Python OpenWeatherMap Web API Tutorial GUI

Contents

Time required: 90 minutes

Here's What I Want You to Do

This project builds on the console-based weather program we created earlier. We will learn how to create a GUI weather program using Python and Tkinter.

You will need your **weather_utils.py** file from the previous project.

Here's Why I Want You to Do It

Demonstrate understanding of:

Tkinter, OOP, API, Input

Part 1: Create the GUI

The first step is to create the GUI (Graphical User Interface).

1. Create a Python program named **weather_gui_1.py**
2. The **__init__** method builds a Tkinter program window.
3. We are importing **weather_utils** and **requests** now, even though we don't need them yet.

```
"""
    Name: weather_gui.py
    Author:
    Created:
    Purpose: OOP Tkinter GUI to get and display OpenWeatherMap data
"""
import weather_utils
from tkinter import *
from tkinter.ttk import *
import requests


class OWMGUI:
    def __init__(self):
        self.root = Tk()
        self.root.title("Weather App")
        self.root.iconbitmap("weather.ico")
        # Call methods to create frames and widgets
        self.create_frames()
        self.create_widgets()
        # Start program main loop
        mainloop()
```

4. The **create_frames()** method creates the containing frames for the widgets

```
24  # ---------------------------- CREATE FRAMES -------------------------------#
25      def create_frames(self):
26          """Create frames"""
27          self.title_frame = Frame(self.root, relief=FLAT)
28          self.entry_frame = LabelFrame(
29              self.root, text="Enter Location", relief=GROOVE)
30          self.weather_frame = LabelFrame(
31              self.root, text="Weather", relief=GROOVE)
32
33          # Pack the frames to the edges of the window
34          self.title_frame.pack(fill=X)
35          self.entry_frame.pack(fill=X)
36          self.weather_frame.pack(fill=X)
37
38          # Works with fill=X to expand the frames to
39          # the edges of the window
40          self.title_frame.pack_propagate(False)
41          self.entry_frame.pack_propagate(False)
42          self.weather_frame.pack_propagate(False)
```

5. The **create_widgets()** method creates the buttons, entry boxes and labels.

```
# ---------------------------- CREATE WIDGETS -----------------------------#
    def create_widgets(self):
        """Create widgets"""
        # Create entry widget and set focus
        self.location_entry = Entry(self.entry_frame, width=25)
        self.location_entry.focus_set()

        # Create button to get weather from location
        self.btn_weather = Button(
            self.entry_frame,
            text="Get Weather"
        )

        # Create description labels
        self.lbl_app_title = Label(
            self.title_frame, text="Bill's Weather App",
                                font=("Arial", 16, "bold"))
        self.lbl_temperature = Label(
            self.weather_frame, text="Temperature:")
        self.lbl_description = Label(
            self.weather_frame, text="Description:")

        # Create value display labels
        self.lbl_temperature_value = Label(
            self.weather_frame, width=20, anchor=W, relief=GROOVE)
        self.lbl_description_value = Label(
            self.weather_frame, width=20, anchor=W, relief=GROOVE)
```

We add all the widgets to the window using the grid layout manager.

```
72          # ---------------------- GRID WIDGETS ----------------------------#
73          self.lbl_app_title.grid(row=0, column=0)
74
75          self.location_entry.grid(row=1, column=0, sticky=W)
76          self.btn_weather.grid(row=1, column=1, sticky=W)
77
78          self.lbl_temperature.grid(row=3, column=0, sticky=E)
79          self.lbl_temperature_value.grid(row=3, column=1, sticky=W)
80
81          self.lbl_description.grid(row=4, column=0, sticky=E)
82          self.lbl_description_value.grid(row=4, column=1, sticky=W)
83
84          # Set padding between frames and the window
85          self.title_frame.pack_configure(padx=10, pady=(10, 0))
86          self.entry_frame.pack_configure(padx=10, pady=(10, 0))
87          self.weather_frame.pack(padx=10, pady=10)
88
89          # Set pad padding for all widgets inside each frame
90          # set ipad padding inside the widgets
91          for widget in self.title_frame.winfo_children():
92              widget.grid_configure(padx=6, pady=6, ipadx=2, ipady=2)
93          for widget in self.entry_frame.winfo_children():
94              widget.grid_configure(padx=6, pady=6, ipadx=2, ipady=2)
95          for widget in self.weather_frame.winfo_children():
96              widget.grid_configure(padx=6, pady=6, ipadx=2, ipady=2)
97
98          # Set focus to the entry box for the next location
99          self.location_entry.focus_set()
100
101
102 # Create program object to start program
103 owm_gui = OWMGUI()
```

