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 [2]:
```
# Dependencies and Setup
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import requests
import time
from pprint import pprint
import json


# Import API key
from api_keys import api_key
from api_keys import darksky_api
from api_keys import gkey

# 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)
```

Generate Cities List

In [3]:
```
# 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[3]: 616

In [4]:
```
len(cities)
```

Out[4]: 616

In [5]:
```
cities_df=pd.DataFrame(cities)
cities_df2=cities_df.rename(columns={0:"City"})
cities_df2.head()
```

Out[5]:

	City
0	pran buri
1	kologriv
2	mehamn
3	vaitupu
4	lagoa

In [6]:
```
#Convert cities to Lat & Lon

#latitude=[]
#longitude=[]
#city_list=[]

#for city in cities:

 #   try:

  #      target_url = ('https://maps.googleapis.com/maps/api/geocode/json
   #     'address={0}&key={1}').format(city, gkey)

    #    geo_data=requests.get(target_url).json()

     #   latitude.append(geo_data["results"][0]["geometry"]["location"]["
      #  longitude.append(geo_data["results"][0]["geometry"]["location"][
       # city_list.append(city)
    #except:
     #   pass
#cities_df3 = pd.DataFrame({"City":city_list,"Latitude":latitude,"Longitu
#cities_df3.head()
```

In [7]:
```
#len(cities_df3)
```

In [8]:
```
# Darkskys api request: https://api.darksky.net/forecast/[key]/[latitude]
#dark_url = "https://api.darksky.net/forecast/"

#response=requests.get(dark_url+darksky_api+"/12.876497,11.031582").json(
#response
```

In [9]:
```
# Build partial query URL
url = "http://api.openweathermap.org/data/2.5/weather?q="
units = "&units=imperial"

# Build partial query URL

query_url = f"&appid={api_key}"

test_city="New York"

response=requests.get(url + test_city + units + query_url).json()
print(response)
```

{'coord': {'lon': -73.99, 'lat': 40.73}, 'weather': [{'id': 800, 'main': 'Clear', 'description': 'clear sky', 'icon': '01n'}], 'base': 'stations', 'main': {'temp': 53.13, 'pressure': 1019, 'humidity': 39, 'temp_min': 51.8, 'temp_max': 55.94}, 'visibility': 16093, 'wind': {'speed': 8.75, 'deg': 308.501}, 'clouds': {'all': 1}, 'dt': 1540936560, 'sys': {'type': 1, 'id': 2121, 'message': 0.0042, 'country': 'US', 'sunrise': 1540898702, 'sunset': 1540936409}, 'id': 5128581, 'name': 'New York', 'cod': 200}

In [10]:

```
url = "http://api.openweathermap.org/data/2.5/weather?q="
units = "&units=imperial"

# Build partial query URL

query_url = f"&appid={api_key}"
#api.openweathermap.org/data/2.5/weather?q={city name}
# set up lists to hold reponse info

name = []
cloudiness = []
country = []
date = []
humidity = []
lat = []
lng = []
temp = []
wind =[]

#city="New York"
#response = requests.get(url + city + query_url).json()

# Loop through the list of cities and perform a request for data on each

for city in cities:

    try:
        response = requests.get(url + city + units + query_url).json()
        wind.append(response['wind']['deg'])
        name.append(response["name"])
        cloudiness.append(response["clouds"]["all"])
        country.append(response["sys"]["country"])
        date.append(response['dt'])
        humidity.append(response['main']['humidity'])
        lat.append(response['coord']['lat'])
        lng.append(response['coord']['lon'])
        temp.append(response['main']['temp_max'])


    except:
        pass

len(date)
```

Out[10]: 536

In [11]:
```
print(len(name))
print(len(cloudiness))
print(len(country))
print(len(date))
print(len(humidity))
print(len(lat))
print(len(lng))
print(len(temp))
print(len(wind))
```

