tchosa-data-collection-through-api

March 13, 2024

1 Using the NCE API

[6]:
```python
import requests

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

1.0.1 See what datasets are available

[8]:
```python
# see what data sets are availvable
response = make_request('datasets', {'startdate':'2024-03-13'})
response.status_code
```

[8]:
```
200
```

1.0.2 Get the keys of the result

[9]:
```python
response.json().keys()
```

[9]:
```
dict_keys(['metadata', 'results'])
```

[10]:
```python
response.json()['metadata']
```

[10]:
```
{'resultset': {'offset': 1, 'count': 11, 'limit': 25}}
```

1.0.3 Figure out what data is in the result

[12]:
```python
response.json()['results'][0].keys()
```

[12]:
```
dict_keys(['uid', 'mindate', 'maxdate', 'name', 'datacoverage', 'id'])
```

1.0.4 Parse the result

[13]:
```
[(data['id'], data['name']) for data in response.json()['results']]
```

[13]:
```
[('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')]
```

1.0.5 Figure out which data category we want

[14]:
```
response = make_request(
    'datacategories',
    payload={
        'datasetid' : 'GHCND'
    }
)
response.status_code
```

[14]:
```
200
```

[16]:
```
response.json()['results']
```

[16]:
```
[{'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'}]
```

1.0.6 Grab the data type ID for the Temperature category

[17]:
```
response = make_request(
    'datatypes',
    payload={
        'datacategoryid' : 'TEMP',
        'limit' : 100
    }
)
response.status_code
```

[17]:
```
200
```

[18]:
```
[(datatype['id'], datatype['name']) for datatype in response.
 ↪json()['results']][-5:]
```

[18]:
```
[('MNTM', 'Monthly mean temperature'),
 ('TAVG', 'Average Temperature.'),
 ('TMAX', 'Maximum temperature'),
 ('TMIN', 'Minimum temperature'),
 ('TOBS', 'Temperature at the time of observation')]
```

[26]:
```
[(datatype['id'], datatype['name']) for datatype in response.
 ↪json()['results']][0:]
```

[26]:
```
[('CITY', 'City'),
 ('CLIM_DIV', 'Climate Division'),
 ('CLIM_REG', 'Climate Region'),
 ('CNTRY', 'Country'),
 ('CNTY', 'County'),
 ('HYD_ACC', 'Hydrologic Accounting Unit'),
 ('HYD_CAT', 'Hydrologic Cataloging Unit'),
 ('HYD_REG', 'Hydrologic Region'),
 ('HYD_SUB', 'Hydrologic Subregion'),
 ('ST', 'State'),
 ('US_TERR', 'US Territory'),
 ('ZIP', 'Zip Code')]
```

1.0.7 Determine which location Category we want

[40]:
```
response = make_request(
    'locationcategories',
    {
        'datasetid' : 'GHCND'
    }
)
response.status_code
```

[40]:
```
200
```

[41]:
```
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'}]}
```

[49]:

```
#for datatype in response.json()['results']
#[(datatype['id'], datatype['name'])][0]
response.json()['results'][0]
```

[49]:

```
{'name': 'City', 'id': 'CITY'}
```

1.0.8 Get NYC Location ID

[31]:

```
def get_item(name, what, endpoint, start=1, end=None):

  mid = (start + (end if end else 1)) // 2

  name = name.lower()

  payload = {
      'datasetid' : 'GHCND',
      'sortfield' : 'name',
      'offset' : mid,
      'limit' : 1
  }

  response = make_request(endpoint, {**payload, **what})

  if response.ok:
    end = end if end else response.json()['metadata']['resultset']['count']

    current_name = response.json()['results'][0]['name'].lower()

    if name in current_name:
      return response.json()['results'][0]
    else:
      if start >= end:
        return{}
      elif name < current_name:
        return get_item(name, what, endpoint, start, mid -1)
      elif name > current_name:
        return get_item(name, what, endpoint, mid + 1, end)

  else:
    print(f'Response not OK, status: {response.status_code}')

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

[32]:

```
nyc = get_location('New York')
nyc
```

[32]:

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

1.0.9 Get the station ID for Central Park

[33]:

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

[33]:

```
{'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}
```

1.0.10 Request the temperature data

[34]:

```
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'],
        'units' : 'metric',
        'limit' : 1000
    }
)
response.status_code
```

[34]:
```
200
```

1.0.11 Create a DataFrame

[35]:
```
import pandas as pd

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

[35]:
```
                  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
```

[36]:
```
df.datatype.unique()
```

[36]:
```
array(['TMAX', 'TMIN'], dtype=object)
```

[48]:
```
if get_item(
    'NY City Central Park', {'locationid' : nyc['id'], 'datatypeid' : 'TOBS'},␣
 ↪'stations'
):
  print('Found!')
```

```
Response not OK, status: 502
```

[50]:
```
laguardia = get_item(
    'LaGuardia', {'locationid' : nyc['id']}, 'stations'
)
laguardia
```

[50]:
```
{'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}
```

[52]:
```
response = make_request(
    'data',
    {
        'datasetid' : 'GHCND',
        'stationid' : laguardia['id'],
        'locationid' : nyc['id'],
        'startdate' : '2018-10-01',
        'enddate' : '2018-10-31',
        'datatypeid' : ['TMIN', 'TMAX', 'TAVG'],
        'units' : 'metric',
        'limit' : 1000
    }
)
response.status_code
```

[52]:
```
200
```

[53]:
```
df = pd.DataFrame(response.json()['results'])
df.head()
```

[53]:
```
                  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
```

[54]:
```
df.datatype.value_counts()
```

[54]:
```
TAVG    31
TMAX    31
TMIN    31
Name: datatype, dtype: int64
```

[55]:
```
df.to_csv('/content/detchosa_nyc_temperatures.csv', index=False)
```

[ ]:
```

```