

AI ASSISTED CODING

Program :B.tech(CSE)
Name :ANANTHA MANIDEEP
En No. :2403A52078
Batch No. :02

LAB ASSIGNMENT-03

Task Description -1:

Try 3 different prompts to generate a factorial function.

Prompt-1:

Write a python function to find the factorial of a number using recursion.

Question:



Code:



The screenshot shows a Jupyter Notebook interface with a file named 'Untitled3.ipynb'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu, there are tabs for 'Commands', '+ Code', '+ Text', and 'Run all'. The main editor area contains the following Python code:

```
def factorial_recursive(n):  
    """Calculates the factorial of a non-negative integer using recursion."""  
    if n == 0:  
        return 1  
    else:  
        return n * factorial_recursive(n-1)  
  
# Example usage:  
number = 5  
result = factorial_recursive(number)  
print(f"The factorial of {number} is {result}")
```

Output:



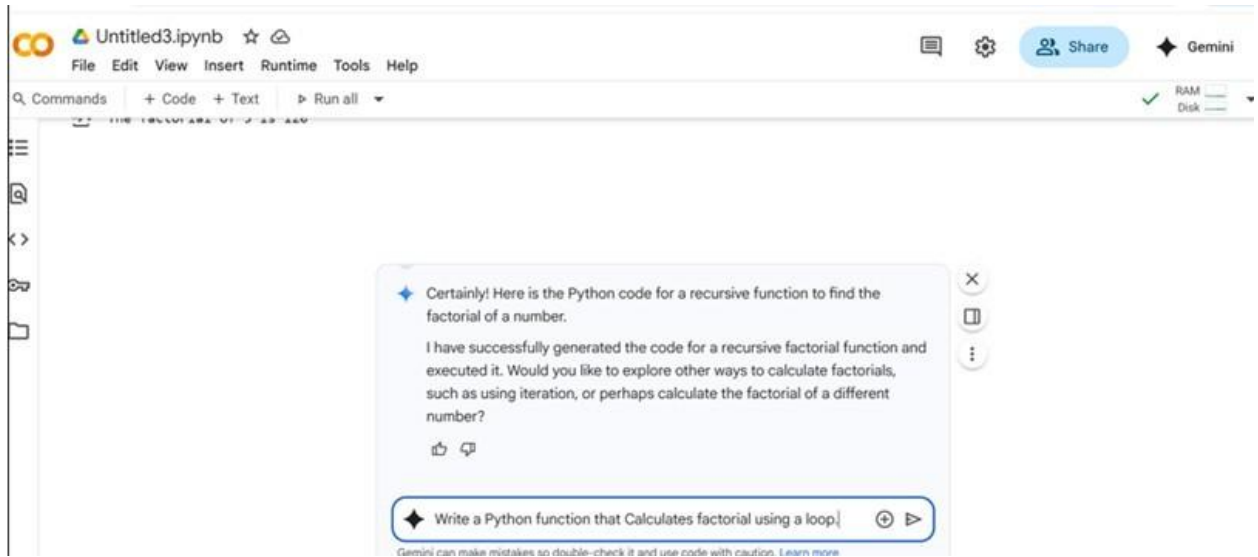
The screenshot shows the same Jupyter Notebook interface, but now the code has been executed. The output of the execution is displayed below the code cell:

```
[1] def factorial_recursive(n):  
    """Calculates the factorial of a non-negative integer using recursion."""  
    if n == 0:  
        return 1  
    else:  
        return n * factorial_recursive(n-1)  
  
# Example usage:  
number = 5  
result = factorial_recursive(number)  
print(f"The factorial of {number} is {result}")  
  
The factorial of 5 is 120
```

Prompt-2:

Write a python function that calculates factorial to find the factorial of a number.

Question:



Code:

```
[ ] 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  
  
# Example usage:  
number = 5  
result = factorial_iterative(number)  
print(f"The factorial of {number} is {result}")  
  
number = -3  
result = factorial_iterative(number)  
print(f"The factorial of {number} is {result}")
```

Output:

The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. The code cell contains a function `factorial_iterative(n)` that calculates the factorial of a non-negative integer `n` using iteration. The function returns 'Factorial is not defined for negative numbers' for `n < 0`, and 1 for `n == 0`. For `n > 0`, it uses a loop to calculate the factorial. Below the function, there are two example usages: one for `number = 5` which prints 'The factorial of 5 is 120', and one for `number = -3` which prints 'The factorial of -3 is Factorial is not defined for negative numbers'. The output cell shows the results of these two calls.

```
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

# Example usage:
number = 5
result = factorial_iterative(number)
print(f"The factorial of {number} is {result}")

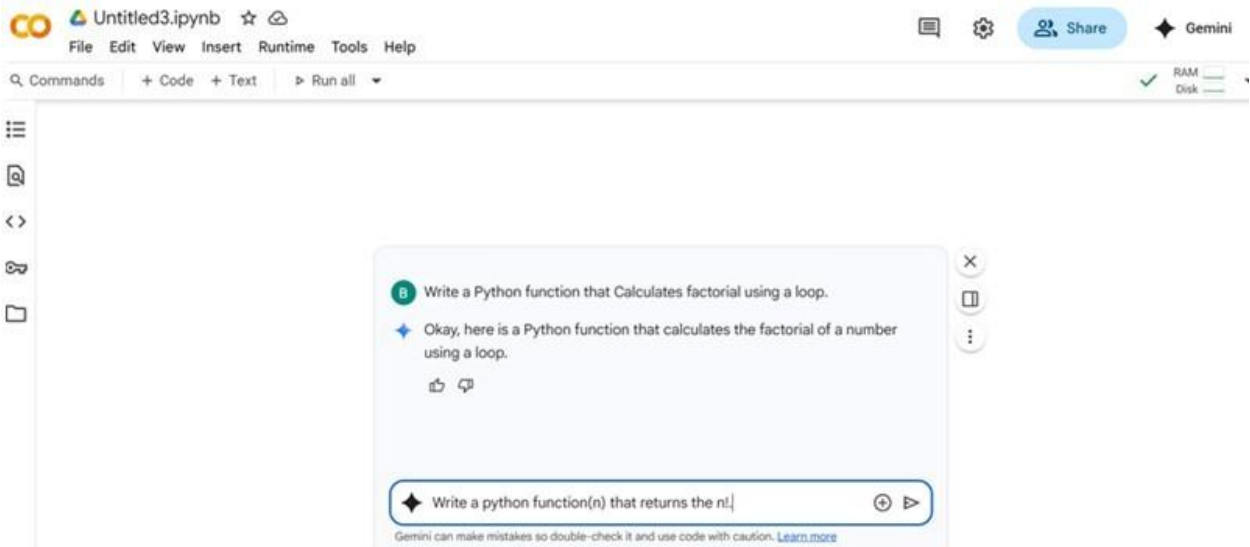
number = -3
result = factorial_iterative(number)
print(f"The factorial of {number} is {result}")
```

