LAB ASSIGNMENT - 1.1

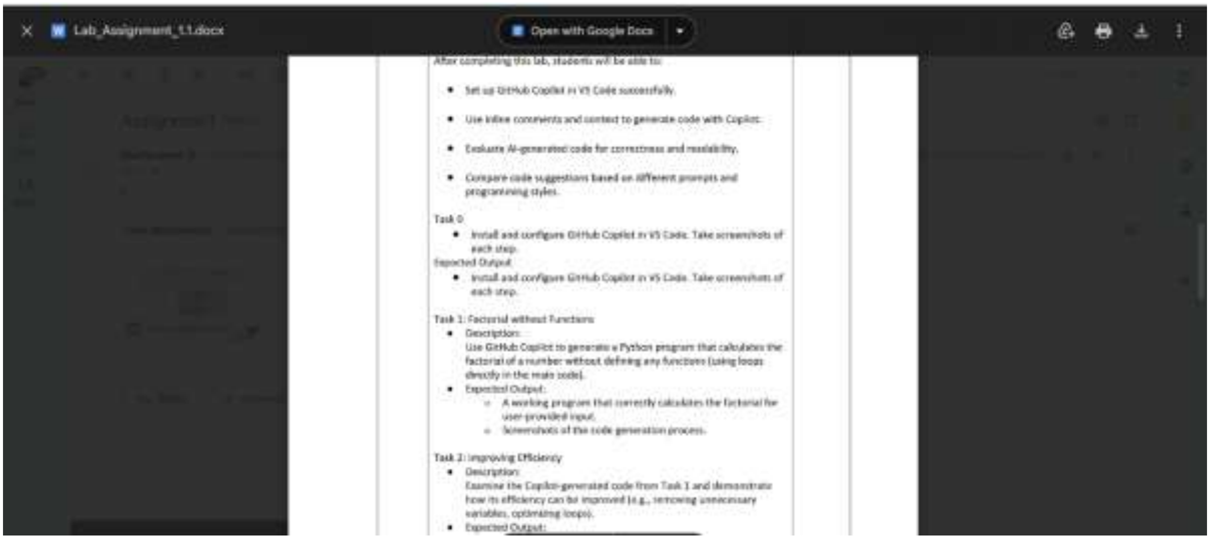
NAME : VADIYALA CHANDANA

ROLL NO : 2403A54103

COURSE : AI ASSISTED CODING

BATCH : 01

QUESTIONS



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

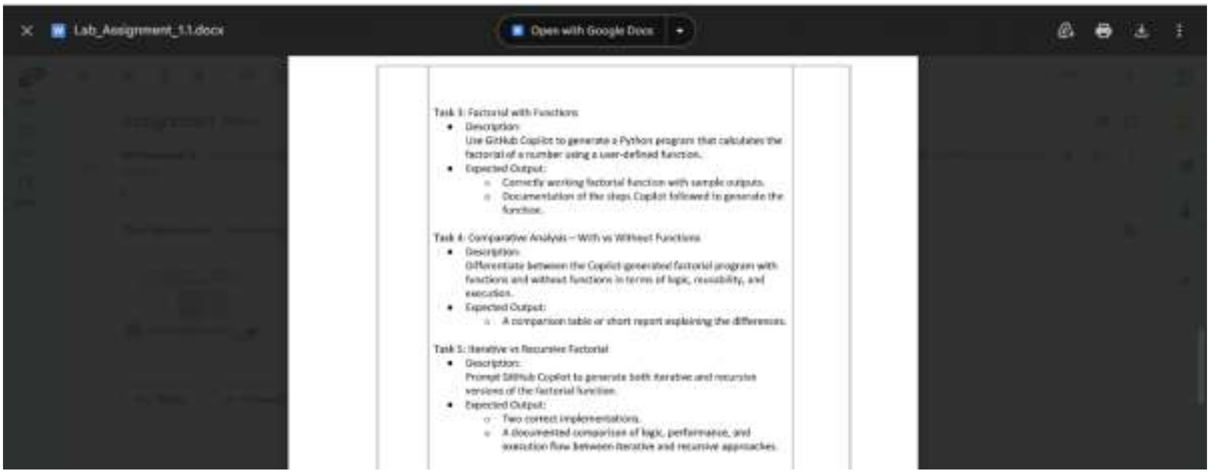
- Install and configure GitHub Copilot in VS Code. Take screenshots of each step.