The following code creates a 2 column by 2 row grid layout. The rows and columns start a 0. **columnspan** spans however many columns are indicated.

sticky=E aligns the widget to the right side of the column. **sticky=W** aligns to the left.

```
self.lbl_app_title.grid(row=0, column=0, columnspan=2)
self.lbl_location.grid(row=1, column=0, sticky=E)
self.city_entry.grid( row=1, column=1, sticky=W)
```

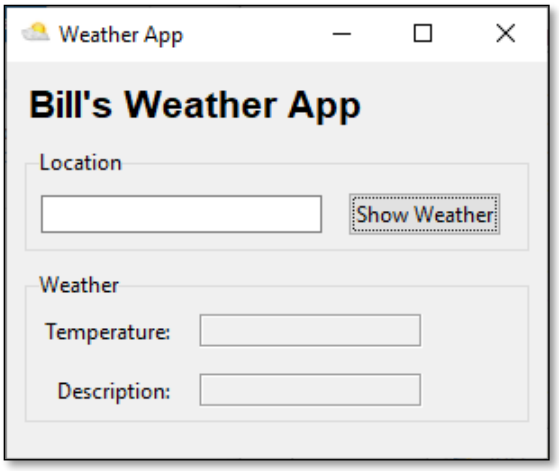
self.lbl_app_title	
self.lbl_city	self.city_entry

When designing a grid layout, it is helpful to draw it out on paper first.

The last piece is to create the program object at the bottom of the file. This starts the program execution.

This finishes the basic GUI for a weather program. Your project does not have to look like the example, you always have license to be creative.

Example run:



Time to get and display some weather.

Part 2: Getting Weather

Save **weather_gui_1.py** as **weather_gui_2.py** The screenshots will only show changes to the first program.

1. The following code gets the weather data.

```python
# ------------------------- GET WEATHER ---------------------------------#
    def get_weather(self, *args):
        try:
            # Get the location
            location = self.location_entry.get()

            # Build the openweathermap request parameters
            # These are added on to the URL to make the complete request
            query_string = {
                "units": "imperial",   # Units of measure ex: Fahrenheit
                "q": location,         # Location for weather
                "appid": weather_utils.API_KEY
            }

            # Get the API JSON data as a Python JSON object
            response = requests.get(
                weather_utils.URL,
                params=query_string
            )

            # Load json response into weather dictionary
            weather_data = response.json()

            # Get current fahrenheit temperature
            self.temperature = weather_data.get("main").get("temp")

            # Get detailed weather status
            self.description = weather_data.get(
                "weather")[0].get("description").title()

            # Call display weather method
            self.display_weather()
        except Exception as e:
            # raise
            print(f"[-] Sorry, there was a problem connecting. {e}")
```

2. The **display_weather()** method populates the display label's with weather data.

```
62  # ------------------------- DISPLAY WEATHER ------------------------------#
63      def display_weather(self):
64          # Set the weather information in the value labels
65          self.lbl_temperature_value.configure(
66              text=f" {self.temperature:.1f}°F")
67          self.lbl_description_value.configure(
68              text=f" {self.description}")
69
70          # Set focus to the entry box for the next location
71          self.location_entry.focus_set()
72          # Select text in entry box for next entry
73          self.location_entry.select_range(0, END)
```

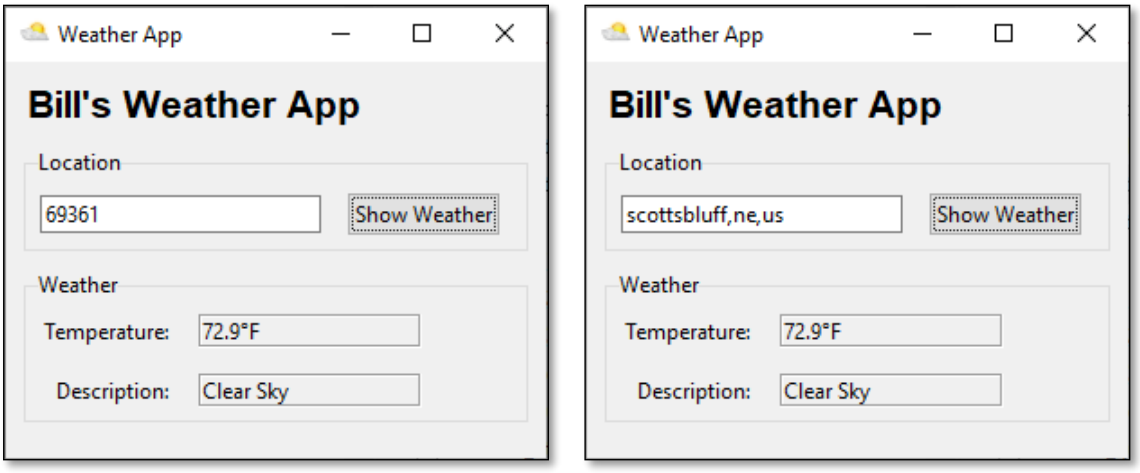
3. A couple more minor details. Add **command=self.get_weather** to the current button code. When we click the **Get Weather** button, we want to call the **get_weather()** method. This is in the **create_widgets()** method.

```
# Create button to get weather from location
self.btn_weather = Button(
    self.entry_frame,
    text="Show Weather",
    command=self.get_weather
)
```

4. Put this code at the end of the **create_widgets()** method. If we press the <Return> (Enter) key, the **get_weather()** method will be called.

```
# Either enter keys will activate the get_weather method
self.root.bind("<Return>", self.get_weather)
self.root.bind("<KP_Enter>", self.get_weather)
```

Example run:



Assignment 1: Add Weather Items

Time to finish our project using the OpenWeatherMap API documentation. The information we are going to use is in this location. This lists the current weather data we can retrieve.

Let's add a couple more weather items to our program. You can add others if you wish.

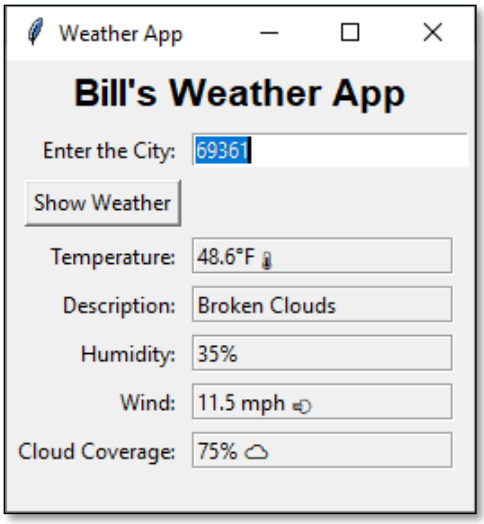
- Humidity
- Wind speed
- Cloud coverage
- Look at the owm.json file pictured below. It is a combination of lists and dictionaries. This is the code to get humidity out of the dictionary.

```
weather_data.get("main").get("humidity")
```

- **.get("main")** – This accesses a list with within a dictionary
- **.get("humidity")** – Gets the specific description element of a dictionary

```
{
    "coord": {
        "lon": -103.6672,
        "lat": 41.8666
    },
    "weather": [
        {
            "id": 803,
            "main": "Clouds",
            "description": "broken clouds",
            "icon": "04n"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 48.47,
        "feels_like": 45.39,
        "temp_min": 46.83,
        "temp_max": 49.32,
        "pressure": 1023,
        "humidity": 37
    },
    "visibility": 10000,
    "wind": {
        "speed": 6.91,
        "deg": 350
    },
    "clouds": {
        "all": 75
    },
    "dt": 1638056547,
    "sys": {
        "type": 1,
        "id": 3415,
        "country": "US",
        "sunrise": 1638021537,
        "sunset": 1638055577
    },
    "timezone": -25200,
    "id": 5699404,
    "name": "Scottsbluff",
    "cod": 200
}
```


Example run:



The little icons are from ASCII Art links in Python Resources. They are Unicode Emoji's.

How to Add Weather Items

Step 1: GET WEATHER - Get the weather items from the weather_data dictionary. You can copy and paste the temperature code and modify it.

```
# Get current fahrenheit temperature
self.temperature = weather_data.get("main").get("temp")
# Get detailed weather status
self.description = weather_data.get(
    "weather")[0].get("description").title()
```

Step 2: CREATE WIDGETS - Create description labels.

```
self.lbl_temperature = Label(
    self.weather_frame, text="Temperature:")
self.lbl_description = Label(
    self.weather_frame, text="Description:")
```

Step 3: CREATE WIDGETS - Create value display labels.

```
# Create value display labels
self.lbl_temperature_value = Label(
    self.weather_frame, width=20, anchor=W, relief=GROOVE)
self.lbl_description_value = Label(
    self.weather_frame, width=20, anchor=W, relief=GROOVE)
```

Step 4: CREATE WIDGETS - Grid the labels to place them on the screen. Increase the row value by one for each row of labels.

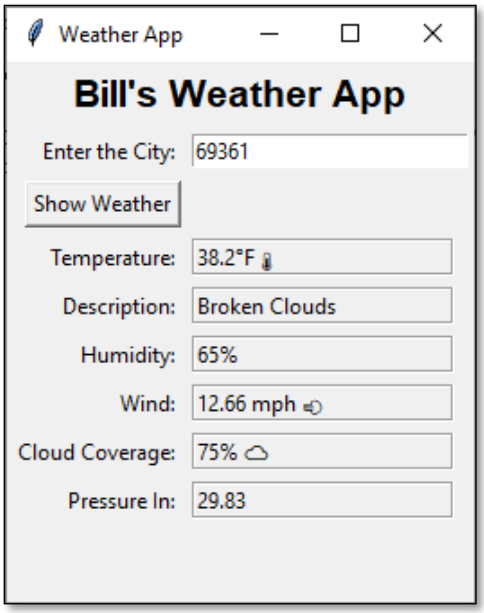
```
self.lbl_temperature.grid(row=2, column=0, sticky=E)
self.lbl_temperature_value.grid(row=2, column=1, sticky=W)

self.lbl_description.grid(row=3, column=0, sticky=E)
self.lbl_description_value.grid(row=3, column=1, sticky=W)
```

Step 5: DISPLAY WEATHER - Display the weather items on the display labels.

```
# Set the weather information in the value labels
self.lbl_temperature_value.configure(
    text=f" {self.temperature:.1f}°F")
self.lbl_description_value.configure(
    text=f" {self.description}")
```

Challenge

If you like a challenge, add more weather information to your program. Let's see what you can come up with!



Optional: Convert GMT Sunrise Sunset to Local Time

If you decide to get sunrise and or sunset, it comes from the API as a GMT Unix Time Stamp. You will have to convert it to your local computer time.

This is how to retrieve sunrise and sunset from the weather_data dictionary.

```
# Get weather items from dictionaries
self.description = self.weather_data.get(
    "weather")[0].get("description").title()
self.temperature = self.weather_data.get("main").get("temp")
self.humidity = self.weather_data.get("main").get("humidity")
self.sunrise = self.weather_data.get("sys").get("sunrise")
self.sunrise = weather_utils.convert_time(self.sunrise)
self.sunset = self.weather_data.get("sys").get("sunset")
self.sunset = weather_utils.convert_time(self.sunset)
```

The following is how to convert the GMT Unix Time stamp to your local computer's time.

1. Add the following function to your **weather_utils.py** module.
2. Import the Python datetime module at the top of this module.
   import datetime
3. Pass the time into the function, it will return a Python datetime object ready for display.

```
self.sunrise = weather_utils.convert_time(self.sunrise)
```

```
35  # ---------------------- CONVERT TIME ----------------------------#
36  def convert_time(time):
37      # Convert GMT Unix timestamp to local Python datetime
38      time = datetime.datetime.fromtimestamp(time)
39
40      # Format the date to hours, minutes, seconds, AM PM
41      time = f"{time:%I:%M:%S %p}"
42
43      # Strip out the leading/left from the hour
44      # 01 becomes 1
45      time = time.lstrip("0")
46
47      # Return GMT Unix as local Python time object
48      return time
```

Assignment Submission

- Attach the program files
- Attach a screenshot of your functioning program.
- Submit the assignment in Blackboard.