The factorial of 5 is 120
The factorial of -3 is Factorial is not defined for negative numbers

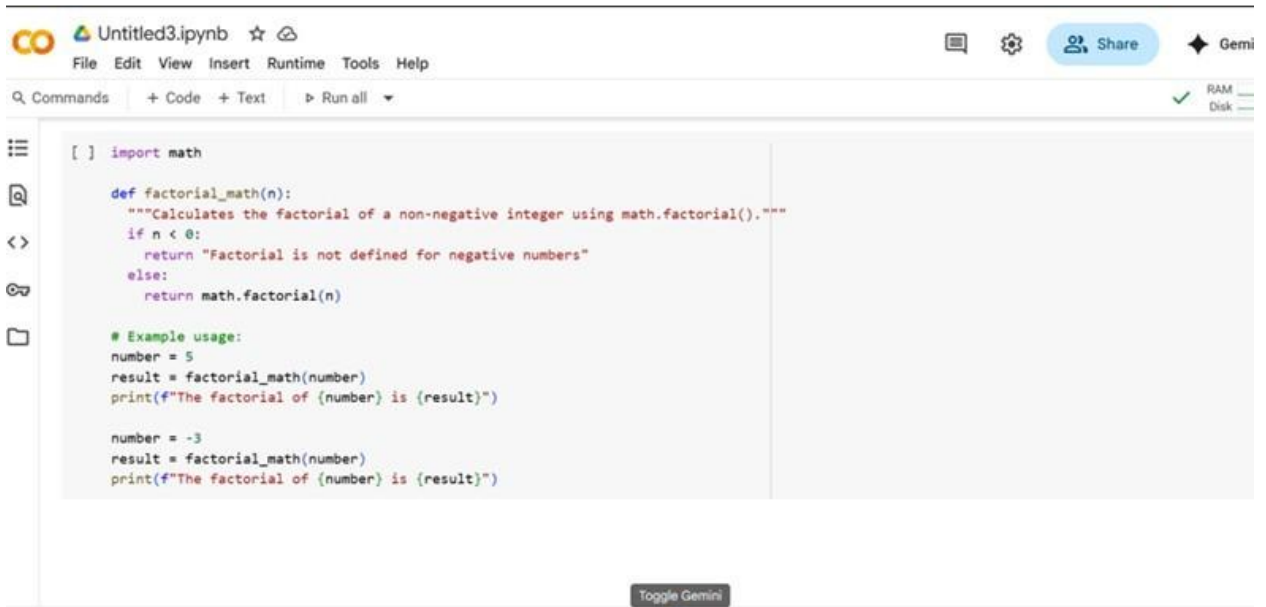
Prompt-3:

Write a python function() that returns the n!

Question:



Code:



The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. The code defines a function `factorial_math(n)` that uses `math.factorial()` to calculate the factorial of a non-negative integer. It includes a docstring, a check for negative numbers, and example usage for `number = 5` and `number = -3`. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with icons for commands, code, text, and running, and a status bar showing RAM and disk usage. A 'Toggle Gemini' button is visible at the bottom.

```
[ ] import math

def factorial_math(n):
    """Calculates the factorial of a non-negative integer using math.factorial()."""
    if n < 0:
        return "Factorial is not defined for negative numbers"
    else:
        return math.factorial(n)

# Example usage:
number = 5
result = factorial_math(number)
print(f"The factorial of {number} is {result}")

number = -3
result = factorial_math(number)
print(f"The factorial of {number} is {result}")
```

Output:



The screenshot shows the same Jupyter Notebook after execution. The code is identical to the previous image. The output area at the bottom displays the results of the function calls: 'The factorial of 5 is 120' and 'The factorial of -3 is Factorial is not defined for negative numbers'. The interface includes a menu bar, a toolbar with additional icons for undo, redo, and search, and a status bar.

```
import math

def factorial_math(n):
    """Calculates the factorial of a non-negative integer using math.factorial()."""
    if n < 0:
        return "Factorial is not defined for negative numbers"
    else:
        return math.factorial(n)

# Example usage:
number = 5
result = factorial_math(number)
print(f"The factorial of {number} is {result}")

number = -3
result = factorial_math(number)
print(f"The factorial of {number} is {result}")
```

The factorial of 5 is 120
The factorial of -3 is Factorial is not defined for negative numbers

