

✓ Collecting weather data from an API

✓ About the data

In this notebook, we will be collecting daily weather data from the [National Centers for Environmental Information \(NCEI\) API](#). We will use the Global Historical Climatology Network - Daily (GHCND) data set; see the documentation [here](#).

Note: The NCEI is part of the National Oceanic and Atmospheric Administration (NOAA) and, as you can see from the URL for the API, this resource was created when the NCEI was called the NCDC. Should the URL for this resource change in the future, you can search for the NCEI weather API to find the updated one.

✓ Using the NCEI API

Paste your token below

```
1 import requests
2 def make_request(endpoint, payload=None):
3     """
4     Make a request to a specific endpoint on the weather API passing headers and optional payload.
5     Parameters:
6         - endpoint: The endpoint of the API you want to make a GET request to.
7         - payload: A dictionary of data to pass along with the request.
8
9     Returns:
10         Response object.
11     """
12
13     return requests.get(
14         f'https://www.ncdc.noaa.gov/cdo-web/api/v2/{endpoint}',
15         headers={
16             'token': 'uPbSRvXwGYFwftSwWzZNLZsxpPKvvaYN'
17         },
18         params=payload
19     )
```

✓ Collect All Data Points for 2018 In NYC (Various Stations)

We can make a loop to query for all the data points one day at a time. Here we create a list of all the results:

```
1 import datetime
2 from IPython import display # for updating the cell dynamically
3
4 current = datetime.date(2018, 1, 1)
5 end = datetime.date(2019, 1, 1)
6
7 results = []
8
9 while current < end:
10     # update the cell with status information
11     display.clear_output(wait=True)
12     display.display(f'Gathering data for {str(current)}')
13
14     response = make_request(
15         'data',
16         {
17             'datasetid': 'GHCND', # Global Historical Climatology Network - Daily (GHCND) dataset
18             'locationid': 'CITY:US360019', # NYC
19             'startdate': current,
20             'enddate': current,
21             'units': 'metric',
```

```

22 ..... 'limit': 1000 # max allowed
23 ..... }
24 ..)
25
26 ..if response.ok:
27 ..... # we extend the list instead of appending to avoid getting a nested list
28 ..... results.extend(response.json()['results'])
29
30 ..# update the current date to avoid an infinite loop
31 ..current += datetime.timedelta(days=1)

    'Gathering data for 2018-12-31'

```

Now, we can create a dataframe with all this data. Notice there are multiple stations with values for each datatype on a given day. We don't know what the stations are, but we can look them up and add them to the data:

```

1 import pandas as pd
2
3 df = pd.DataFrame(results)
4 df.head()

```

	date	datatype	station	attributes	value
0	2018-01-01T00:00:00	PRCP	GHCND:US1CTFR0039	„N,0800	0.0
1	2018-01-01T00:00:00	PRCP	GHCND:US1NJBG0015	„N,1050	0.0
2	2018-01-01T00:00:00	SNOW	GHCND:US1NJBG0015	„N,1050	0.0
3	2018-01-01T00:00:00	PRCP	GHCND:US1NJBG0017	„N,0920	0.0
4	2018-01-01T00:00:00	SNOW	GHCND:US1NJBG0017	„N,0920	0.0

Save this data to a file:

```

1 df.to_csv('/content/data/nyc_weather_2018.csv', index=False)

```

and write it to the database:

```

1 import sqlite3
2
3 with sqlite3.connect('/content/data/weather.db') as connection:
4     df.to_sql(
5         'weather', connection, index=False, if_exists='replace'
6     )

```

For learning about merging dataframes, we will also get the data mapping station IDs to information about the station:

```

1 response = make_request(
2     'stations',
3     {
4         'datasetid' : 'GHCND', # Global Historical Climatology Network - Daily (GHCND) dataset
5         'locationid' : 'CITY:US360019', # NYC
6         'limit' : 1000 # max allowed
7     }
8 )
9
10 stations = pd.DataFrame(response.json()['results'])[['id', 'name', 'latitude', 'longitude', 'elevation']]
11 stations.to_csv('/content/data/weather_stations.csv', index=False)
12
13 with sqlite3.connect('/content/data/weather.db') as connection:
14     stations.to_sql(
15         'stations', connection, index=False, if_exists='replace'
16     )

```

Comments and Conclusions

- I understand that this module is all about collecting all the weather data in NYC in 2018 by using a loop and saving the results into a list, then using pandas to save it as a dataframe and saving it as a .csv file.