

LAB ASSIGNMENT – 3

Program	: B-TECH
Specialization	: AIML
Name of Student	: AKULA MANOJ
Enrollment-No	: 2403A52031
Batch No	: 02
Date	: 20-08-2025

TRY 3 DIFFERENT PROMPTS TO GENERATE A FACTORIAL FUNCTION.

WRITE A PYTHON CODE TO GENERATE FACTORIAL FUNCTION IN EASY WAY AND IT SHOULD TAKE THE INPUT FROM THE USER.

[illegible]

PROMPT -2:- WRITE A PYTHON PROGRAM THAT DEFINES A FACTORIAL FUNCTION IN THE SIMPLEST WAY POSSIBLE. THE PROGRAM SHOULD ASK THE USER TO ENTER A NUMBER AND THEN PRINT THE FACTORIAL OF THAT NUMBER.

GENERATED CODE:

```
lab 3 task-1 prompt 2.py > ...  
1 def factorial(n):  
2     if n == 0 or n == 1:  
3         return 1  
4     return n * factorial(n - 1)  
5  
6 num = int(input("Enter a number: "))  
7 print("Factorial:", factorial(num))
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
PS C:\Users\manoj akula\OneDrive\Documents\Desktop\AIAC> & "C:/Users/manoj akula/AppData/Local/Microsoft/WindowsApp  
top/AIAC/lab 3 task-1 prompt 2.py"  
Enter a number: 5  
Factorial: 120  
PS C:\Users\manoj akula\OneDrive\Documents\Desktop\AIAC> |
```

PROMPT -3:-

WRITE A PROMT TO GENERATE FACTORIAL OF LIST OF NUMBERS AND IT SHOULD TAKE INPUT FROM USER AND MAKE IT SIMPLE AND STRUCTURED MANNER

GENERATED CODE:

```
1 # Program to calculate the factorial of a list of numbers entered by the user
2
3 def factorial(n):
4     result = 1
5     for i in range(2, n+1):
6         result *= i
7     return result
8
9 # Take input from user
10 numbers = input("Enter numbers separated by spaces: ").split()
11 numbers = [int(num) for num in numbers]
12
13 # Calculate and display factorial for each number
14 for num in numbers:
15     print(f"Factorial of {num} is {factorial(num)}")
```

OUTPUT:

```
PS C:\Users\manoj_akula\OneDrive\Documents\Desktop\AIAC> & "C:/Users/manoj_akula/AppData/Local/Microsoft/windowsApps/python3.13.exe" "c:/Users/manoj_akula/OneDrive/Documents/Desktop/AIAC/lab 3 task-1 prompt 2.py"
Enter a number: 5
Factorial: 120
PS C:\Users\manoj_akula\OneDrive\Documents\Desktop\AIAC> & "C:/Users/manoj_akula/AppData/Local/Microsoft/windowsApps/python3.13.exe" "c:/Users/manoj_akula/OneDrive/Documents/Desktop/AIAC/lab-3 task-1 p-3.py"
Enter numbers separated by spaces: 3 4 6 7
Factorial of 3 is 6
Factorial of 4 is 24
Factorial of 6 is 720
Factorial of 7 is 5040
PS C:\Users\manoj_akula\OneDrive\Documents\Desktop\AIAC> |
```

COMPARISON:

THE AI I USED IS GITHUB COPILOT

THE SAME PROMPT IS GIVEN IN DIFFERENT WAYS AND BY ADDING DIFFERENT CONDITIONS. IT CHANGED ITS CODE BUT THE OUTPUT IS THE SAME. IT IS GENERATING THE FACTORIAL OF THE GIVEN NUMBER BY TAKING THE USER INPUT AND ABOVE ALL OF THESE, MY PERSONALLY I LIKED THE 3RD PROMPT BECAUSE IT CREATED PROPERLY COMPARED TO THE ABOVE 2 PROMPTS.

