
Program	:B.tech(CSE)
Specialization	:AIML
Course Title	:AI Assisted Coding
Course Code	:24CS002PC215
Semester	:3 rd semester
Academic Session	:2025-2026
Name of Student	:BachanagariDeekshitha
Enrollment No.	:2403A52032
Batch No.	:02
Date	:20/08/2025

#LAB ASSIGNMENT-3

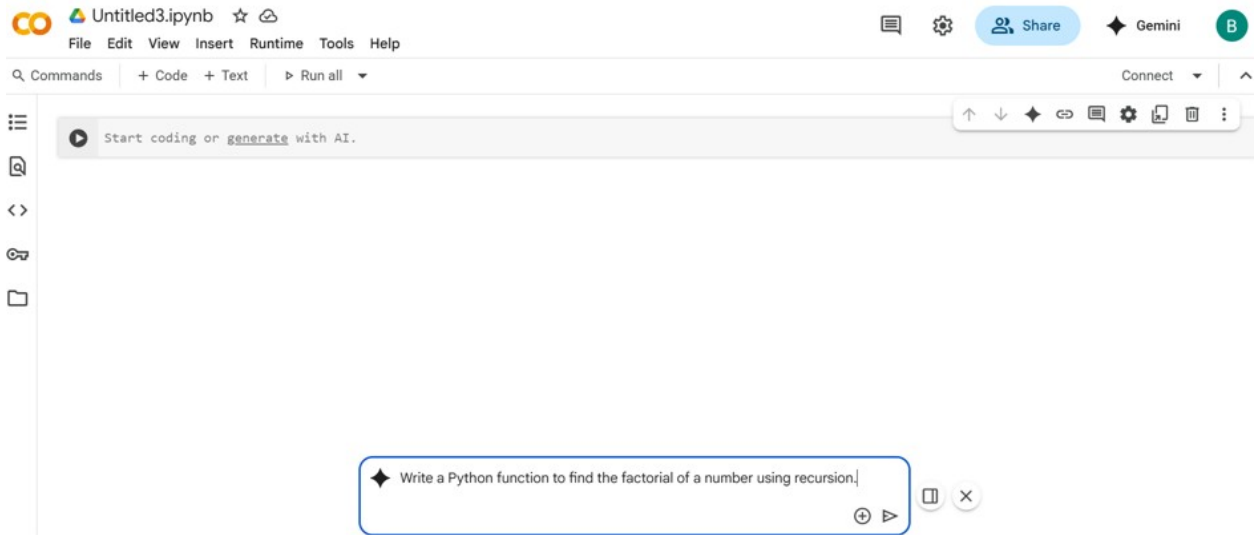
Task Description -1:

Try 3 different prompts to generate a factorial function.

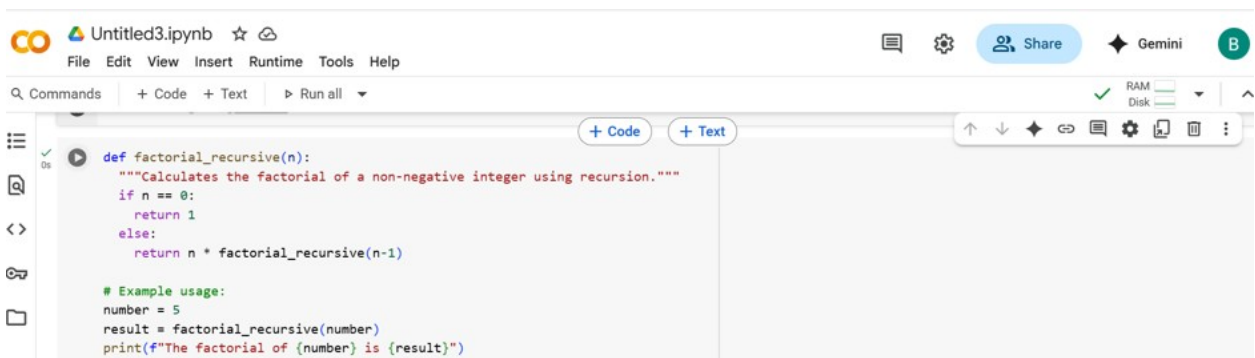
#PROMPT:

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

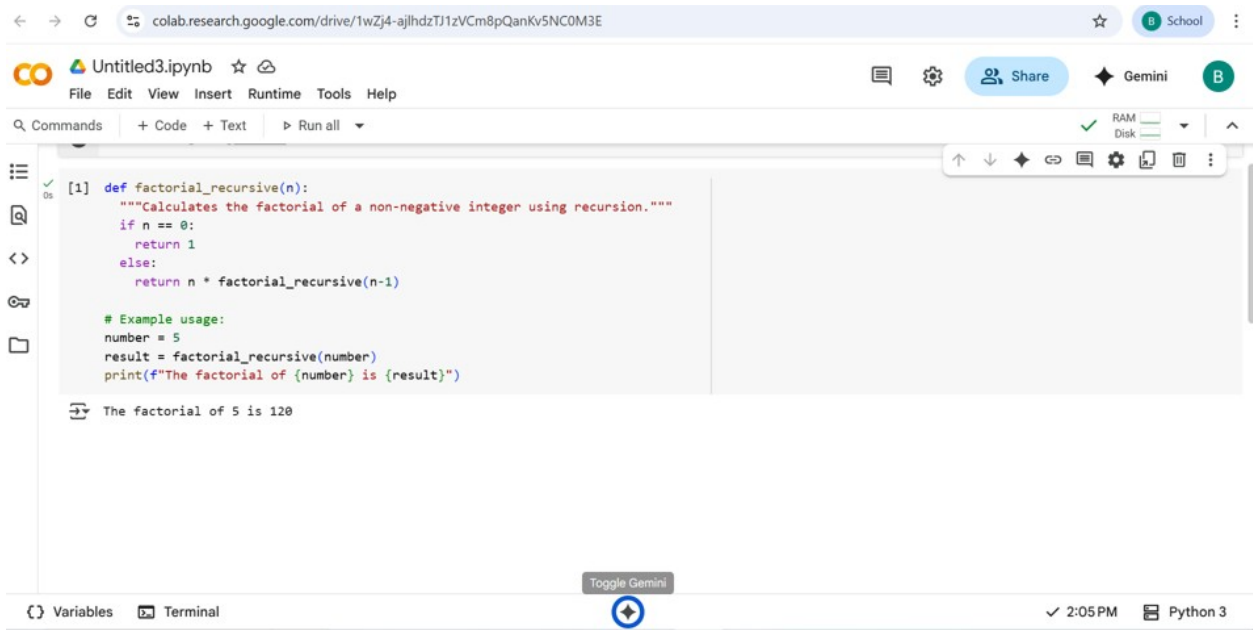
QUESTION:



CODE:



OUTPUT:



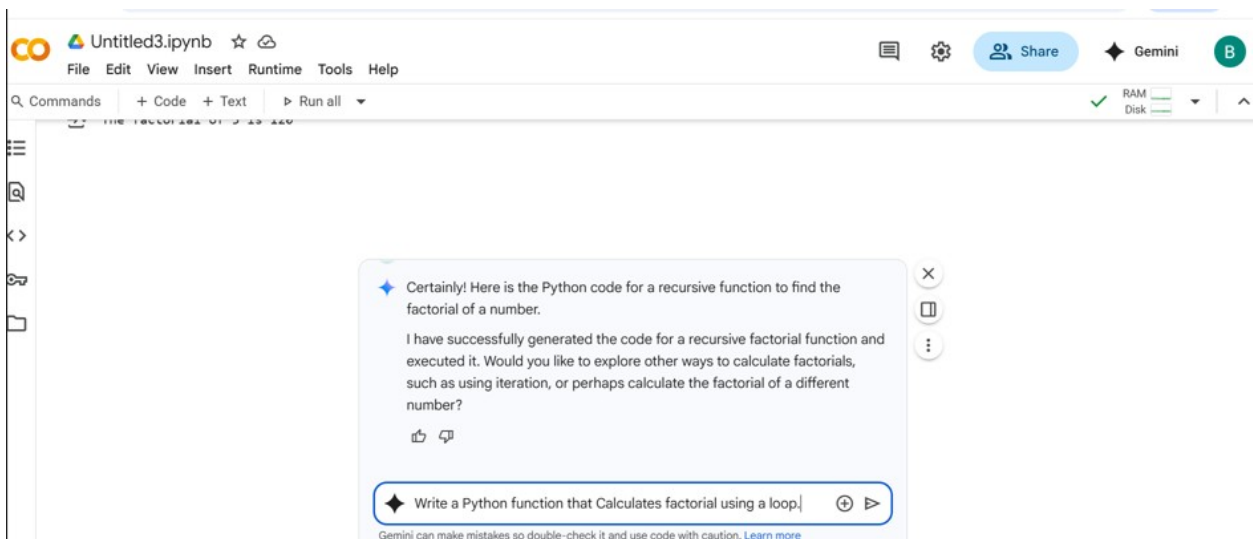
The screenshot shows a Google Colab notebook titled 'Untitled3.ipynb'. The code cell contains a recursive function `factorial_recursive(n)` that calculates the factorial of a non-negative integer `n`. The function uses a base case of `n == 0` returning 1, and a recursive case where it returns `n * factorial_recursive(n-1)`. An example usage is provided: `number = 5`, `result = factorial_recursive(number)`, and `print(f"The factorial of {number} is {result}")`. The output of the cell is 'The factorial of 5 is 120'. 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 at the bottom showing 'Variables', 'Terminal', 'Toggle Gemini', '2:05 PM', and 'Python 3'.

```
[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:



The screenshot shows a Gemini chat interface. The prompt is: 'Write a Python function that Calculates factorial using a loop'. The response from Gemini is: 'Certainly! Here is the Python code for a recursive function to find the factorial of a number. I have successfully generated the code for a recursive factorial function and executed it. Would you like to explore other ways to calculate factorials, such as using iteration, or perhaps calculate the factorial of a different number?'. 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 at the bottom showing 'Variables', 'Terminal', 'Toggle Gemini', '2:05 PM', and 'Python 3'.

Certainly! Here is the Python code for a recursive function to find the factorial of a number.

I have successfully generated the code for a recursive factorial function and executed it. Would you like to explore other ways to calculate factorials, such as using iteration, or perhaps calculate the factorial of a different number?

Write a Python function that Calculates factorial using a loop

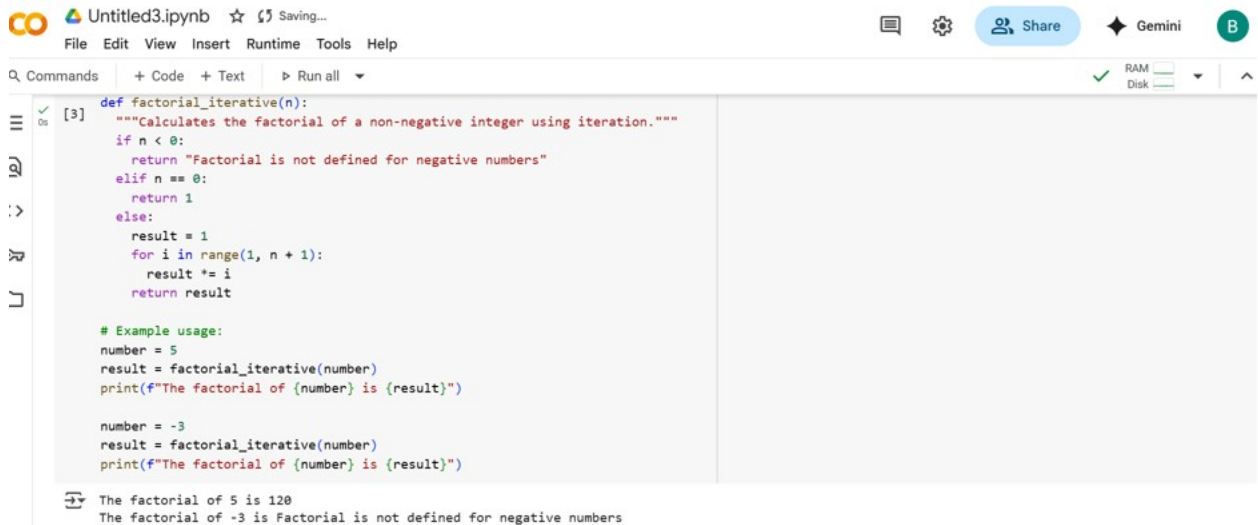
Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

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:



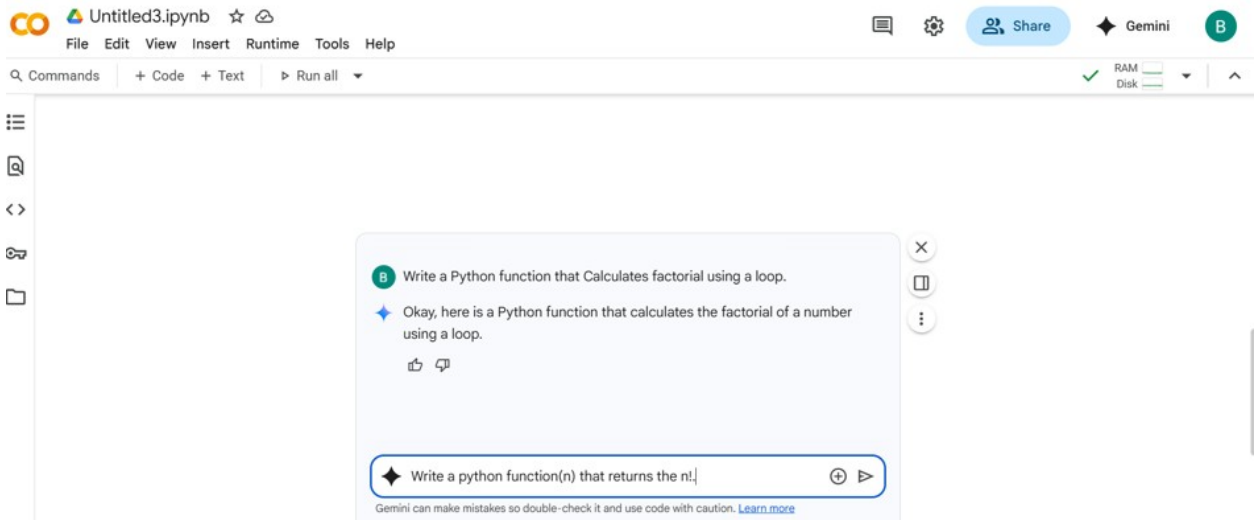
```
[3] def factorial_iterative(n):  
    """Calculates the factorial of a non-negative integer using iteration."""  
    if n < 0:  
        return "Factorial is not defined for negative numbers"  
    elif n == 0:  
        return 1  
    else:  
        result = 1  
        for i in range(1, n + 1):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

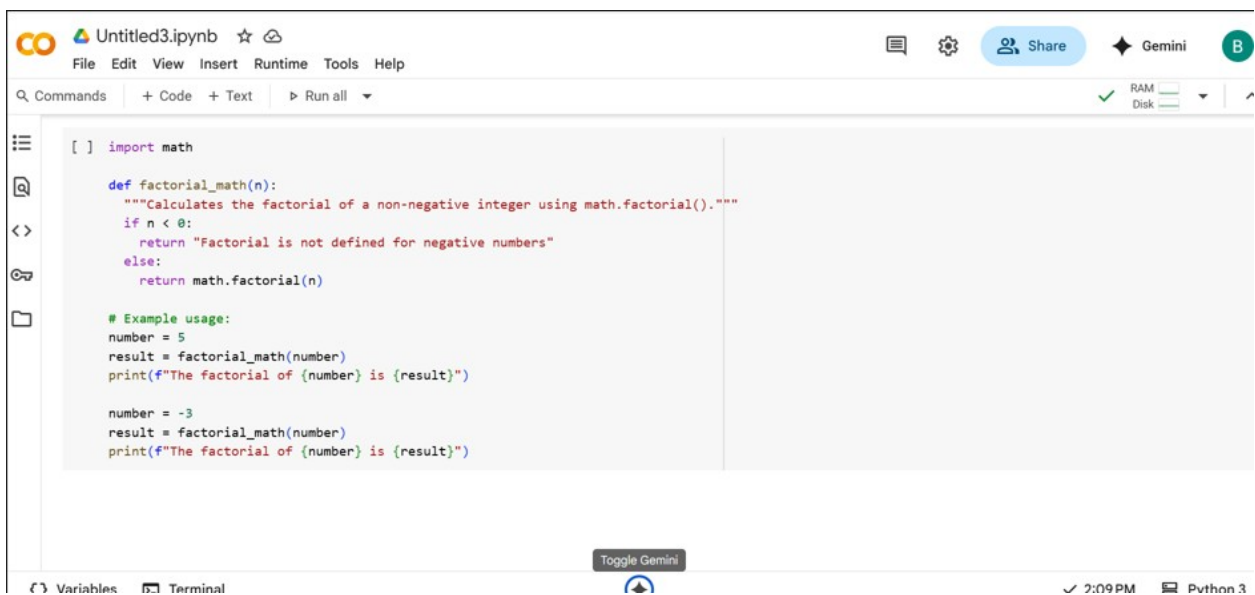
#PROMPT-3:

➤ Write a python function() that returns the n!

QUESTION:



CODE:



OUTPUT:

The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. The code defines a function `factorial_math(n)` that uses `math.factorial()` for non-negative integers and returns an error message for negative numbers. It includes example usage for `number = 5` and `number = -3`. The output at the bottom shows the factorial of 5 is 120 and the factorial of -3 is 'Factorial is not defined for negative numbers'.

```
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:



QUESTION:

The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. A Gemini chat window is open in the center, displaying a message from Gemini: 'Okay, here is another method using Python's built-in `math.factorial()` function.' Below this is a text input box containing the prompt: 'Write a python program for sorting a function with clear example of input-output'. The interface includes a top menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the left is a sidebar with icons for file operations. On the right, there are status indicators for RAM and Disk usage, and a 'Share' button. At the bottom, there are tabs for 'Variables' and 'Terminal', and a status bar showing '2:17 PM' and 'Python 3'.

CODE:

The screenshot shows the same Jupyter Notebook interface, but now with Python code entered in a code cell. The code defines a function `get_unsorted_numbers()` that returns a list of unsorted numbers. It then calls this function to get an `unsorted_list`, prints it, sorts it, and prints the `sorted_list`. The code is as follows:

```
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}")
```

The interface elements are consistent with the previous screenshot, including the top menu bar, left sidebar, right status indicators, and bottom tabs.

OUTPUT:

The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. The interface includes a top bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus. Below the menu is a toolbar with 'Commands', '+ Code', '+ Text', and 'Run all' buttons. On the right, there are icons for chat, settings, a 'Share' button, 'Gemini', and a user profile icon. The main area contains a code cell with the following Python code:

```
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, 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]
```

The bottom of the interface shows a 'Variables' tab, a 'Terminal' tab, and a status bar indicating '2:24 PM' and 'Python 3'.

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:

colab.research.google.com/drive/1wZj4-ajlhdzTJ1zVCm8pQanKv5NC0M3E#scrollTo=eff76352

Untitled3.ipynb

File Edit View Insert Runtime Tools Help

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 Python 3

CODE:

Untitled3.ipynb

File Edit View Insert Runtime Tools Help

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

Variables Terminal 2:24 PM Python 3

```
# 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:

```
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:

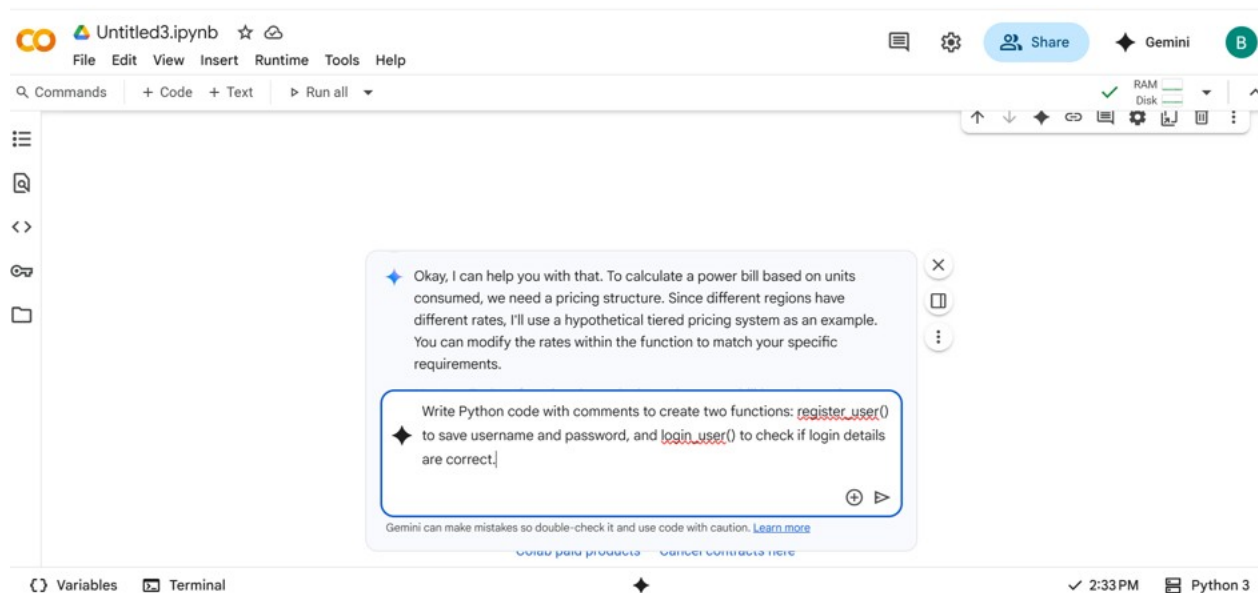
- Write structured comments to help AI generate two linked functions (e.g., login_user() and register_user()).

#PROMPT:

- 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:



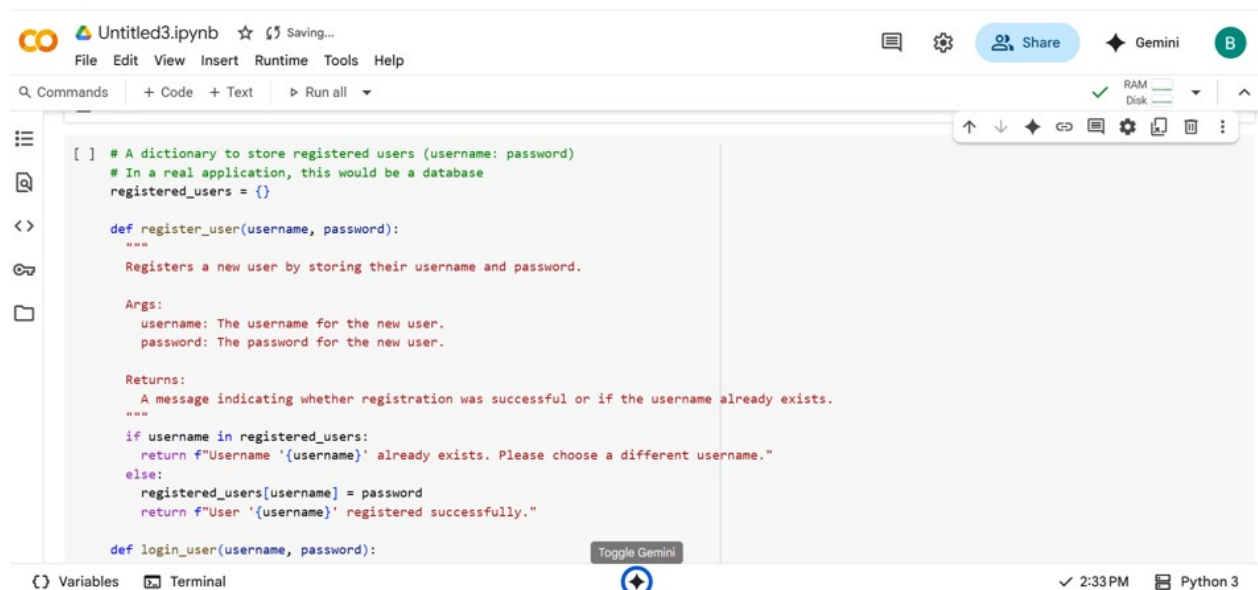
The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb' with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar. A Gemini AI chat window is open, displaying a message from the AI and a user prompt. The AI message says: 'Okay, I can help you with that. To calculate a power 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.' The user prompt asks for Python code to create two functions: `register_user()` to save username and password, and `login_user()` to check login details. The chat window also includes a disclaimer: '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 with the Python code from the Gemini AI prompt pasted into a code cell. The code defines a dictionary for registered users and two functions: `register_user()` and `login_user()`. The `register_user()` function checks if a username already exists and registers the user if it's new. The `login_user()` function is defined but its implementation is not visible in the screenshot.

```
[ ] # 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):
```

Untitled3.ipynb ☆ ☁

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

RAM Disk

```
[ ] 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"))
```

Toggle Gemini

{ } Variables Terminal 2:33 PM Python 3

Untitled3.ipynb ☆ ☁

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

RAM Disk

```
# 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  
  
[ ] Start coding or generate with AI.
```

{ } Variables Terminal 2:33 PM Python 3

OUTPUT:

The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. 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
```

The output of the code is displayed below the code cells:

```
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.
```

The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with icons for commands, code, text, and running all cells, and a status bar at the bottom showing '2:39 PM' and 'Python 3'.

TASK Description-5:

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

#PROMPT:

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

QUESTION:

The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb' with a Gemini chat window open. The chat window contains the following text:

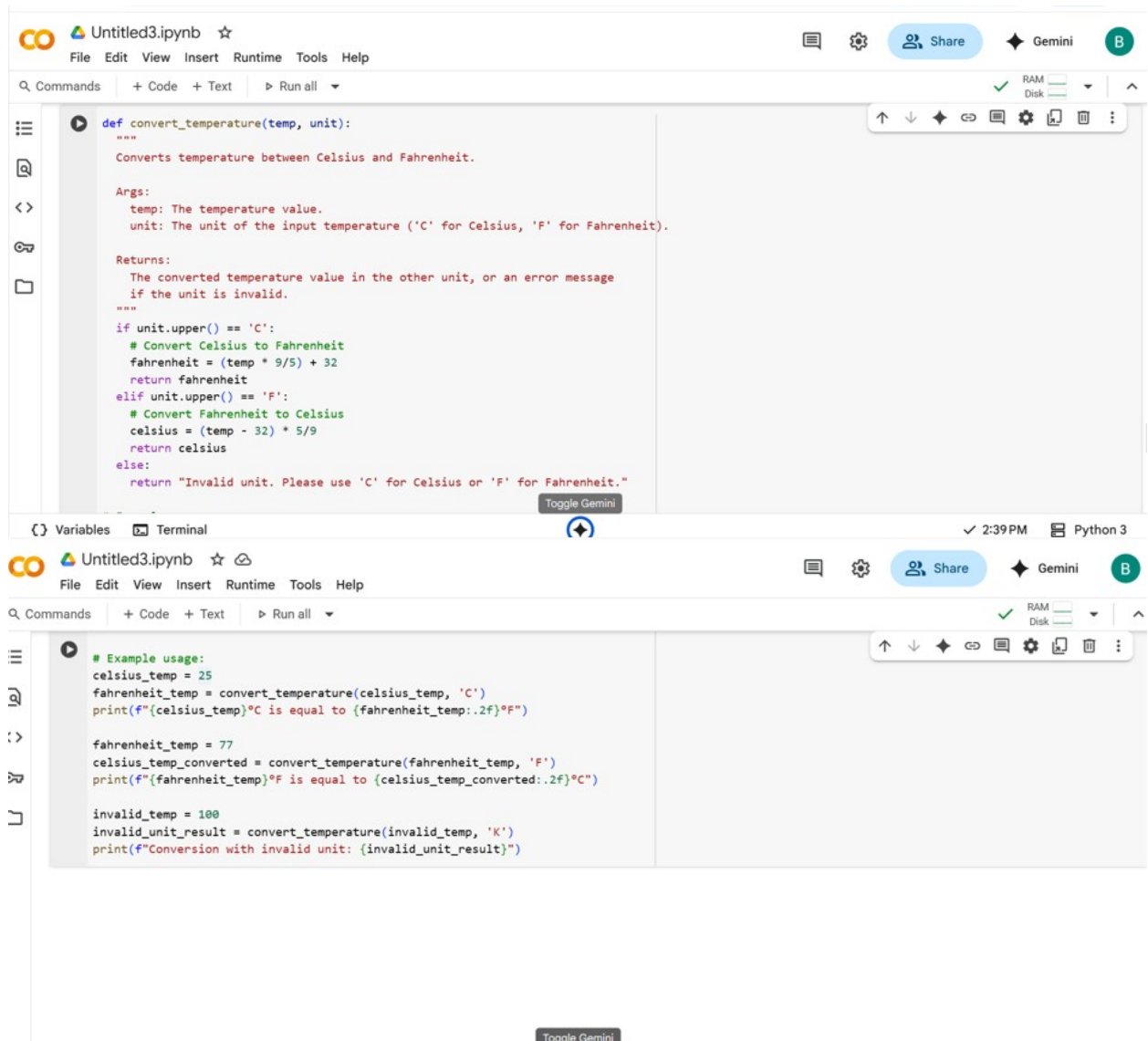
◆ Certainly! Here is the Python code with comments to create `register_user()` and `login_user()` functions. This example uses a dictionary to store user data in memory. For a real-world application, you would typically use a database.

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](#)

The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with icons for commands, code, text, and running all cells, and a status bar at the bottom showing '2:39 PM' and 'Python 3'.

CODE:



The image displays two screenshots of a Jupyter Notebook interface, likely Google Colab, showing the definition and usage of a temperature conversion function.


Top Screenshot: Function Definition

```
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."
```

Bottom Screenshot: Example Usage

```
# 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:



The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. The interface includes a top menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu is a toolbar with icons for running code, saving, and other functions. The notebook contains a code cell with the following Python code:

```
# 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}")
```

The output of the code cell is displayed below the code:

```
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.
```

THANK YOU