

Name – divy vishwakarma

Registration No. – 25MIB10044

Email –

divyvishwakarma.25mib10044@vitbhopal.ac.in

***Project – creating a program for weather
and time by using python language***

Content

- ***Problem statement***
- ***Requirement***
- ***Design***
- ***Algorithm***
- ***Flowchart***
- ***Code & implementation***
- ***Discussion***
- ***Resources***
- ***reference***

Problem Statement

This project aims to build a basic Python program that displays up-to-date weather info for any city the user types in. The program needs to link to an online weather service, grab live data, and show helpful details like temperature, weather conditions, city name, country, and local time. Users should be able to look up several cities without having to restart the program. It also needs to deal with problems like wrong city names or internet troubles.

Requirement

1. Software Requirements

- Python (version 3.x)
- VS Code or any Python-supported editor
- Internet connection (you need this to get weather data)
- Requests library (pip install requests)

2. Functional Requirements

- The program lets users type in any city name.
- It grabs weather data from the wttr.in API.

It shows:

- City name
- Country

- Temperature (°C)
- Weather condition
- Local observation time
- The program keeps running until the user types "quit".

3. Error Handling Requirements

- Show a message if the city name isn't valid.
- Deal with network or connection problems.
- Handle missing or incomplete JSON data.
- Manage empty input from the user.

4. System Requirements

- Windows / Mac / Linux (any OS that runs Python)
- Steady internet connection for API communication

Design

1. Weather Function

There's a function called `get_weather (city_name)` that takes care of everything needed to fetch the weather for a city.

Here's how the function works:

- A URL is formed using the city name.
- The program sends a request to `wtr.in` to get the weather data.
- The data is converted into JSON format.
- Important details like temperature, weather condition, city, country, and local time are taken from the JSON.

- The function prints these details for the user.
- It also handles errors if the city is wrong or if there is no internet.

2. Main Program Loop

The main part of the program:

- Shows a welcome message.
- Keeps asking the user to enter a city name.
- If the user types "quit", the program stops.
- If a valid city name is entered, it calls the `get_weather()` function to show the weather.
- If the input is empty, it asks the user again.

