

AI-Assisted Coding

Name: P.Srikanth

Htno:2403A52002

Task-1:

Prompt-1: Generate a Python function to calculate the factorial of a number. The number should be taken as user input.

Code&Output:

The screenshot shows a Google Colab notebook titled "Untitled1.ipynb". The code cell contains the following Python code:

```
def factorial(n):
    """Calculates the factorial of a non-negative integer."""
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

# Get input from the user
try:
    num = int(input("Enter a non-negative integer: "))
    if num < 0:
        print("Factorial is not defined for negative numbers.")
    else:
        print(f"The factorial of {num} is {factorial(num)}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

The output cell shows the result of running the code with input 6:

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

A Gemini sidebar is open on the right, displaying the following conversation:

- V Generate a Python function to calculate the factorial of a number. The number should be taken as user input.
- ◆ Sure, I can help with that. Here is a Python function to calculate the factorial of a number, taking the number as user input.

Below the sidebar, there are three options:

1. Modify the existing code?
2. Work on a different problem?
3. Learn more about factorials or related mathematical concepts?

At the bottom of the sidebar, there is a "What can I help you build?" input field and a "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)" note.

Prompt-2: Write a Python program to calculate the factorial of a user-provided number. Include two

different implementations: one using a loop (iterative) and another using recursion.

Code&Output:

The screenshot shows a Google Colab notebook titled "Untitled1.ipynb". The code cell [1] contains an iterative factorial function and handles user input for non-negative integers. The code cell [2] contains a recursive factorial function. The code cell [3] demonstrates both methods. The Gemini AI sidebar suggests writing a program to calculate the factorial of a user-provided number using both iterative and recursive methods, comparing their performance, and exploring error handling for large outputs.

```
[1]
else:
    print(f"The factorial of {num} is {factorial(num)}")
except ValueError:
    print("Invalid input. Please enter an integer.")

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

[2]
# Iterative implementation of factorial
def factorial_iterative(n):
    """Calculates the factorial of a non-negative integer iteratively."""
    if n < 0:
        return "Factorial is not defined for negative numbers."
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

# Get input from the user for the iterative method
try:
    num_iterative = int(input("Enter a non-negative integer for iterative calculation: "))
    print(f"The factorial of {num_iterative} using iteration is {factorial_iterative(num_iterative)}")
except ValueError:
    print("Invalid input. Please enter an integer.")

# Enter a non-negative integer for iterative calculation: 10
The factorial of 10 using iteration is 3628800

[3]
# Recursive implementation of factorial
def factorial_recursive(n):
    """Calculates the factorial of a non-negative integer recursively."""
    if n < 0:
        return "Factorial is not defined for negative numbers."
    if n == 0:
        return 1
```

Prompt-3: Write a factorial function in Python and show the result for 5.

Code&Output:

The screenshot shows a Google Colab interface. In the code cell, a recursive factorial function is defined:

```
[4] def factorial(n):
    """Calculates the factorial of a non-negative integer."""
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

# Calculate and print the factorial of 5
print(f"The factorial of 5 is: {factorial(5)}")
```

The output of the code is: "The factorial of 5 is: 120".

A Gemini sidebar is open on the right, showing a conversation with the AI:

- Write a factorial function in Python and show the result for 5.
- Okay, here is a Python function to calculate the factorial of a number and the result for 5.
- I have provided the factorial function and shown the result for 5.
- Would you like to:
 - Calculate the factorial for a different number?
 - Implement the factorial function using a different method (like iteration)?
 - Explore other mathematical functions in Python?

At the bottom of the sidebar, there's a button "What can I help you build?" and a note: "Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)".

The system tray at the bottom shows the date and time: 2:27PM, Python 3, ENG IN, 14:28, 20-08-2025.

Observation: All three methods give the same result for calculating factorial.

- The recursive method is great for learning, but not ideal for very large numbers due to stack limits.
- The iterative method is practical and commonly used.
- The math module is best for performance and simplicity—just plug and play!

Task-2:

Prompt: Generate a Python function that returns the sorted list operation give input by the user

Code&Output:

The screenshot shows a Google Colab interface. On the left, a code cell contains a Python function named `sort_list` that sorts a list of numbers entered by the user. The code includes error handling for non-numeric input. On the right, a Gemini AI sidebar provides a response to the prompt, suggesting a function that sorts a list and includes code to get the list from the user. It also lists three items for improvement: handling potential errors, implementing sorting for mixed data types, and learning different ways to take list input from the user.