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:



Task 3: 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.

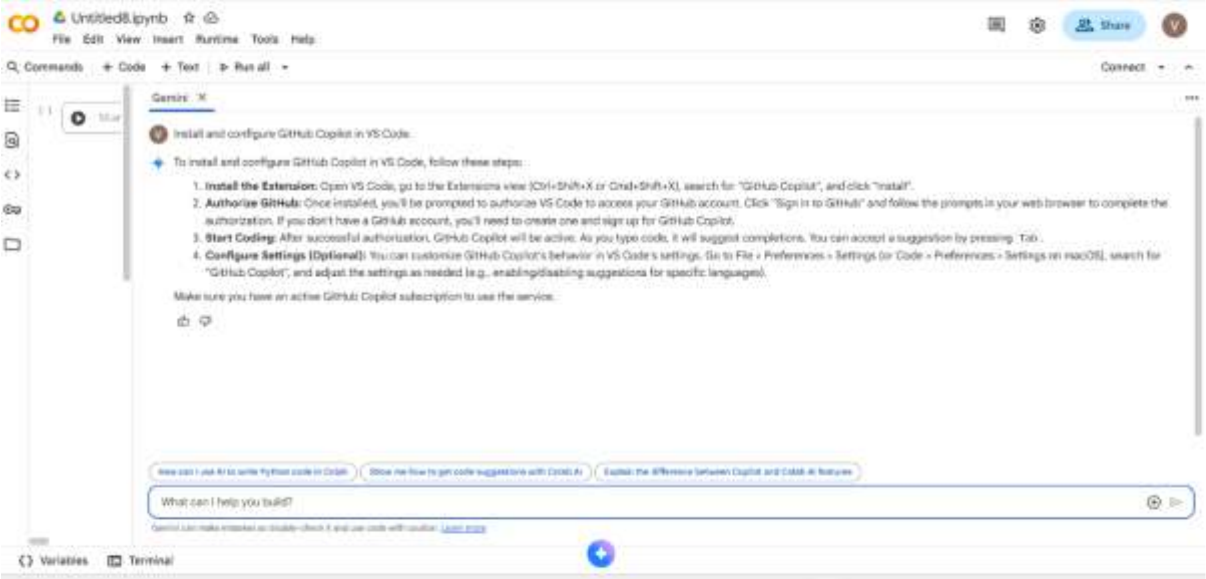
Task 4: Comparative Analysis – With vs Without Functions

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

Task 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.

TASK 0:


Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
Connect
Gemini
How can I use AI to write Python code in Colab
Show me how to get code suggestions with Colab AI
Explain the difference between Copilot and Colab AI features
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal

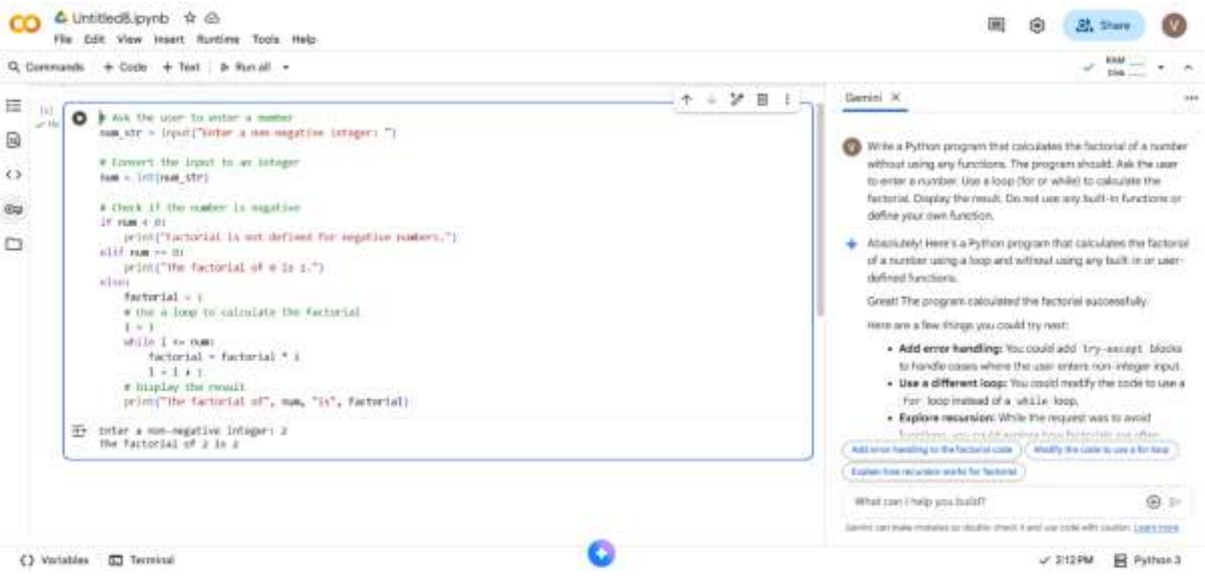
Install and configure GitHub Copilot in VS Code.