TASK DESCRIPTION 2:

PROVIDE A CLEAR EXAMPLE INPUT-OUTPUT PROMPT TO GENERATE A SORTING FUNCTION

PROMPT:

WRITE AN EFFECTIVE AND SIMPLE STRUCTURED CODE, PROVIDE A CLEAR EXAMPLE INPUT-OUTPUT TO GENERATE A SORTING FUNCTION. IT SHOULD TAKE INPUT FROM THE USER

FUNCTIONAL SORTING CODE FROM AI :

```
task-4 lab-3 assig -3.3.py >...
1 def sort_numbers():
2     # Ask user for input
3     user_input = input("Enter numbers separated by spaces: ")
4     # Convert input string to a list of integers
5     numbers = list(map(int, user_input.strip().split()))
6     # Sort the list
7     sorted_numbers = sorted(numbers)
8     # Print the sorted list
9     print("Sorted numbers:", sorted_numbers)
10
11 # Example input-output:
12 # Input: 5 2 9 1 7
13 # Output: Sorted numbers: [1, 2, 5, 7, 9]
14
15 if __name__ == "__main__":
16     sort_numbers()
```

OUTPUT:-

```
PS C:\Users\manoj_akula\OneDrive\Documents\Desktop\AIAC> & "C:/Users/manoj_akula/AppData/Local/Microsoft/WindowsApps/python3.13.exe" "c:/Users/manoj_akula/OneDrive/Documents/Desktop/AIAC/task-4 lab-3 assig -3.3.py"
Enter numbers separated by spaces: 1 2 3 1 2 1
Sorted numbers: [1, 1, 1, 2, 2, 3]
PS C:\Users\manoj_akula\OneDrive\Documents\Desktop\AIAC> |
```

EXPLANATION:

I ASKED COPILOT TO GIVE CODE ACCORDING TO THE PROMPT AND BASED ON THAT IT TAKEN INPUT FROM THE USER WHICH ARE SEPARATED BY SPACES AND PROVIDED THE SORTED ,THE VALID OUTPUT IS GIVEN BY SORTING THE GIVEN NUMBERS

OBSERVATION:

I OBSERVED THE SIMPLE WAY OF WRITING PROMPTS TO GET THE DESIRED OUTPUT, AND BY THE GIVEN PROMPT, THE COPILOT PROVIDED FULLY FUNCTIONING CODE

TASK DESCRIPTION 3:

START WITH THE VAGUE PROMPT “GENERATE PYTHON CODE TO CALCULATE POWER BILL” AND IMPROVE IT STEP-BY-STEP

PROMPT:-

GENERATE A PYTHON CODE TO CALCULATE POWER BILL AND IT SHOULD TAKE INPUT FROM US AND MAKE IT SIMPLER AND EFFICIENT AND IMPROVE IT STEP BY STEP.

CODE GENERATED:-

```
1  # Step 1: Simple version to calculate power bill
2
3  # Take input from user
4  units = float(input("Enter the number of units consumed: "))
5  rate = float(input("Enter the rate per unit: "))
6
7  # Calculate bill
8  bill = units * rate
9
10 # Display result
11 print(f"Total Power Bill: ₹{bill:.2f}")
```

OUTPUT:-

```
PS C:\Users\manoj_aku\OneDrive\Documents\Desktop\AIAC> & "C:/Users/manoj_aku/AppData/Local/Microsoft/WindowsApps/python3.13.exe" "c:/Users/manoj_aku/OneDrive/Documents/Desktop/AIAC/lab-3_assi-3.3_task3.py"
Enter the number of units consumed: 120
Enter the rate per unit: 2
Total Power Bill: ₹240.00
PS C:\Users\manoj_aku\OneDrive\Documents\Desktop\AIAC> |
```

EXPLANATION:

THE CODE IS PERFECT AND GENERATED IN SIMPLE CODE AND WITH MY REQUIREMENTS BY THIS WE CAN CALCULATE THE POWERBILL NATIONALLY BECAUSE WE ARE GIVING THE DATA MANUALLY

CONCLUSION:

I OBSERVED THE SIMPLE WAY OF WRITING PROMPTS TO GET THE DESIRED OUTPUT, AND BY THE GIVEN PROMPT, THE COPILOT PROVIDED FULLY FUNCTIONING CODE

TASK DESCRIPTION 4:

WRITE STRUCTURED COMMENTS TO HELP AI GENERATE TWO LINKED FUNCTIONS (E.G., LOGIN_USER() AND REGISTER_USER()).

PROMPT:

WRITE CLEAN PYTHON CODE WITH TWO LINKED FUNCTIONS: REGISTER_USER() TO SECURELY CREATE AND STORE USER CREDENTIALS.REGISTER USER IT SHOULD ASK DATE OF BIRTH

AND PLACE, FATHER NAME AND PHONE NUMBER AND LOGIN_USER() TO VALIDATE LOGIN BY CHECKING ENTERED CREDENTIALS AGAINST STORED DATA, WITH PROPER COMMENTS AND ERROR HANDLING.

