WeatherPy

Analysis

- As expected, the weather becomes significantly warmer as one approaches the equator (0 Deg. Latitude). More interestingly, however, is the fact that the southern hemisphere tends to be warmer this time of year than the northern hemisphere. This may be due to the tilt of the earth.
- There is no strong relationship between latitude and cloudiness. However, it is interesting to see that a strong band of cities sits at 0, 80, and 100% cloudiness.
- There is no strong relationship between latitude and wind speed. However, in northern hemispheres there is a flurry of cities with over 20 mph of wind.

Note

- Instructions have been included for each segment. You do not have to follow them exactly, but they are included to help you think through the steps.

In [19]:

```
# Dependencies and Setup
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import requests
import time

# Import API key
import api_keys

# Incorporated citipy to determine city based on latitude and longitude
from citipy import citipy

# Output File (CSV)
output_data_file = "output_data/cities.csv"

# Range of latitudes and longitudes
lat_range = (-90, 90)
lng_range = (-180, 180)
```

In [ ]:

```
```

In [ ]:

```

```

Generate Cities List

In [20]:

```python
# List for holding lat_lngs and cities
lat_lngs = []
cities = []

# Create a set of random lat and lng 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)
lat_lngs = zip(lats, lngs)

# Identify nearest city for each lat, lng combination
for lat_lng in lat_lngs:
    city = citipy.nearest_city(lat_lng[0], lat_lng[1]).city_name
    
    # If the city is unique, then add it to a our cities list
    if city not in cities:
        cities.append(city)

# Print the city count to confirm sufficient count
len(cities)
```

Out[20]:

```
623
```

In [21]:

```python
cities_df = pd.DataFrame(cities)
cities_df.head()
```

Out[21]:

	0
0	chuy
1	pisco
2	kapaa
3	cape town
4	rawson

In [ ]:

```

```

Perform API Calls

- Perform a weather check on each city using a series of successive API calls.
- Include a print log of each city as it'sbeing processed (with the city number and city name).

In [22]:

```
url = "http://api.openweathermap.org/data/2.5/weather?"
units = "imperial"
api_key = api_keys.api_key  #from api.py


# Build partial query URL

query_url = f"{url}appid={api_key}&units={units}&q="
query_url
```

Out[22]:

```
'http://api.openweathermap.org/data/2.5/weather?appid=3974be9fa6e0d0ba
48004fb47c9abbeb&units=imperial&q='
```

In [23]:

```
#sample a city (London) to see how json data is formated
city = "London"
london_url = query_url + city
weather_response = requests.get(london_url)
weather_json = weather_response.json()

# Get the temperature from the response
print(f"The weather API responded with: {weather_json}.")
```

```
The weather API responded with: {'coord': {'lon': -0.13, 'lat': 51.51}
, 'weather': [{'id': 802, 'main': 'Clouds', 'description': 'scattered
clouds', 'icon': '03n'}], 'base': 'stations', 'main': {'temp': 46.56,
'pressure': 1014, 'humidity': 87, 'temp_min': 44.6, 'temp_max': 48.2},
'visibility': 10000, 'wind': {'speed': 9.17, 'deg': 130}, 'clouds': {'
all': 32}, 'dt': 1541713800, 'sys': {'type': 1, 'id': 5091, 'message':
0.0065, 'country': 'GB', 'sunrise': 1541660866, 'sunset': 1541694010},
'id': 2643743, 'name': 'London', 'cod': 200}.
```

In [ ]:

```

```

In [*]:

```
'''try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block. '''


#Perform a weather check on each city using a series of successive API calls.
#city list


city = []
cloudiness = []
country = []
date = []
humidity = []
lat = []
lng = []
max_temp = []
wind_speed = []

#use except from Activities/08-Stu_MakingExceptions

#starting record so we have a number id for cities
record = 1

for city in cities:
    try:
        response = requests.get(query_url + city).json()

        city.append(response['name'])
        cloudiness.append(response['clouds'])
        country.append(response['sys']['country'])
        date.append(response['dt'])
        humidity.append(response['main']['humidity'])
        lat.append(response['coord']['lat'])
        lng.append(response['coord']['lng'])
        max_temp.append(response['main']['temp_max'])
        wind_speed.append(response['wind']['speed'])

        #Include a print log of each city as it's being processed (with the city nu
        #assigning a record number:
```

```python
        record += 1
        #naming the response
        city_data = response['name']


        # prevent overload https://www.tutorialspoint.com/python/time_sleep.htm
        #300 miliseconds
        #time.sleep(1.1)

        #read out of data using record variable
        print(f"City data is {record} | {city_data}")
        print(f"{url}&q={city}")

    except:
        print("No information, skipping")
    continue
```

```
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
No information, skipping
```

Convert Raw Data to DataFrame

- Export the city data into a .csv.
- Display the DataFrame

In [ ]:

```

```

In [ ]:

```

```

Plotting the Data

- Use proper labeling of the plots using plot titles (including date of analysis) and axes labels.
- Save the plotted figures as .pngs.

Latitude vs. Temperature Plot

In [ ]:

```

```

Latitude vs. Humidity Plot

In [ ]:

```

```

Latitude vs. Cloudiness Plot

In [ ]:

```

```

Latitude vs. Wind Speed Plot

In [ ]:

```

```

In [ ]:

```

```