Background

Data generated from a variety of sources is growing exponentially. For example, one autonomous car can generate as much as 4000 GB of data per day. IoT sensors, mobile devices, social media, and websites generate large amounts of data requiring storage, management, analysis, and security. Large datasets or big data can be analyzed and turned into information to assist companies with decision making. This course project covers data analysis using Python that includes analysis of weather data downloaded from a cloud data source, stored in a database, extracted and processed. The design and development process of the system will include planning, software setup, programming, and data analysis. It will encompass the many aspects of the software development process and prepare you for your future career in technology.

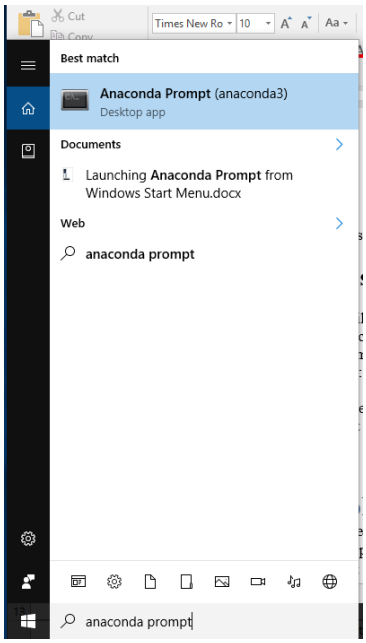
Scenario

Develop a software system to download weather data from a cloud source, store it in a database, and extract desired subsets for processing. Use programming and data analytics to analyze data and develop charts and predictions

a. Download weather data to a database.
b. Extract weather data from database into a comma separated file with python
c. Cleanse weather data
d. Use Excel to manipulate data
e. Use python data analytics modules to develop graphical models

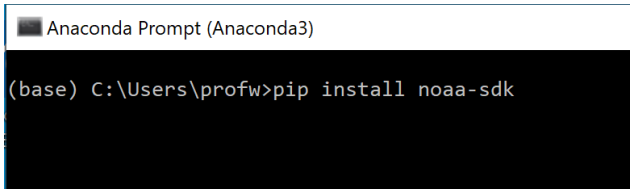
2. You will be using a Python program to download weather observations from the US government National Oceanic and Atmospheric Administration (NOAA) weather data service using a cloud-based Application Programming Interface (API). Perform the following steps to install the library module required to use this service:

   a. In the search box on the left side of the Windows taskbar at the bottom of your screen, enter Anaconda Prompt (without pressing Enter). In the search results, right click on Anaconda Prompt (anaconda3) and choose **Run as Administrator** to open the Anaconda Prompt command window.

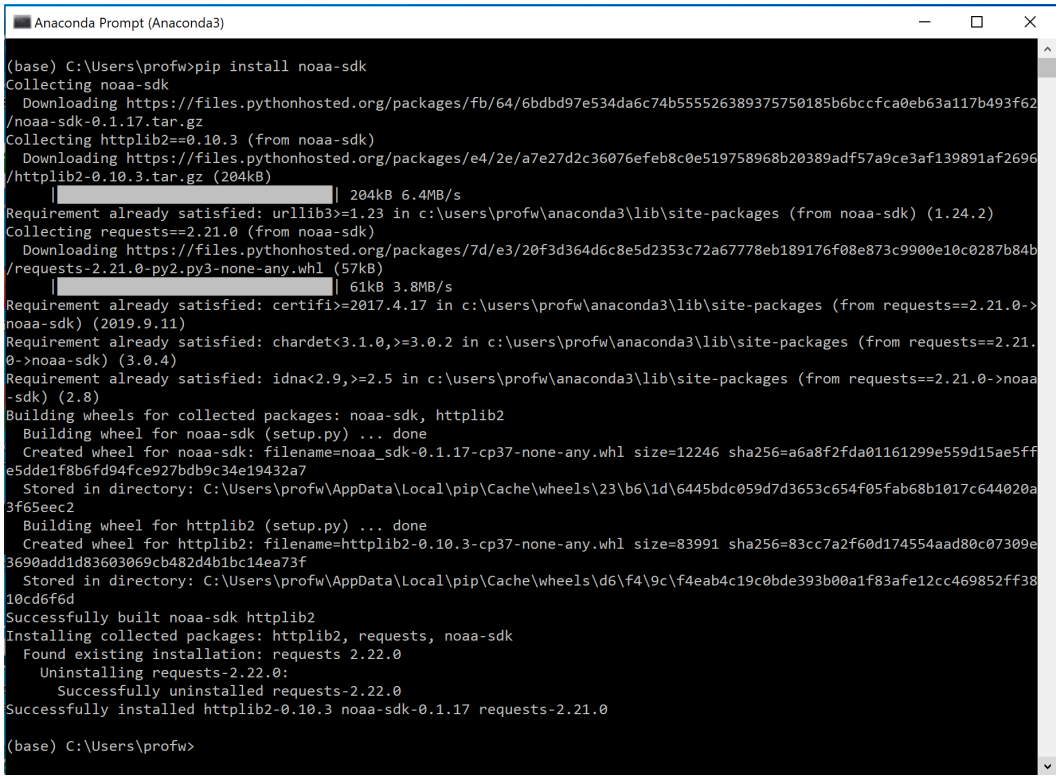


For macOS, open Launchpad, then click the terminal icon. For Linux, open a terminal window.

b. At the prompt, enter the following command and Enter: **pip install noaa-sdk**



The command may take a few minutes to execute. You should see output similar to the following. As long as the last line of the output says “Successfully installed noaa-sdk” you are good!



c. Before closing the window, capture this result in a screenshot to document successful installation.

Downloading Weather Data

Objectives

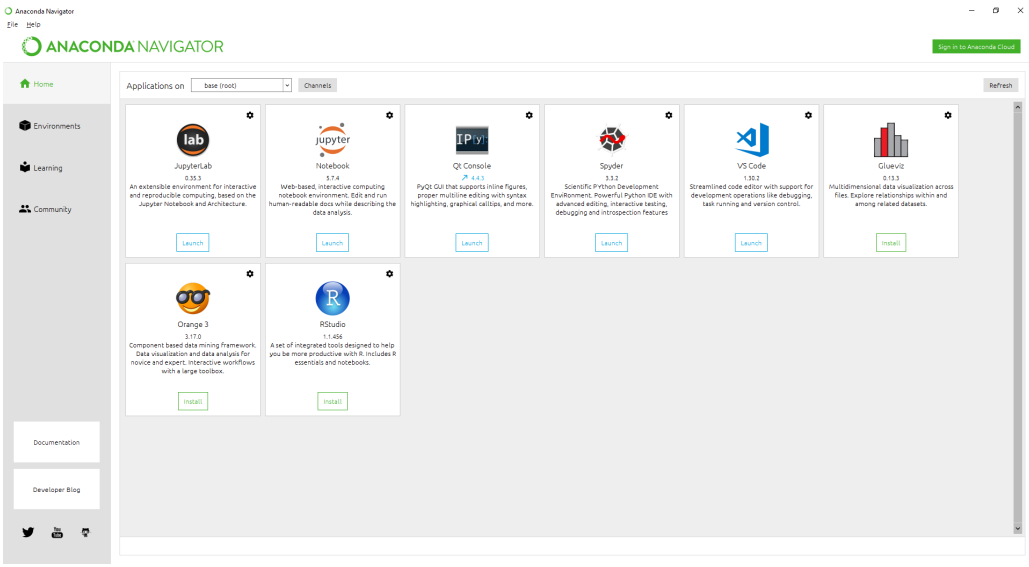
- To practice writing and executing Python programs using Anaconda Spyder IDE
- To learn how to use download data from the cloud using an API
- To learn how to create a relational database
- To learn how to save data into a database table using SQL and Python

Downloading Weather Data

The US National Oceanic and Atmospheric Administration (NOAA) provides free access to nationwide weather observations via a cloud-based Application Programming Interface (API). You will create and run a Python program to download a data set of recent weather observations for your location. Your Python program will create a database on your computer's hard drive and store the weather data in a table for later analysis.

Steps

1. Open Anaconda Navigator and click Launch on the Spyder icon:

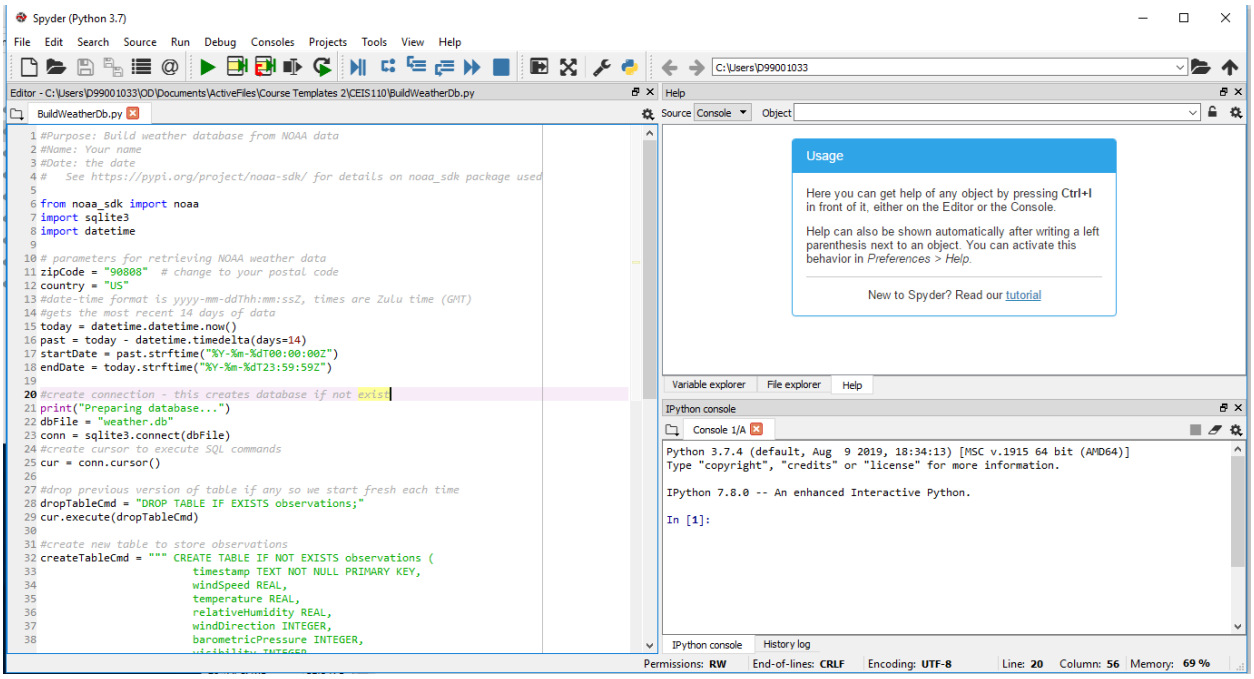


2. Copy the following code into the window. You will need to modify a few things:
   a. Change the name and date to your own
   b. Change the zip code to your own local 5-digit zip code
   c. Save your program as BuildWeatherDb.py. **NOTE:** You should create a class folder, if you have not already done so, and save this and all your other Python files for this project into that folder. Your weather database will be created in the same folder where your Python code files are saved. All Python programs must be in the same folder as the database in order to access the data.
   d. Please pay attention to the create table command in the following code. In the createTableCmd you will be creating an observations table with the fields: timestamp, windSpeed, temperature, relativeHumidity, windDirection, barometricPressure, visibility, and textDescription. The data types of each field are listed next to the field names below. The insert command enters the data into the table.

```
#Purpose: Build weather database from NOAA data
#Name: Your name
#Date: the date
#   See https://pypi.org/project/noaa-sdk/ for details on noaa_sdk package used

from noaa_sdk import noaa
import sqlite3
import datetime

# parameters for retrieving NOAA weather data
zipCode = "90808"  # change to your postal code
country = "US"
#date-time format is yyyy-mm-ddThh:mm:ssZ, times are Zulu time (GMT)
```

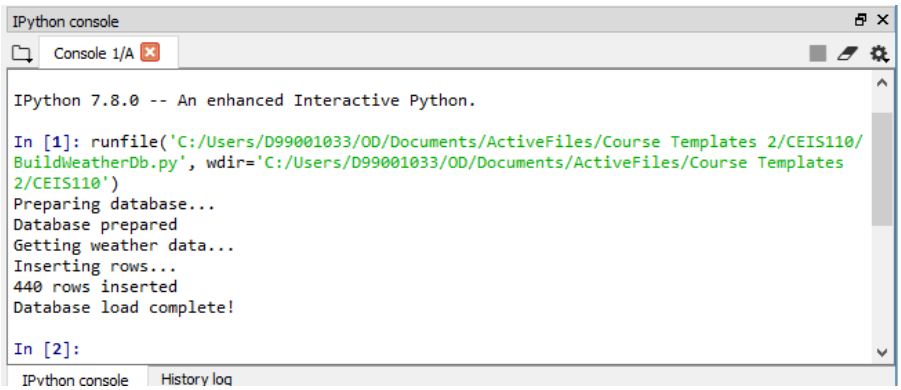
```
#gets the most recent 14 days of data
today = datetime.datetime.now()
past = today - datetime.timedelta(days=14)
startDate = past.strftime("%Y-%m-%dT00:00:00Z")
endDate = today.strftime("%Y-%m-%dT23:59:59Z")

#create connection - this creates database if not exist
print("Preparing database...")
dbFile = "weather.db"
conn = sqlite3.connect(dbFile)
#create cursor to execute SQL commands
cur = conn.cursor()

#drop previous version of table if any so we start fresh each time
dropTableCmd = "DROP TABLE IF EXISTS observations;"
cur.execute(dropTableCmd)

#create new table to store observations
createTableCmd = """ CREATE TABLE IF NOT EXISTS observations (
                    timestamp TEXT NOT NULL PRIMARY KEY,
                    windSpeed REAL,
                    temperature REAL,
                    relativeHumidity REAL,
                    windDirection INTEGER,
                    barometricPressure INTEGER,
                    visibility INTEGER,
                    textDescription TEXT
                ) ; """
cur.execute(createTableCmd)
print("Database prepared")

# Get hourly weather observations from NOAA Weather Service API
print("Getting weather data...")
n = noaa.NOAA()
observations =  n.get_observations(zipCode,country,startDate,endDate)

#populate table with weather observations
print("Inserting rows...")
insertCmd = """ INSERT INTO observations
                (timestamp, windSpeed, temperature, relativeHumidity,
                 windDirection, barometricPressure, visibility, textDescription)
              VALUES
                (?, ?, ?, ?, ?, ?, ?, ?) """
count = 0
for obs in observations:
   insertValues = (obs["timestamp"],
```

```
                    obs["windSpeed"]["value"],
                    obs["temperature"]["value"],
                    obs["relativeHumidity"]["value"],
                    obs["windDirection"]["value"],
                    obs["barometricPressure"]["value"],
                    obs["visibility"]["value"],
                    obs["textDescription"])
            cur.execute(insertCmd, insertValues)
            count += 1
        if count > 0:
            cur.execute("COMMIT;")
        print(count, "rows inserted")
print("Database load complete!")
```



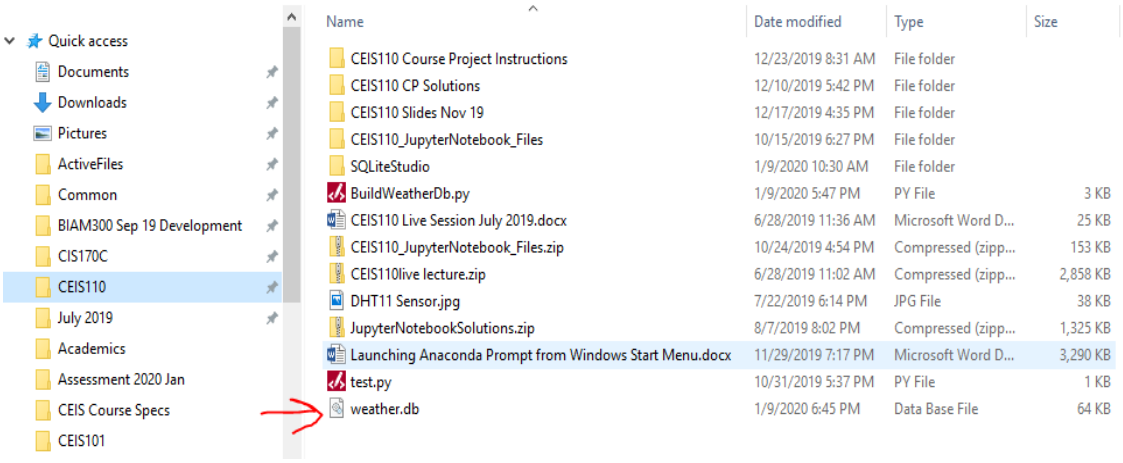
3. After saving, run your Python code by clicking the Run button (green triangle in the toolbar) or going to Run > Run (F5) in the menus.

   The program may take a few minutes to run. When it completes successfully, you should see messages similar to the following in the Python console at the lower right of the Spyder window:



Your number of rows inserted may be different from what is shown above, but it should be a few hundred rows. Capture a screenshot of this output to document that your program ran successfully.

4. Check the contents of your folder (or wherever your BuildWeatherDb.py program file was saved) to verify that a database file named weather.db was created in this folder. Take a screenshot showing the listing for this file. (You will not be able to open the file and view its contents, just take a screenshot of Windows Explorer to show that the file is there.)



Next Objective

- To connect to a database using Python
- To execute a simple SQL SELECT query
- To view data retrieved from the database using a query

Querying the Database

Structured Query Language (SQL) is a specialized programming language for working with a relational database. Database systems include client tools to allow developers to issue SQL commands to a database and view the results.

Programs written in a general-purpose programming language like Python, issue SQL commands to the database "under the hood" and receive and display the results to the user.

You will use Python to issue SQL query commands to your database and view the results. In the next stage, you will use a similar SQL query embedded in a Python program to retrieve data from the database and store it in Excel file.

Steps

1. Open Spyder and create a new file.
5. Copy the following code into the window.
    e. Change the name and date to your own
    f. Save your program as QueryWeatherDb.py. **NOTE:** You should save this and all your other Python files for this project into the folder created . Your weather database will be created in the same folder where your Python code files are saved. All Python programs must be in the same folder as the database in order to access the data.

```
#Purpose: Query database using SQL
#Name: Your name
#Date: Your date
#   Run BuildWeatherDB.py to build weather database before running this program

import sqlite3
import pandas as pd


#file names for database and output file
dbFile = "weather.db"

#format output
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
pd.set_option('display.expand_frame_repr', False)

#connect to and query weather database
conn = sqlite3.connect(dbFile)
#Create SQL command
selectCmd = " SELECT * FROM observations ORDER BY timestamp; "


#print out the query
result = pd.read_sql_query(selectCmd, conn)
print(result)
```

2. After saving your program, run your python code by clicking the Run button (green triangle in the toolbar). If you get an error that the database is not found. Please go to File -> Save As and save the python program in the same directory as weather.db.
3. You should see the weather data appear in your output window (make sure your output window is wide enough to see all columns):

```
In [40]: runfile('C:/CEIS110/sqlcoding.py', wdir='C:/CEIS110')
                  timestamp  windSpeed  temperature  relativeHumidity  windDirection  barometricPressure  visibility textDescription
0   2021-04-19T00:53:00+00:00      25.92         26.1         18.904128          260.0              101390     16090.0           Clear
1   2021-04-19T01:53:00+00:00        NaN         24.4         22.636684            NaN              101390     16090.0           Clear
2   2021-04-19T02:53:00+00:00        NaN         22.8         23.878012            NaN              101420     16090.0           Clear
3   2021-04-19T03:53:00+00:00      11.16         21.7         29.862534          260.0              101520     16090.0           Clear
4   2021-04-19T04:53:00+00:00      12.96         19.4         45.123994          290.0              101560     16090.0           Clear
5   2021-04-19T05:53:00+00:00        NaN         17.8         55.982540            3.1              101590     16090.0           Clear
6   2021-04-19T06:53:00+00:00       9.36         17.2         60.134186          310.0              101560     16090.0           Clear
7   2021-04-19T07:53:00+00:00       9.36         16.1         67.133567          300.0              101560     16090.0           Clear
8   2021-04-19T08:53:00+00:00        NaN         15.0         74.978034            0.0              101560     16090.0           Clear
9   2021-04-19T09:53:00+00:00       0.00         15.0         74.978034            0.0              101560     16090.0           Clear
10  2021-04-19T10:53:00+00:00       0.00         13.9         74.290523            0.0              101560     16090.0           Clear
11  2021-04-19T11:53:00+00:00        NaN         13.3         77.246796            0.0              101590     16090.0           Clear
12  2021-04-19T12:53:00+00:00       0.00         12.8         79.811182            0.0              101560     16090.0           Clear
13  2021-04-19T13:53:00+00:00       0.00         13.9         77.340598            0.0              101630     16090.0           Clear
14  2021-04-19T14:53:00+00:00       0.00         16.7         55.702677            0.0              101660     16090.0           Clear
15  2021-04-19T15:53:00+00:00       0.00         20.0         37.350835            0.0              101660     16090.0           Clear
16  2021-04-19T16:53:00+00:00       7.56         21.7         39.158254          160.0              101660     16090.0           Clear
17  2021-04-19T17:53:00+00:00        NaN         23.3         37.016196            2.6              101660     16090.0           Clear
18  2021-04-19T18:53:00+00:00      11.16         23.9         39.783076          170.0              101630     16090.0           Clear
19  2021-04-19T19:53:00+00:00      12.96         23.9         38.465645          200.0              101590     16090.0           Clear
20  2021-04-19T20:53:00+00:00       9.36         26.1         26.921778          170.0              101520     16090.0           Clear
21  2021-04-19T21:53:00+00:00       9.36         29.4         27.821532            NaN              101420     16090.0           Clear
22  2021-04-19T22:53:00+00:00      20.52         27.2         24.203008          300.0              101360     16090.0           Clear
23  2021-04-19T23:53:00+00:00        NaN         26.7               NaN            7.7              101320     16090.0           Clear
24  2021-04-20T00:53:00+00:00      14.76         24.4         29.787994          280.0              101320     16090.0           Clear
```

The command: SELECT * FROM observations; will retrieve all rows and all columns.

4. To see the lowest and highest temperatures observed in this data set, change the query to the following:

```
                    #Purpose: Query database using SQL
#Name: Your name
#Date: Your date
#   Run BuildWeatherDB.py to build weather database before running this program

import sqlite3
import pandas as pd


#file names for database and output file
dbFile = "weather.db"

#format output
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
pd.set_option('display.expand_frame_repr', False)

#connect to and query weather database
conn = sqlite3.connect(dbFile)
#Create SQL command
selectCmd = " SELECT MIN(temperature), MAX(temperature) FROM observations; "


#print out the query
result = pd.read_sql_query(selectCmd, conn)
print(result)
```

and run the program. Note that the temperatures supplied by the NOAA weather service use the Celsius scale. Capture a second screenshot showing your lowest and highest temperatures.

5. Write another query to find the temperature and windspeed of the records with a textDescription of clear. (Please note: you may not have any rows). Change the select command to the following:

   selectCmd = "SELECT temperature, windspeed, textDescription FROM observations where textDescription = 'Clear'; "

   Experiment to see what other weather data you can retrieve with similar queries. For example, can you write a query to find the lowest and highest relative humidity values in the data set? Hint: relativeHumidity is the field name you need to use. You can see the names of all the fields by looking at the full list (select * from observations;)

Deliverable :

- write a query to find the lowest and highest relative humidity values in the data set
- Write the names of all the fields by looking at the full list
- Include a screenshot of the query that retrieves all rows and columns and its results
- Include a screenshot of the query that finds the lowest and highest temperature and its results
- Include a screenshot of the query that finds the clear days.
- Put the screen shots in ppt .
- Submit this ppt and code file .