

This is your **last** free member-only story this month. [Sign up for Medium and get an extra one](#)



Ritvik Kharkar
Jun 10, 2019 · 4 min read · Listen



Getting Weather Data in 3 Easy Steps



Photo by [Max LaRoche](#) on [Unsplash](#)

Continuing our data-gathering journey, this time we'll be looking at how to gather **weather data**. The applications of weather data are pretty endless from standalone weather prediction applications to projects where weather analysis is just one component of a larger objective.

I personally found the need to access weather data during a project where my team was helping the City of Cincinnati in Ohio to improve their Emergency Medical Response system. My team needed to use daily weather data as a predictor in a model to predict level of emergency response throughout the city.

We'll be getting our weather data from the most reliable source, **NOAA** (National Oceanic and Atmospheric Administration).

Since I now live in beautiful **Long Beach, California**, let's guide our exploration by trying to answer the following question:

How has average daily temperature in Long Beach changed since 2015?

Let's get started!

1. Request an Access Token from NOAA

In order to request data from NOAA, we will need to request an **access token**, which basically affirms that we are a trusted source asking for data. Navigate to the [Request a Token website](#).

Request Web Services Token

To gain access to [NCDC CDO Web Services](#), register with your email address. An email will be sent with a unique token which will allow access RESTful services. For more information about CDO Web Services [read the documentation for CDO Web Services guide](#).

Please enter your email address

SUBMIT

Request an Access Token from NOAA

Fill in an email address and you'll shortly get an email with your access token. Store this somewhere safe and keep it **private**. Otherwise, others could make requests to NOAA on your behalf and you don't necessarily want that.

2. Figure Out Which Weather Station to Access

The next thing you'll probably want to do is figure out which weather station to gather data from. The easiest way to find this station would be to navigate to the [Find a Station website](#)

Enter Location
Long Beach, CA, USA

Select Dataset
Daily Summaries

Select Date Range
2015-01-01 to 2019-06-01

Data Categories
☒ Air Temperature
☐ Precipitation
☐ Land
☐ Sky cover & clouds

Fill out the fields based on your **preferred location**, **dataset preference** (pick "Daily Summaries" for daily data), **time range** (the longer range you pick the fewer stations will have the entirety of that data), and **data categories** (we care just about air temperature for the task at hand).

The map on the right of the page will auto-update with available stations.

STATION DETAILS

Name: LONG BEACH DAUGHERTY AIRPORT, CA US

ID: GHNCNDUSW00023129

Lat/Lon: 33.8116, -118.1463

PERIOD OF RECORD

Start/End: 1943-01-01 to 2019-06-03

Coverage: 91%

FULL DETAILS ADD TO CART

We'll pick the Long Beach Airport station for this application. Make sure to note the station ID. For us, it's **GHNCND:USW00023129**.

3. Grab the Data!

Now that we have our access token and we know the ID of the station we wish to access, all that is left to do is request the data from NOAA. We'll be using Python's **requests** library for this task.

First let's import the needed libraries in Python.

```
1 #needed to make web requests
2 import requests
3
4 #store the data we get as a dataframe
5 import pandas as pd
6
7 #convert the response as a structured json
8 import json
9
10 #mathematical operations on lists
11 import numpy as np
12
13 #parse the datetimes we get from NOAA
14 from datetime import datetime
15
16 #add the access token you got from NOAA
17 token = 'YOUR_ACCESS_TOKEN'
18
19 #Long Beach Airport station
20 station_id = 'GHNCND:USW00023129'
```

In order to request data from NOAA we need to use NOAA's API (Application Programming Interface) which is basically just the specific language to communicate with NOAA to ask for data.

Let's take a look at the anatomy of a NOAA API request.

The base request looks like:

<https://www.ncdc.noaa.gov/cdo-web/api/v2/data?>

After the question mark, '?', we will need to put all the options specifying the exact data we are looking for. These will include:

- datasetid**, which for us will be **GHNCND** (Global Historical Climatology Network Daily).
- datatypeid**, which is a list of the variables we are after. For us, that will be **TAVG** (average temperature)
- limit**, specifying the maximum number of items to include in the response. The default is 25 and the maximum is 1000, which is what we will set.
- stationid**, specifying which station(s) we would like data from. Our station id is **GHNCND:USW00023129**
- startdate** and **enddate**, specifying the date range we wish to get data from. We will be calling the API once for each year because 1000 items is not enough for our full 3.5 year date range from 2015 until now.