```
536
536
536
536
536
536
536
536
536
```

In [12]:
```
# create a data frame from cities, lat, and temp
weather_dict = {
    "City": name,
    "Cloudiness": cloudiness,
    "Country": country,
    "Date": date,
    "Humidity":humidity,
    "Lat": lat,
    "Lng":lng,
    "Max Temp": temp,
    "Wind Speed":wind
}
weather_data = pd.DataFrame(weather_dict)
weather_data.head()
```

Out[12]:

	City	Cloudiness	Country	Date	Humidity	Lat	Lng	Max Temp	Wind Speed
0	Pran Buri	0	TH	1540938683	100	12.39	99.90	83.26	59.5037
1	Kologriv	0	RU	1540938683	65	58.83	44.31	6.49	242.0040
2	Mehamn	0	NO	1540936200	46	71.03	27.85	30.20	200.0000
3	Lagoa	20	PT	1540936800	76	37.14	-8.45	53.60	290.0000
4	Busselton	0	AU	1540938451	100	-33.64	115.35	54.90	154.0010

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 [ ]:

```

```

In [13]:

```
weather_data.to_csv("Weather Data")
```

Convert Raw Data to DataFrame

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

In [14]:

```
weather_data.head()
```

Out[14]:

	City	Cloudiness	Country	Date	Humidity	Lat	Lng	Max Temp	Wind Speed
0	Pran Buri	0	TH	1540938683	100	12.39	99.90	83.26	59.5037
1	Kologriv	0	RU	1540938683	65	58.83	44.31	6.49	242.0040
2	Mehamn	0	NO	1540936200	46	71.03	27.85	30.20	200.0000
3	Lagoa	20	PT	1540936800	76	37.14	-8.45	53.60	290.0000
4	Busselton	0	AU	1540938451	100	-33.64	115.35	54.90	154.0010

In [ ]:

```

```

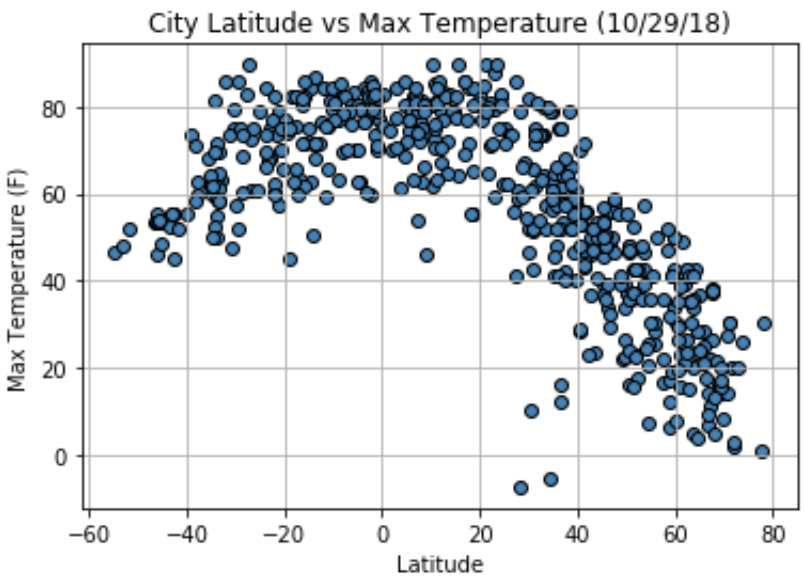
Plotting the Data

- 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 [15]:
```
fig1, ax1 = plt.subplots()

ax1.set_xlabel("Latitude")
ax1.set_ylabel("Max Temperature (F)")
ax1.set_title("City Latitude vs Max Temperature (10/29/18)")

x1 = weather_data["Lat"]
y1 = weather_data["Max Temp"]
plt.grid()

ax1.scatter(x1, y1, marker="o",edgecolors="black",color="steelblue")
```

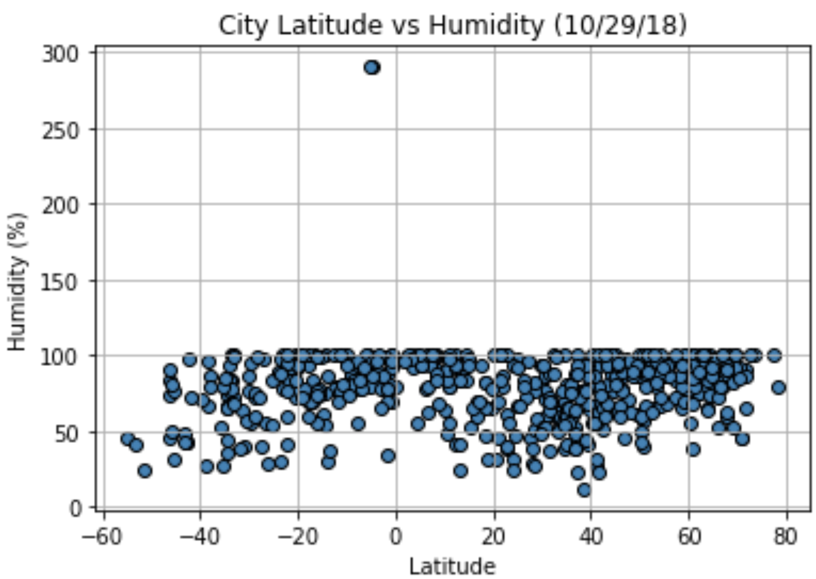
Out[15]: <matplotlib.collections.PathCollection at 0x10dcb6a90>



Latitude vs. Humidity Plot

In [16]:
```
fig2, ax2 = plt.subplots()

