8.1 Weather Data Collection

Name: Calingo, Christian Lei

Section: CPE22S3

Course: Computational Thinking With Python

Course Code: CPE 311

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

+ Code + Text

```
import datetime
from IPython import display # for updating the cell dynamically
current = datetime.date(2018, 1, 1)
end = datetime.date(2019, 1, 1)
results = []
while current < end:
# update the cell with status information
  display.clear_output(wait=True)
  display.display(f'Gathering data for {str(current)}')
  response = make_request('data',
   {
       'datasetid' : 'GHCND', # Global Historical Climatology Network - Daily (GHCND) dataset
       'locationid' : 'CITY:US360019', # NYC
       'startdate' : current,
       'enddate' : current,
       'units' : 'metric',
       'limit' : 10 # max allowed
   })
  if response.ok:
# we extend the list instead of appending to avoid getting a nested list
    results.extend(response.json()['results'])
# update the current date to avoid an infinite loop
    current += datetime.timedelta(days=1)
```

```
'Gathering data for 2018-12-31'
```

```
import pandas as pd
df = pd.DataFrame(results)
df
```

	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
...	...	...	...	...	...
3645	2018-12-31T00:00:00	PRCP	GHCND:US1NJBG0017	T,,N,0815	0.0
3646	2018-12-31T00:00:00	SNOW	GHCND:US1NJBG0017	T,,N,0815	0.0
3647	2018-12-31T00:00:00	WESF	GHCND:US1NJBG0017	T,,N,0815	0.0
3648	2018-12-31T00:00:00	PRCP	GHCND:US1NJBG0018	,,N,0900	0.0
3649	2018-12-31T00:00:00	SNOW	GHCND:US1NJBG0018	,,N,0900	0.0

3650 rows × 5 columns

Next steps: View recommended plots

Trying different values for the limit

In the code below, I tried changing the limit to 100 instead of 10 and 1000. As you can see, the limit affects the number of rows.

```
import datetime
from IPython import display # for updating the cell dynamically
current = datetime.date(2018, 1, 1)
end = datetime.date(2019, 1, 1)
results2 = []
while current < end:
# update the cell with status information
  display.clear_output(wait=True)
  display.display(f'Gathering data for {str(current)}')
  response = make_request('data',
   {
       'datasetid' : 'GHCND', # Global Historical Climatology Network - Daily (GHCND) dataset
       'locationid' : 'CITY:US360019', # NYC
       'startdate' : current,
       'enddate' : current,
       'units' : 'metric',
       'limit' : 100 # max allowed
   })
  if response.ok:
# we extend the list instead of appending to avoid getting a nested list
    results.extend(response.json()['results'])
# update the current date to avoid an infinite loop
    current += datetime.timedelta(days=1)
```

```
'Gathering data for 2018-12-31'
```

In here, since we have 100 as the limit, we have 40,150 rows, unlike the previous one, we have 3650 rows

```
import pandas as pd
df2 = pd.DataFrame(results)
df2
```

	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
...	...	...	...	...	...
40145	2018-12-31T00:00:00	SNOW	GHCND:US1NJUN0003	,,N,0900	0.0
40146	2018-12-31T00:00:00	PRCP	GHCND:US1NJUN0010	,,N,0730	0.0
40147	2018-12-31T00:00:00	SNOW	GHCND:US1NJUN0010	,,N,0730	0.0

Next steps: View recommended plots

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

```
import sqlite3
with sqlite3.connect('data/weather.db') as connection:
  df.to_sql('weather', connection, index=False, if_exists='replace')
```

```
response = make_request(
    'stations',
     {
         'datasetid' : 'GHCND', # Global Historical Climatology Network - Daily (GHCND) dataset
         'locationid' : 'CITY:US360019', # NYC
         'limit' : 1000 # max allowed
     })
stations = pd.DataFrame(response.json()['results'])[['id', 'name', 'latitude', 'longitude', 'elevation']]
stations.to_csv('data/weather_stations.csv', index=False)
with sqlite3.connect('data/weather.db') as connection:
  stations.to_sql(
      'stations', connection, index=False, if_exists='replace')
```

Conclusions

• In this activity, we saved a dataframe to the built-in database of Python, which is the sqlite3. Once the dataframe has been saved to the database, we can use DML or Data Manipulation Language to change or manipulate our database.