3. Input and Output

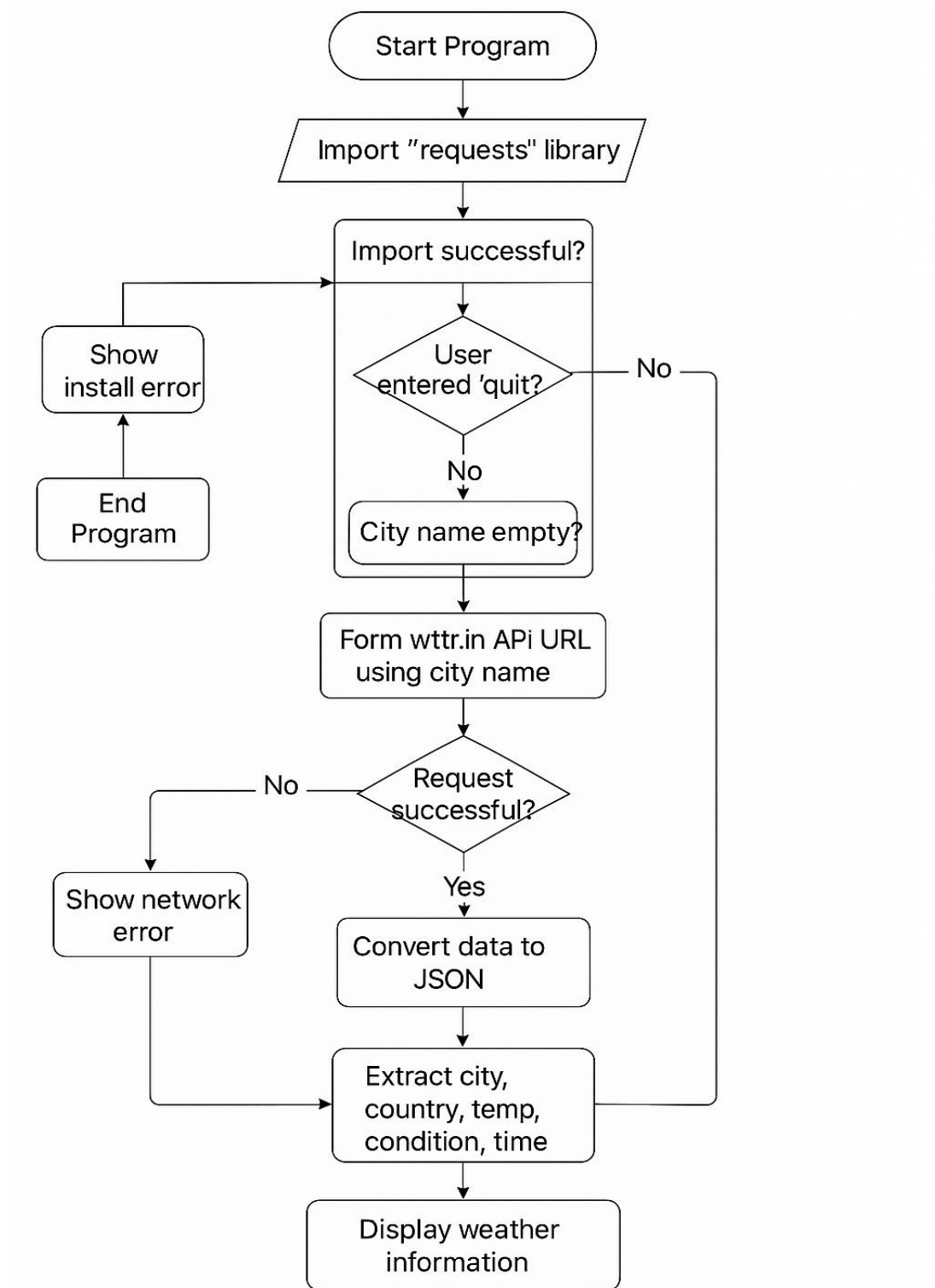
- Input: City name typed by the user.
- Output: Weather report including temperature, weather condition, city name, country, and local time.

4. Error Handling

- If the weather service cannot be reached or gives no data, the program shows an error instead of crashing.