CODE GENERATED :-

```
lab-3 task-4 assin.3.3.py > ...
1 import hashlib
2 import getpass
3
4 # Dictionary to store user data
5 users_db = {}
6
7 def hash_password(password):
8     """Hash a password for secure storage."""
9     return hashlib.sha256(password.encode()).hexdigest()
10
11 def register_user():
12     """
13     Register a new user by collecting credentials and personal information.
14     Stores the data securely in users_db.
15     """
16     print("=== User Registration ===")
17     username = input("Enter username: ").strip()
18     if username in users_db:
19         print("Error: Username already exists.")
20         return
21
22     password = getpass.getpass("Enter password: ")
23     confirm_password = getpass.getpass("Confirm password: ")
24     if password != confirm_password:
25         print("Error: Passwords do not match.")
26         return
27
28     dob = input("Enter date of birth (YYYY-MM-DD): ").strip()
29     place = input("Enter place: ").strip()
```



```

father_name = input("Enter father's name: ").strip()
phone = input("Enter phone number: ").strip()

# Store user data securely
users_db[username] = {
    'password_hash': hash_password(password),
    'dob': dob,
    'place': place,
    'father_name': father_name,
    'phone': phone
}
print("Registration successful!\n")

def login_user():
    """
    Validate user login by checking entered credentials against stored data.
    """
    print("=== User Login ===")
    username = input("Enter username: ").strip()
    if username not in users_db:
        print("Error: Username not found.")
        return

    password = getpass.getpass("Enter password: ")
    password_hash = hash_password(password)
    if users_db[username]['password_hash'] == password_hash:
        print("Login successful!")
        # Optionally, display user info (excluding password)

```

```

def login_user():
    if users_db[username]['password_hash'] == password_hash:
        print("Login successful!")
        # Optionally, display user info (excluding password)
        print(f"Welcome, {username}!")
        print(f"Date of Birth: {users_db[username]['dob']}")
        print(f"Place: {users_db[username]['place']}")
        print(f"Father's Name: {users_db[username]['father_name']}")
        print(f"Phone: {users_db[username]['phone']}")
    else:
        print("Error: Incorrect password.")

# Example usage
if __name__ == "__main__":
    while True:
        print("\n1. Register\n2. Login\n3. Exit")
        choice = input("Choose an option: ").strip()
        if choice == '1':
            register_user()
        elif choice == '2':
            login_user()
        elif choice == '3':
            print("Goodbye!")
            break
        else:
            print("Invalid option. Please try again.")

```

OUTPUT :-

```
PS C:\Users\manoj_akula\OneDrive\Documents\Desktop\AIAC> & "C:/Users/manoj_akula/AppData/Local/Microsoft/WindowsApps/python3.13.exe" "c:/Users/manoj_akula/OneDrive/Documents/Desktop/AIAC/lab-3 task-4 ass1n.3.3.py"
```

```
1. Register
2. Login
3. Exit
Choose an option: 1
=== User Registration ===
Enter username: manoj
Enter password:
Confirm password:
Enter date of birth (YYYY-MM-DD): 27-10-2006
Enter place: warangal
Enter father's name: narsaiah
Enter phone number: 9949307229
Registration successful!
```

```
1. Register
2. Login
3. Exit
Choose an option: 2
=== User Login ===
Enter username: manoj
Enter password:
Login successful!
Welcome, manoj!
Date of Birth: 27-10-2006
Place: warangal
Father's Name: narsaiah
Phone: 9949307229
```

```
1. Register
2. Login
3. Exit
Choose an option: █
```

EXPLANATION:

The `register_user()` function adds a new user to a database (a dictionary) by checking if the username already exists—if it doesn't, it stores the username and password. The `login_user()` function checks if the username exists and if the given password matches the stored one, allowing access

only if both are correct. Together, these functions create a simple user authentication system with basic validation.

OBSERVATION:

IT PROVIDED SIMPLE AND STRUCTURED .IT PROVIDED THE OUTPUT AS EXPECTED.

