Weather Forecast Report Application in Python

Introduction

A weather forecast report application allows users to fetch real-time weather data for a given city using the **OpenWeatherMap API**. This application is built using Python and provides weather details like temperature, humidity, wind speed, and weather conditions in a graphical user interface (GUI). The project aims to help users get instant weather updates for travel, outdoor activities, or daily planning.

Objectives

- To develop a Python-based weather forecast application.
- To fetch real-time weather data using the OpenWeatherMap API.
- To create an interactive GUI using Tkinter.
- To enhance the application with a 5-day forecast feature.
- To visualize weather trends using charts.

Requirements

- **Python 3.x**
- **Libraries Needed:**
  - `requests` (for API calls)
  - `tkinter` (for GUI)

Step 1: Getting Started

1. Obtain OpenWeatherMap API Key

Sign up at fhttps://api.openweathermap.org/data/2.5/weather?q={city}&appid=7d890a62d9d759226b0ec88c803d273d to get your API key.

2. Create a Python Script

Step 1: Writing the Python Code

Fetching Weather Data

```
def data_get():
    city = city_name.get()



    if city:
        try:


            url =
f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid=7d890a62d9d
759226b0ec88c803d273d"
            response = requests.get(url)
            data = response.json()



            if data["cod"] == 200:


                w_label1.config(text=data["weather"][0]["main"])
                wb_label1.config(text=data["weather"][0]["description"])
                temp_label1.config(text=str(int(data["main"]["temp"] - 273.15)))
                per_label1.config(text=data["main"]["pressure"])
```

```
        else:

            w_label1.config(text="City not found")
            wb_label1.config(text="")
            temp_label1.config(text="")
            per_label1.config(text="")
    except requests.exceptions.RequestException as e:

        print(f"Error fetching data: {e}")
        w_label1.config(text="Error")
        wb_label1.config(text="")
        temp_label1.config(text="")
        per_label1.config(text="")
else:

    w_label1.config(text="Please select a city")
    wb_label1.config(text="")
    temp_label1.config(text="")
    per_label1.config(text="")
```

Building the GUI with Tkinter

```
win = Tk()

win.title("Wscube Weather App")

win.config(bg="blue")

win.geometry("500x570")



name_label = Label(win, text="Wscube Weather App", font=("Times New Roman", 30, "bold"))

name_label.place(x=25, y=50, height=50, width=450)



city_name = StringVar()

list_name = ["Andhra Pradesh", "Arunachal Pradesh", "Assam", "Bihar", "Chhattisgarh", "Goa", "Gujarat", "Haryana",

        "Himachal Pradesh", "Jammu and Kashmir", "Jharkhand", "Karnataka", "Kerala", "Madhya Pradesh", "Maharashtra",

        "Manipur", "Meghalaya", "Mizoram", "Nagaland", "Odisha", "Punjab", "Rajasthan", "Sikkim", "Tamil Nadu",

        "Telangana", "Tripura", "Uttar Pradesh", "Uttarakhand", "West Bengal", "Andaman and Nicobar Islands",

        "Chandigarh", "Dadra and Nagar Haveli", "Daman and Diu", "Lakshadweep", "National Capital Territory of Delhi",

        "Puducherry"]

com = ttk.Combobox(win, values=list_name, font=("Times New Roman", 20, "bold"), textvariable=city_name)

com.place(x=25, y=120, height=50, width=450)
```

```
w_label = Label(win, text="Weather Climate", font=("Times New Roman", 17))

w_label.place(x=25, y=260, height=50, width=210)

w_label1 = Label(win, text="", font=("Times New Roman", 20))

w_label1.place(x=250, y=260, height=50, width=210)


wb_label = Label(win, text="Weather Description", font=("Times New Roman",
16))

wb_label.place(x=25, y=330, height=50, width=210)

wb_label1 = Label(win, text="", font=("Times New Roman", 17))

wb_label1.place(x=250, y=330, height=50, width=210)


temp_label = Label(win, text="Temperature", font=("Times New Roman", 20))

temp_label.place(x=25, y=400, height=50, width=210)

temp_label1 = Label(win, text="", font=("Times New Roman", 20))

temp_label1.place(x=250, y=400, height=50, width=210)


per_label = Label(win, text="Pressure", font=("Times New Roman", 20))

per_label.place(x=25, y=470, height=50, width=210)

per_label1 = Label(win, text="", font=("Times New Roman", 20))

per_label1.place(x=250, y=470, height=50, width=210)
```

```
done_button = Button(win, text="Done", font=("Times New Roman", 20, "bold"), command=data_get)

done_button.place(y=190, height=50, width=100, x=200)



win.mainloop()
```

Step 3: Implementing

```
com = ttk.Combobox(win, values=list_name, font=("Times New Roman", 20, "bold"), textvariable=city_name)

com.place(x=25, y=120, height=50, width=450)



w_label = Label(win, text="Weather Climate", font=("Times New Roman", 17))

w_label.place(x=25, y=260, height=50, width=210)

w_label1 = Label(win, text="", font=("Times New Roman", 20))

w_label1.place(x=250, y=260, height=50, width=210)


wb_label = Label(win, text="Weather Description", font=("Times New Roman", 16))

wb_label.place(x=25, y=330, height=50, width=210)

wb_label1 = Label(win, text="", font=("Times New Roman", 17))

wb_label1.place(x=250, y=330, height=50, width=210)


temp_label = Label(win, text="Temperature", font=("Times New Roman", 20))

temp_label.place(x=25, y=400, height=50, width=210)
```

```
temp_label1 = Label(win, text="", font=("Times New Roman", 20))

temp_label1.place(x=250, y=400, height=50, width=210)


per_label = Label(win, text="Pressure", font=("Times New Roman", 20))

per_label.place(x=25, y=470, height=50, width=210)

per_label1 = Label(win, text="", font=("Times New Roman", 20))

per_label1.place(x=250, y=470, height=50, width=210)



done_button = Button(win, text="Done", font=("Times New Roman", 20, "bold"),
command=data_get)

done_button.place(y=190, height=50, width=100, x=200)



win.mainloop()
```

Step 5: Additional Features

Enhancing the GUI

- **Adding Icons**: Use images to represent weather conditions.
- **Search History**: Allow users to view recent searches.
- **Multiple Units**: Enable switching between Celsius and Fahrenheit.

Step 6: Testing and Debugging

1. **Error Handling:** If a city is not found, display an error message.
2. **Testing with Different Cities:** Test with various city names to verify data accuracy.
3. **Check API Response:** Ensure API returns correct weather data format.
4. **GUI Performance:** Test for responsiveness and usability.

Step 7: Enhancements and Future Scope

- Adding **graphical representations** of weather trends.
- Including **alerts for extreme weather conditions**.
- Developing a **mobile app version**.
- Implementing **voice command integration**.
- Storing **weather history for analysis**.

Case Study: Real-World Applications

- **Aviation**: Pilots rely on weather data for flight planning.
- **Agriculture**: Farmers use forecasts to determine irrigation schedules.
- **Disaster Management**: Helps authorities prepare for storms and extreme conditions.

Conclusion

This project successfully demonstrates how Python can be used to fetch and display real-time weather data. By integrating APIs and GUI components, we can create a user-friendly weather forecasting application. Future improvements could include real-time notifications and AI-based weather predictions.

Note: Replace `your_api_key` with a valid API key from OpenWeatherMap before running the script.