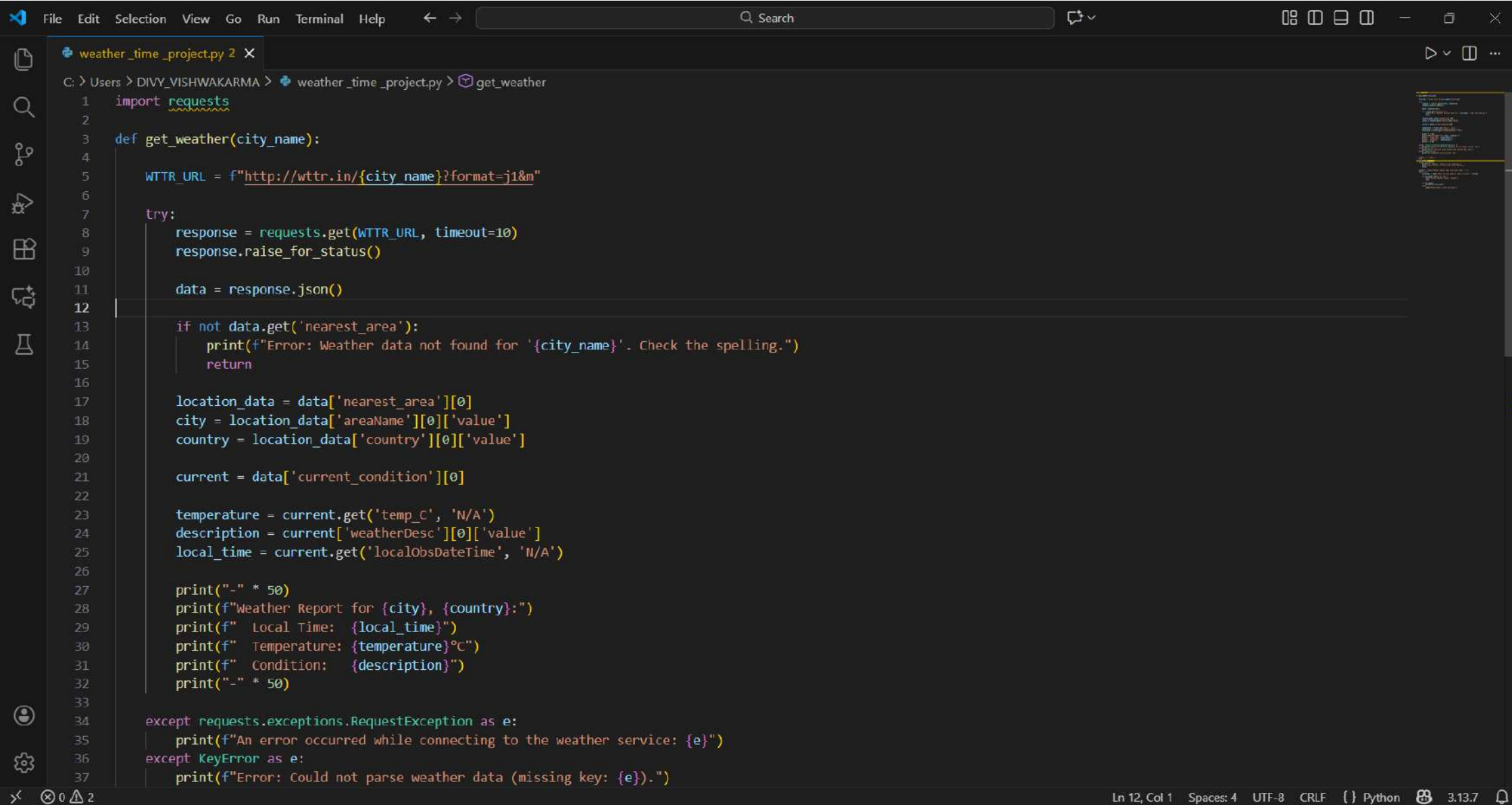
ALGORITHM

- 1) Start the program.
- 2) Try to import the requests library.
If it's not installed, tell the user to install it and stop.
- 3) Show a welcome message for the weather checker.
- 4) Keep asking the user to type a city name.
If they type "quit", then stop the program.
- 5) When the user enters a city name:
 - If the input is empty, ask them again.
 - Otherwise continue.
- 6) Create the wttr.in link using the city name.
- 7) Send a request to the website to get the weather details.
If any network error happens, show an error message.
- 8) Convert the received data into JSON form.
- 9) Check if the website actually returned weather information for that city.
If not, tell the user the city is not found.
- 10) If the data is valid, take out the useful details:
 - City name
 - Country
 - Temperature
 - Weather condition
 - Local observation time
- 11) Show all these details nicely on the screen.
- 12) Go back to asking for the next city until the user quits.
- 13) End