TASK DESCRIPTION 5:

ANALYZING PROMPT SPECIFICITY: IMPROVING TEMPERATURE CONVERSION FUNCTION WITH CLEAR INSTRUCTIONS

PROMPT-1:-

WRITE A PYTHON CODE CLEARLY FOR IMPROVING TEMPERATURE CONVERSION FUNCTION WITH CLEAR INSTRUCTIONS

CODE GENERATED:-

```
1 def convert_temperature():
2     """
3     Converts temperature between Celsius and Fahrenheit.
4     Instructions:
5     1. Enter the temperature value as a number.
6     2. Enter the unit you want to convert from: 'C' for Celsius or 'F' for Fahrenheit.
7     3. The function will display the converted temperature.
8     """
9     try:
10        temp = float(input("Enter the temperature value: "))
11        unit = input("Enter the unit to convert from (C/F): ").strip().upper()
12        if unit == 'C':
13            converted = (temp * 9/5) + 32
14            print(f"{temp}°C is {converted:.2f}°F")
15        elif unit == 'F':
16            converted = (temp - 32) * 5/9
17            print(f"{temp}°F is {converted:.2f}°C")
18        else:
19            print("Invalid unit. Please enter 'C' for Celsius or 'F' for Fahrenheit.")
20    except ValueError:
21        print("Invalid input. Please enter a numeric temperature value.")
22
23    # Run the function
24    convert_temperature()
```

OUTPUT:-

```
• Enter the temperature value: 40
  Enter the unit to convert from (C/F): c
  40.0°C is 104.00°F
```

PROMPT-2:-

CREATE A IMPROVING TEMPARATURE CONVERSATIONAL FUNCTION WITH NEAT CODE. WHICH TAKE WHEATER CONDITIN FROM THE USER

CODE GENERATED:-

```
1 def improve_temperature_conversation():
2     print("Hi! Let's talk about the weather and temperature.")
3     weather = input("How is the weather today? (e.g., sunny, rainy, cloudy): ").strip().lower()
4     temp = input("What is the current temperature (in °C)? ").strip()
5
6     try:
7         temp = float(temp)
8     except ValueError:
9         print("Sorry, I couldn't understand the temperature value.")
10        return
11
12    if weather in ['sunny', 'clear']:
13        if temp > 30:
14            print("It's a hot and sunny day! Stay hydrated and wear sunscreen.")
15        elif temp > 20:
16            print("It's a pleasant sunny day. Enjoy the weather!")
17        else:
18            print("Sunny but a bit cool. A light jacket might help.")
19    elif weather in ['rainy', 'raining']:
20        if temp < 20:
21            print("Rainy and cool. Don't forget your umbrella and a warm drink!")
22        else:
23            print("Rainy but warm. An umbrella should be enough.")
24    elif weather in ['cloudy', 'overcast']:
25        if temp < 15:
26            print("Cloudy and chilly. Dress warmly!")
27        else:
28            print("Cloudy but comfortable. Have a nice day!")
29    else:
30        print("Thanks for sharing! Whatever the weather, hope you have a great day.")
31
32    # Example usage:
33    if __name__ == "__main__":
34        improve_temperature_conversation()
```

OUTPUT:-

```
PS C:\Users\ROUSHIK\OneDrive\Desktop\web> & C:/Users/ROUSHIK/AppData/Local/Programs/Python/Python310/python.exe
Hi! Let's talk about the weather and temperature.
How is the weather today? (e.g., sunny, rainy, cloudy): cloudy
What is the current temperature (in °C)? 26
Hi! Let's talk about the weather and temperature.
How is the weather today? (e.g., sunny, rainy, cloudy): cloudy
What is the current temperature (in °C)? 26
How is the weather today? (e.g., sunny, rainy, cloudy): cloudy
What is the current temperature (in °C)? 26
What is the current temperature (in °C)? 26
Cloudy but comfortable. Have a nice day!
```

DIFFERENCE:-

THE CODE IS COMPLETELY CHANGED BECAUSE IN THE FIRST PROMPT, IT CONVERTED THE CELSIUS TO FARENHEIT AND VICE VERSA, BUT IN THE SECOND FROM I WAS TOLD TO TAKE THE WEATHER AS AN INPUT, THEN IT ASKED ABOUT WHETHER AND TEMPERATURE, AND PROVIDED A CLEAN OUTPUT.

CONCLUSION:-

THE CODE PROVIDED IS CHANGED BY CHANGING THE PROMPTS, AND IT FOLLOWED MY PROMPT. I TRIED TO IMPROVE CODE QUALITY. IT CHANGED THE CODE AND BOTH CODES ARE PERFECT ACCORDING TO THE PROMPT.

THANK YOU