To install and configure GitHub Copilot in VS Code, follow these steps:

1. **Install the Extension:** Open VS Code, go to the Extensions view (Ctrl+Shift+X or Cmd+Shift+X), search for "GitHub Copilot", and click "Install".
2. **Authorize GitHub:** Once installed, you'll be prompted to authorize VS Code to access your GitHub account. Click "Sign in to GitHub" and follow the prompts in your web browser to complete the authorization. If you don't have a GitHub account, you'll need to create one and sign up for GitHub Copilot.
3. **Start Coding:** After successful authorization, GitHub Copilot will be active. As you type code, it will suggest completions. You can accept a suggestion by pressing `Tab`.
4. **Configure Settings (Optional):** You can customize GitHub Copilot's behavior in VS Code's settings. Go to File > Preferences > Settings (or Code > Preferences > Settings on macOS), search for "GitHub Copilot", and adjust the settings as needed (e.g., enabling/disabling suggestions for specific languages).

Make sure you have an active GitHub Copilot subscription to use the service.

TASK 1 :


Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Share
RAM
Disk
Gemini
Add error handling to the factorial code
Modify the code to use a for loop
Explain how recursion works for factorial
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
3:12 PM
Python 3

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

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

# 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
    i = 1
    while i <= num:
        factorial = factorial * i
        i = i + 1
    # Display the result
    print("The factorial of", num, "is", factorial)
```

```
Enter a non-negative integer: 2
The factorial of 2 is 2
```

Write a Python program that calculates the factorial of a number without using any functions. The program should: Ask the user to enter a number. Use a loop (for or while) to calculate the factorial. Display the result. Do not use any built-in functions or define your own function.

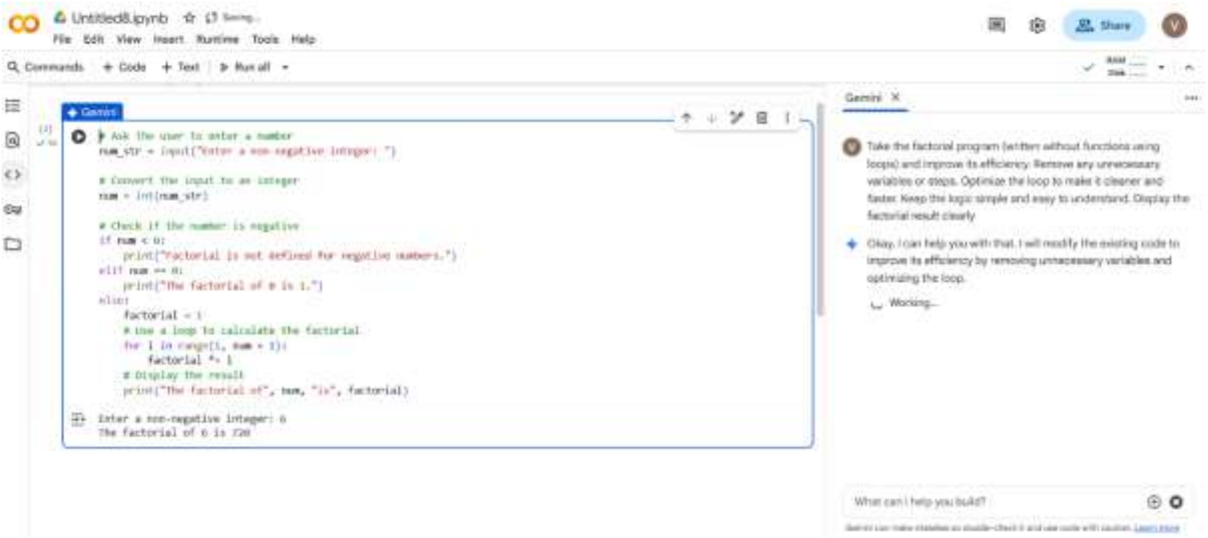
Absolutely! Here's a Python program that calculates the factorial of a number using a loop and without using any built-in or user-defined functions.

Great! The program calculated the factorial successfully.

Here are a few things you could try next:

- **Add error handling:** You could add `try-except` blocks to handle cases where the user enters non-integer input.
- **Use a different loop:** You could modify the code to use a `for` loop instead of a `while` loop.
- **Explore recursion:** While the request was to avoid functions, you could explore how factorials are often

TASK 2 :



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

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

# 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)
```