```
def sort_list(input_list):
    """Sorts a list and returns the sorted list."""
    return sorted(input_list)

# Get input from the user
try:
    user_input = input("Enter a list of numbers separated by commas (e.g., 3, 1, 4, 1, 5): ")
    # Convert the input string into a list of integers
    input_list = [int(x.strip()) for x in user_input.split(',')]

    # Sort the list and print the result
    sorted_list = sort_list(input_list)
    print(f"Original list: {input_list}")
    print(f"Sorted list: {sorted_list}")

except ValueError:
    print("Invalid input. Please enter a list of numbers separated by commas.")
```

gemini generate a Python function that returns the sorted list operation give input by the user

Certainly, I can provide a Python function that sorts a list, and also include code to get the list as input from the user.

I have successfully generated the function to sort a list and included the code to get the list as input from the user.

Would you like to:

- Handle potential errors for non-numeric input more robustly?
- Implement sorting for lists containing mixed data types?
- Learn about different ways to take list input from the user?

Observation: This Python code uses the Bubble Sort algorithm to sort a list of numbers in ascending order. It compares each pair of adjacent elements and swaps them if they're out of order. This process repeats until the entire list is sorted.

Task-3:

Prompt: Generate a Python program to calculate a power bill. The bill should be based on kilowatt-hours (kWh) consumed.

Code&Output:

The screenshot shows a Google Colab interface with three tabs: 'Untitled2.ipynb' (active), 'Untitled1.ipynb - Colab', and 'Untitled2.ipynb - Colab'. The main area displays the following Python code:

```
def calculate_power_bill(kwh_consumed, rate_per_kwh):
    """
    Calculates the power bill based on kWh consumed and the rate per kWh.

    Args:
        kwh_consumed: The total kilowatt-hours consumed.
        rate_per_kwh: The cost per kilowatt-hour.

    Returns:
        The total power bill amount.
    """
    return kwh_consumed * rate_per_kwh

# Example usage:
kwh_used = 500 # Replace with actual kWh consumed
cost_per_unit = 0.15 # Replace with the actual rate per kWh

total_bill = calculate_power_bill(kwh_used, cost_per_unit)

print(f"Total kWh consumed: {kwh_used}")
print(f"Rate per kWh: ${cost_per_unit}")
print(f"Total power bill: ${total_bill:.2f}")
```

The output pane shows the results of running the code:

```
Total kWh consumed: 500
Rate per kWh: $0.15
Total power bill: $75.00
```

A Gemini sidebar is open on the right, showing the prompt and the generated response:

v Generate a Python program to calculate a power bill. The bill should be based on kilowatt-hours (kWh) consumed.

◆ Certainly, I can help with that. Here is a Python program that calculates a power bill based on kilowatt-hours (kWh) consumed.

I have generated a Python program to calculate a power bill based on kWh consumed.

Would you like to explore adding more features to the program, such as:

- Incorporating tiered pricing based on consumption levels?
- Accounting for taxes or other fees?
- Allowing the user to input their consumption and rate?

Observation: -Prompt now specifies exact billing logic.

-Covers all necessary components of a realistic bill.

Code is ready to run and easy to adapt.

This Python function calculates a basic electricity bill by multiplying the number of

kilowatt-hours consumed (consumption_kwh) with the cost per kilowatt-hour (rate_per_kwh). It includes a clear docstring explaining its purpose and usage. The example shows a consumption of 500 kWh at a rate of \$0.15 per kWh, resulting in a total bill of \$75.00.

Task-4:

Prompt: Create a python program with two functions:register_user() and login_user().Use a dictionary{username:password}to store users.Add a menu-driven loop where users can register like username,password,phone number,age,clg name.login ,or quit

Code&Output:

The screenshot shows a Google Colab notebook titled "Untitled4.ipynb". The code defines two functions: `register_user` and `login_user`. `register_user` takes a list of users, checks if the username already exists, and if not, creates a new user entry with password, phone number, age, and college name. `login_user` checks if the provided username and password match any in the users dictionary.

```

def register_user(users):
    """Registers a new user."""
    username = input("Enter username: ")
    if username in users:
        print("Username already exists. Please choose a different username.")
        return

    password = input("Enter password: ")
    phone = input("Enter phone number: ")
    age = input("Enter age: ")
    college = input("Enter college name: ")

    users[username] = {
        'password': password,
        'phone': phone,
        'age': age,
        'college': college
    }
    print("User registered successfully!")

def login_user(users):
    """Logs in an existing user."""
    username = input("Enter username: ")
    password = input("Enter password: ")

    if username in users and users[username]['password'] == password:
        print("Login successful!")
    else:
        print("Invalid credentials. Please try again.")

if __name__ == "__main__":
    main()

```

The Gemini sidebar provides a summary of the task and generates code for a menu-driven loop:

- V** Create a python program with two functions:`register_user()` and `login_user()`.Use a dictionary(`username:password`)to store users.Add a menu-driven loop where users can register like `username,password,phone number,age,clg name,login ,or quit`
- ◆** Okay, I can help you create a Python program with `register_user()` and `login_user()` functions, a dictionary to store user data, and a menu-driven loop for registration, login, and quitting. Here is the code:

```

while True:
    print("\n--- User Management Menu ---")
    print("1. Register")
    print("2. Login")
    print("3. Quit")

    choice = input("Enter your choice: ")

    if choice == '1':
        register_user(users)
    elif choice == '2':
        login_user(users)
    elif choice == '3':
        print("Exiting program.")
        break
    else:
        print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()

```

The Gemini sidebar also lists additional features and asks what can be built:

- Would you like to explore some additional features, such as:
 - Saving and loading user data to a file?
- What can I help you build?

At the bottom, the status bar shows the time as 7:47PM, date as 20-08-2025, and environment as Python 3.

Observation: The code provides a simple command-line user management system with

registration and login functionality. It uses a dictionary (users) to store user data in memory. The structure is clear and functional for basic scenarios, but lacks features like data persistence (saving to a file), password hashing for security, and input validation (e.g., checking phone number format or age type).

Task-5:

Prompt: Write a python code on improving Temperature Conversion Function with Clear Instruction

Code&Output:

The screenshot shows a Jupyter Notebook interface in Google Colab. The code cell contains a Python function named `convert_temperature`. The function takes three parameters: `value`, `from_unit`, and `to_unit`. It includes docstrings for arguments and returns, and handles unit conversions between Celsius and Fahrenheit. The output pane shows the execution results, including a call to the function and its output. A Gemini AI sidebar is visible on the right, providing assistance and suggestions related to the code.

```
# --- Instructions for use --#
# To convert Celsius to Fahrenheit:
celsius_temp = 25
fahrenheit_temp = convert_temperature(celsius_temp, 'celsius', 'fahrenheit')
if fahrenheit_temp is not None:
    print(f"{celsius_temp} Celsius is equal to {fahrenheit_temp} Fahrenheit")

# To convert Fahrenheit to Celsius:
fahrenheit_temp_2 = 77
celsius_temp_2 = convert_temperature(fahrenheit_temp_2, 'fahrenheit', 'celsius')
if celsius_temp_2 is not None:
    print(f"{fahrenheit_temp_2} Fahrenheit is equal to {celsius_temp_2} Celsius")

# Example with invalid units:
convert_temperature(100, 'kelvin', 'celsius')
```

Output:

```
25 Celsius is equal to 77.0 Fahrenheit
77 Fahrenheit is equal to 25.0 Celsius
Invalid units specified. Please use 'celsius' or 'fahrenheit'.
```

Gemini X

V write a python code on improving Temperature Conversion Function with Clear Instruction

◆ Sure, I can help with that! Here's a Python function to convert temperatures between Celsius and Fahrenheit, along with clear instructions on how to use it.

The code has been executed successfully and you can see the output above.

Would you like to:

1. Add more temperature units (e.g., Kelvin)?
2. Add input validation to ensure the value is a number?
3. Explore other types of unit conversions?

◆ What can I help you build?

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

Observation: The `convert_temperature` function correctly converts temperatures between Celsius and Fahrenheit. It handles two valid conversion paths and returns `None` with an error message if the input units are invalid. Example usages are clear and demonstrate functionality, including an invalid case for robustness.