Python OpenWeatherMap API Tutorial CLI

Contents

Time required: 180 minutes

Please read all the directions carefully before beginning the assignment.

- Comment your code as show in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Tutorial Purpose

This tutorial will explore the powerful OpenWeatherMap API. This API provides access to using weather data in your programs.

This is a step-by-step guide to developing your own API programs. Start with the basics, hard code the request information, be successful, build from there.

Week 12: Part 1

Tutorial 1: Get an OpenWeatherMap Web API Key

Some API's, like OpenWeatherMap, require a key for authentication. This enables the API provider to identify and track the API usage.

1. Create an account at the OpenWeatherMap website. Choose a free account.
2. After you've signed up on OpenWeatherMap's website, you'll see this at the top of the page:

3. Click on **API keys** and you'll see your API key.
4. Create a folder for your weather program.
5. Create a Python module file called **weather_utils.py**

```python
"""
    Name: weather_utils.py
    Author:
    Created:
    Purpose: Store OpenWeatherMap API key and URL for
    easy import into other OpenWeatherMap programs
"""

# OpenWeatherMap API Key
API_KEY = "PUT YOUR API KEY HERE"

# URL to access current weather Openweathermap API
URL = "https://api.openweathermap.org/data/2.5/weather"
```

Here is a copy of the URL you can copy and paste.

```
URL = "https://api.openweathermap.org/data/2.5/weather"
```

This file allows you to import your OpenWeatherMap API key and URL into other weather projects. The URL is the OpenWeatherMap JSON endpoint.

Tutorial 2: Simple Weather Project

Time to start coding our weather project. The first version will be just enough to test that our API program code and key work and we can successfully get weather information.

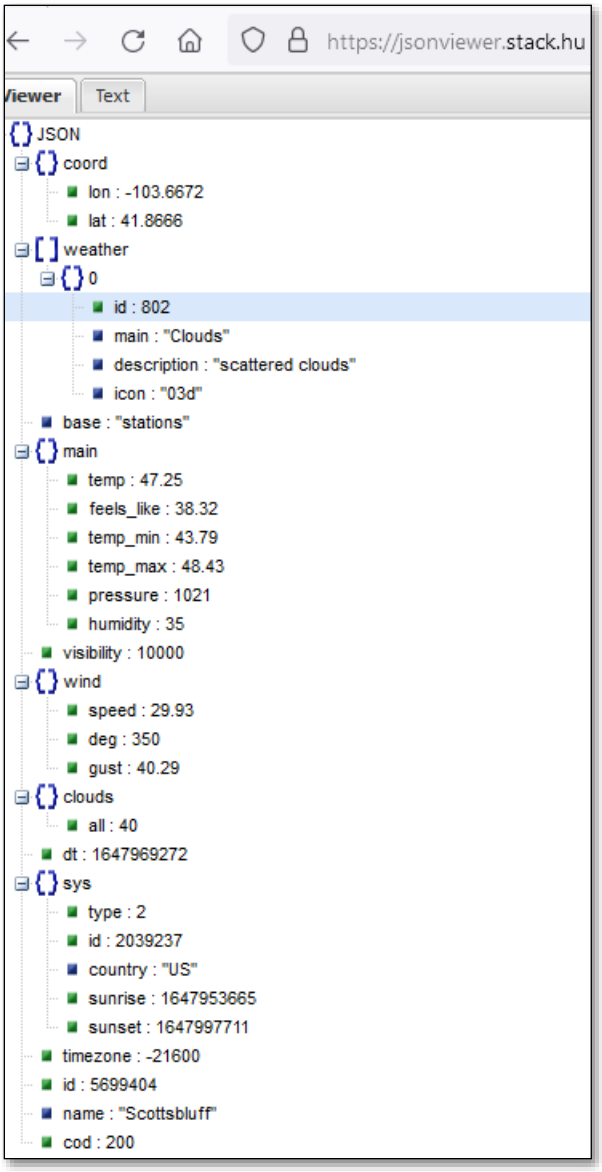
1. Import the requests module
    a. Open a command prompt
    b. **pip install requests**
2. Create a program file named: **weather_1.py**
3. Add the following code:

```python
"""
    Name: weather_1.py
    Author: William A Loring
    Created: 11/27/2021
    Purpose: Get weather dictionary from Openweathermap.org
"""
# pip install requests
import requests
# Import Openweather map api key and URL
import weather_utils

# Hard code location for testing
# Replace with your location for local weather
location = "Scottsbluff, NE, US"

# 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
)

# Print raw JSON data for testing
print(response.text)
```

