AI Assisted coding

➢ Task Description#1

Try 3 different prompts to generate a factorial function.

➢ Expected Output#1

Comparison of AI-generated code styles

➢ **Prompt1:** create a python program of factorial numbers using recursion method.

```
[1]  # Method 1: Using recursion
     def factorial_recursive(n):
         if n == 0:
             return 1
         else:
             return n * factorial_recursive(n - 1)

     # Example usage:
     num = 5
     print(f"Factorial of {num} using recursion: {factorial_recursive(num)}")
```

```
Factorial of 5 using recursion: 120
```

➢ Explanation of code:

- It defines a function factorial_recursive(n)
- If n is 0, it returns 1 — this is the base case.
- Otherwise, it returns n * factorial_recursive(n - 1), which keeps calling itself with smaller values until it reaches 0.

➢ **Prompt2:** create a python program of factorial numbers using iteration method.

```
[2]  # Method 2: Using iteration
     def factorial_iterative(n):
         result = 1
         for i in range(1, n + 1):
             result *= i
         return result

     # Example usage:
     num = 5
     print(f"Factorial of {num} using iteration: {factorial_iterative(num)}")
```

```
Factorial of 5 using iteration: 120
```

➢ Explanation of code:

- factorial_iterative(n) takes an integer n as input.
- It starts with result = 1.
- Then it loops from 1 to n, multiplying result by each number in that range.
- After the loop finishes, it returns the final value of result, which is the factorial of n

➢ **Prompt3:** create a python program of factorial numbers using math module.

```
[3]  # Method 3: Using the math module
     import math

     def factorial_math(n):
         return math.factorial(n)

     # Example usage:
     num = 5
     print(f"Factorial of {num} using math module: {factorial_math(num)}")
```

```
Factorial of 5 using math module: 120
```

➢ ***Explanation of code:***

- import math brings in Python's standard math library.
- The function factorial_math(n) simply returns math.factorial(n), which is a built-in method that computes the factorial of n.
- Then, it sets num = 5 and prints the result using an f-string

➢ **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 Description#2***

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

➢ **Expected Output#2**

Functional sorting code from AI

➢ **Prompt:** create a python program of sorting functions with input and output with example usage.

```python
def bubble_sort(arr):
  """Sorts an array using Bubble Sort."""
  n = len(arr)
  for i in range(n):
    # Last i elements are already in place
    for j in range(0, n - i - 1):
      # traverse the array from 0 to n-i-1
      # Swap if the element found is greater than the next element
      if arr[j] > arr[j + 1]:
        arr[j], arr[j + 1] = arr[j + 1], arr[j]
  return arr

# Example usage
my_list = [64, 34, 25, 12, 22, 11, 90]
print("Original list:", my_list)
sorted_list = bubble_sort(my_list.copy()) # Use a copy to keep the original list
print("Sorted list:", sorted_list)
```

```
Original list: [64, 34, 25, 12, 22, 11, 90]
Sorted list: [11, 12, 22, 25, 34, 64, 90]
```

➢ **Explanation of code :**

The function bubble_sort(arr) takes a list as input.

- It loops through the list multiple times.
- In each pass, it compares adjacent elements and swaps them if they're in the wrong order.
- After each pass, the largest unsorted element "bubbles up" to its correct position.
- Finally, it returns the sorted list.

➢ **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 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:** create a python program to calculate power bill with example usage.

```python
def calculate_basic_power_bill(consumption_kwh, rate_per_kwh):
  """Calculates a basic power bill based on consumption and a fixed rate.

  Args:
    consumption_kwh: The amount of electricity consumed in kilowatt-hours.
    rate_per_kwh: The cost per kilowatt-hour.

  Returns:
    The total power bill amount.
  """
  total_bill = consumption_kwh * rate_per_kwh
  return total_bill

# Example usage:
consumption = 500  # kWh
rate = 0.15  # $/kWh
bill = calculate_basic_power_bill(consumption, rate)
print(f"Basic power bill for {consumption} kWh: ${bill:.2f}")
```

```
Basic power bill for 500 kWh: $75.00
```

➢ **Explanation of code:**

The function calculate_basic_power_bill(consumption_kwh, rate_per_kwh) takes two inputs:

- consumption_kwh: how many kilowatt-hours of electricity were used.
- rate_per_kwh: the cost of one kilowatt-hour.
- It multiplies these two values to get the total bill.
- Then it returns the result.

