Lab Assignment - 5.1

NAME : RISHIK.P
BATCH : 12
ROLLNO: 2303A51778

Task 1: Privacy in API Usage

Scenario

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

Prompt used:

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

Code:

```python
#Generate code to fetch weather data securely without exposing API keys in the code.
import os
import requests
from dotenv import load_dotenv
from pathlib import Path

# Load environment variables from .env file
env_path = Path(__file__).parent / ".env"
load_dotenv(dotenv_path=env_path)

API_KEY = os.getenv("WEATHER_API_KEY")
BASE_URL = "https://api.openweathermap.org/data/2.5/weather"

def get_weather(city_name):
    if not API_KEY:
        raise ValueError("API key is not set. Please set WEATHER_API_KEY.")

    params = {
        "q": city_name,
        "appid": API_KEY,
        "units": "metric"
    }

    response = requests.get(BASE_URL, params=params)
    if response.status_code == 200:
        return response.json()
    else:
        response.raise_for_status()

def display_weather_info(weather_data):
    city = weather_data["name"]
    country = weather_data["sys"]["country"]
    temperature = weather_data["main"]["temp"]
    description = weather_data["weather"][0]["description"]

    print(f"Weather in {city}, {country}")
    print(f"Temperature: {temperature}°C")
    print(f"Description: {description.capitalize()}")

try:
    city_name = input("Enter city name: ")
    weather_data = get_weather(city_name)
    display_weather_info(weather_data)
except Exception as e:
    print("Error fetching weather data:", e)
```

Task 2: Privacy & Security in File Handling

Scenario

Use an AI tool to generate a Python script that stores user data (name, email, password) in a file Check if the AI stores sensitive data in plain text or without encryption.

Prompt used:

Create a Python program to store user name, email, and password in a file. Identify privacy risks and rewrite it using secure password storage.

Code:

```python
#Create a Python program to store user name, email, and password in a file.
#Identify privacy risks and rewrite it using secure password storage.
import hashlib

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def save_user_details(name, email, hashed_password, filename="user_details.txt"):
    with open(filename, "a") as file:
        file.write(f"Name: {name}, Email: {email}, Hashed Password: {hashed_password}\n")

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

hashed_password = hash_password(password)
save_user_details(name, email, hashed_password)

print("User details saved securely.")
```

Output:

```
sushanth@Sushanth-2 Ai Coding % /usr/bin/python3 "/Users/sushanth/Downloads/College/Ai Coding/lab5_1"
Enter your name: sushanth
Enter your email: sush@gmail.com
Enter your password: honey123
User details saved securely.
sushanth@Sushanth-2 Ai Coding %
```

Explanation:

The slicing method simplifies the logic and removes unnecessary variables and loops. The slicing method is faster in practice due to internal optimizations in Python.
Readability is significantly improved.

Task 3: Transparency in Algorithm Design

Scenario

The string reversal logic is reused in multiple parts of an application

Prompt used:

Generate a Python function that checks whether a given number is an Armstrong number.
Include clear comments explaining each step.

Code:

```
def is_armstrong_number(number):
    digits = str(number)
    num_digits = len(digits)
    sum_of_powers = 0

    for digit in digits:
        sum_of_powers += int(digit) ** num_digits

    return sum_of_powers == number

try:
    user_input = int(input("Enter a number to check if it is an Armstrong number: "))
    if is_armstrong_number(user_input):
        print(f"{user_input} is an Armstrong number.")
    else:
        print(f"{user_input} is not an Armstrong number.")
except ValueError:
    print("Please enter a valid integer.")
save_user_details
```

Output:

```
Enter a number to check if it is an Armstrong number: 123
123 is not an Armstrong number.
```

Explanation

1. The function is_armstrong_number takes an integer input number.
2. It converts the number to a string to easily access each digit and calculates the number of digits.
3. It initializes a variable sum_of_powers to store the sum of each digit raised to the power of the number of digits.
4. It iterates through each digit, converts it back to an integer, raises it to the power of num_digits, and adds it to sum_of_powers.
5. Finally, it checks if sum_of_powers is equal to the original number and returns True or False accordingly.
6. The user is prompted to input a number, and the program checks if it is an Armstrong number, printing the result.

Task 4: Transparency in Algorithm Comparison

Scenario: Use AI to create a product recommendation system

Prompt used:
Generate Python code for QuickSort and BubbleSort, with comments explaining how each works.

Code:

```
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

def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

try:
    user_input = input("Enter numbers to sort, separated by spaces: ")
    arr = list(map(int, user_input.split()))

    print("Bubble Sort Result:", bubble_sort(arr.copy()))
    print("Quick Sort Result:", quick_sort(arr.copy()))
except ValueError:
    print("Please enter valid integers.")
```

Output:

```
Enter numbers to sort, separated by spaces: 1 3 5 7 9
Bubble Sort Result: [1, 3, 5, 7, 9]
Quick Sort Result: [1, 3, 5, 7, 9]
```

Explanation:
QuickSort works by selecting a 'pivot' element and partitioning the other elements into two sub-arrays according to whether they are less than or greater than the pivot.

Task 5: Transparency in AI Recommendations

Scenario: Use AI to create a product recommendation system.

Prompt used:

```
#Generate a recommendation system that also provides reasons for each suggestion.
products = [
    {"name": "Wireless Mouse", "category": "Electronics", "interests": {"technology", "gadgets"}},
    {"name": "Yoga Mat", "category": "Fitness", "interests": {"health", "wellness"}},
    {"name": "Cookbook", "category": "Books", "interests": {"cooking", "food"}},
    {"name": "Smartphone", "category": "Electronics", "interests": {"technology", "communication"}},
    {"name": "Running Shoes", "category": "Fitness", "interests": {"health", "sports"}},
    {"name": "Mystery Novel", "category": "Books", "interests": {"reading", "entertainment"}}
]

def recommend_products(user_interests):
    recommendations = []

    for product in products:
        product_interests = product.get("interests", set())
        if any(interest in user_interests for interest in product_interests):
            explanation = (
                f"Recommended because it matches your interest in "
                f"{', '.join(set(user_interests).intersection(product_interests))}."
            )
            recommendations.append({
                "product": product["name"],
                "category": product["category"],
                "explanation": explanation
            })
    return recommendations

user_input = input("Enter your interests (comma-separated): ")
user_interests = [interest.strip().lower() for interest in user_input.split(",")]

recommended_products = recommend_products(user_interests)

if recommended_products:
    print("Recommended Products:")
    for rec in recommended_products:
        print(f"- {rec['product']} ({rec['category']}): {rec['explanation']}")
else:
    print("No recommendations available based on your interests.")
```

Output:

```
sushanth@Sushanth-2 Ai Coding % /usr/bin/python3 "/Users/sushanth/Downloads/College/Ai Coding/lab5_1"
Enter your interests (comma-separated): smartphones
No recommendations available based on your interests.
sushanth@Sushanth-2 Ai Coding % /usr/bin/python3 "/Users/sushanth/Downloads/College/Ai Coding/lab5_1"
Enter your interests (comma-separated): gadgets
Recommended Products:
- Wireless Mouse (Electronics): Recommended because it matches your interest in gadgets.
```

Explanation: The function recommend movies takes user preferences and a movie database as input. It initializes empty lists for recommendations and reasons. It iterates through each movie in the database, calculating a score based on how well the movie matches user preferences. If the score meets a certain threshold, the movie is added to the recommendations list along the reasons for its recommendation. Finally, the main block demonstrates how to use the function with sample data and prints the recommendations along with their reasons.