# **Contents**

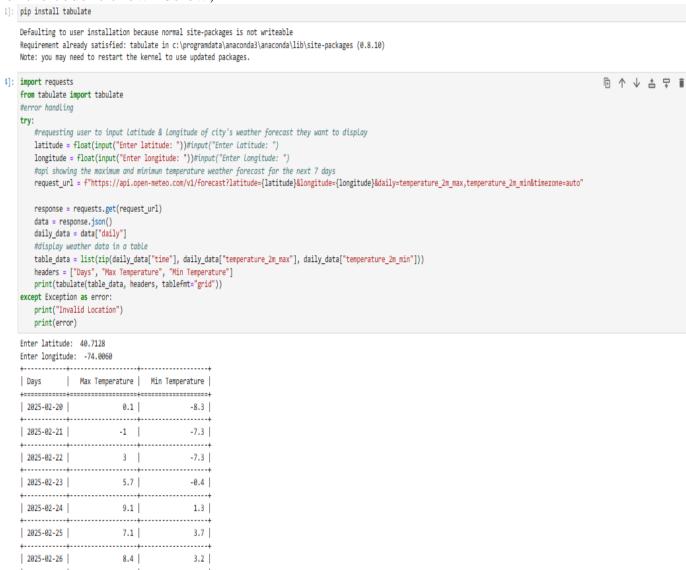
1	Project Title	2
	Project Code	
	Description	
	Technical Details	
	Improved Project Code	
	Error Handling	

# 1 Project Title

Weather Forecast Application

### 2 Project Code

I had some issues running my replit account. In order not to waste so much time dwelling on how to fix it, I ended up using jupyter lab so therefore, a screenshot of the code is shown below;



### 3 Description

Using latitude and longitude, this Python script retrieves and shows the sevenday weather prediction, including the maximum and minimum temperatures, for a user-specified location. The tabulate package is used to format the output into a legible table when the data is obtained from the Open-Meteo API. As seen in the screenshot above, I made sure to install tabulate to display data in tabular form.

#### Key Features:

- Utilises an API request to obtain real-time weather data.
- Displays results in a formatted table for improved readability.
- Uses error handling to handle invalid inputs or request failures.
- Allows users to input geographic coordinates (latitude and longitude) to retrieve weather data.

### 4 Technical Details

Libraries used:

Requests -- Responds to HTTP requests to retrieve meteorological information.

tabulate -- Creates a table with weather data and formats it.

Integrated features:

input() -- Takes latitude and longitude input from the user.

float() -- Transforms input into numbers.

Potential errors (invalid input, API failures, etc.) are handled by the try-except function.

#### API Usage:

Weather Data API: Open-Meteo's weather forecast API.

Request URL Format

Response Structure (JSON Format)

The script extracts the "time", "temperature\_2m\_max", and

"temperature\_2m\_min" values and displays them in a grid table.

### Error Handling:

- If the user enters non-numeric latitude/longitude, a ValueError occurs.
- If the API fails or returns an unexpected response, an exception is caught.
- Errors print "Invalid Location" along with the actual error message.

# 5 Improved Project Code

With the use of Geocoding API to fetch city coordinates instead of the use of latitude and longitude, all you need to do is input the city.

```
import requests
                                                                                                                                                                    □↑↓占♀ⅰ
    from tabulate import tabulate
    def get_coordinates(city):
        #"""Fetch Latitude and Longitude of a city using Open-Meteo Geocoding API."""
           geo_url = f"https://geocoding-api.open-meteo.com/v1/search?name={city}&count=1&language=en&format=json"
           response = requests.get(geo_url)
           if response.status_code != 200:
               print("Error fetching city coordinates!")
               return None, None
           weather_data = response.json()
           if "results" not in weather_data or len(weather_data["results"]) == 0:
               print(f"City '{city}' not found!")
               return None, None
           city_data = weather_data["results"][0]
           return city_data["latitude"], city_data["longitude"]
       except requests.exceptions.RequestException as e:
           print("Network error while fetching coordinates:", e)
           return None, None
    def get_weather_forecast(city):
        #"""Fetch and display weather forecast for a given city."""
       latitude, longitude = get_coordinates(city)
       if latitude is None or longitude is None:
           # Fetch weather data using coordinates
           weather_url = f"https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitude={longitude}&daily=temperature 2m max,temperature 2m min&timezone=auto"
           response = requests.get(weather_url)
           if response.status_code != 200:
               print(f"API request failed! Status code: {response.status_code}")
               return
           weather_data = response.json()
           if "daily" not in data:
               print("Invalid API response: 'daily' data not found")
```

```
if "daily" not in data:
         print("Invalid API response: 'daily' data not found")
      daily_data = weather_data["daily"]
      table_data = list(zip(daily_data["time"], daily_data["temperature_2m_max"], daily_data["temperature_2m_min"]))
      headers = ["Days", "Max Temperature", "Min Temperature"]
      print(f"\nWeather forecast for {city.capitalize()}:\n")
      print(tabulate(table_data, headers, tablefmt="grid"))
   except requests.exceptions.RequestException as e:
      print("Network error:", e)
   except KeyError as e:
      print(f"Unexpected API response structure. Missing key: {e}")
   except Exception as error:
      print("An unexpected error occurred:", error)
city_name = input("Enter city name: ").strip()
get_weather_forecast(city_name)
Enter city name: lagos
Weather forecast for Lagos:
| Days | Max Temperature | Min Temperature |
+=====+===++=======+
2025-02-25
                     32.5
2025-02-26
                     31.2
2025-02-27 32.7
                                      27.5
2025-03-01
                     32.6
                                        28
2025-03-02 | 33.1 |
                                       28.1
2025-03-03
```

# 6 Error Handling

- Invalid City Name Prints "City '<name>' not found!"
- Network Errors Catches requests.exceptions.RequestException
- API Response Issues Handles missing keys using KeyError
- Invalid JSON Response Prints "Unexpected API response structure".