

Python OpenWeatherMap API Tutorial CLI

Contents

Python OpenWeatherMap API Tutorial CLI	1
Tutorial Purpose.....	2
Week 13: Part 1.....	2
Tutorial 1: Get an OpenWeatherMap Web API Key	2
Tutorial 2: Simple Weather Project	3
What is JSON (JavaScript Object Notation)?	5
Why is it Used?	5
JSON Format Overview	6
Tutorial 3: Display Weather	6
Assignment Submission.....	11
Week 14: Part 2.....	11
Tutorial 4: Add User Input	11
Tutorial 5: Add Rich Text Formatting and Exception Handling.....	13
Assignment Submission.....	16
Week 15: Part 3.....	16
Tutorial 6: Convert GMT Sunrise Sunset to Local Time	16
Assignment 1: Expand the Program	18
Challenges.....	18
Assignment Submission.....	19

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 13: 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**

```
1 """
2     Name: weather_utils.py
3     Author:
4     Created:
5     Purpose: Store OpenWeatherMap API key and URL for
6             easy import into other OpenWeatherMap programs
7 """
8
9 # OpenWeatherMap API Key
10 API_KEY = "PUT YOUR API KEY HERE"
11
12 # URL to access current weather Openweathermap API
13 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:

```

1 """
2     Name: weather_1.py
3     Author: William A Loring
4     Created: 11/27/2021
5     Purpose: Get weather dictionary from Openweathermap.org
6 """
7 # pip install requests
8 import requests
9 # Import Openweather map api key and URL
10 import weather_utils
11
12 # Hard code location for testing
13 # Replace with your location for local weather
14 location = "Scottsbluff, NE, US"
15
16 # Build the openweathermap request parameters
17 # These are added on to the URL to make the complete request
18 query_string = {
19     "units": "imperial",           # Units of measure ex: Fahrenheit
20     "q": location,                # Location for weather
21     "appid": weather_utils.API_KEY
22 }
23
24 # Get the API JSON data as a Python JSON object
25 response = requests.get(
26     weather_utils.URL,
27     params=query_string
28 )
29
30 # Print raw JSON data for testing
31 print(response.text)

```

Example run:

```
{"coord":{"lon":-103.6672,"lat":41.8666}, "weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01d"}], "base":"stations", "main":{"temp":34.03,"feels_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":1639502684, "sys":{"type":1,"id":3415,"country":"US","sunrise":1639491293,"sunset":1639524257}, "timezone":-25200, "id":5699404, "name":"Scottsbluff", "cod":200}
```

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

The screenshot shows a JSON viewer interface with the URL <https://jsonviewer.stack.hu>. The JSON structure is as follows:

```
{ "coord": { "lon": -103.6672, "lat": 41.8666 }, "weather": [ { "id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03d" }, {"base": "stations"} ], "main": { "temp": 47.25, "feels_like": 38.32, "temp_min": 43.79, "temp_max": 48.43, "pressure": 1021, "humidity": 35, "visibility": 10000 }, "wind": { "speed": 29.93, "deg": 350, "gust": 40.29 }, "clouds": { "all": 40 }, "dt": 1647969272, "sys": { "type": 2, "id": 2039237, "country": "US", "sunrise": 1647953665, "sunset": 1647997711, "timezone": -21600, "id": 5699404, "name": "Scottsbluff", "cod": 200 } }
```

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 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.

```
1  {
2      "coord": {
3          "lon": -103.6672,
4          "lat": 41.8666
5      },
6      "weather": [
7          {
8              "id": 803,
9              "main": "Clouds",
10             "description": "broken clouds",
11             "icon": "04n"
12         }
13     ],
14     "base": "stations",
15     "main": {
16         "temp": 48.47,
17         "feels_like": 45.39,
18         "temp_min": 46.83,
19         "temp_max": 49.32,
20         "pressure": 1023,
21         "humidity": 37
22     },
23     "visibility": 10000,
24     "wind": {
25         "speed": 6.91,
26         "deg": 350
27     },
28     "clouds": {
29         "all": 75
30     },
31     "dt": 1638056547,
32     "sys": {
33         "type": 1,
34         "id": 3415,
35         "country": "US",
36         "sunrise": 1638021537,
37         "sunset": 1638055577
38     },
39     "timezone": -25200,
40     "id": 5699404,
41     "name": "Scottsbluff",
42     "cod": 200
43 }
```

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
    },
    "clouds": {
        "all": 75
    }
}
```

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

```
1  """
2      Name: weather_2.py
3      Author: William A Loring
4      Created: 11/27/2021
5      Purpose: OOP to Display Openweathermap weather from hard coded location
6  """
7
8  # pip install requests
9  import requests
10 # Import Openweather map api key and URL
11 import weather_utils
12
13
14 class Weather:
15     def __init__(self):
16         # Hard code location for testing
17         # Replace with your location for local weather
18         self.location = "Scottsbluff, NE, US"
19
20         # Build the openweathermap request parameters
21         # These are added on to the URL to make the complete request
22         params = {
23             "units": "imperial",          # Units of measure ex: Fahrenheit
24             "q": self.location,          # Location for weather
25             "appid": weather_utils.API_KEY
26         }
27
28         # Get the API JSON data as a Python JSON object
29         response = requests.get(
30             weather_utils.URL,
31             params=params
32         )
33
34         # Get response into a Python dictionary
35         self.weather_data = response.json()
```

We use a separate method to get the weather data.

```
35     # ----- GET WEATHER ----- #
36     def get_weather(self):
37         """Get weather data from Openweathermap."""
38         # Get weather items from dictionaries
39         self.description = (
40             self.weather_data.get("weather")[0].get("description").title()
41         )
42
43         self.temperature = self.weather_data.get("main").get("temp")
44         self.feels_like = self.weather_data.get("main").get("feels_like")
45         self.humidity = self.weather_data.get("main").get("humidity")
46         self.clouds = self.weather_data.get("clouds").get("all")
47
```

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.

```
48     # ----- DISPLAY WEATHER ----- #
49     def display_weather(self):
50         """Display weather data."""
51         print(f"    Location: {self.location}")
52         print(f"Description: {self.description}")
53         print(f"Temperature: {self.temperature}°F")
54         print(f"    Feels Like: {self.feels_like}°F")
55         print(f"    Humidity: {self.humidity}%")
56         print(f"    Clouds: {self.clouds}%")
```

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

```
59     # ----- MAIN PROGRAM ----- #
60     """The main program starts here. Create a Weather program object """
61     weather = Weather()
62     weather.get_weather()
63     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 14: Part 2

Tutorial 4: Add User Input

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

```
1 """
2 Name: weather_3.py
3 Author: William A Loring
4 Created: 11/27/2021
5 Purpose: Display Openweathermap weather from user input
6 """
7
8 # pip install requests
9 import requests
10 import weather_utils
11
12
13 class Weather:
14     def __init__(self):
15         # Skip the method for now
16         pass
17
18     # ----- GET LOCATION ----- #
19     def get_location(self):
20         """Get location from user."""
21         self.city = input("Enter city: ")
22         self.state = input("Enter state: ")
23         self.country = input("Enter country: ")
24
25         # Build the weather query
26         self.location = f"{self.city},{self.state},{self.country}"
```

```
27 # ----- GET WEATHER -----#
28     def get_weather(self):
29         """Get weather data from Openweathermap."""
30         # Build the openweathermap request parameters
31         # These are added on to the URL to make the complete request
32         query_string = {
33             "units": "imperial",          # Units of measure ex: Fahrenheit
34             "q": self.location,          # Location for weather
35             "appid": weather_utils.API_KEY
36         }
37
38         # Get the API JSON data as a Python JSON object
39         response = requests.get(
40             weather_utils.URL,
41             params=query_string
42         )
43
44         # Get json response into a Python dictionary
45         self.weather_data = response.json()
46
47         # Get weather items from dictionaries
48         self.description = self.weather_data.get(
49             "weather")[0].get("description").title()
50         self.temperature = self.weather_data.get("main").get("temp")
51         self.humidity = self.weather_data.get("main").get("humidity")
```

```
53 # ----- DISPLAY WEATHER -----#
54     def display_weather(self):
55         # Display current weather
56         print(f"\n Current weather in {self.city.title()}")
57         print(f" Description: {self.description}")
58         print(f" Temperature: {self.temperature:.1f}°F")
59         print(f" Humidity:    {self.humidity} %")
60
61
62 # ----- MAIN PROGRAM -----#
63 """The main program starts here. Create a Weather program object """
64 weather = Weather()
65 weather.get_location()
66 weather.get_weather()
67 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()
```

```
23 class Weather:
24     def __init__(self):
25         console.print(
26             Panel.fit(
27                 " -- Bill's OpenWeatherMap App -- ",
28                 style="bold blue",
29                 subtitle="By William Loring",
30             )
31         )
32
33 # ----- GET LOCATION ----- #
34 def get_location(self):
35     """Get location from user."""
36     self.city = input("Enter city: ")
37     state = input("Enter state: ")
38     country = input("Enter country: ")
39
40     # Build the weather query
41     self.location = f"{self.city},{state},{country}"
```

```

43     # ----- GET WEATHER ----- #
44     def get_weather(self):
45         """Get weather data from Openweathermap."""
46         try:
47             # Build the openweathermap request parameters
48             # These are added on to the URL to make the complete request
49             query_string = {
50                 "units": "imperial", # Units of measure ex: Fahrenheit
51                 "q": self.location, # Location for weather
52                 "appid": weather_utils.API_KEY,
53             }
54
55             # Get the API JSON data as a Python JSON object
56             response = requests.get(weather_utils.URL, params=query_string)
57
58             # If the status_code is 200, successful connection and data
59             if response.status_code == 200:
60
61                 # Get json response into a Python dictionary
62                 self.weather_data = response.json()
63
64                 # Let user know the connection was successful
65                 print("\n [+] Connection successful.")
66             else:
67                 print(f" Response code: {response.status_code}")
68                 print(" You may have typed an invalid location.")
69                 print(" Please try again.")
70                 self.get_location()
71
72             # Get weather items from dictionaries
73             self.description = (
74                 self.weather_data.get("weather")[0].get("description").title()
75             )
76             self.temperature = self.weather_data.get("main").get("temp")
77             self.feels_like = self.weather_data.get("main").get("feels_like")
78             self.humidity = self.weather_data.get("main").get("humidity")
79             self.clouds = self.weather_data.get("clouds").get("all")
80
81         except:
82             # Handle any exceptions
83             print("[-] Sorry, there was a problem connecting.")

```

Example run:

```
-- Bill's OpenWeatherMap App --
By William Loring
Enter city: Scottsbluff
Enter state: NE
Enter country: US

[+] Connection successful.

Current weather in Scottsbluff
Description: Overcast Clouds
Temperature: 31.71°F
Feels Like: 27.1°F
Humidity: 83%
Clouds: 100%
```

Assignment Submission

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

Week 15: 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
49

```

Example run (with a bit more Rich formatting):

```
-- Bill's OpenWeatherMap App --
By William Loring

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

[+] Connection successful.

Current weather in Scottsbluff
Description: Overcast Clouds
Temperature: 31.7°F
Humidity: 83 %
Clouds: 100 %
Wind: 5 mph
Wind Dir: 350°
Sunrise: 6:40:12 AM
Sunset: 7:17:33 PM
Another location? (y/n) n
```

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:

```
-- Bill's OpenWeatherMap App --
By William Loring
Enter city: Scottsbluff
Enter state: NE
Enter country: US

[+] Connection successful.

Current weather in Scottsbluff
Description: Overcast Clouds
Temperature: 31.7°F
    Humidity: 83 %
    Clouds: 100 %
    Wind: 5 mph
    Wind Dir: 350°
    Sunrise: 6:40:12 AM
    Sunset: 7:17:33 PM
Another location? (y/n) n
```

Assignment Submission

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