➢ **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 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.

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 ,phonenumber, age, collage name,login, or quit.

```python
def register_user(users):
    """Registers a new user."""
    username = input("Enter username: ")
    if username in users:
        print("Username already exists.")
        return
    password = input("Enter password: ")
    phone_number = input("Enter phone number: ")
    age = input("Enter age: ")
    college_name = input("Enter college name: ")
    users[username] = {'password': password, 'phone_number': phone_number, 'age': age, 'college_name': college_name}
    print("Registration successful!")

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 username or password.")

def main():
    """Main function to run the user management system."""
    users = {}  # Dictionary to store user data
```

```
    while True:
        print("\nMenu:")
        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()
```

```
Menu:
1. Register
2. Login
3. Quit
Enter your choice: 1
Enter username: Akhila
Enter password: Ammu@123
Enter phone number: 9160165775
Enter age: 19
Enter college name: SR University
Registration successful!

Menu:
1. Register
2. Login
3. Quit
Enter your choice: 2
Enter username: Akhila
Enter password: Ammu@123
Login successful!

Menu:
1. Register
2. Login
3. Quit
Enter your choice: 3
Exiting program.
```

Explanation of code:

- `register_user()` Function

To register a new user by collecting their details and storing them in the users dictionary.

Steps:

- Prompts the user to enter a **username**.
- Checks if the username already exists in the users dictionary:
  - If it does, it prints a message and exits the function.
- If the username is new, it asks for:
  - password
  - phone number
  - age
  - college name
- Stores all the collected data in the users dictionary under the entered username.
- Prints "**Registration successful!**" to confirm the process.

- login_user() Function

To allow an existing user to log in by verifying their username and password.

Steps:

- Prompts the user to enter their **username** and **password**.
- Checks if:
  - The username exists in the users dictionary.
  - The entered password matches the stored password.
- If both conditions are true:
  - Prints "**Login successful!**"
  - Displays all the user's stored details using a loop.
- If the username doesn't exist or the password is incorrect:
  - Prints "**Invalid username or password.**"

Observation:

- The code is **interactive**, using input() to collect user data.
- It uses **conditional logic** (if statements) to handle menu choices.
- It stores user data temporarily (likely in variables, not persistent storage).
- There's **no data validation** (e.g., checking if phone number is numeric).

✓ **Task Description#5**

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

➢ **Expected Output#5**

Code quality difference analysis for various prompts.

➢ **Prompt:** Write a Python function that converts temperatures between Celsius and Fahrenheit. The function should take two arguments: the temperature value and the unit ('C' or 'F'). It should return the converted temperature with the correct unit.

Gemini

```
def celsius_to_fahrenheit(celsius):
  """Converts Celsius to Fahrenheit."""
  fahrenheit = (celsius * 9/5) + 32
  return fahrenheit

# Example usage:
celsius_temp = 25
fahrenheit_temp = celsius_to_fahrenheit(celsius_temp)
print(f"{celsius_temp}°C is equal to {fahrenheit_temp}°F")
```

```
25°C is equal to 77.0°F
```

➢ **Explanation of code:**

The function celsius_to_fahrenheit(celsius) uses the formula:

[(\text{Celsius} \times \frac{9}{5}) + 32]

- It takes a Celsius value, calculates the Fahrenheit equivalent, and returns it.

- Then it prints the result using an example value of 25°C.

➢ **Observation:**

- The function is **well-documented** with a docstring: """Converts Celsius to Fahrenheit."""
- The code is **clean, readable, and efficient**.
- It uses **type-safe arithmetic** and avoids unnecessary conversions.
- The output includes **degree symbols (°)** for clarity and professionalism.