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

Gemini

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

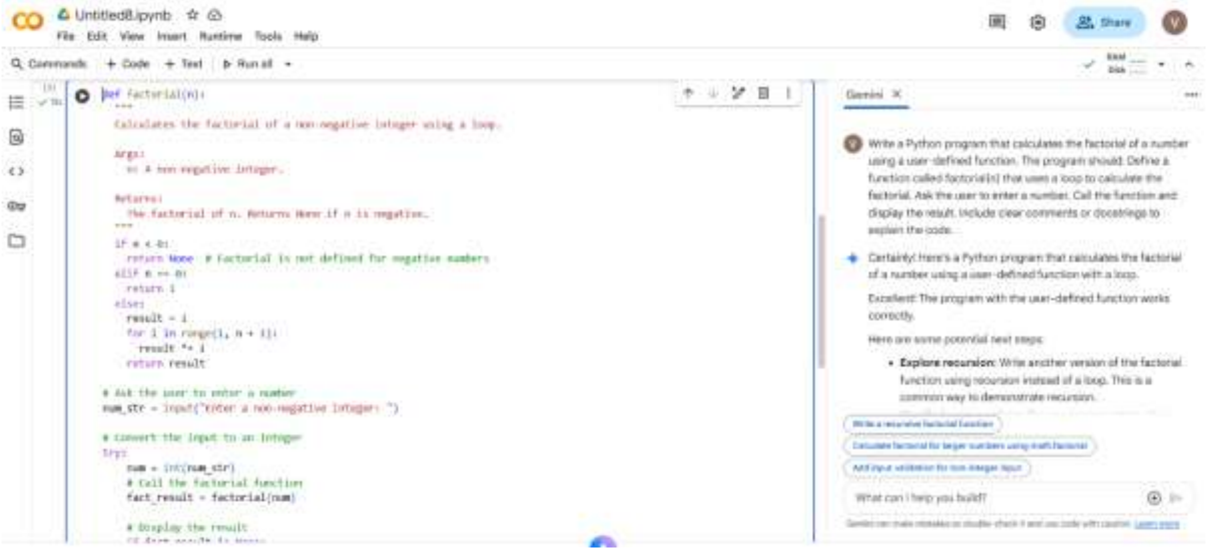
Okay, I can help you with that. I will modify the existing code to improve its efficiency by removing unnecessary variables and optimizing the loop.

Working...

What can I help you build?

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

TASK 3 :



```
def factorial(n):
  """
  Calculates the factorial of a non-negative integer using a loop.

  Args:
    n: A non-negative integer.

  Returns:
    The factorial of n. Returns None if n is negative.
  """
  if n < 0:
    return None  # 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

# 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:
```

Gemini

Write a Python program that calculates the factorial of a number using a user-defined function. The program should: Define a function called factorial(n) 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 docstrings to explain the code.

Certainly! 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 recursion:** Write another version of the factorial function using recursion instead of a loop. This is a common way to demonstrate recursion.