In all, our first request for all data in 2015 will look like this:

<https://www.ncdc.noaa.gov/cdo-web/api/v2/data?datasetid=GHNCND&datatypeid=TAVG&limit=1000&stationid=GHNCND:USW00023129&startdate=2015-01-01&enddate=2015-12-31>

Let's take a look at the code to make the API calls:

```
1 #initialize lists to store data
2 dates_temp = []
3 dates_prop = []
4 temps = []
5 prop = []
6
7 #for each year from 2015-2019 ...
8 for year in range(2015, 2020):
9     year = str(year)
10    print('working on year ' + year)
11
12    #make the api call
13    r = requests.get('https://www.ncdc.noaa.gov/cdo-web/api/v2/data?datasetid=GHNCND&datatypeid=TAVG&limit=1000&stationid=GHNCND:USW00023129&startdate=' + year + '-01-01&enddate=' + year + '-12-31')
14    #load the api response as a json
15    d = json.loads(r.text)
16
17    #get all items in the response which are average temperature readings
18    avg_temps = [item for item in d['results'] if item['datatype']=='TAVG']
19    #get the date field from all average temperature readings
20    dates_temp += [item['date']] for item in avg_temps]
21    #get the actual average temperature from all average temperature readings
22    temps += [item['value']] for item in avg_temps]
```

And, now the code to store the collected data as a dataframe:

```
1 #initialize dataframe
2 df_temp = pd.DataFrame()
3
4 #populate date and average temperature fields (cast string date to datetime and convert
5 df_temp['date'] = [datetime.strptime(d, "%Y-%m-%dT%H:%M:%S") for d in dates_temp]
6 df_temp['avgTemp'] = [float(v)/10.0+32 for v in temps]
```

A chart made from the resulting dataframe looks like this:

Nice! So we successfully requested, processed, and reported on the average daily temperature in Long Beach, California since January 2015.

In truth, there are a **ton** more options built into the NOAA API. I figured it would be a good start to walk through one complete specific use case of the NOAA API.

~ Happy data gathering! :)

290 7

290 7

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

Get this newsletter

More from Towards Data Science

Your home for data science. A Medium publication sharing concepts, ideas and code.

Connor Shorten · Jun 10, 2019

Deep Compression

In their current form, Deep Neural Networks require enormous memory to fund their massive over-parameterization. Classic Neural Networks such as AlexNet and VGG-16 require around 240 and 552 MB, respectively. Many...

Deep Learning · 3 min read

Benny Friedman · Jun 10, 2019

Creating a TensorFlow CNN in C++ (Part 2)

In this post I'll show how to create, train and test a Convolutional Neural Network using the TensorFlow C++ API — Background In the last two years, Google's TensorFlow has been gaining popularity. It is by far the m...

Machine Learning · 13 min read

Prajakta KN · Jun 10, 2019

The next frontier in Big Data Analytics

Cross-industry analytics and its three pillars — In our world, data is ubiquitous but has diminishing marginal returns. More data does not necessarily translate into new information, in fact, it may sometimes only...

Big Data · 2 min read

Ryan Gross · Jun 10, 2019

Reinforcement learning is going mainstream. Here's what to expect.

The same tech that's beating world champions in games will soon revolutionize anything that can be simulated (and because everything is...

Machine Learning · 6 min read

Ashutosh Tripathi · Jun 10, 2019

Feature Selection Techniques in Regression Model

Feature selection is a way to reduce the number of features and hence reduce the computational complexity of the model. Many times feature selection becomes very useful to overcome with overfitting problem. It...

Machine Learning · 7 min read

[Read more from Towards Data Science](#)

Recommended from Medium

Milica

Gender gap in IT

Adresults live

How to predict 4d numbers accurately in Malaysia

Ramya Subramanya

Predicting Personal Loan defaulter

erin malone

Using Mapping to Scope a System

Lawrence Kuria... In Gojek Produ...

Efficient Experimentation at Gojek

Sabita Langkam

Descriptive Statistics for Data Science Part-2

IFRC

Logging data: 7 Factors for Maximizing Digital Opportunities in the Mining...

Additor

Misinterpretations and Legends About Sadness