Example run:

```
{"coord":{"lon":-103.6672,"lat":41.8666},"weather":[{"id":800,"main":"Clear","de
scription":"clear sky","icon":"01d"}],"base":"stations","main":{"temp":34.03,"fe
els_like":34.03,"temp_min":29.52,"temp_max":34.03,"pressure":1011,"humidity":69}
,"visibility":10000,"wind":{"speed":0,"deg":0},"clouds":{"all":1},"dt":163950268
4,"sys":{"type":1,"id":3415,"country":"US","sunrise":1639491293,"sunset":1639524
257},"timezone":-25200,"id":5699404,"name":"Scottsbluff","cod":200}
```

Copy and paste these results into https://jsonviewer.stack.hu



What is JSON (JavaScript Object Notation)?

JSON is a format for encoding data in human readable format for storing and sending over a network. Although it started in JavaScript, it is used in all modern programming languages.

Why is it Used?

JSON is used because it makes it easy to store and transfer arrays, lists, dictionaries, and objects as text data. JSON has become a standard for data transfer.

JSON Format Overview

JSON stores data as:

- key/value pairs
- Data is separated using commas
- Text data is enclosed in double quotes
- Numerical data has no quotes.
- Arrays or Lists are enclosed in square brackets []
- Objects or dictionaries are enclosed in curly brackets {}
- The output is a text string

Tutorial 3: Display Weather

Let's get our weather information into a more readable format.

Look at the **owm.json** file. It is a combination of lists and dictionaries. This is the code to get specific data out of the dictionary.

```
self.weather_data.get("weather")[0].get("description").title()
```

.get("**weather**"**)[0]** – This accesses a list with within a dictionary

.get("**description**"**)** – Gets the specific description element of a dictionary

.title() – Python string method to return a string with title case

The **owm.json** file.

```json
{
    "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
}
```

The following text file shows methods to access different JSON data elements.

```
# Main dictionary, weather_data.get()
# All data is a dictionary within a dictionary
# or a dictionary inside a list inside a dictionary
{

   # Dictionary, weather_data.get("coord").get("lon")
   "coord": {
        "lon": -103.6672,
        "lat": 41.8666
    },

   # Dictionary with list, weather_data.get("weather"[0]).get("description")
    "weather": [
        {
            "id": 803,
            "main": "Clouds",
            "description": "broken clouds",
            "icon": "04n"
        }
    ],
    "base": "stations",

    # Dictionary, weather_data.get("main").get("temp"
    "main": {
        "temp": 48.47,
        "feels_like": 45.39,
        "temp_min": 46.83,
        "temp_max": 49.32,
        "pressure": 1023,
        "humidity": 37
    },

    # Dictionary, weather_data.get("visibility")
    "visibility": 10000,
    "wind": {
        "speed": 6.91,
        "deg": 350
    },
```

Open **weather_1.py** Save it as **weather_2.py**

```python
"""
    Name: weather_2.py
    Author: William A Loring
    Created: 11/27/2021
    Purpose: OOP to Display Openweathermap weather from hard coded location
"""

# pip install requests
import requests
# Import Openweather map api key and URL
import weather_utils


class Weather:
    def __init__(self):
        # Hard code location for testing
        # Replace with your location for local weather
        self.location = "Scottsbluff, NE, US"

        # Build the openweathermap request parameters
        # These are added on to the URL to make the complete request
        params = {
            "units": "imperial",        # Units of measure ex: Fahrenheit
            "q": self.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=params
        )

        # Get response into a Python dictionary
        self.weather_data = response.json()
```

We use a separate method to get the weather data.

```python
    # ------------------------- GET WEATHER ------------------------------ #
    def get_weather(self):
        """Get weather data from Openweathermap."""
        # 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.feels_like = self.weather_data.get("main").get("feels_like")
        self.humidity = self.weather_data.get("main").get("humidity")
        self.clouds = self.weather_data.get("clouds").get("all")
```

Displaying the weather in it's own method. Notice how all variables have self. in the name. That allows us to use those variables anywhere in the class.

```python
    # ------------------------ DISPLAY WEATHER -------------------------- #
    def display_weather(self):
        """Display weather data."""
        print(f"   Location: {self.location}")
        print(f"Description: {self.description}")
        print(f"Temperature: {self.temperature}°F")
        print(f" Feels Like: {self.feels_like}°F")
        print(f"   Humidity: {self.humidity}%")
        print(f"     Clouds: {self.clouds}%")
```

At the end of the program we create a Weather() and run our new methods.

```python
# ------------------------- MAIN PROGRAM ------------------------------ #
"""The main program starts here. Create a Weather program object """
weather = Weather()
weather.get_weather()
weather.display_weather()
```

Example run:

```
   Location: Scottsbluff, NE, US
Description: Overcast Clouds
Temperature: 31.71°F
 Feels Like: 25.21°F
   Humidity: 84%
     Clouds: 100%
```

Assignment Submission

1. Attach the program files.
2. Attach screenshots showing the successful operation of the program.
3. Submit in Blackboard.

Week 13: Part 2

Tutorial 4: Add User Input

This version finds the location by user input of city, state code, and country code.

```
"""
Name: weather_3.py
Author: William A Loring
Created: 11/27/2021
Purpose: Display Openweathermap weather from user input
"""

# pip install requests
import requests
import weather_utils


class Weather:
    def __init__(self):
        # Skip the method for now
        pass

    # ------------------------- GET LOCATION ----------------------------- #
    def get_location(self):
        """Get location from user."""
        self.city = input("Enter city: ")
        self.state = input("Enter state: ")
        self.country = input("Enter country: ")

        # Build the weather query
        self.location = f"{self.city},{self.state},{self.country}"
```

```python
    # ------------------------- GET WEATHER ---------------------------------#
    def get_weather(self):
        """Get weather data from Openweathermap."""
        # 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": self.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
        )

        # Get json response into a Python dictionary
        self.weather_data = response.json()

        # 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")
```

```python
    # ------------------------- DISPLAY WEATHER -----------------------------#
    def display_weather(self):
        # Display current weather
        print(f"\n Current weather in {self.city.title()}")
        print(f" Description: {self.description}")
        print(f" Temperature: {self.temperature:.1f}°F")
        print(f" Humidity:    {self.humidity} %")


# ---------------------------- MAIN PROGRAM -------------------------------#
"""The main program starts here. Create a Weather program object """
weather = Weather()
weather.get_location()
weather.get_weather()
weather.display_weather()
```

Example run:

```
Enter city: Scottsbluff
Enter state: NE
Enter country: US

Current weather in Scottsbluff
Description: Overcast Clouds
Temperature: 31.71°F
 Feels Like: 25.21°F
   Humidity: 84%
     Clouds: 100%
```

Tutorial 5: Add Rich Text Formatting and Exception Handling

Get creative!

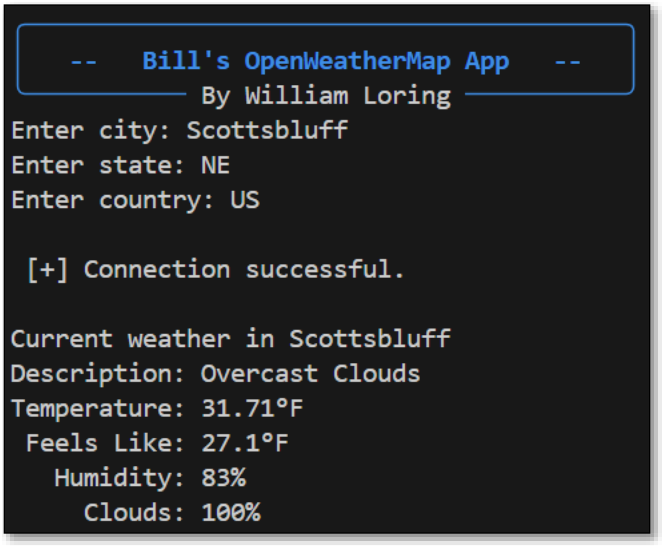
Exception handling is a very good idea when dealing with Web API's. A few flourishes with the Rich library have been added to the program. These are not necessary but are fun to do. They make your program look more finished. Get creative with Rich!

```
1   """
2       Name: weather_4.py
3       Author: William A Loring
4       Created: 11/27/2021
5       Purpose: Display Openweathermap weather from input location
6   """
7
8   # pip install requests
9   import requests
10  import weather_utils
11  # pip install rich
12  # Import Console for console printing
13  from rich.console import Console
14  # Import Panel for title displays
15  from rich.panel import Panel
16  # Initialize rich.console
17  console = Console()
```

```python
class Weather:
    def __init__(self):
        console.print(
            Panel.fit(
                "  --   Bill's OpenWeatherMap App   --  ",
                style="bold blue",
                subtitle="By William Loring",
            )
        )

    # ------------------------- GET LOCATION ----------------------------- #
    def get_location(self):
        """Get location from user."""
        self.city = input("Enter city: ")
        state = input("Enter state: ")
        country = input("Enter country: ")

        # Build the weather query
        self.location = f"{self.city},{state},{country}"
```

```python
    # ------------------------- GET WEATHER ------------------------------ #
    def get_weather(self):
        """Get weather data from Openweathermap."""
        try:
            # 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": self.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)

            # If the status_code is 200, successful connection and data
            if response.status_code == 200:

                # Get json response into a Python dictionary
                self.weather_data = response.json()

                # Let user know the connection was successful
                print("\n [+] Connection successful.")
            else:
                print(f" Response code: {response.status_code}")
                print(" You may have typed an invalid location.")
                print(" Please try again.")
                self.get_location()

            # 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.feels_like = self.weather_data.get("main").get("feels_like")
            self.humidity = self.weather_data.get("main").get("humidity")
            self.clouds = self.weather_data.get("clouds").get("all")

        except:
            # Handle any exceptions
            print("[-] Sorry, there was a problem connecting.")
```

Example run:



Assignment Submission

1. Attach the program files.
2. Attach screenshots showing the successful operation of the program.
3. Submit in Blackboard.

Week 14: Part 3

Tutorial 6: 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.feels_like = self.weather_data.get("main").get("feels_like")
self.humidity = self.weather_data.get("main").get("humidity")
self.clouds = self.weather_data.get("clouds").get("all")

# Get sunrise and sunset
self.sunrise = self.weather_data.get("sys").get("sunrise")
self.sunset = self.weather_data.get("sys").get("sunset")

# Convert GMT to local computer time
self.sunrise = weather_utils.convert_time(self.sunrise)
self.sunset = weather_utils.convert_time(self.sunset)
```

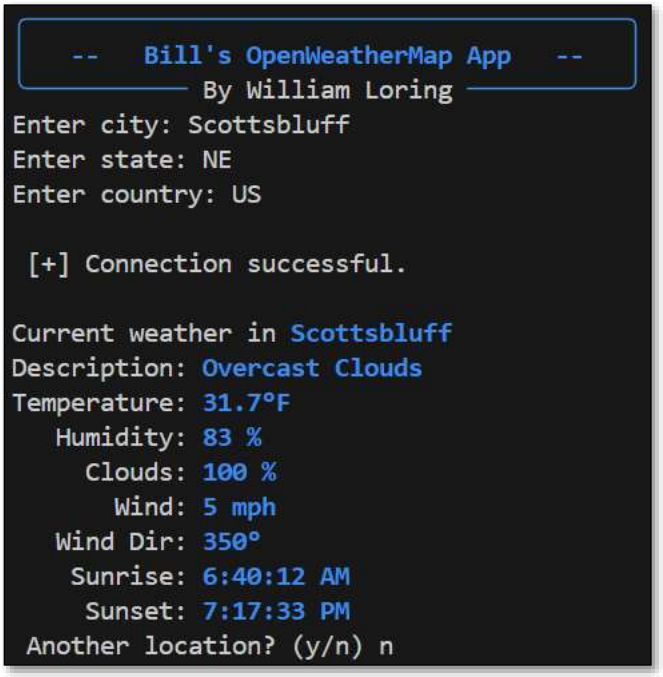
The following is how to convert the GMT Unix Time stamp to your local computer's time.

1. Add the following function to your **weather_utils.py** module.
2. Import the Python datetime module at the top of this module.
   from datetime import datetime
3. When you pass the Unix 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.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
```

Example run (with a bit more Rich formatting):



Assignment 1: Expand the Program

- Add a menu loop that allows the user to choose to get weather from another location or end the program.
- Create the Weather object once, right before the menu loop.
- This is the location where you would put the menu loop.

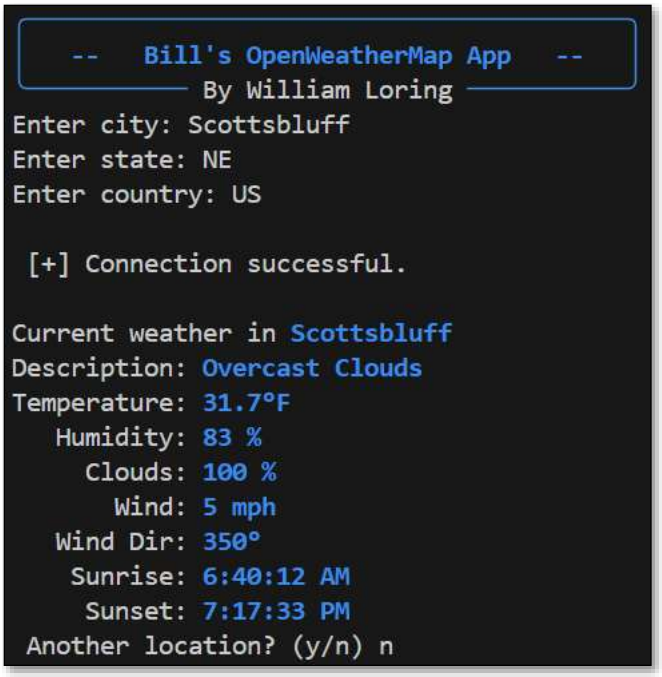
```
# ---------------------------- MAIN PROGRAM -------------------------------#
"""The main program starts here. Create a Weather program object."""
weather = Weather()
weather.get_location()
weather.get_weather()
weather.display_weather()
```

- Add three or more weather dictionary items to the program. Look at the earlier **Tutorial 3 Display Weather** to see what dictionary items you might get.

Challenges

- Use the emojis library to add some character.

Example run:



Assignment Submission

1. Attach the program files.
2. Attach screenshots showing the successful operation of the program.
3. Submit in Blackboard.