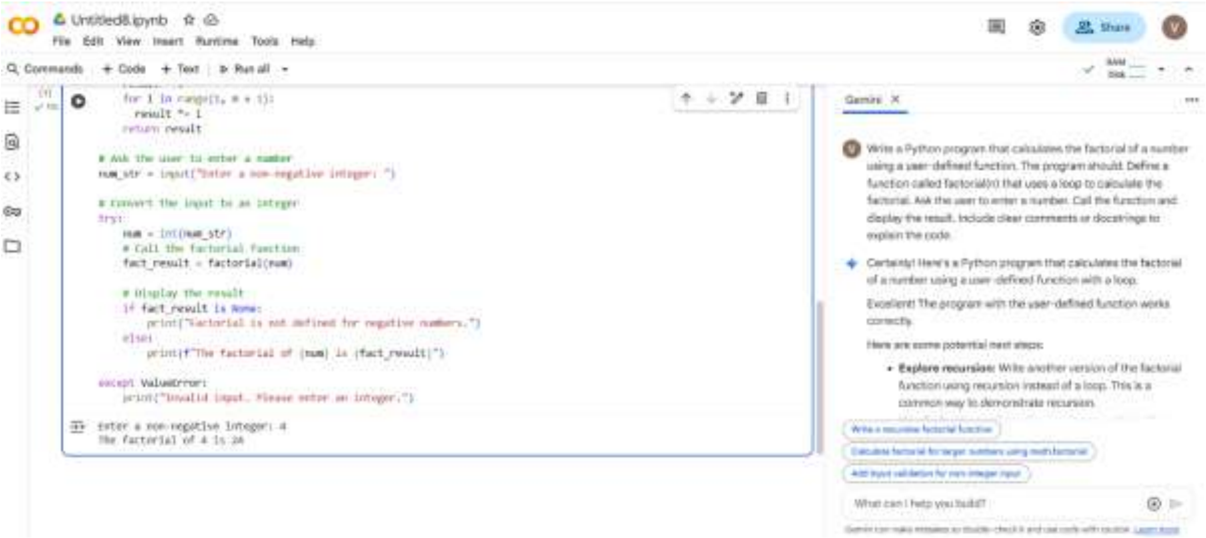
Write a recursive factorial function

Calculate factorial for larger numbers using math.factorial

Add input validation for non-integer input

What can I help you build?

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



Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
Share
RAM
Disk
Gemini ✕

```
    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
```

Write a Python program that calculates the factorial of a number using a user-defined function. The program should: Define a function called factorial(n) 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 docstrings to explain the code.

Certainly! Here's a Python program that calculates the factorial of a number using a user-defined function with a loop.

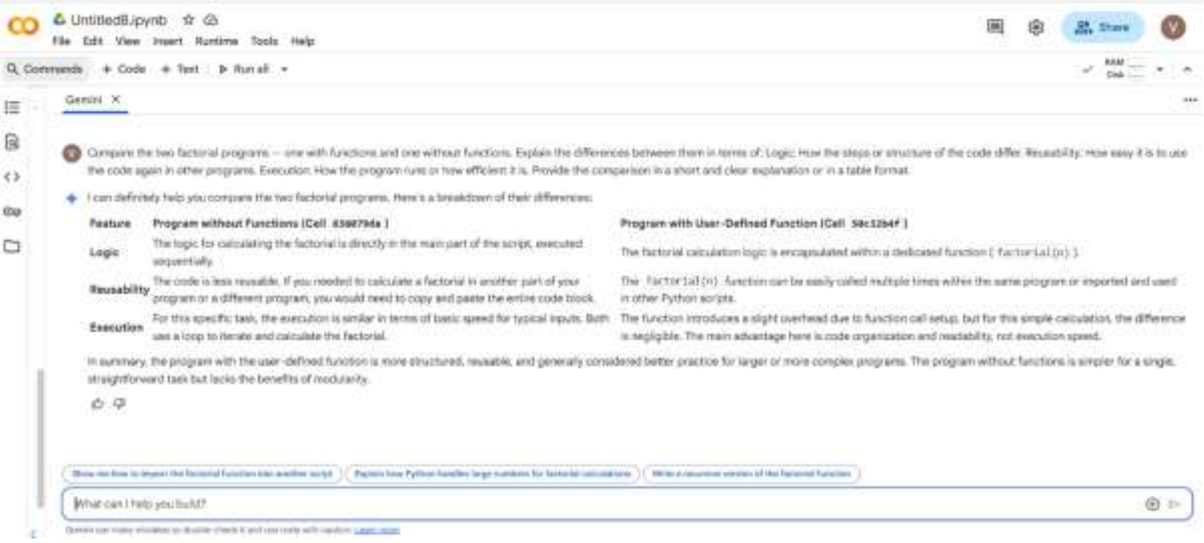
Excellent! The program with the user-defined function works correctly.

