making a request

In [1]:
```
import requests

def make_request(endpoint,payload=None) :
    return requests. get(
        f'https://www.ncei.noaa.gov/cdo-web/api/v2/{endpoint}',
        headers = {
            'token' : 'HMFjKrAcDVrobiWoVhKslmzbdKGjYsAY'
        },params = payload
    )
```

see what datasets are available

In [2]:
```
response = make_request('datasets')
response.status_code # if the output of the status code is 200 meaning the request is successful
```

Out[2]:
```
200
```

get the keys of the result

In [3]:
```
response.json().keys()
```

Out[3]:
```
dict_keys(['metadata', 'results'])
```

In [4]:
```
response.json()['metadata']
```

Out[4]:
```
{'resultset': {'offset': 1, 'count': 11, 'limit': 25}}
```

Figure out what data is in the result

In [5]:
```
response.json()['results'][0].keys()
```

Out[5]:
```
dict_keys(['uid', 'mindate', 'maxdate', 'name', 'datacoverage', 'id'])
```

Parse the result

In [6]:
```
[(data['id'], data['name']) for data in response.json()['results']] # getting the id and the name of the data set result
```

Out[6]:
```
[('GHCND', 'Daily Summaries'),
 ('GSOM', 'Global Summary of the Month'),
 ('GSOY', 'Global Summary of the Year'),
 ('NEXRAD2', 'Weather Radar (Level II)'),
 ('NEXRAD3', 'Weather Radar (Level III)'),
 ('NORMAL_ANN', 'Normals Annual/Seasonal'),
 ('NORMAL_DLY', 'Normals Daily'),
 ('NORMAL_HLY', 'Normals Hourly'),
 ('NORMAL_MLY', 'Normals Monthly'),
 ('PRECIP_15', 'Precipitation 15 Minute'),
 ('PRECIP_HLY', 'Precipitation Hourly')]
```

Figure out which data category we want

In [7]:
```
response = make_request(
    'datacategories',
    payload={
    'datasetid' : 'GHCND'
    }
    )
response.status_code # if the output returns 200 meaning the data set with the id you've given is available
```

Out[7]:
```
200
```

In [8]:
```
response.json()['results']
```

Out[8]:
```
[{'name': 'Evaporation', 'id': 'EVAP'},
 {'name': 'Land', 'id': 'LAND'},
 {'name': 'Precipitation', 'id': 'PRCP'},
 {'name': 'Sky cover & clouds', 'id': 'SKY'},
 {'name': 'Sunshine', 'id': 'SUN'},
 {'name': 'Air Temperature', 'id': 'TEMP'},
 {'name': 'Water', 'id': 'WATER'},
 {'name': 'Wind', 'id': 'WIND'},
 {'name': 'Weather Type', 'id': 'WXTYPE'}]
```

Grab the data type ID for the Temperature category

In [9]:
```
response = make_request(
    'datatypes',{'datacategoryid' : 'TEMP','limit' : 100})
response.status_code
```

Out[9]:
```
200
```

In [10]:
```
#get the data type id
import pandas as pd
a = pd.DataFrame([(data['id'],data['name']) for data in response.json()['results']][-5:], columns = ['id','name'])
a.index += 1
a
```

Out[10]:

	id	name
1	MNTM	Monthly mean temperature
2	TAVG	Average Temperature.
3	TMAX	Maximum temperature
4	TMIN	Minimum temperature
5	TOBS	Temperature at the time of observation

Determine which Location Category we want

In [11]:
```
#getting the location category
response = make_request(
'locationcategories',
{
'datasetid' : 'GHCND'
}
)
response.status_code
```

Out[11]:
```
200
```

In [12]:
```
import pprint
pprint.pprint(response.json())
```

```
{'metadata': {'resultset': {'count': 12, 'limit': 25, 'offset': 1}},
 'results': [{'id': 'CITY', 'name': 'City'},
             {'id': 'CLIM_DIV', 'name': 'Climate Division'},
             {'id': 'CLIM_REG', 'name': 'Climate Region'},
             {'id': 'CNTRY', 'name': 'Country'},
             {'id': 'CNTY', 'name': 'County'},
             {'id': 'HYD_ACC', 'name': 'Hydrologic Accounting Unit'},
             {'id': 'HYD_CAT', 'name': 'Hydrologic Cataloging Unit'},
             {'id': 'HYD_REG', 'name': 'Hydrologic Region'},
             {'id': 'HYD_SUB', 'name': 'Hydrologic Subregion'},
             {'id': 'ST', 'name': 'State'},
             {'id': 'US_TERR', 'name': 'US Territory'},
             {'id': 'ZIP', 'name': 'Zip Code'}]}
```