FLOWCHART

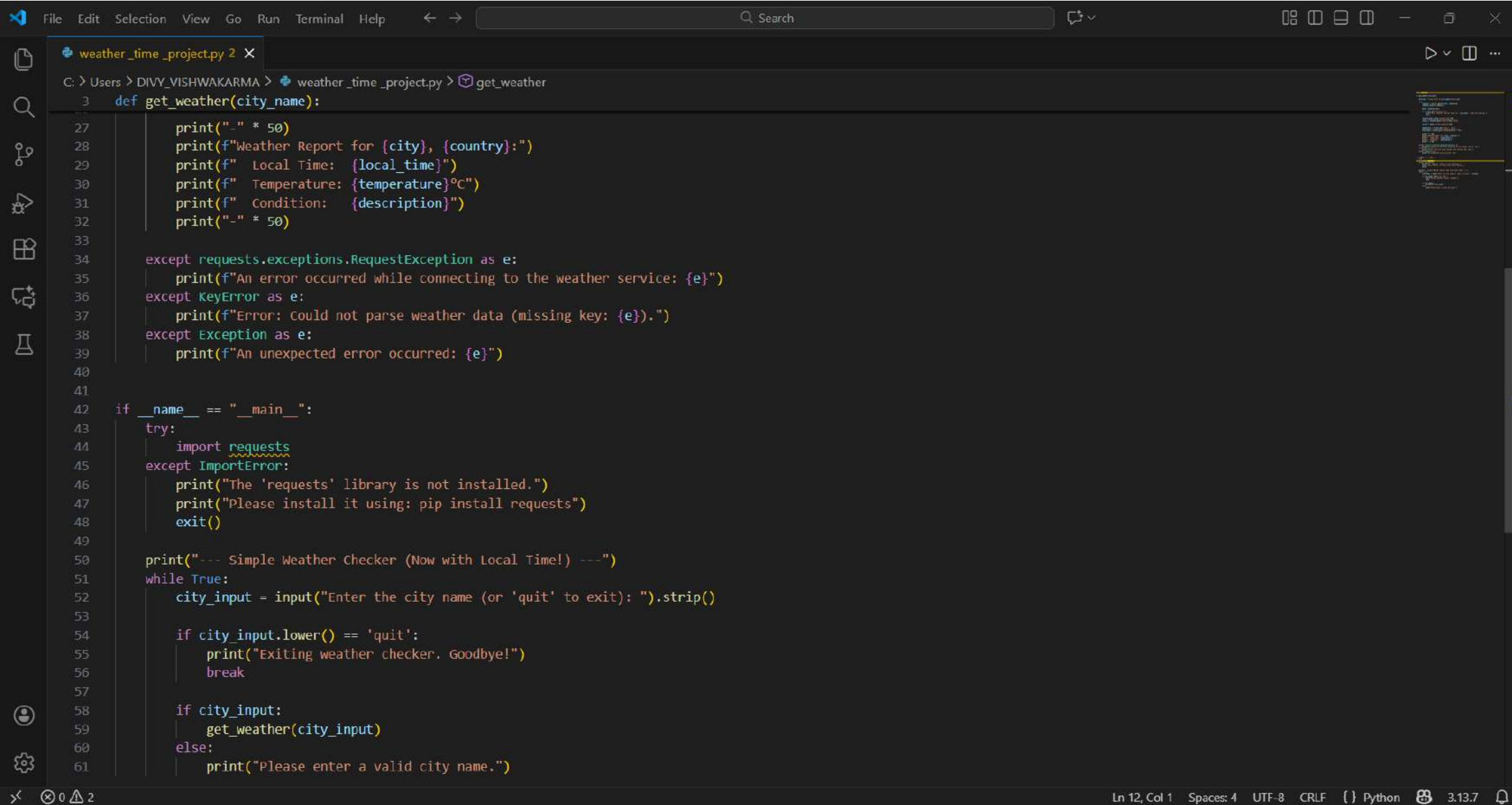
Code/implementation



The image shows a Visual Studio Code editor window with a Python script named `weather_time_project.py`. The script is designed to fetch weather data from a specific URL and format it into a report. It includes error handling for network exceptions and missing keys in the JSON response. The script is currently at line 12, column 1.

```
1 import requests
2
3 def get_weather(city_name):
4
5     WTTR_URL = f"http://wttr.in/{city_name}?format=j1&m"
6
7     try:
8         response = requests.get(WTTR_URL, timeout=10)
9         response.raise_for_status()
10
11         data = response.json()
12
13         if not data.get('nearest_area'):
14             print(f"Error: Weather data not found for '{city_name}'. Check the spelling.")
15             return
16
17         location_data = data['nearest_area'][0]
18         city = location_data['areaName'][0]['value']
19         country = location_data['country'][0]['value']
20
21         current = data['current_condition'][0]
22
23         temperature = current.get('temp_c', 'N/A')
24         description = current['weatherDesc'][0]['value']
25         local_time = current.get('localObsDateTime', 'N/A')
26
27         print("-" * 50)
28         print(f"Weather Report for {city}, {country}:")
29         print(f"  Local Time: {local_time}")
30         print(f"  Temperature: {temperature}°C")
31         print(f"  Condition: {description}")
32         print("-" * 50)
33
34     except requests.exceptions.RequestException as e:
35         print(f"An error occurred while connecting to the weather service: {e}")
36     except KeyError as e:
37         print(f"Error: Could not parse weather data (missing key: {e}).")
```

The status bar at the bottom indicates the current position is Line 12, Column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.13.7, and 2 warnings.



```
File Edit Selection View Go Run Terminal Help
C:\Users\DIVY_VISHWAKARMA > weather_time_project.py > get_weather
3 def get_weather(city_name):
27     print("-" * 50)
28     print(f"Weather Report for {city}, {country}:")
29     print(f"    Local Time: {local_time}")
30     print(f"    Temperature: {temperature}°C")
31     print(f"    Condition: {description}")
32     print("-" * 50)
33
34     except requests.exceptions.RequestException as e:
35         print(f"An error occurred while connecting to the weather service: {e}")
36     except KeyError as e:
37         print(f"Error: Could not parse weather data (missing key: {e}).")
38     except Exception as e:
39         print(f"An unexpected error occurred: {e}")
40
41
42 if __name__ == "__main__":
43     try:
