# Assignment: Python Programming for GUI Development

# **Name:** P. Sonu

# **Register Number:** 192365057

# **Department:** CSE (cyber-security)

# **Date of Submission:** 26 -08-2024

# Problem 1: Real-Time Weather Monitoring System

# 1.Data Flow Diagram:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Input  (City name) |  | Fetch whether Data From API  (OpenWhetherMap) |  | Store data in  Database  (SQLite) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Valid Input |  | Process API Data |  | Save data to DB |

|  |
| --- |
| Plot Data  (Matplotlib) |

2. Implementation:

Here’s the complete Python code for the real-time weather monitoring system. The application fetches weather data from the OpenWeatherMap API, processes it, and displays the current weather information.

Pseudocode:

1.Initialize:

* Set up API credentials.
* Create a function to fetch weather data from the API.
* Create a function to parse the API response and extract relevant information.
* Create a function to display the weather data to the user.

2.User Input:

* Prompt the user to input a city name.
* Call the API fetch function with the user’s input.

3.Fetch and Display Data:

* Fetch the weather data from the API.
* Parse the data.
* Display the weather information to the user.

import requests

Python code implementation:

import requests

def get\_weather(api\_key, city):

# Base URL for OpenWeatherMap API

base\_url = "https://api.openweathermap.org/data/2.5/weather"

# Parameters for the API request

params = {

'q': city,

'appid': api\_key,

'units': 'metric' # For temperature in Celsius; use 'imperial' for Fahrenheit

}

# Send GET request to OpenWeatherMap

response = requests.get(base\_url, params=params)

# Checking if the request was successful or not

if response.status\_code == 200:

# Parse the JSON response

data = response.json()

# Extract weather information

main = data['main']

weather = data['weather'][0]

temperature = main['temp']

weather\_description = weather['description']

# weather information

print(f"City: {city}")

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

print(f"Weather: {weather\_description.capitalize()}")

else:

print("Error:", response.status\_code, response.reason)

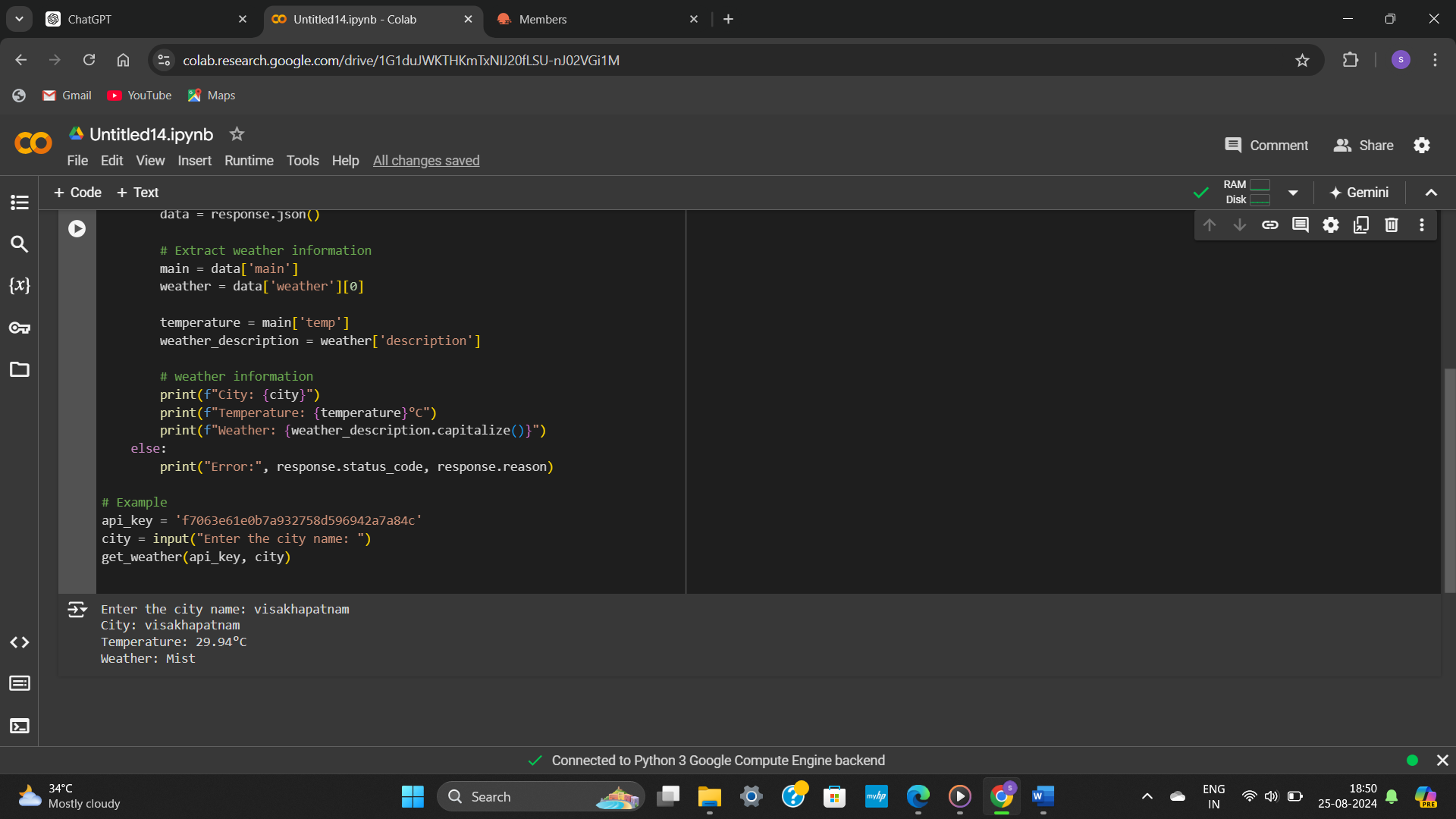
# Example

api\_key = 'f7063e61e0b7a932758d596942a7a84c'

city = input("Enter the city name: ")

get\_weather(api\_key, city)

Working example from collab:



# 3.Display the Current weather information:

# Enter the city name: Visakhapatnam

# City: Visakhapatnam

Temperature: 29.94 C

Weather : Mist

# 4.User Input:

The code allows users to input the city name directly in the terminal. It then fetches and displays the corresponding weather data.

# 5.Documentation:

**API Integration:**

* **API Used:** OpenWeatherMap API
* **Endpoint:** **https://api.openweathermap.org/data/2.5/weather**
* **Parameters:**
  + **q**: City name (e.g., **q=visakhapatnam**)
  + **appid**: API key (e.g., **appid=YOUR\_API\_KEY\_HERE**)
  + **units**: Metric (for temperatures in Celsius; use **imperial** for Fahrenheit)

**Methods:**

* **get\_weather(api\_key, city)**: Fetches weather data from the API, parses it, and displays it in a human-readable format.

**Assumptions:**

* The API key is valid and active.
* The city name input by the user is valid and correctly formatted.

**Potential Improvements:**

* **Error Handling:** Enhance error handling to manage network issues, invalid inputs, or API errors more gracefully.
* **GUI Interface:** Develop a graphical user interface (GUI) using a library like Tkinter or PyQt for a more user-friendly experience.
* **Unit Testing:** Add unit tests to verify the functionality of data fetching and parsing.

This implementation provides a straightforward approach to fetching and displaying real-time weather data, making it suitable for learning and further development.