Get NYC Location ID

In [13]:
```
def get_item(name,what,endpoint,start=1,end=None):
    """
    parameters:
        name: item you are looking for

    """
    # find the midpoint which we use to cut the data in half each time
    mid = (start + (end if end else 1)) // 2

    # lowercase the name so this is not case-sensitive
    name = name.lower()
```

```
    # define the payload we will send with each request
    payload = {
        'datasetid' : 'GHCND',
        'sortfield' : 'name',
        'offset' : mid, # we will change the offset each time
        'limit' : 1 # we only want one value back
    }

    # make our request adding any additional filter parameters from `what`
    response = make_request(endpoint, {**payload, **what})

    if response.ok:
        # if response is ok, grab the end index from the response metadata the first time through
        end = end if end else response.json()['metadata']['resultset']['count']
        # grab the lowercase version of the current name
        current_name = response.json()['results'][0]['name'].lower()
        # if what we are searching for is in the current name, we have found our item
        if name in current_name:
            return response.json()['results'][0] # return the found item
        else:
            if start >= end:
            # if our start index is greater than or equal to our end, we couldn't find it
                return {}
            elif name < current_name:
                # our name comes before the current name in the alphabet, so we search further to the left
                return get_item(name, what, endpoint, start, mid - 1)
            elif name > current_name:
                # our name comes after the current name in the alphabet, so we search further to the right
                return get_item(name, what, endpoint, mid + 1, end)
    else:
        # response wasn't ok, use code to determine why
        print(f'Response not OK, status: {response.status_code}')
```

In [14]:
```
def get_location(name):
    return get_item(name, {'locationcategoryid' : 'CITY'}, 'locations')
```

In [16]:
```
nyc = get_location('New York')
nyc
```

Out[16]:
```
{'mindate': '1869-01-01',
 'maxdate': '2024-03-11',
 'name': 'New York, NY US',
 'datacoverage': 1,
 'id': 'CITY:US360019'}
```

Get the station ID for Central Park

In [19]:
```
central_park = get_item('NY City Central Park', {'locationid' : nyc['id']}, 'stations')
central_park
```

Out[19]:
```
{'elevation': 42.7,
 'mindate': '1869-01-01',
 'maxdate': '2024-03-10',
 'latitude': 40.77898,
 'name': 'NY CITY CENTRAL PARK, NY US',
 'datacoverage': 1,
 'id': 'GHCND:USW00094728',
 'elevationUnit': 'METERS',
 'longitude': -73.96925}
```

Request the temperature data

In [21]:
```
response = make_request(
    'data',
    {
        'datasetid' : 'GHCND',
        'stationid' : central_park['id'],
        'locationid' : nyc['id'],
        'startdate' : '2018-10-01',
        'enddate' : '2018-10-31',
        'datatypeid' : ['TMIN', 'TMAX', 'TOBS'], # temperature at time of observation, min, and max
        'units' : 'metric',
        'limit' : 1000
    }
)
response.status_code
```

Out[21]:
```
200
```

Create a DataFrame

In [22]:
```
import pandas as pd
df = pd.DataFrame(response.json()['results'])
df.head()
```

Out[22]:

	date	datatype	station	attributes	value
0	2018-10-01T00:00:00	TMAX	GHCND:USW00094728	,,W,2400	24.4
1	2018-10-01T00:00:00	TMIN	GHCND:USW00094728	,,W,2400	17.2
2	2018-10-02T00:00:00	TMAX	GHCND:USW00094728	,,W,2400	25.0
3	2018-10-02T00:00:00	TMIN	GHCND:USW00094728	,,W,2400	18.3
4	2018-10-03T00:00:00	TMAX	GHCND:USW00094728	,,W,2400	23.3

In [23]:
```
df.datatype.unique()
```

Out[23]:
```
array(['TMAX', 'TMIN'], dtype=object)
```

In [24]:
```
if get_item(
'NY City Central Park', {'locationid' : nyc['id'], 'datatypeid': 'TOBS'}, 'stations'
):
    print('Found!')
```

```
Found!
```

Using a different station

In [25]:
```
laguardia = get_item(
    'LaGuardia', {'locationid' : nyc['id']}, 'stations'
)
laguardia
```

Out[25]:
```
{'elevation': 3,
 'mindate': '1939-10-07',
 'maxdate': '2024-03-11',
 'latitude': 40.77945,
 'name': 'LAGUARDIA AIRPORT, NY US',
 'datacoverage': 1,
 'id': 'GHCND:USW00014732',
 'elevationUnit': 'METERS',
 'longitude': -73.88027}
```

In [26]:
```
response = make_request(
    'data',
    {
        'datasetid' : 'GHCND',
        'stationid' : laguardia['id'],
        'locationid' : nyc['id'],
        'startdate' : '2018-10-01',
        'enddate' : '2018-10-31',
        'datatypeid' : ['TMIN', 'TMAX', 'TAVG'], # temperature at time of observation, min, and max
        'units' : 'metric',
        'limit' : 1000
    }
)
response.status_code
```

Out[26]:
```
200
```

In [27]:
```
df = pd.DataFrame(response.json()['results'])
df.head()
```

Out[27]:

	date	datatype	station	attributes	value
0	2018-10-01T00:00:00	TAVG	GHCND:USW00014732	H,,S,	21.2
1	2018-10-01T00:00:00	TMAX	GHCND:USW00014732	,,W,2400	25.6
2	2018-10-01T00:00:00	TMIN	GHCND:USW00014732	,,W,2400	18.3
3	2018-10-02T00:00:00	TAVG	GHCND:USW00014732	H,,S,	22.7
4	2018-10-02T00:00:00	TMAX	GHCND:USW00014732	,,W,2400	26.1

In [29]:
```
df.to_csv('nyc_temperatures.csv', index=False)
```