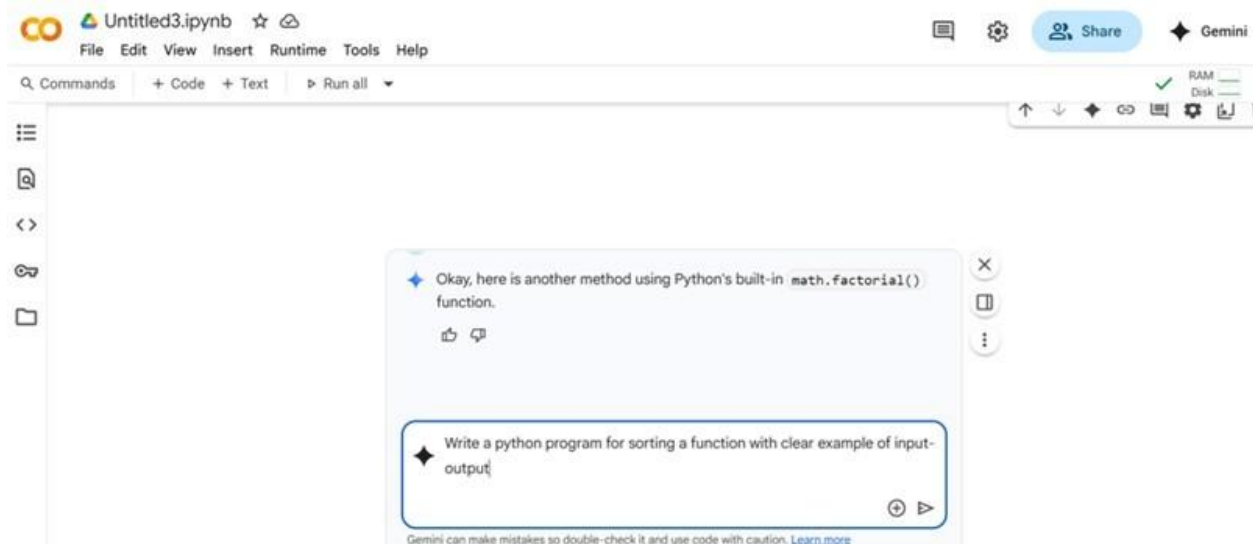
Task Description-2:

Provide a clear example input-output prompt to generate a sorting function.

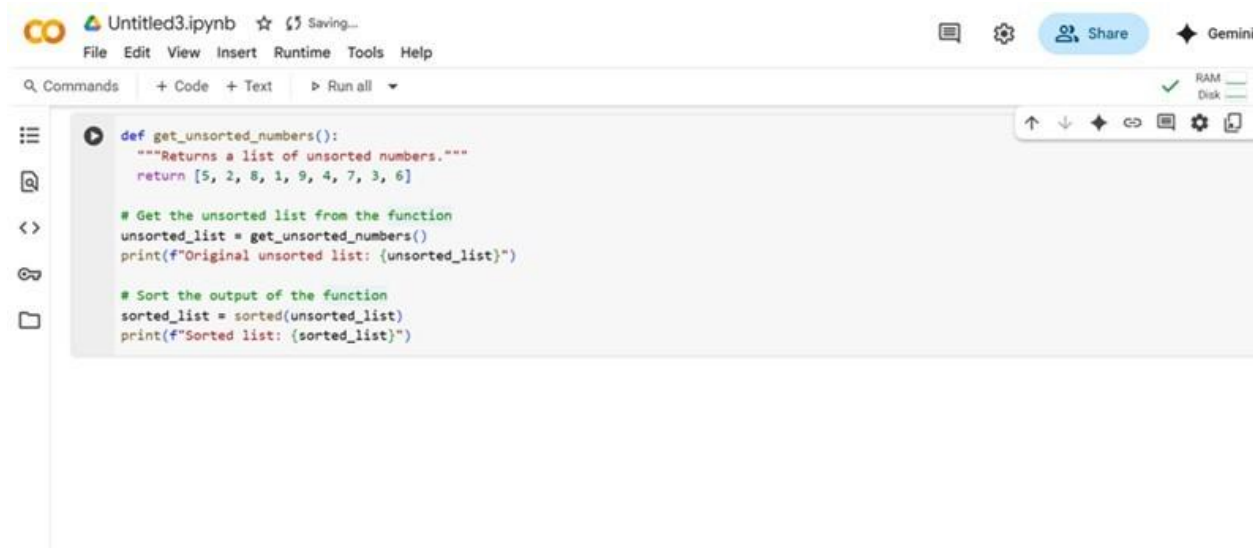
Prompt-2:

Write a python program using python built-in
math.factorial() function

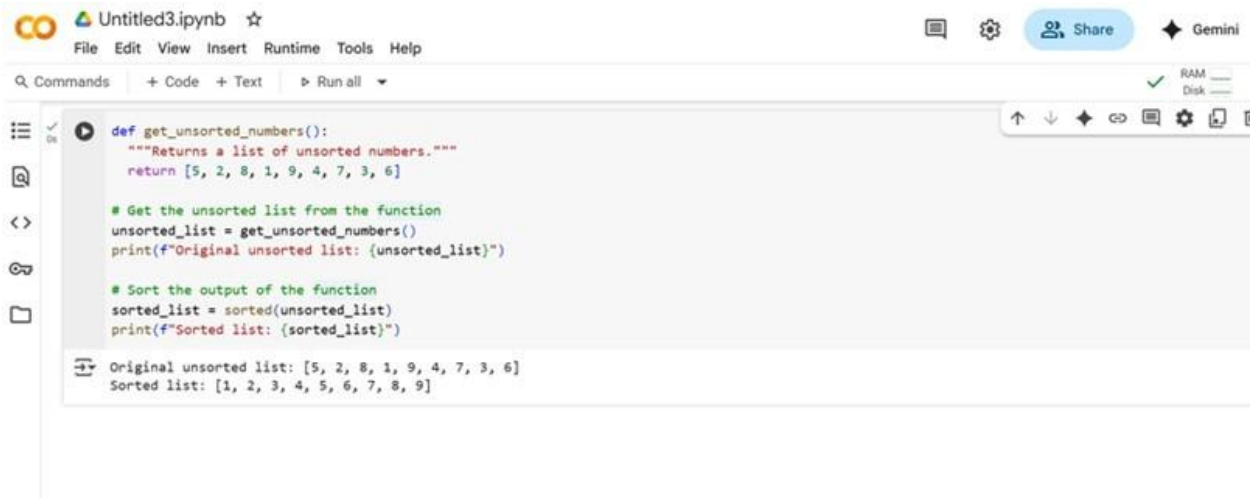
Question:



Code:



Output:



The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. The interface includes a top bar with a 'Share' button and a 'Gemini' icon. Below the top bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. A 'Commands' search bar is also present. The main area displays a Python code cell with the following content:

```
def get_unsorted_numbers():  
    """Returns a list of unsorted numbers."""  
    return [5, 2, 8, 1, 9, 4, 7, 3, 6]  
  
# Get the unsorted list from the function  
unsorted_list = get_unsorted_numbers()  
print(f"Original unsorted list: {unsorted_list}")  
  
# Sort the output of the function  
sorted_list = sorted(unsorted_list)  
print(f"Sorted list: {sorted_list}")
```

Below the code cell, the output is displayed:

```
Original unsorted list: [5, 2, 8, 1, 9, 4, 7, 3, 6]  
Sorted list: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Task Description-3:

Start with the vague prompt “Generate python code to calculate power bill” and improve it step-by-step.

Prompt:

Write a python function to calculate power bill based on units consumed.

Question:

Untitled3.ipynb ☆ ☁

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

✓ RAM
Disk

☰

🔍

<>

🔑

📁

✦ Great, so you'd like to see an example of sorting the output of a function. I can show you how to sort a list of numbers generated by a function.

👍 🗨

✦ Write a Python function to calculate power bill based on units consumed|

⊕ ▶

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

() Variables 📄 Terminal

✦

✓ 2:24 PM 📅 P

Code:

Untitled3.ipynb ☆ ☁

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

✓ RAM
Disk

☰

🔍

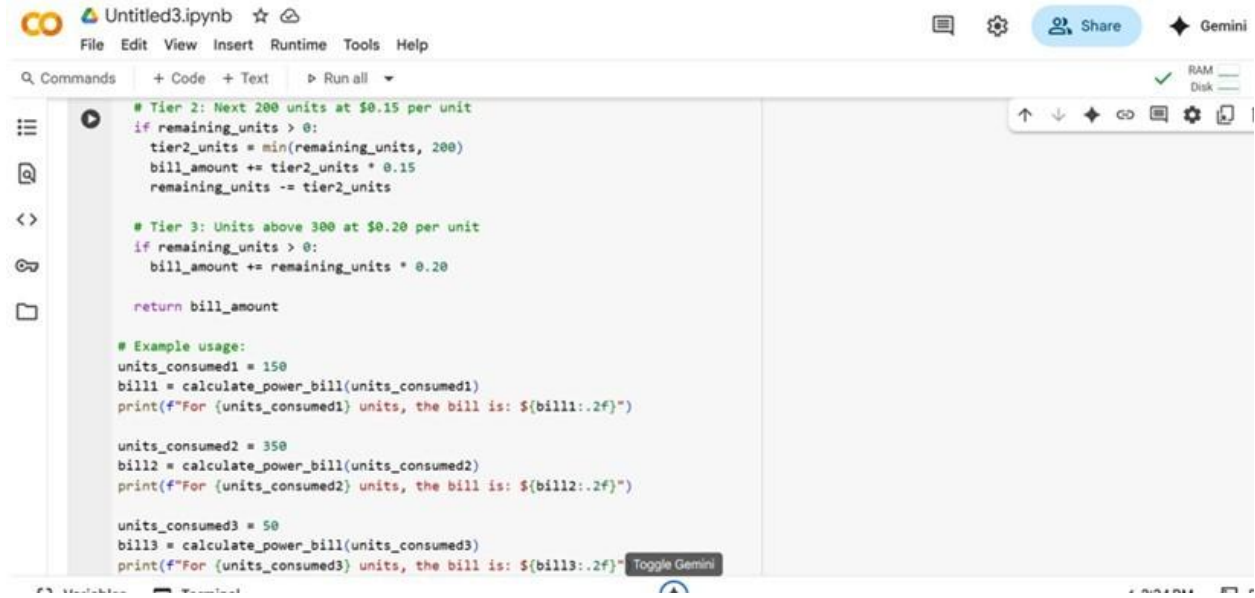
<>

🔑

📁

```
def calculate_power_bill(units):  
    """  
    Calculates the power bill based on units consumed using a tiered pricing structure.  
  
    Args:  
        units: The number of units of electricity consumed (non-negative).  
  
    Returns:  
        The calculated power bill amount, or an error message if input is invalid.  
    """  
    if units < 0:  
        return "Units consumed cannot be negative."  
  
    bill_amount = 0  
    remaining_units = units  
  
    # Tier 1: First 100 units at $0.10 per unit  
    if remaining_units > 0:  
        tier1_units = min(remaining_units, 100)  
        bill_amount += tier1_units * 0.10  
        remaining_units -= tier1_units  
  
    # Tier 2: Next 200 units at $0.15 per unit
```

Toggle Gemini



The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. The code defines a function to calculate a bill amount based on units consumed, with three tiers: Tier 1 (first 200 units at \$0.15), Tier 2 (next 200 units at \$0.15), and Tier 3 (units above 300 at \$0.20). It includes example usage for 150, 350, and 50 units.

```
# Tier 2: Next 200 units at $0.15 per unit
if remaining_units > 0:
    tier2_units = min(remaining_units, 200)
    bill_amount += tier2_units * 0.15
    remaining_units -= tier2_units

# Tier 3: Units above 300 at $0.20 per unit
if remaining_units > 0:
    bill_amount += remaining_units * 0.20

return bill_amount

# Example usage:
units_consumed1 = 150
bill1 = calculate_power_bill(units_consumed1)
print(f"For {units_consumed1} units, the bill is: ${bill1:.2f}")

units_consumed2 = 350
bill2 = calculate_power_bill(units_consumed2)
print(f"For {units_consumed2} units, the bill is: ${bill2:.2f}")

units_consumed3 = 50
bill3 = calculate_power_bill(units_consumed3)
print(f"For {units_consumed3} units, the bill is: ${bill3:.2f}")
```

Output:



The screenshot shows the same Jupyter Notebook interface, but now displaying the output of the code. The output shows the bill amounts for 150, 350, and 50 units, and an error message for -10 units.

```
units_consumed3 = 50
bill3 = calculate_power_bill(units_consumed3)
print(f"For {units_consumed3} units, the bill is: ${bill3:.2f}")

units_consumed4 = -10
bill4 = calculate_power_bill(units_consumed4)
print(f"For {units_consumed4} units, the bill is: {bill4}")

For 150 units, the bill is: $17.50
For 350 units, the bill is: $50.00
For 50 units, the bill is: $5.00
For -10 units, the bill is: Units consumed cannot be negative.
```

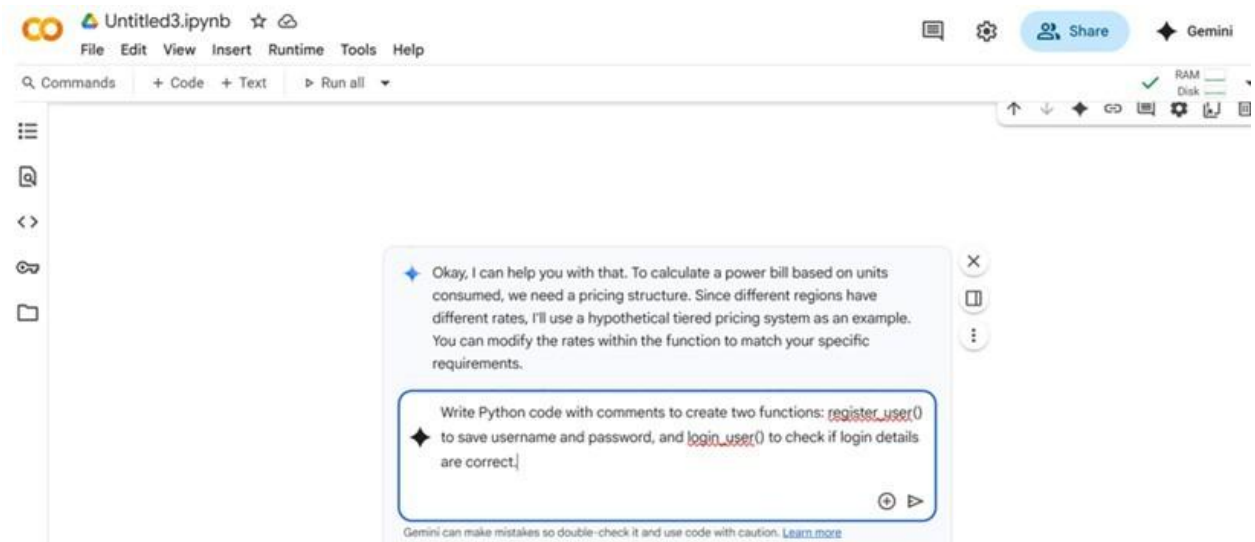
Task Description-4:

Write structured comments to help AI generate two linked functions (e.g., `login_user()` and `register_user()`).

Prompt-4:

Write Python code with comments to create two functions :`register user()` to save username and password ,and `login_user()` to check if login details are correct.

Question:



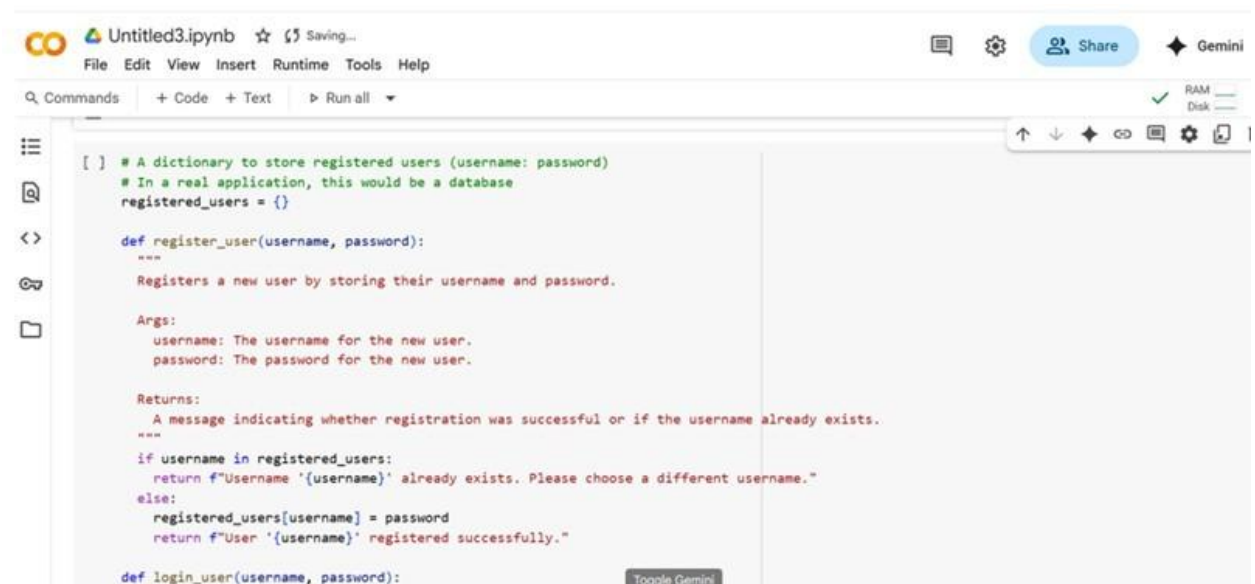
The screenshot shows a Jupyter Notebook titled "Untitled3.ipynb" with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar (Commands, + Code, + Text, Run all). A Gemini chat window is open, displaying a response to a query about calculating a power bill. The response includes a detailed explanation of a hypothetical tiered pricing system and a request to write Python code for user registration and login. The chat window has a close button (X), a copy button, and a scroll bar. Below the chat window, a disclaimer states: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

Okay, I can help you with that. To calculate a power bill based on units consumed, we need a pricing structure. Since different regions have different rates, I'll use a hypothetical tiered pricing system as an example. You can modify the rates within the function to match your specific requirements.

Write Python code with comments to create two functions: `register_user()` to save username and password, and `login_user()` to check if login details are correct.

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

Code:



The screenshot shows the same Jupyter Notebook interface with the Gemini chat window closed. The code cell contains the following Python code:

```
[ ] # A dictionary to store registered users (username: password)
# In a real application, this would be a database
registered_users = {}

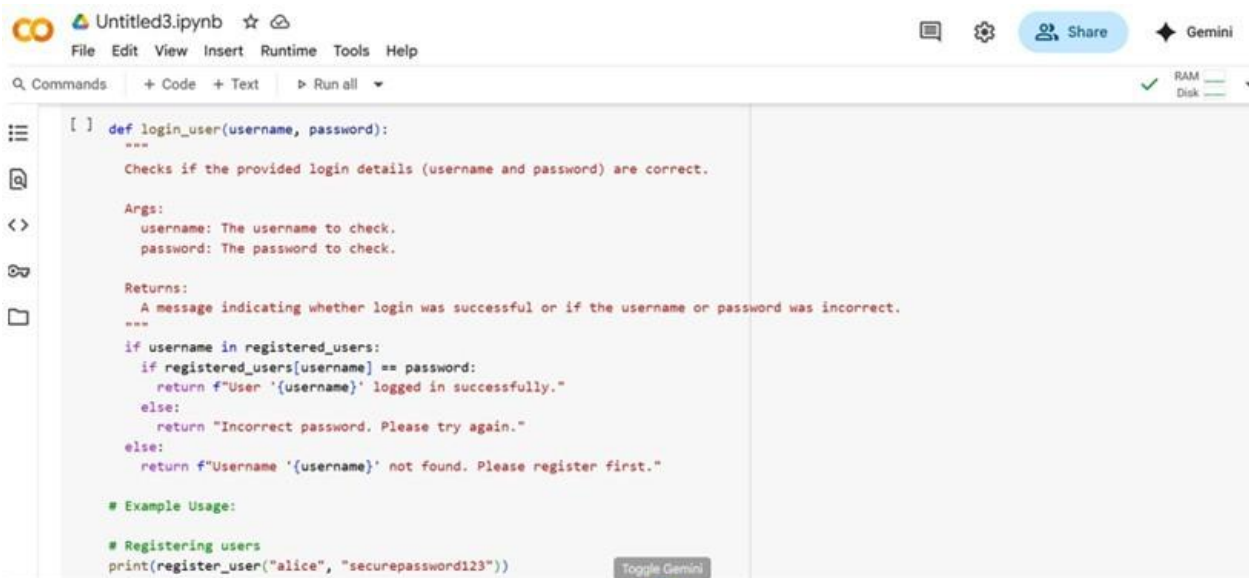
def register_user(username, password):
    """
    Registers a new user by storing their username and password.

    Args:
        username: The username for the new user.
        password: The password for the new user.

    Returns:
        A message indicating whether registration was successful or if the username already exists.
    """
    if username in registered_users:
        return f"Username '{username}' already exists. Please choose a different username."
    else:
        registered_users[username] = password
        return f"User '{username}' registered successfully."

def login_user(username, password):
```

The code is written in a code cell with a "Toggle Gemini" button at the bottom right.



The screenshot shows a Jupyter Notebook interface with the title "Untitled3.ipynb". The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu bar, there are tabs for "Commands", "+ Code", "+ Text", and "Run all". The main code area contains the following Python code:

```
[ ] def login_user(username, password):  
    """  
    Checks if the provided login details (username and password) are correct.  
  
    Args:  
        username: The username to check.  
        password: The password to check.  
  
    Returns:  
        A message indicating whether login was successful or if the username or password was incorrect.  
    """  
    if username in registered_users:  
        if registered_users[username] == password:  
            return f"User '{username}' logged in successfully."  
        else:  
            return "Incorrect password. Please try again."  
    else:  
        return f"Username '{username}' not found. Please register first."  
  
# Example Usage:  
  
# Registering users  
print(register_user("alice", "securepassword123"))
```

At the bottom right of the code area, there is a "Toggle Gemini" button. The right sidebar shows "RAM" and "Disk" usage indicators.

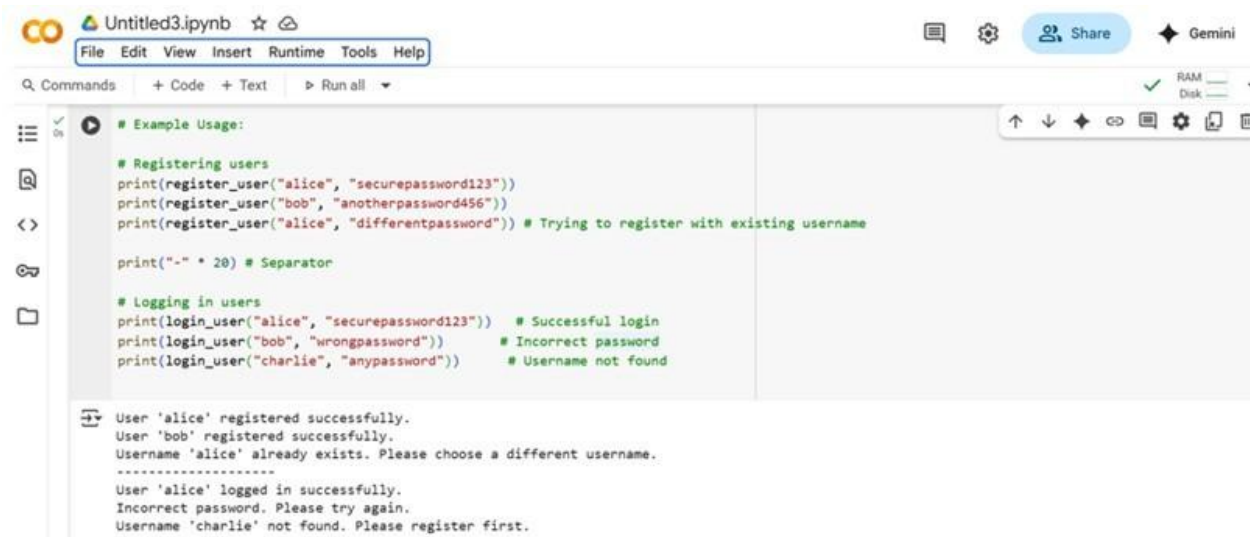


The screenshot shows the same Jupyter Notebook interface, but now displaying the execution of the code. The code is as follows:

```
# Example Usage:  
  
# Registering users  
print(register_user("alice", "securepassword123"))  
print(register_user("bob", "anotherpassword456"))  
print(register_user("alice", "differentpassword")) # Trying to register with existing username  
  
print("-" * 20) # Separator  
  
# Logging in users  
print(login_user("alice", "securepassword123")) # Successful login  
print(login_user("bob", "wrongpassword")) # Incorrect password  
print(login_user("charlie", "anypassword")) # Username not found
```

Below the code area, there is a prompt: "[] Start coding or generate with AI." The right sidebar shows "RAM" and "Disk" usage indicators, along with a toolbar with various icons.

Output:



```
# Example Usage:

# Registering users
print(register_user("alice", "securepassword123"))
print(register_user("bob", "anotherpassword456"))
print(register_user("alice", "differentpassword")) # Trying to register with existing username

print("-" * 20) # Separator

# logging in users
print(login_user("alice", "securepassword123")) # Successful login
print(login_user("bob", "wrongpassword")) # Incorrect password
print(login_user("charlie", "anypassword")) # Username not found

User 'alice' registered successfully.
User 'bob' registered successfully.
Username 'alice' already exists. Please choose a different username.
-----
User 'alice' logged in successfully.
Incorrect password. Please try again.
Username 'charlie' not found. Please register first.
```

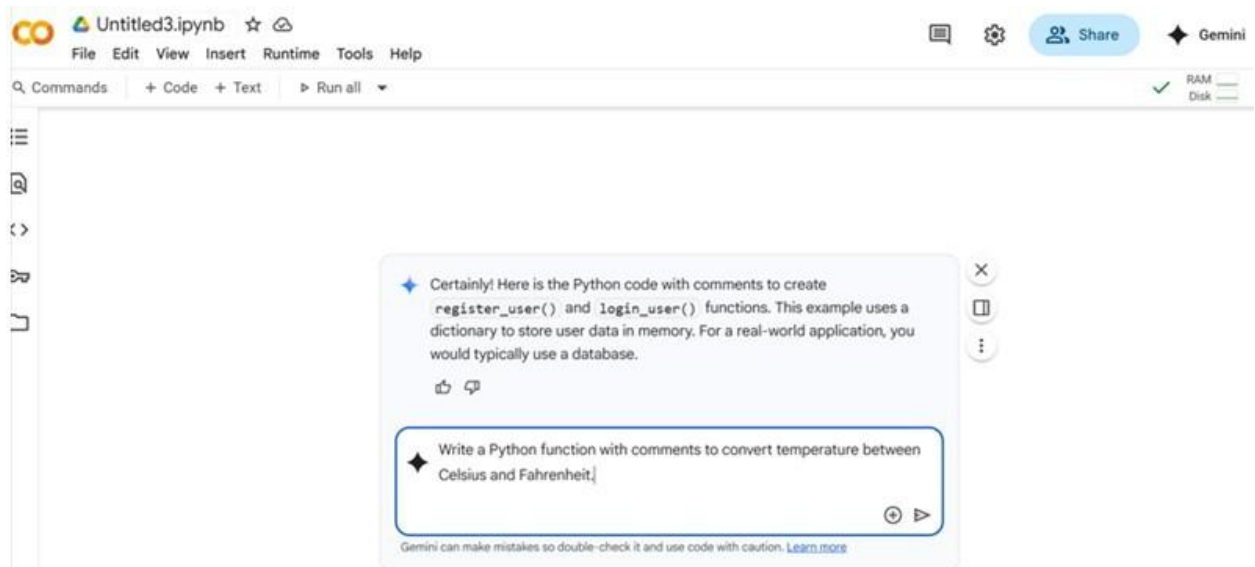
Task Description-5:

Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions.

Prompt-5:

Write a function with comments to convert temperature between Celsius and Fahrenheit.

Question:



Untitled3.ipynb

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text Run all

RAM Disk

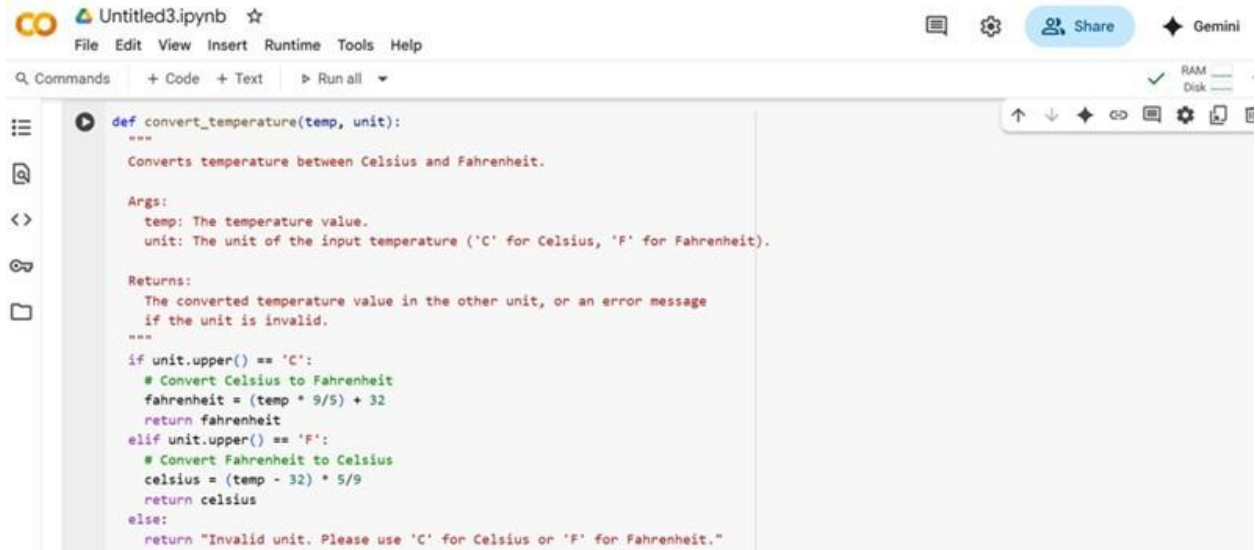
Gemini

Certainly! Here is the Python code with comments to create `register_user()` and `login_user()` functions. This example uses a dictionary to store user data in memory. For a real-world application, you would typically use a database.

Write a Python function with comments to convert temperature between Celsius and Fahrenheit.

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

Code:



Untitled3.ipynb

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

```
def convert_temperature(temp, unit):  
    """  
    Converts temperature between Celsius and Fahrenheit.  
  
    Args:  
        temp: The temperature value.  
        unit: The unit of the input temperature ('C' for Celsius, 'F' for Fahrenheit).  
  
    Returns:  
        The converted temperature value in the other unit, or an error message  
        if the unit is invalid.  
    """  
    if unit.upper() == 'C':  
        # Convert Celsius to Fahrenheit  
        fahrenheit = (temp * 9/5) + 32  
        return fahrenheit  
    elif unit.upper() == 'F':  
        # Convert Fahrenheit to Celsius  
        celsius = (temp - 32) * 5/9  
        return celsius  
    else:  
        return "Invalid unit. Please use 'C' for Celsius or 'F' for Fahrenheit."
```



Untitled3.ipynb

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

```
# Example usage:  
celsius_temp = 25  
fahrenheit_temp = convert_temperature(celsius_temp, 'C')  
print(f"{celsius_temp}°C is equal to {fahrenheit_temp:.2f}°F")  
  
fahrenheit_temp = 77  
celsius_temp_converted = convert_temperature(fahrenheit_temp, 'F')  
print(f"{fahrenheit_temp}°F is equal to {celsius_temp_converted:.2f}°C")  
  
invalid_temp = 100  
invalid_unit_result = convert_temperature(invalid_temp, 'K')  
print(f"Conversion with invalid unit: {invalid_unit_result}")
```

Output:



Untitled3.ipynb

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

```
# Example usage:  
celsius_temp = 25  
fahrenheit_temp = convert_temperature(celsius_temp, 'C')  
print(f"{celsius_temp}°C is equal to {fahrenheit_temp:.2f}°F")  
  
fahrenheit_temp = 77  
celsius_temp_converted = convert_temperature(fahrenheit_temp, 'F')  
print(f"{fahrenheit_temp}°F is equal to {celsius_temp_converted:.2f}°C")  
  
invalid_temp = 100  
invalid_unit_result = convert_temperature(invalid_temp, 'K')  
print(f"Conversion with invalid unit: {invalid_unit_result}")
```

25°C is equal to 77.00°F
77°F is equal to 25.00°C
Conversion with invalid unit: Invalid unit. Please use 'C' for Celsius or 'F' for Fahrenheit.

---END---

