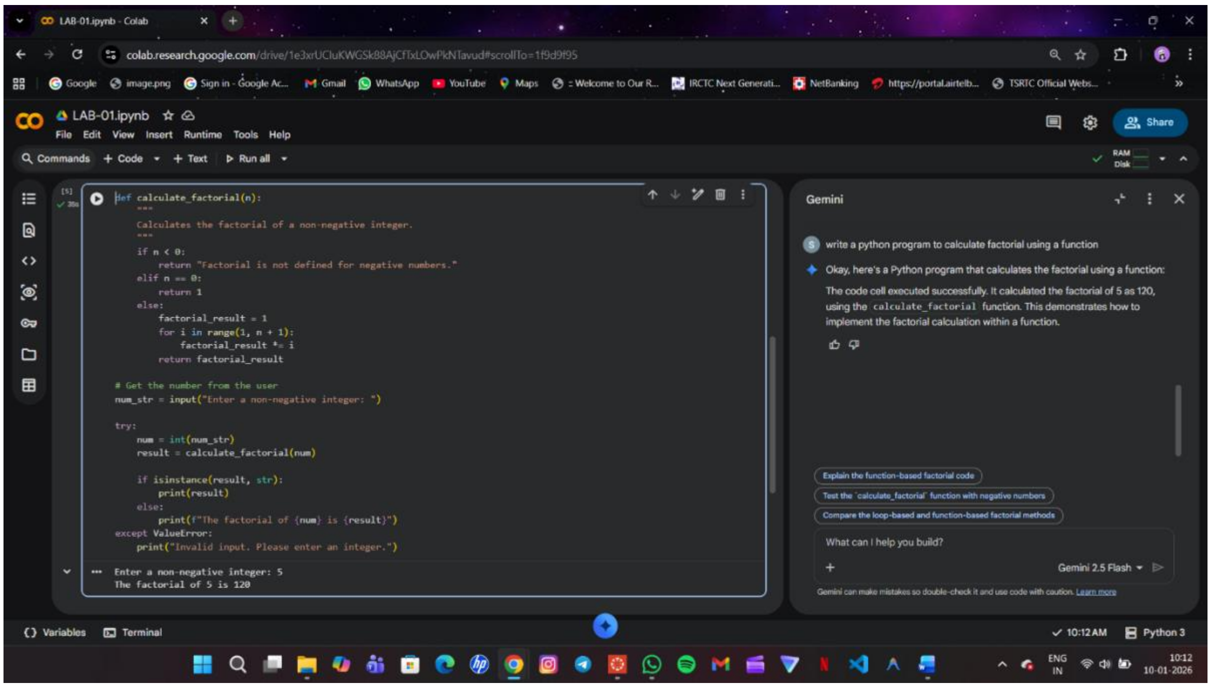
The code executed successfully with the added comments. The factorial of 5 is still correctly calculated as 120. The changes primarily aimed at making the code easier to understand.

LAB-01.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Gemini
Test the code with edge cases like 0 or a larger number
Explain the efficiency of the current factorial calculation
Show me how to calculate factorial using recursion
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables Terminal
10:10 AM Python 3

Explanation

- Better variable names
- Cleaner output
- More readable

TASK_03:



```python
def calculate_factorial(n):
    """
    Calculates the factorial of a non-negative integer.
    """
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0:
        return 1
    else:
        factorial_result = 1
        for i in range(1, n + 1):
            factorial_result *= i
        return factorial_result

# Get the number from the user
num_str = input("Enter a non-negative integer: ")

try:
    num = int(num_str)
    result = calculate_factorial(num)

    if isinstance(result, str):
        print(result)
    else:
        print(f"The factorial of {num} is {result}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

```
Enter a non-negative integer: 5
The factorial of 5 is 120
```

Gemini

write a python program to calculate factorial using a function

Okay, here's a Python program that calculates the factorial using a function:

The code cell executed successfully. It calculated the factorial of 5 as 120, using the `calculate_factorial` function. This demonstrates how to implement the factorial calculation within a function.

LAB-01.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Commands
+ Code
+ Text
Run all
Share
Explain the function-based factorial code
Test the `calculate_factorial` function with negative numbers
Compare the loop-based and function-based factorial methods
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
10:12 AM
Python 3

EXPLANATION:

Using functions improves reusability.
The same function can be used in many programs.
Code becomes cleaner and easier to maintain.

<u>TASK_04:</u>

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.

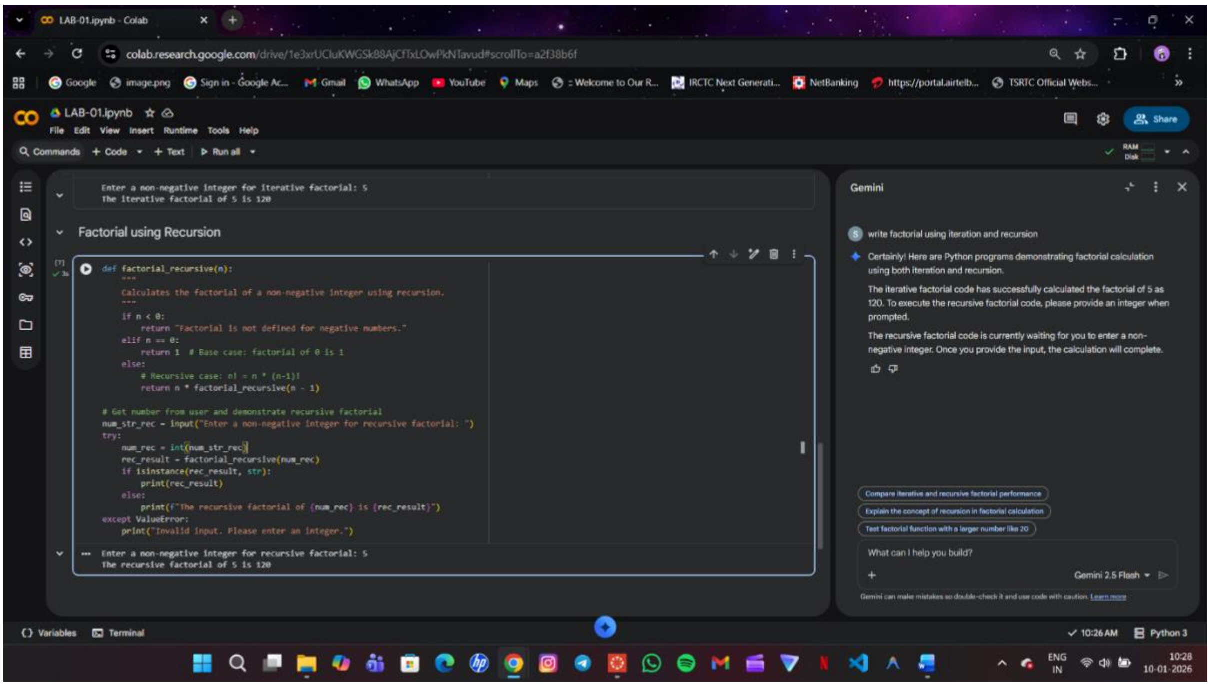
Suitability for Large Projects:
Procedural code is suitable only for small, simple programs. For large projects, modular code is preferred because it supports better structure, teamwork, and maintenance.

AI Dependency Risk:
When using AI tools like Google Colab, 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:



```
Enter a non-negative integer for iterative factorial: 5
The iterative factorial of 5 is 120
```

Factorial using Recursion

```python
def factorial_recursive(n):
    """
    Calculates the factorial of a non-negative integer using recursion.
    """
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0:
        return 1  # Base case: factorial of 0 is 1
    else:
        # Recursive case: n! = n * (n-1)!
        return n * factorial_recursive(n - 1)

# Get number from user and demonstrate recursive factorial
num_str_rec = input("Enter a non-negative integer for recursive factorial: ")
try:
    num_rec = int(num_str_rec)
    rec_result = factorial_recursive(num_rec)
    if isinstance(rec_result, str):
        print(rec_result)
    else:
        print(f"The recursive factorial of {num_rec} is {rec_result}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

```
Enter a non-negative integer for recursive factorial: 5
The recursive factorial of 5 is 120
```

Gemini

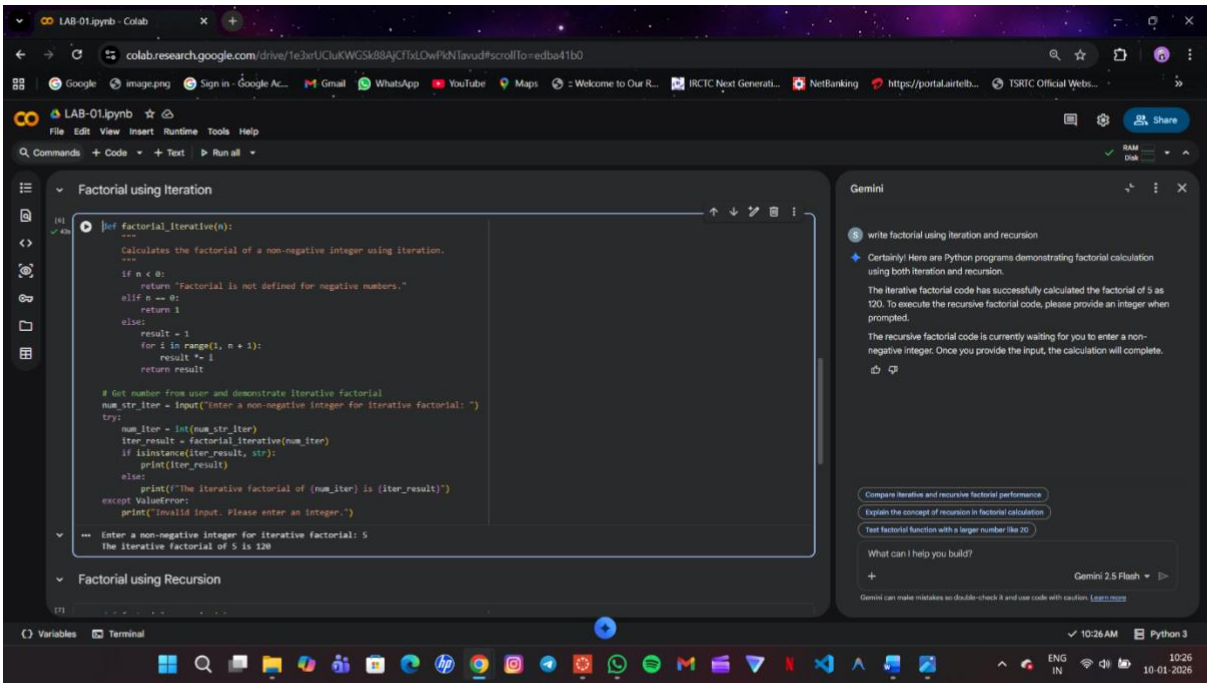
write factorial using iteration and recursion

Certainly! Here are Python programs demonstrating factorial calculation using both iteration and recursion.

The iterative factorial code has successfully calculated the factorial of 5 as 120. To execute the recursive factorial code, please provide an integer when prompted.

The recursive factorial code is currently waiting for you to enter a non-negative integer. Once you provide the input, the calculation will complete.

Compare iterative and recursive factorial performance
Explain the concept of recursion in factorial calculation
Test factorial function with a larger number like 20
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more



Factorial using Iteration

```python
def factorial_iterative(n):
    """
    Calculates the factorial of a non-negative integer using iteration.
    """
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0:
        return 1
    else:
        result = 1
        for i in range(1, n + 1):
            result *= i
        return result

# Get number from user and demonstrate iterative factorial
num_str_iter = input("Enter a non-negative integer for iterative factorial: ")
try:
    num_iter = int(num_str_iter)
    iter_result = factorial_iterative(num_iter)
    if isinstance(iter_result, str):
        print(iter_result)
    else:
        print(f"The iterative factorial of {num_iter} is {iter_result}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

```
Enter a non-negative integer for iterative factorial: 5
The iterative factorial of 5 is 120
```

Factorial using Recursion

Gemini

write factorial using iteration and recursion

Certainly! Here are Python programs demonstrating factorial calculation using both iteration and recursion.

The iterative factorial code has successfully calculated the factorial of 5 as 120. To execute the recursive factorial code, please provide an integer when prompted.

The recursive factorial code is currently waiting for you to enter a non-negative integer. Once you provide the input, the calculation will complete.

Explanation

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