In [1]:

```python
#Import the dependencies.
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#import citipy as citipy

import numpy as np
import json
import requests
import timeit
```

In [2]:

```python
#Create a set of random latitude and longitude combinations.
lats = np.random.uniform(low=-90.000, high=90.000, size=1500)
lngs = np.random.uniform(low=-180.000, high=180.000,size=1500)
```

In [3]:

```python
#lat_lngs zip into list create a set of random latitudes and longitudes.
lat_lngs = zip(lats,lngs)
lat_lngs
```

Out[3]:

```
<zip at 0x7f99c2c0cf40>
```

In [4]:

```python
# Create a practice set of random latitude and longituted combinations.
lats = [25.12903625, 25.92017388, 26.62509167,-59.9896938437,37.30571269]
lngs = [-67.59741259, 11.09532135,74.84233102, -76.89176677,-61.13376282]
lat_lngs = zip(lat_lngs)
```

In [5]:

```python
#Add the latitudes and longitutes to a list.
coordinates = list(lat_lngs)
```

In [6]:

```python
# Use the print() fuction to display the latitude and longitude combinations.
for coordinate in coordinates:
    print(coordinate[0], coordinates[1])
```

```
(-87.60590929658393, -139.6971932834026) ((-7.469205367311062, 28.058994001609506),)
(-7.469205367311062, 28.058994001609506) ((-7.469205367311062, 28.058994001609506),)
(20.11645768295415, -128.16373707158795) ((-7.469205367311062, 28.058994001609506),)
(-47.04978734378678, 120.9516454825694) ((-7.469205367311062, 28.058994001609506),)
(25.704270404333286, -165.35695339146875) ((-7.469205367311062, 28.058994001609506),)
(-53.61035188423602, -78.12210476456822) ((-7.469205367311062, 28.058994001609506),)
(36.79371362278616, -98.23354472896995) ((-7.469205367311062, 28.058994001609506),)
(46.48667094608689, -150.70253488721534) ((-7.469205367311062, 28.058994001609506),)
(-86.82928207174727, 71.00744640615204) ((-7.469205367311062, 28.058994001609506),)
(8.566133331760753, 48.83235652226236) ((-7.469205367311062, 28.058994001609506),)
(2.8425169319432655, -123.94946238243944) ((-7.469205367311062, 28.058994001609506),)
(33.25062011598904, 65.75437194495402) ((-7.469205367311062, 28.058994001609506),)
(-70.9238263624479, 46.734661836569586) ((-7.469205367311062, 28.058994001609506),)
(-2.881116116431855, -25.827592588639163) ((-7.469205367311062, 28.058994001609506),)
(-28.593995357912398, 168.22810181841993) ((-7.469205367311062, 28.058994001609506),)
(-40.04381903484868, 104.0672693949897) ((-7.469205367311062, 28.058994001609506),)
(75.613089120431, 22.238550632067188) ((-7.469205367311062, 28.058994001609506),)
(-16.730474303739115, -21.289220384385885) ((-7.469205367311062, 28.058994001609506),)
(-36.05257517752878, -113.7692148395428) ((-7.469205367311062, 28.058994001609506),)
(-16.754316185780922, -143.2644658213485) ((-7.469205367311062, 28.058994001609506),)
```

In [ ]:

```
!pip install citipy
```

In [13]:

```python
#use the citipy mdoule to determine city based on latitude and longitude.
from citipy import citipy
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Input In [13], in <cell line: 2>()
      1 #use the citipy mdoule to determine city based on latitude and longitude.
----> 2 from citipy import citipy

ModuleNotFoundError: No module named 'citipy'
```

In [ ]:

```
for citipy.nearest_city()
    citipy.nearest_city()

```

In [10]:

```
#Identify the nearest city for each latitude and longtitude combination.
for coordinate in coordinates:
    citipy = citipy.nearest_city(coordinates[0],coordinates[1]).city_name

    #If the city is unique, then we will add to the cities list.
    if city not in cities:
        cities.append(city)
       #Print the city count to confirl sufficient count.
    len(cities)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [10], in <cell line: 2>()
      1 #Identify the nearest city for each latitude and longtitude combination.
      2 for coordinate in coordinates:
----> 3     citipy = citipy.nearest_city(coordinates[0],coordinates[1]).city_name
      5     #If the city is unique, then we will add to the cities list.
      6     if city not in cities:

NameError: name 'citipy' is not defined
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
#Starting URL for Weather Map API CAll.
import requests
from config import api_key
```

In [ ]:

```
url = "http//api.openweathermap.org/data/2.5/weather?units=Imperial&APPID + (weather_api_key)
print(url)
```

In [ ]:

```
#use the print() function to display the latitude and longitude combination.
for coordinate in coordinates:
    print(citipy.nearest_city(coordinate[0], coordinate[1].city_name,
           citypy.nearest_city(coordinates[1].coordinates[1].cuntry_code)
```

In [ ]:

```
#Create an endpoint URL for a city
city_url = +"&q=" + "Bston"
city_weather = requests.get(city_url)
city_weather

```

In [ ]:

```
#Get the text of the "Get' request.
city_weather.text
```

In [ ]:

```
#Get the JSON text of the 'Get'request.
city_weather.json()
```

In [ ]:

```
#Create an endpoint URL for a city.

city_url = url +"&q=" + "Boston"
city_weather = requests.get(city_url)
if city_weather.status_code == 200:
    print(f"City Weather found.")
else:
    print(f"City weather not found.")
```

In [ ]:

```
#Create an endpoint URL for a city.
city_url = url + "&q=" + "Bston"
city_weather = requests.get(city_url)
if city_weather.status_code == 200:
    print(f"City Weather found.")
else:
    print(f"City weather not found.")
```

In [ ]:

```
#Create an endpoint URL for a city.
city_url = url + "&q=" + "Boston"
city_weather = requests.get(city_url)
city_weather.json()
```

In [ ]:

```
#Get the JSON data.
boston_data = city_weather.json()

```

In [ ]:

```
lat = boston_data["coord"]["lat"]
lng = boston_data["coord"]["lon"]
max_temp = boston_data["main"]["temp-max"]
humidity = boston_data["main"]["humidity"]
clouds = boston_data["clouds"]["all"]
wind = boston_data["wind"]["speed"]
print(lat,lng,max-temp,humidity,clouds,)
```

In [ ]:

```
#Get the date from the JSON file
date = boston_data["dt"]
```

In [ ]:

```
#Convert the UTC date to a date format with year, month, day, hours, minutes, and seconds.
datetime.utcfromtimestamp(date)
```

In [ ]:

```
strftime('%Y-%m-%d %H:%M:%S')
```

In [ ]:

```
datetime.utcfromtimstamp(date).strftime('%Y-%m-%d %H:%M:%S')
```

In [ ]:

```
#Import the time library and the datetime module from the datetime library
import time
from datetime import datetime
```

In [ ]:

```
#Create an empty list to hold the weather data.
city_data =[]

```

In [ ]:

```
#print the beginning of the logging.
print("Beginning Data Retreieval       ")
print("--------------------------------")
```

In [ ]:
```
#Create counters
record_count = 1

set count = 1
```

In [ ]:
```
#Loop through all the cities in our list.
for i range (len(cities)):
```

In [ ]:
```
#Group cities in sets of 50 for logging purposes.
    if (i %50 ==0 and i >= 50):
        set_count += 1
        recond_count = 1
        time.sleep(60)
```

In [ ]:
```
#Create endpoint URL with each city.
city_url = url + "&q=" + citites[i]
```

In [ ]:
```
for i, item in enumerate(list):
```

In [ ]:
```
#Loop through all the cities in the list.
for i, city in enimerate(cities):
```

In [ ]:
```
#Group cities in sets of 50 for logging purposes.
if (i % 50 == 0 and i >= 50):
        set_count += 1
        recond_count = 1
        time.sleep(60)
```

In [ ]:
```
#Create endpoint URL with each city.
city_url = url + "&q=" + city.replace(" ","+")

```

In [ ]:
```
#Log the URL, record, and set numbers and the city.
print(f"Processing Record {record_count} of Set {set_count}  {city}")
#Add 1 to the record count.
record_ count += 1

```

In [ ]:

```
3Run an API request for each of the cities.

try:

    # Parse the JSON and retreive data
    city_weather = requests.get(city_url).json()



```

In [ ]:

```
#parse out the needed data.
city_lat = city_weather["coord"]["lat"]
city_lng = city_weather["coord"]["lon"]
city_max_temp = city_weather["main"]["temp_max"]
city_humidity = city_weather["main"]["humidity"]
city_clouds = city_weather["clouds"]["all"]
city_wind = city_weather["wind"]["speed"]
city_country = city_wather["sys"]["county"]

```

In [ ]:

```
#Convert the date to ISO standard.
city_date = datetime.utcfromtimestamp(city_weather)["dt"])strftime('%Y-%m-%d %H:%M:%S')
```

In [ ]:

```
#IF an error is experienced, skip the city,
except:
    print("City not found. Skipping...")
    pass
```

In [ ]:

```
#Indicate that Data Loading is complete.
print("-----------------------------")
print("Data Retrieval Complete      ")
print("-----------------------------")
```

In [ ]:

```
#Convert the aray of dictionaries to a Panda DataFrame
city_date_df = pd.DataFrame(city_data)
city_data_df.head(10)

```

In [ ]:

```
#Create the output file (CSV).
output_data_file = "weather_data/cities.csv"

```

In [ ]:

```
#Export the City_Data into a CSV.
city_data.df.to_csv(output_data_file,index_label = "City_ID")

```

In [ ]:

```

```