ax2.set_xlabel("Latitude")
ax2.set_ylabel("Humidity (%)")
ax2.set_title("City Latitude vs Humidity (10/29/18)")

x2 = weather_data["Lat"]
y2 = weather_data["Humidity"]
plt.grid()

ax2.scatter(x2, y2, marker="o",edgecolors="black",color="steelblue")
```

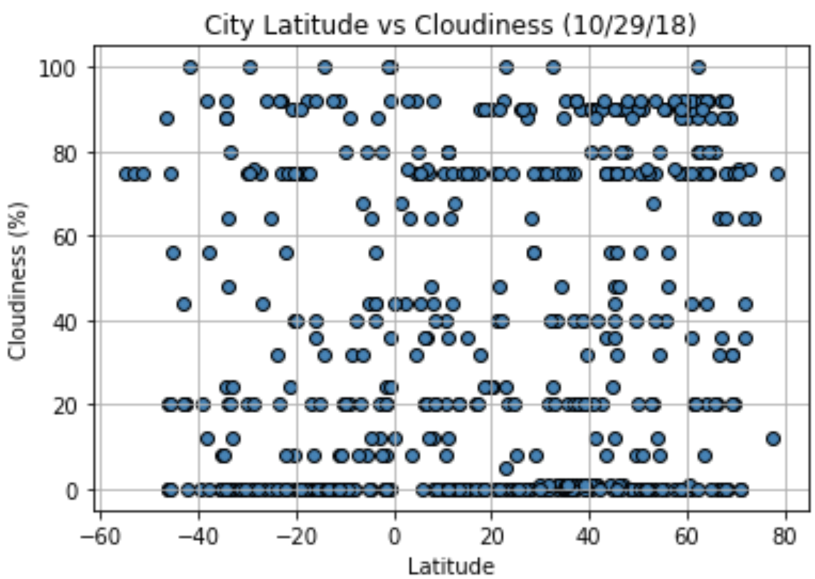
Out[16]: <matplotlib.collections.PathCollection at 0x110a8d828>



Latitude vs. Cloudiness Plot

In [17]:
```
fig3, ax3 = plt.subplots()

ax3.set_xlabel("Latitude")
ax3.set_ylabel("Cloudiness (%)")
ax3.set_title("City Latitude vs Cloudiness (10/29/18)")

x3 = weather_data["Lat"]
y3 = weather_data["Cloudiness"]
plt.grid()

ax3.scatter(x3, y3, marker="o",edgecolors="black",color="steelblue")
```

Out[17]: <matplotlib.collections.PathCollection at 0x110af8c88>

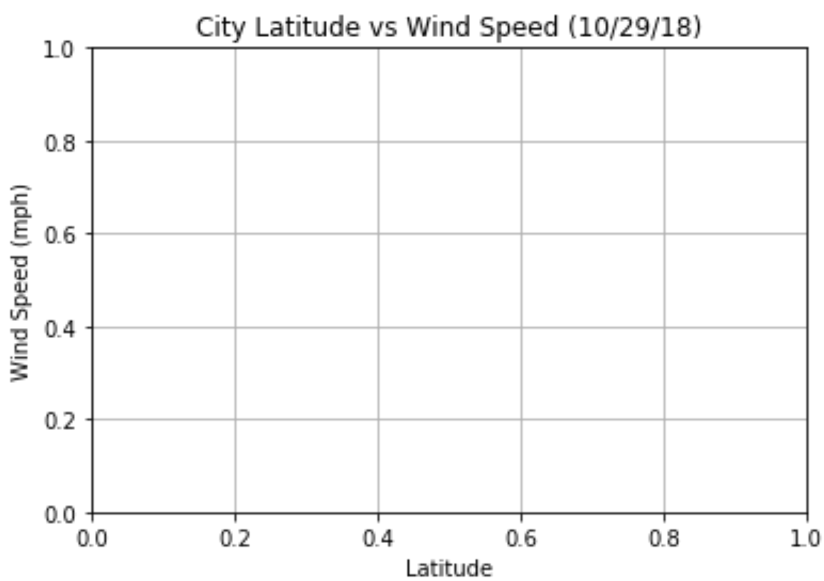


In [18]:
```
print(weather_data["Wind Speed"])
```

