# 6.2.5 Parse a Response from an API

**Congratulations** on passing the test! You can pat yourself on the back for passing your own test with flying colors. Now you can confidently start mining the JSON file to retrieve specific weather data for each city and add it to a DataFrame.

For each city in our `lats_lngs` list, we need to retrieve the following data and add it to a DataFrame:

- City, country, and date
- Latitude and longitude
- Maximum temperature
- Humidity
- Cloudiness
- Wind speed

Our final DataFrame should look like the following.

| | City | Country | Date | Lat | Lng | Max Temp | Humidity | Cloudiness | Wind Speed |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Tuktoyaktuk | CA | 2019-08-08 22:26:20 | 69.44 | -133.03 | 42.80 | 87 | 75 | 5.82 |
| **1** | Hermanus | ZA | 2019-08-08 22:26:21 | -34.42 | 19.24 | 54.00 | 73 | 0 | 1.01 |
| **2** | Bluff | AU | 2019-08-08 22:26:21 | -23.58 | 149.07 | 56.08 | 54 | 1 | 4.94 |
| **3** | Port Alfred | ZA | 2019-08-08 22:26:21 | -33.59 | 26.89 | 54.00 | 77 | 0 | 4.00 |
| **4** | Outjo | NA | 2019-08-08 22:26:21 | -20.11 | 16.16 | 62.60 | 12 | 26 | 1.12 |
| **5** | Agadez | NE | 2019-08-08 22:26:21 | 16.97 | 7.99 | 98.24 | 25 | 0 | 4.79 |
| **6** | Paso De Los Toros | UY | 2019-08-08 22:26:22 | -32.81 | -56.52 | 48.20 | 93 | 90 | 19.46 |
| **7** | Avarua | CK | 2019-08-08 22:26:22 | -21.21 | -159.78 | 77.00 | 78 | 90 | 6.93 |
| **8** | Taltal | CL | 2019-08-08 22:26:22 | -25.41 | -70.49 | 54.14 | 67 | 33 | 5.32 |
| **9** | Airai | TL | 2019-08-08 22:26:23 | -8.93 | 125.41 | 67.40 | 82 | 12 | 4.72 |

Before we collect weather data from more than 500 cities, we'll walk through how to get the weather data from Boston. First, correct the spelling for the city of `Boston` to get a valid URL. Then, in a new cell, add the following code and run the cell.

```
# Create an endpoint URL for a city.
city_url = url + "&q=" + "Boston"
city_weather = requests.get(city_url)
city_weather.json()
```

After running the cell, the output will be the JSON-formatted data from the city of Boston.

```
{'coord': {'lon': -71.06, 'lat': 42.36},
 'weather': [{'id': 801,
   'main': 'Clouds',
   'description': 'few clouds',
   'icon': '02d'}],
 'base': 'stations',
 'main': {'temp': 61.7,
  'pressure': 1020,
  'humidity': 59,
  'temp_min': 57,
  'temp_max': 66.2},
 'visibility': 16093,
 'wind': {'speed': 14.99, 'deg': 80},
 'rain': {},
 'clouds': {'all': 20},
 'dt': 1571678675,
 'sys': {'type': 1,
  'id': 3486,
  'country': 'US',
  'sunrise': 1571655818,
  'sunset': 1571694830},
 'timezone': -14400,
 'id': 4930956,
 'name': 'Boston',
 'cod': 200}
```

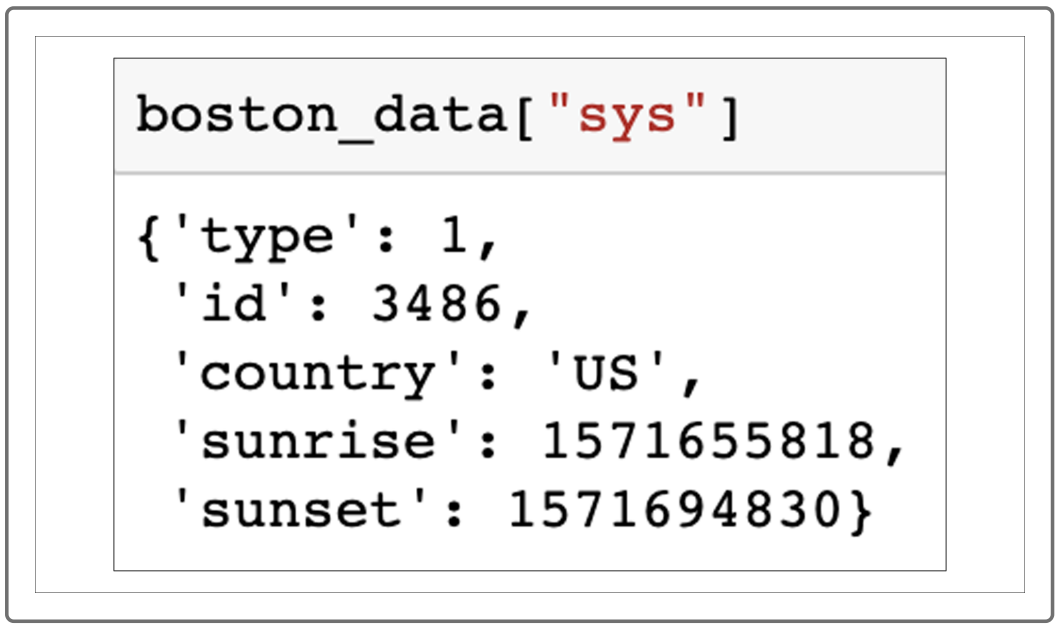
First, let's get something simple, like the country code from the JSON formatted data, which is in a nested dictionary where the first dictionary starts with `sys`.

```
'clouds': {'all': 20},
'dt': 1571678675,
'sys': {'type': 1,
 'id': 3486,
 'country': 'US',
 'sunrise': 1571655818,
 'sunset': 1571694830},
'timezone': -14400,
'id': 4930956,
'name': 'Boston',
```

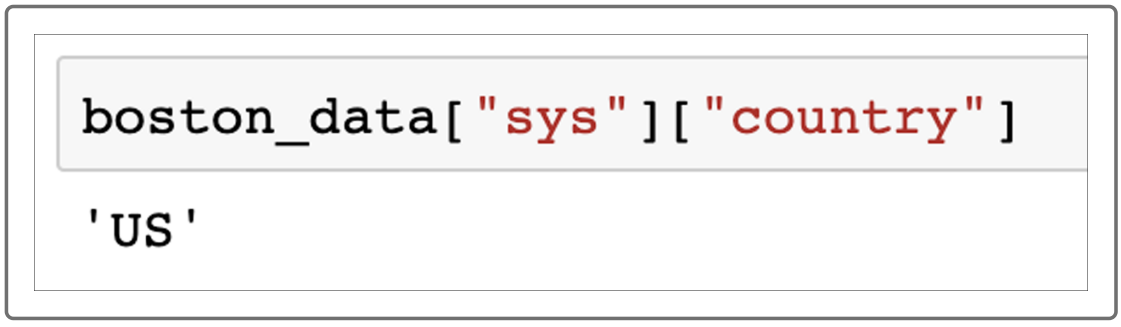
1. In a new cell, let's assign a variable to the `city_weather.json()` data to the variable "boston_data" and run the cell.

```
# Get the JSON data.
boston_data = city_weather.json()
```

2. Next, using the `sys` key to get the corresponding value, we type `boston_data['sys']` in a new cell and run the cell. The output is another dictionary as shown in the following image.



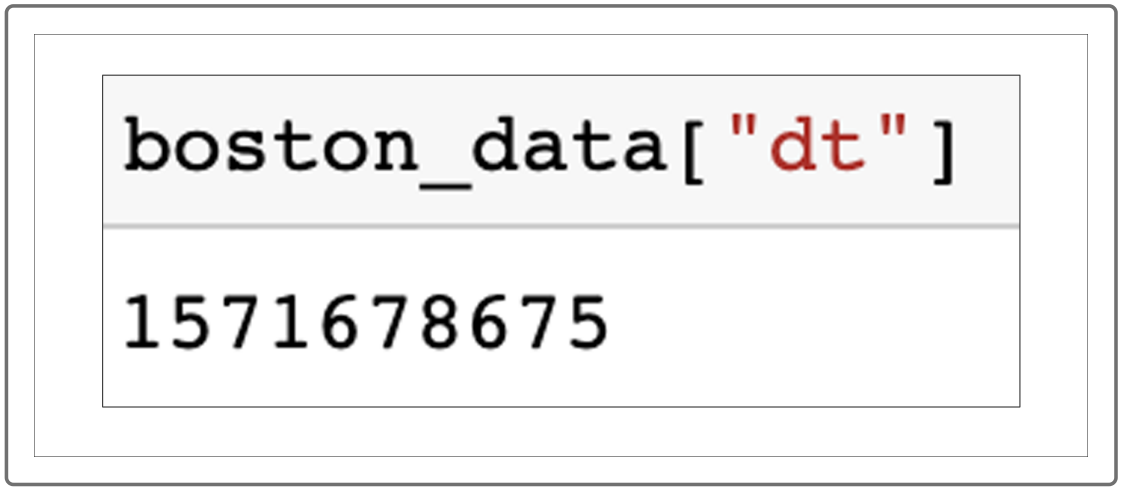

3. If we add the `country` key in brackets after the `sys` key, and run the cell again, `'US'` will be returned in the output.



**NOTE**

> When we used `boston_data["sys"]`, there was a key for `sunrise` and a key for `sunset` in the output. The value for these keys is the time of day in seconds in a database timestamp format.

If we want to retrieve the date in the weather data, we would add the `dt` key to the `boston_data` variable like this: `boston_data["dt"]`.




How would you get the latitude value from the Boston JSON data?

- `boston_data["lat"]`
- ✔ `boston_data["coord"]["lat"]`
- `boston_data["latitude"]`
- `boston_data["sys"]["lat"]`

Retake

Using similar syntax to get the time of day, we can get the latitude, longitude, maximum temperature, humidity, percent cloudiness, and wind speed. Add the following code to a new cell and run the cell.

```
lat = boston_data["coord"]["lat"]
lng = boston_data["coord"]["lon"]
max_temp = boston_data["main"]["temp_max"]
humidity = boston_data["main"]["humidity"]
clouds = boston_data["clouds"]["all"]
wind = boston_data["wind"]["speed"]
print(lat, lng, max_temp, humidity, clouds, wind)
```

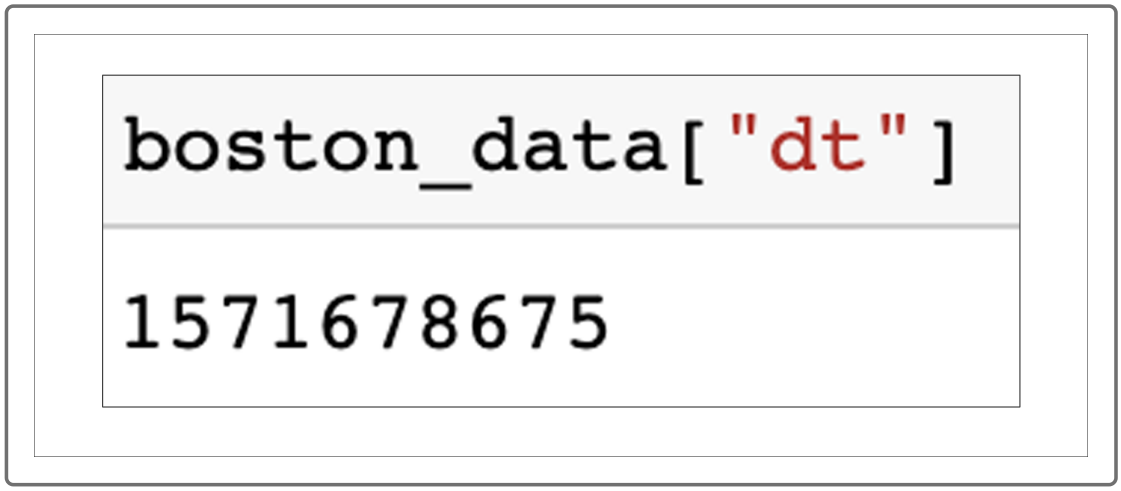
The output will be all of the weather parameters, with the units for maximum temperature in degrees Fahrenheit, the humidity and clouds as a percentage, and the wind in miles per hour.

```
print(lat, lng, max_temp, humidity, clouds, wind)
```

```
42.36 -71.06 66.2 59 20 14.99
```

## Convert the Date Timestamp

The date format will appear in seconds, as we saw when we ran this code.




This format is called Coordinated Universal Time (UTC) or Greenwich Mean Time (GMT). If we want to convert the timestamp to the International Organization for Standardization (ISO) format, or YYYY-MM-DD-HH-MM-SS, we need to use the Python datetime module.

Let's convert the date from the Boston weather data in the JSON format to the ISO format.

Add the following code to a new cell in the `API_practice` file and run the cell.

```
# Import the datetime module from the datetime library.
from datetime import datetime
# Get the date from the JSON file.
date = boston_data["dt"]
# Convert the UTC date to a date format with year, month, day, hours, minute
datetime.utcfromtimestamp(date)
```

When we run this code, the output is now in the ISO format with the, year, month, date, hour, minute, and seconds offset by commas.

```
datetime.datetime(2019, 10, 21, 17, 24, 35)
```

We can convert this datetime format to 2019-10-21 17:24:35 using the Python string format method `strftime()` and adding how we want the string to look inside the parentheses. In our case, we would use `strftime('%Y-%m-%d %H:%M:%S')`.

Add `.strftime('%Y-%m-%d %H:%M:%S')` to the end of the conversion: `datetime.utcfromtimestamp(date).strftime('%Y-%m-%d %H:%M:%S')`. Rerun the cell. The output should look like the following.

```
datetime.utcfromtimestamp(date).strftime('%Y-%m-%d %H:%M:%S')
'2019-10-21 17:24:35'
```

Now that we know how to get all the weather data from a JSON response, we can iterate through our cities list and retrieve the data from each city.

NOTE

> For more information about the datetime library and `strftime()`, see the documentation:
>
> - **datetime** (https://docs.python.org/3.7/library/datetime.html)
> - **strftime()** (https://docs.python.org/3.7/library/datetime.html#strftime-and-strptime-behavior)