**Problem 1: Real-Time Weather Monitoring System**

**Scenario:**

You are developing a real-time weather monitoring system for a weather forecasting company. The system needs to fetch and display weather data for a specified location.

**Tasks:**

1. Model the data flow for fetching weather information from an external API and displaying it to the user.
2. Implement a Python application that integrates with a weather API (e.g., OpenWeatherMap) to fetch real-time weather data.
3. Display the current weather information, including temperature, weather conditions, humidity, and wind speed.
4. Allow users to input the location (city name or coordinates) and display the corresponding weather data.

**Deliverables:**

* Data flow diagram illustrating the interaction between the application and the API.
* Pseudocode and implementation of the weather monitoring system.
* Documentation of the API integration and the methods used to fetch and display weather data.
* Explanation of any assumptions made and potential improvements.

**Approach:**

The data flow diagram illustrates the interaction between the application and the OpenWeatherMap API. The application receives user input for the location, fetches the weather data from the API, and displays the current weather information to the user.

**+---------------+**

**| User Input |**

**+---------------+**

**|**

**| Location**

**v**

**+---------------+**

**| Fetch Weather |**

**| Data from API |**

**+---------------+**

**|**

**| Weather Data**

**v**

**+---------------+**

**| Display Weather |**

**| Information to |**

**| User |**

**+---------------+**

**Pseudocode:**

1. Create `state\_cities.py` file with the following content:

- Dictionary `STATE\_CITIES` mapping states to their cities

2. Main script:

- Import necessary modules:

- `requests` for making API requests

- `STATE\_CITIES` from `state\_cities.py` for accessing the dictionary

- Define constants:

- `API\_KEY`: Your API key for the weather service

- `BASE\_URL`: Base URL for the weather API

- Define function `fetch\_weather(city)`:

- Construct URL for API request using the city name, `API\_KEY`, and `BASE\_URL`

- Try to send the request to the API

- If successful, return the JSON response

- If an error occurs, print the error message and return `None`

- Define function `display\_weather(weather\_data, city)`:

- If `weather\_data` is not `None`:

- Extract relevant weather information from `weather\_data`

- Print the weather information (temperature, conditions, humidity, wind speed) for the city

- If `weather\_data` is `None`, print that weather data is not found for the city

- Define function `get\_cities\_for\_state(state)`:

- Return the list of cities for the given state from `STATE\_CITIES`

- If the state is not found, return an empty list

- Define main function `main(location)`:

- If `location` is found in `STATE\_CITIES` (treated as a state):

- Get the list of cities for the state

- If no cities are found, print a message and return

- For each city in the list:

- Fetch the weather data for the city

- Display the weather information for the city

- If `location` is not found in `STATE\_CITIES` (treated as a single city):

- Fetch the weather data for the city

- If weather data is found, display the weather information for the city

- If no data is found, print a message that no data is found for the city

- Prompt the user to enter a city or state name

- Call `main(location)` with the user input

**Detailed explanation of the actual code:**

1. **fetch\_weather**: This function takes a city name as input and fetches the weather data from the OpenWeatherMap API. It constructs the URL using the city name, API key, and base URL. Using the requests library, it sends a GET request to the API. If the response is successful (200 status code), it parses the JSON response and returns the data. If there is an error fetching the data, it prints an error message and returns None.
2. **display\_weather**: This function takes the weather data and city name as inputs and displays the current weather information to the user. If the weather data is available, it extracts and prints the temperature, weather conditions, humidity, and wind speed. If the weather data is not available, it prints an error message indicating that the weather data was not found for the specified city.
3. **get\_cities\_for\_state**: This function takes a state name as input and returns the list of cities for the given state from the STATE\_CITIES dictionary. If the state is not found in the dictionary, it returns an empty list.
4. **main**: This function handles user input and calls the fetch\_weather and display\_weather functions. It prompts the user to enter a location, which can be either a city name or a state name. If the location is found in the STATE\_CITIES dictionary (treated as a state), it fetches and displays the weather data for each city in that state. If the location is not found in the dictionary (treated as a single city), it fetches and displays the weather data for the city.

**Assumptions made (if any):**

 The user will always enter a valid location (either a valid city name or a valid state name).

 The OpenWeatherMap API will always return data in the expected format.

**Limitations:**

1. The application does not handle errors in the API response other than printing an error message.
2. It does not validate user input for location (e.g., it does not check if the user input is a valid city or state name before making the API request).
3. It does not provide any additional features like historical weather data or weather forecasts.

**Assumptions made (if any):**

1. The user will always enter a valid location.
2. The OpenWeatherMap API will always return data in the expected format.

**Limitations:**

 The application does not handle errors in the API response other than printing an error message.

 It does not validate user input for location (e.g., it does not check if the user input is a valid city or state name before making the API request).

 It does not provide any additional features like historical weather data or weather forecasts.

**Code:**

import requests

from state\_cities import STATE\_CITIES # Import the dictionary from the created file

# Constants

API\_KEY = '0c975279379f26973f06107a2186312b'

BASE\_URL = 'http://api.openweathermap.org/data/2.5/weather'

def fetch\_weather(city):

# Construct the URL for the API request

url = f"{BASE\_URL}?q={city}&appid={API\_KEY}&units=metric"

try:

# Send the request to the API

response = requests.get(url)

response.raise\_for\_status() # Raise an error for bad status codes

return response.json()

except requests.exceptions.RequestException as e:

print(f"Error fetching data for {city}: {e}")

return None

def display\_weather(weather\_data, city):

if weather\_data:

# Extract data from the response

main = weather\_data.get('main', {})

weather = weather\_data.get('weather', [{}])[0]

wind = weather\_data.get('wind', {})

temperature = main.get('temp', 'N/A')

humidity = main.get('humidity', 'N/A')

conditions = weather.get('description', 'N/A')

wind\_speed = wind.get('speed', 'N/A')

# Display the weather information

print(f"Weather in {city}:")

print(f" Temperature: {temperature}°C")

print(f" Weather Conditions: {conditions.capitalize()}")

print(f" Humidity: {humidity}%")

print(f" Wind Speed: {wind\_speed} m/s")

print() # Blank line for readability

else:

print(f"Weather data not found for {city}.")

print() # Blank line for readability

def get\_cities\_for\_state(state):

return STATE\_CITIES.get(state, [])

def main(location):

if location in STATE\_CITIES:

cities = get\_cities\_for\_state(location)

if not cities:

print(f"No cities found for state {location}.")

return

for city in cities:

weather\_data = fetch\_weather(city)

display\_weather(weather\_data, city)

else:

# Treat location as a single city

weather\_data = fetch\_weather(location)

if weather\_data and weather\_data.get("cod") == 200:

display\_weather(weather\_data, location)

else:

print(f"No data found for {location}. Please ensure the city/state name is correct.")

# Example usage:

location = input("Enter a city name or Indian state name: ")

main(location)

**Sample Output / Screen Shots**