```
0      59.50370
1     242.00400
2     200.00000
3     290.00000
4     154.00100
5     300.00000
6     218.00100
7      77.00370
8     250.00000
9     120.00000
10    240.00000
11    240.00000
12    115.50400
13     67.50060
14    124.50100
15    107.50400
16    170.00000
```

```
16     170.00000
17     360.00000
18      29.50370
19     330.00000
20      70.00000
21     330.00000
22     228.00400
23     187.50400
24     264.00400
25     340.00000
26     110.00000
27     120.00000
28     100.00000
29      17.50060
         ...
506      1.50372
507    307.00400
508    130.00400
509     58.50370
510    177.50100
511    178.00400
512    260.00000
513    236.00400
514    200.50400
515    280.00000
516    142.00400
517    250.50100
518    344.00400
519     85.50370
520    121.50400
521    135.50400
522    220.00000
523    170.00000
524    114.00400
525    160.00000
526    340.00000
527    263.00400
528     50.00000
529    220.00000
530    320.00000
531    307.00400
532    235.50400
533    260.00000
534    111.00400
535    310.00000
Name: Wind Speed, Length: 536, dtype: float64
```

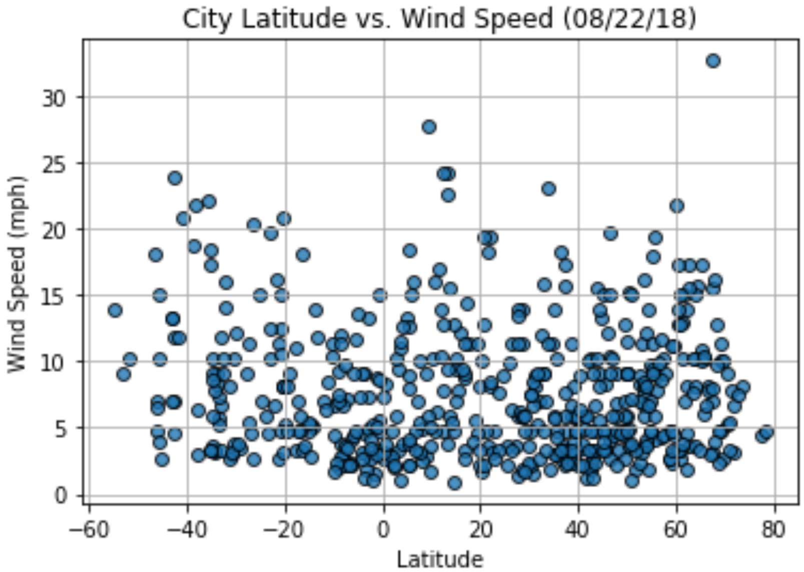
In [22]:
```
fig4, ax4 = plt.subplots()

ax4.set_xlabel("Latitude")
ax4.set_ylabel("Wind Speed (mph)")
ax4.set_title("City Latitude vs Wind Speed (10/29/18)")

x4 = weather_data["Lat"]
y4 = weather_data["Wind Speed"]
plt.grid()

ax3.scatter(x4, y4, marker="o",edgecolors="black",color="steelblue")
plt.show()

# No Idea why the graph won't appear. I tried using the variables from pr
```



In [ ]:

Latitude vs. Wind Speed Plot

In [9]:


City Latitude vs. Wind Speed (08/22/18)
Wind Speed (mph)
Latitude
0
5
10
15
20
25
30
−60
−40
−20
0
20
40
60
80

In [ ]: