ASSIGNMENT-5.1

NAME: G.Likhitha Rao

H.T.NO:2303A52487

BATCH:35

Task Description #1 (Privacy in API Usage)

Task: Use an AI tool to generate a Python program that connects to a weather API.

Prompt:

"Generate code to fetch weather data securely without exposing API keys in the code."

Expected Output:

• Original AI code (check if keys are hardcoded).

• Secure version using environment variables.

CODE:

```
import os
import requests

# Set your API key here. Replace 'YOUR_API_KEY' with your actual OpenWeatherMap API key.
os.environ["WEATHER_API_KEY"] = "YOUR_API_KEY"
API_KEY = os.getenv("WEATHER_API_KEY")
CITY = "Hyderabad"

if not API_KEY:
    raise ValueError("API key not found in environment variables")

url = f"https://api.openweathermap.org/data/2.5/weather?q={CITY}&appid={API_KEY}"
response = requests.get(url)

print(response.json())
```

OUTPUT:

```
{'cod': 401, 'message': 'Invalid API key. Please see https://openweathermap.org/faq#error401 for more info.'}
```

Task Description #2 (Privacy & Security in File Handling)

Task: Use an AI tool to generate a Python script that stores user data (name, email, password) in a file.

Analyze: Check if the AI stores sensitive data in plain text or without encryption.

Expected Output:

• Identified privacy risks.

• Revised version with encrypted password storage (e.g., hashing).

CODE:

```
import hashlib

name = input("Enter your name: ")
email = input("Enter your email: ")
password = input("Enter your password: ")

# Encode the password to bytes before hashing
encoded_password = password.encode('utf-8')

# Create a SHA-256 hash object and update it with the encoded password
hashed_password = hashlib.sha256(encoded_password).hexdigest()

file_name = 'user_data_hashed.txt'
with open(file_name, 'w') as f:
    f.write(f"Name: {name}\n")
    f.write(f"Email: {email}\n")
    f.write(f"Hashed Password: {hashed_password}\n")

print(f"User data with hashed password saved to {file_name}")
```

```
Enter your name: mounika
Enter your email: mounika@gmail.com
Enter your password: 123@123
User data with hashed password saved to user_data_hashed.txt
```

Task Description #3 (Transparency in Algorithm Design)

Objective: Use AI to generate an Armstrong number checking function

with comments and explanations.

Instructions:

1. Ask AI to explain the code line-by-line.

2. Compare the explanation with code functionality.

Expected Output:

• Transparent, commented code.

• Correct, easy-to-understand explanation.

CODE:

```python
num = int(input("Enter a number: "))
temp = num
sum = 0

digits = len(str(num))

while temp > 0:
    digit = temp % 10
    sum = sum + digit ** digits
    temp = temp // 10

if sum == num:
    print("Armstrong number")
else:
    print("Not an Armstrong number")
```

```
Enter a number: 153
Armstrong number
```

Task Description #4 (Transparency in Algorithm Comparison)

Task: Use AI to implement two sorting algorithms (e.g., QuickSort and BubbleSort).

Prompt:

"Generate Python code for QuickSort and BubbleSort, and include comments explaining step-by-step how each works and where they differ."

Expected Output:

• Code for both algorithms.

• Transparent, comparative explanation of their logic and efficiency.

CODE:

QUICK SORT:

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr

    pivot = arr[-1]
    left = []
    right = []

    for i in range(len(arr) - 1):
        if arr[i] <= pivot:
            left.append(arr[i])
        else:
            right.append(arr[i])

    return quick_sort(left) + [pivot] + quick_sort(right)

numbers = [64, 34, 25, 12, 22, 11, 90]
print(quick_sort(numbers.copy()))
```

BUBBLE SORT:

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr


numbers = [64, 34, 25, 12, 22, 11, 90]
print(bubble_sort(numbers.copy()))
```

COMPARISON TABLE:

Feature	Bubble Sort	Quick Sort
Method	Repeated swapping	Divide & conquer
Time Complexity	$O(n^2)$	O(n log n)
Efficiency	Slow	Fast
Use Case	Small data	Large datasets

Task Description #5 (Transparency in AI Recommendations)

Task: Use AI to create a product recommendation system.

Prompt:

"Generate a recommendation system that also provides reasons for each suggestion."

Expected Output:

• Code with explainable recommendations.

• Evaluation of whether explanations are understandable.

CODE:

```
products = [
    {"name": "Laptop", "category": "Electronics", "price": 50000, "rating": 4.5},
    {"name": "Smartphone", "category": "Electronics", "price": 30000, "rating": 4.3},
    {"name": "Headphone", "category": "Electronics", "price": 2000, "rating": 4.1}
]

results = recommend_products(user_preferences, products)

if results:
    for item in results:
        print("\nRecommended Product:", item["product"])
        print("Reason(s):")
        for reason in item["reasons"]:
            print("-", reason)
else:
    print("\nNo products match your preferences.")
```

```
Enter liked categories (comma-separated): music
Enter maximum budget: 5000
Enter minimum rating: 5

Recommended Product: Headphone
Reason(s):
- fits within your budget
```