Here are some potential next steps:

- **Explore recursion:** 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.factorial
Add input validation for non-integer input
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more

TASK 4 :



Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
Share
RAM
Disk
Gemini ✕

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. Reusability: 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.

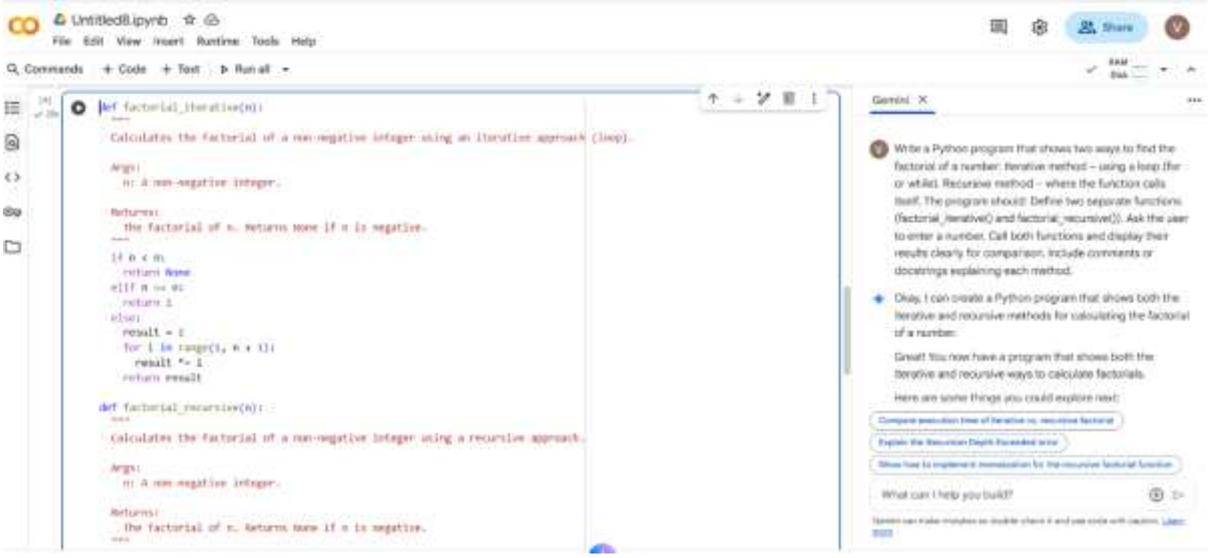
I can definitely help you compare the two factorial programs. Here's a breakdown of their differences:

Feature	Program without Functions (Cell 6388794a)	Program with User-Defined Function (Cell 58c3264f)
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 (factorial(n)).
Reusability	The code is less reusable. 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 factorial(n) 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 basic 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 calculation, 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.

Show me how to import the factorial function into another script
Explain how Python handles large numbers for factorial calculations
Write a recursive version of the factorial function
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more

TASK 5 :



Untitled8.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▶ Run all
Share
RAM Disk
Gemini

```
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.
  """
```

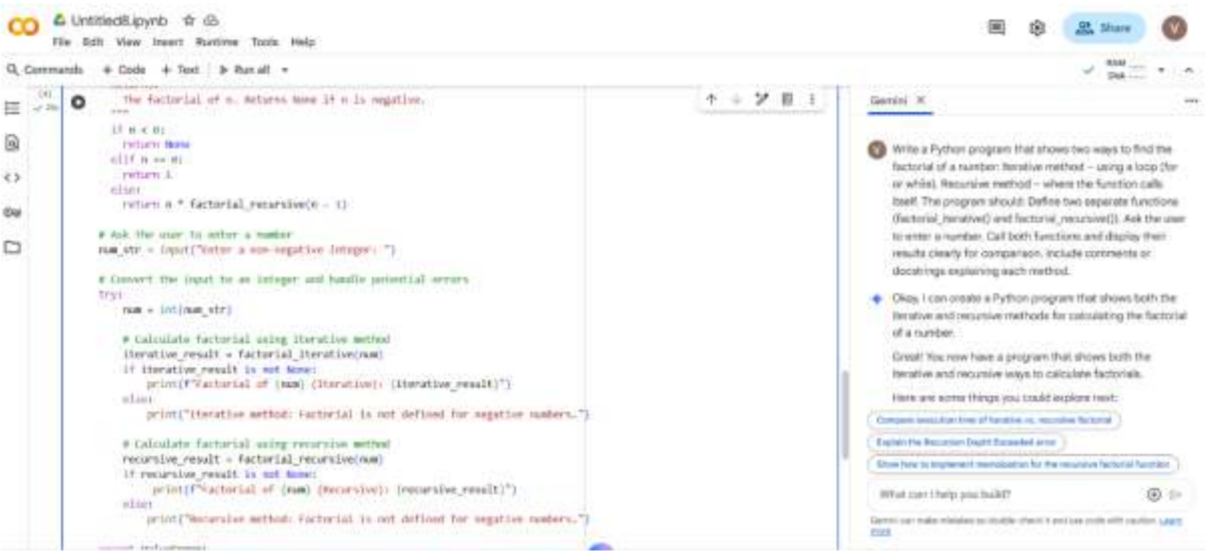
Write a Python program that shows two ways to find the factorial of a number: Iterative method – using a loop (for or while). Recursive method – where the function calls itself. The program should: Define two separate functions (factorial_iterative() and factorial_recursive()). Ask the user to enter 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
Explain the Recursion Depth Exceeded error
Show how to implement memoization for the recursive factorial function
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more



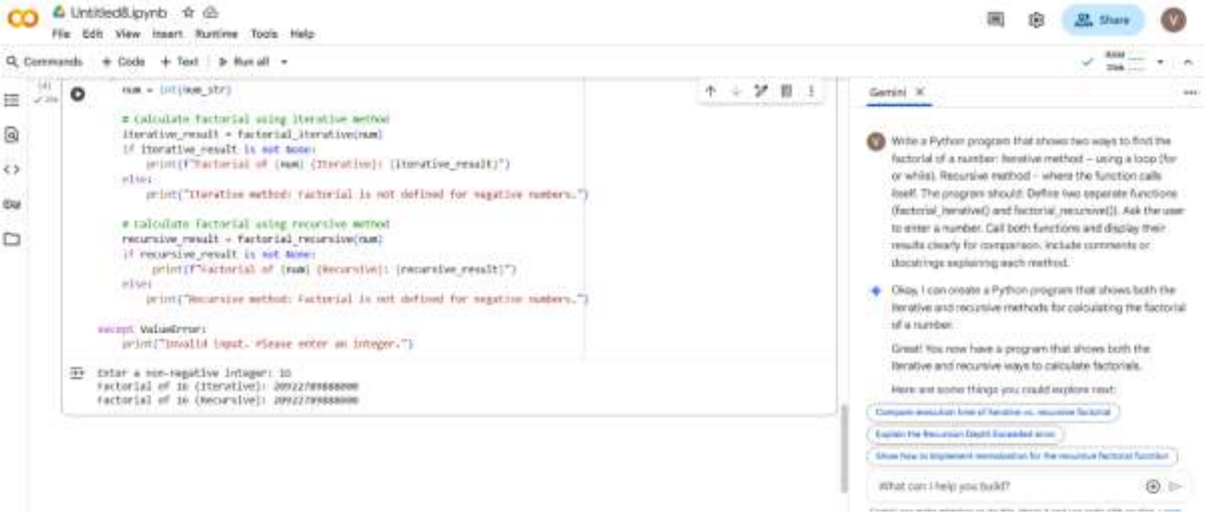
```
    The factorial of n. Returns None if n is negative.
  """
  if n < 0:
    return None
  elif n == 0:
    return 1
  else:
    return n * factorial_recursive(n - 1)

# 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.")
```

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:



```
    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: 16
Factorial of 16 (Iterative): 20922789888000
Factorial of 16 (Recursive): 20922789888000
```

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: