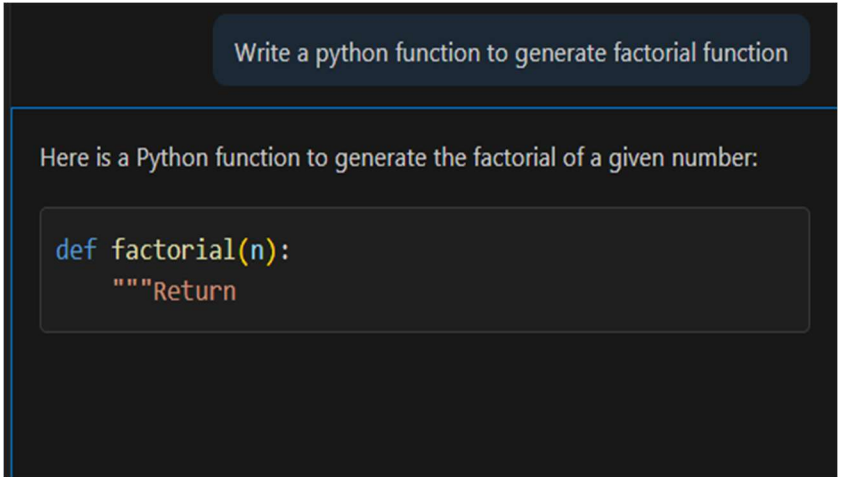


	AI ASSISTED CODING NAME:J.KEERTHI PRIYA ROLL NO : 2403A510G4 ASSIGNMENT:3.3	Expected Time to complete
1	<p>Lab Outcomes (LOs): After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> • Generate Python code using Google Gemini in Google Colab. • Analyze the effectiveness of code explanations and suggestions by Gemini. • Set up and use Cursor AI for AI-powered coding assistance. • Evaluate and refactor code using Cursor AI features. • Compare AI tool behavior and code quality across different platforms. <p>Task Description#1</p> <ul style="list-style-type: none"> • Try 3 different prompts to generate a factorial function. <p>Expected Output#1</p> <ul style="list-style-type: none"> • Comparison of AI-generated code styles <p>PROMPT 1:</p>  <p>The screenshot shows a dark-themed code editor. At the top, a light blue prompt box contains the text "Write a python function to generate factorial function". Below this, the editor displays the following Python code:</p> <pre>Here is a Python function to generate the factorial of a given number: def factorial(n): """Return</pre> <p>PROMPT 2:</p>	03.08.2025 EOD



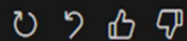
Restore Checkpoint

write a python code to genrate factorial function using one example

Here's a Python function to generate the factorial of a number, along with an example:

```
def factorial(n):  
    """Return the factorial of n (n!)."""  
    if n < 0:  
        raise ValueError("Factorial is not defined for  
        result = 1  
    for i in range(2, n + 1):  
        result *= i  
    return result  
  
# Example usage:  
print(factorial(5)) # Output: 120
```

This code defines a function `factorial` and demonstrates its usage with the input `5`.



PROMPT 3:

the input 5.



Restore Checkpoint

write a python function to generate factorial function dynamically

If you want to generate a factorial function dynamically (for example, by creating the function at runtime), you can use Python's `exec` to define the function as a string and then execute it. Here's how you can do it:

```
factorial_code = """
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
"""

exec(factorial_code)

# Example usage:
print(factorial(6)) # Output: 720
```

This approach creates the `factorial` function dynamically at runtime.

Task Description#2

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

Expected Output#2

- Functional sorting code from AI

PROMPT AND CODE:



Restore Checkpoint

Write a python function to generate sorting function with clear input and output

Here's a Python function that sorts a list of numbers. The input is a list, and the output is a sorted list:

```
def sort_numbers(numbers):  
    """  
    Sorts a list of numbers in ascending order.  
  
    Args:  
        numbers (list): List of numbers to sort.  
  
    Returns:  
        list: Sorted list of numbers.  
    """  
    return sorted(numbers)  
  
# Example usage:  
input_list = [5, 2, 9, 1, 5, 6]  
output_list = sort_numbers(input_list)  
print("Input:", input_list)  
print("Sorted Output:", output_list)
```

Task Description#3

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

Expected Output#3

- Enhanced AI output with clearer prompts

PROMPT AND CODE:



Generate python code to calculate power bill How much consumed last month and how much cost with input and output

```
File Edit Selection View Go Run Terminal Help
task3.py X
C:\Users\keerthi priya > Desktop > AI > task3.py > ...
1 def calculate_power_bill(previous_reading, current_reading, cost_per_unit):
2     """
3     Calculates the power bill based on meter readings and cost per unit.
4
5     Args:
6         previous_reading (float): Last month's meter reading.
7         current_reading (float): This month's meter reading.
8         cost_per_unit (float): Cost per unit of electricity.
9
10    Returns:
11        tuple: (units_consumed, total_cost)
12    """
13    units_consumed = current_reading - previous_reading
14    total_cost = units_consumed * cost_per_unit
15    return units_consumed, total_cost
16
17 # Example usage:
18 prev = 1200.5
19 curr = 1350.75
20 rate = 5.5 # cost per unit
21
22 units, cost = calculate_power_bill(prev, curr, rate)
23 print("Units Consumed:", units)
24
```

```
File Edit Selection View Go Run Terminal Help
task3.py X
C:\Users\keerthi priya > Desktop > AI > task3.py > ...
14 total_cost = units_consumed * cost_per_unit
15 return units_consumed, total_cost
16
17 # Example usage:
18 prev = 1200.5
19 curr = 1350.75
20 rate = 5.5 # cost per unit
21
22 units, cost = calculate_power_bill(prev, curr, rate)
23 print("Units Consumed:", units)
24 print("Total Cost:", cost)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\keerthi priya> & "C:\Users\keerthi priya\AppData\Local\Microsoft\WindowsApps\python3.11.exe" "c:/Users/
I/task3.py"
Units Consumed: 150.25
Total Cost: 826.375
PS C:\Users\keerthi priya>
```

Task Description#4

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

Expected Output#4

- Consistent functions with shared logic

CODE WITH OUTPUT:

CO Untitled7.ipynb ☆ ☁
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▼

☰ 0s ▶

```
def login_user(username, password):  
    """  
    Logs in a user with the provided username and password.  
  
    Args:  
        username (str): The username of the user.  
        password (str): The password of the user.  
  
    Returns:  
        bool: True if the login is successful, False otherwise.  
    """  
    # This is a placeholder. In a real application, you would  
    # check against a database of registered users.  
    if username == "test_user" and password == "password123":  
        print("Login successful!")  
        return True  
    else:  
        print("Invalid username or password.")  
        return False  
  
def register_user(username, password):  
    """  
    Registers a new user with the provided username and password.  
  
    Args:  
        username (str): The desired username for the new user.  
        password (str): The desired password for the new user.  
  
    Returns:  
        bool: True if the registration is successful, False otherwise.
```

CO Untitled7.ipynb ☆ ☁
File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▼

☰ 0s [1]

```
        bool: True if the registration is successful, False otherwise.  
    """  
    # This is a placeholder. In a real application, you would  
    # add the new user to a database and check for existing users.  
    print(f"User '{username}' registered successfully!")  
    return True  
  
    # Example usage:  
    register_user("new_user", "new_password")  
    login_user("test_user", "password123")  
    login_user("wrong_user", "wrong_password")
```

↗
User 'new_user' registered successfully!
Login successful!
Invalid username or password.
False

Task Description#5

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

Expected Output#5

- Code quality difference analysis for various prompts

Write a Python script that defines a function to convert temperatures between Fahrenheit, Celsius, and Kelvin. The function should be named `convert_temperature` and take three parameters:

Here's a Python script that defines a function `convert_temperature` to convert temperatures between Fahrenheit, Celsius, and Kelvin. The function takes three parameters: `value`, `from_unit`, and `to_unit`.

```
def convert_temperature(value, from_unit, to_unit):  
    """  
    Converts temperature between Fahrenheit, Celsius,  
  
    Args:  
        value (float): The temperature value to convert  
        from_unit (str): The unit of the input temperature  
        to_unit (str): The unit to convert to ('C', 'F', 'K')  
  
    Returns:  
        float: Converted temperature value.  
    """  
  
    # Convert input to Celsius  
    if from_unit == 'C':  
        celsius = value  
    elif from_unit == 'F':  
        celsius = (value - 32) * 5 / 9  
    elif from_unit == 'K':  
        celsius = value - 273.15  
    else:
```

```
if from_unit == 'C':
    celsius = value
elif from_unit == 'F':
    celsius = (value - 32) * 5 / 9
elif from_unit == 'K':
    celsius = value - 273.15
else:
    raise ValueError("Invalid from_unit. Use 'C', 'F', or 'K'")

# Convert Celsius to target unit
if to_unit == 'C':
    return celsius
elif to_unit == 'F':
    return celsius * 9 / 5 + 32
elif to_unit == 'K':
    return celsius + 273.15
else:
    raise ValueError("Invalid to_unit. Use 'C', 'F', or 'K'")

# Example usage:
temp = 100
print("Celsius to Fahrenheit:", convert_temperature(temp, 'C', 'F'))
print("Fahrenheit to Kelvin:", convert_temperature(32, 'F', 'K'))
print("Kelvin to Celsius:", convert_temperature(300, 'K', 'C'))
```

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks
Factorial Function (Task#1)	0.5
Sorting Function (Task#2)	0.5
Vogue Vs. Specific Prompting (Task #3)	0.5
Linked Functions (Task #4)	0.5
Temperature Conversion Function (Task #5)	0.5
Total	2.5 Marks