44         import requests
45     except ImportError:
46         print("The 'requests' library is not installed.")
47         print("Please install it using: pip install requests")
48         exit()
49
50     print("--- Simple Weather Checker (Now with Local Time!) ---")
51     while True:
52         city_input = input("Enter the city name (or 'quit' to exit): ").strip()
53
54         if city_input.lower() == 'quit':
55             print("Exiting weather checker. Goodbye!")
56             break
57
58         if city_input:
59             get_weather(city_input)
60         else:
61             print("Please enter a valid city name.")
```

Ln 12, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.7

Discussion

This program checks the weather for any city the user types in. It uses an online weather service called wttr.in that gives weather data in JSON format. The code sends a request to this website and gets the data back using the requests library.

The program loops, so users can type in different city names without starting over. When someone enters a city name, the program builds a URL with that name and asks the server for information. If something goes wrong, like no internet or the website not answering, the program shows an error message instead of crashing.

If the server responds, the program turns the data into JSON and makes sure the city exists. For real cities, it gathers details such as temperature, weather condition , city name, country, and local time. It then prints these details in an easy-to-read format for the user.

The program continues to operate until the user enters "quit," which brings the loop to an end and closes the program. All in all, the code is easy to use straight forward, and demonstrates how to get up-to-date weather data using API calls and JSON information.

Resources Used

1. Python Programming Language

Used to write the entire program and handle user input, loops, error handling, and function creation.

2. Requests Library (Python)

This library was used to send HTTP requests to the weather API and receive data from the server.

3. wttr.in Weather API

This online service provided real-time weather data in JSON format for any city entered by the user.

4. VS Code (Visual Studio Code)

Used as the code editor to write, edit, and run the Python program.

References

1. wttr.in – Weather Information Service

Source: <https://wttr.in> (public weather API)

2. Requests Library Documentation.

Source: <https://requests.readthedocs.io/>

3. Python Official Documentation.

Source: <https://docs.python.org/>

4. VS Code (Visual Studio Code) Documentation.

Source: <https://code.visualstudio.com/docs>

THANK YOU

