Python OpenWeatherMap API Tutorial CLI

Contents

Time required: 120 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.

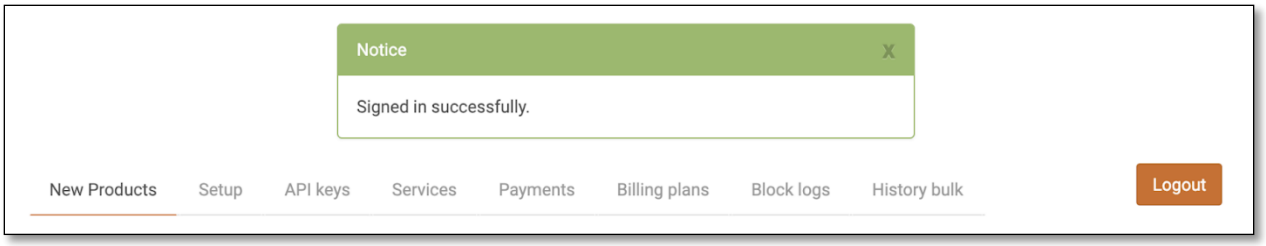
This is a step-by-step guide to developing your own API programs. Start with the basics, hard code the request information, be successful, build from there.

Week 12

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**

```
"""
    Name: weather_utils.py
    Author:
    Created:
    Purpose: Store OpenWeatherMap API key adn 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}
```

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.

```
{
    "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: 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
        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 response into a Python dictionary
        self.weather_data = response.json()

#------------------------- 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.humidity = self.weather_data.get("main").get("humidity")

        # Display current weather
        print(f"{self.location}")
        print(f"Description: {self.description}")
        print(f"Temperature: {self.temperature}")
        print(f"Humidity:    {self.humidity}")


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

Example run:

```
Scottsbluff, NE, US
Description: Broken Clouds
Temperature: 37.85
Humidity:    75
```

Week 13

Tutorial 4: Add User Input

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

```python
"""
    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):
        pass

#------------------------- 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}"

#------------------------- 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")
```

```
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: Fog
Temperature: 22.2°F
Humidity:    86 %
```

Tutorial 5: Add Exception Handling

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

```
"""
    Name: weather_4.py
    Author: William A Loring
    Created: 11/27/2021
    Purpose: Display Openweathermap weather from input location
"""

# pip install requests
import requests
import weather_utils
# pip install rich
# Import Console for console printing
from rich.console import Console
# Import Panel for title displays
from rich.panel import Panel
# Initialize rich.console
console = Console()


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"The response status code was: {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.humidity = self.weather_data.get("main").get("humidity")
        except:
            print("[-] Sorry, there was a problem connecting.")
            # raise is for troubleshooting a program by
            # raising the exception as if the try except block didn't exist
            # raise
```

```
82 #------------------------- DISPLAY WEATHER ----------------------------------#
83     def display_weather(self):
84         """Display current weather at console."""
85         console.print(
86             f"\n Current weather in [bold blue]{self.city.title()}[/bold blue]")
87         console.print(
88             f" Description: [bold blue]{self.description}[/bold blue]")
89         console.print(
90             f" Temperature: [bold blue]{self.temperature:.1f}°F[/bold blue]")
91         console.print(f" Humidity:    [bold blue]{self.humidity} %[/bold blue]")
92 
93 
94 #---------------------------- MAIN PROGRAM ------------------------------------#
95 """The main program starts here. Create a Weather program object."""
96 weather = Weather()
97 weather.get_location()
98 weather.get_weather()
99 weather.display_weather()
```

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:   Clear Sky
Temperature:   69.5°F
Humidity:      28 %
Clouds:        0 %
Wind:          5 mph
Sunrise:       7:01:15 AM
Sunset:        6:21:43 PM
Another location? (y/n)
```

Week 14

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.

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.

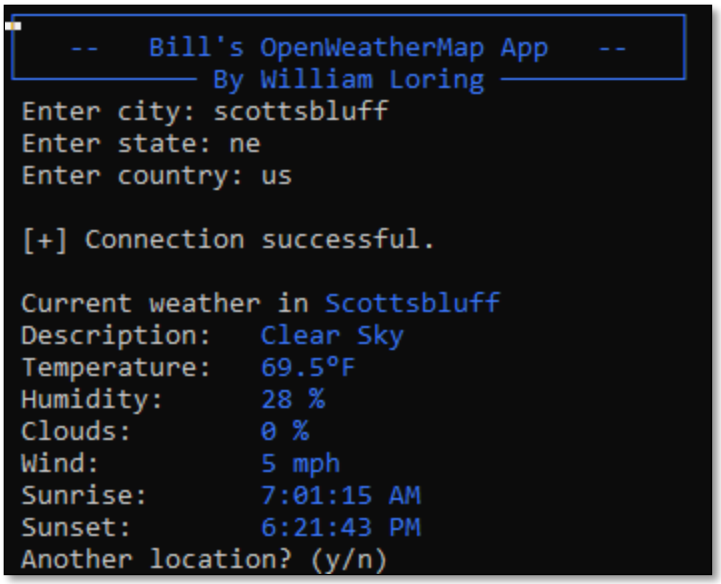
```
self.sunrise = weather_utils.convert_time(self.sunrise)
```

```
47 #---------------------- CONVERT TIME ----------------------------#
48 def convert_time(time):
49     # Convert GMT Unix timestamp to local Python datetime
50     time = datetime.datetime.fromtimestamp(time)
51
52     # Format the date to hours, minutes, seconds, AM PM
53     time = f"{time:%I:%M:%S %p}"
54
55     # Strip out the leading/left from the hour
56     # 0: 01 becomes 1
57     time = time.lstrip("0")
58
59     # Return GMT Unix as local Python time object
60     return time
```

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. This would be around the main program method calls at the end of the program.
- Create the Weather object once, right before the menu loop
- Add three or more weather items to the program.

Example run:



Assignment Submission

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