**WeatherPy[¶](https://render.githubusercontent.com/view/ipynb?commit=2e67ec608169ad9cd46d4d42fa3e8125ebaea490&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f6c6b616c616c61302f776561746865725f70792f326536376563363038313639616439636434366434643432666133653831323565626165613439302f5765617468657250792d636865636b706f696e742e6970796e62&nwo=lkalala0%2Fweather_py&path=WeatherPy-checkpoint.ipynb&repository_id=199238556&repository_type=Repository" \l "WeatherPy)**

**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 [1]:

*# Dependencies and Setup*

**import** **matplotlib.pyplot** **as** **plt**

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **requests**

**import** **time**

**import** **json**

*# Import API key*

**from** **api\_keys** **import** api\_key

*# 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 [2]:

*# 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

cities.append(city)

*# If the city is unique, then add it to a our cities list*

*# if city not in cities:*

*# cities.append(city)*

*#cities*

In [3]:

new\_city = pd.DataFrame(cities)

new\_city = new\_city.drop\_duplicates()

*#new\_city*

In [4]:

*##### I want to get sample cities of 500*

sample\_cities = new\_city.sample(500)

sample\_cities = sample\_cities.reset\_index(drop=**True**)

sample\_cities.columns = ['City']

*#sample\_cities*

In [5]:

*# add new columns*

sample\_cities['Cloudiness'] = " "

sample\_cities['Country'] = " "

sample\_cities['Date'] = " "

sample\_cities['Humidity'] = " "

sample\_cities['Lat'] = " "

sample\_cities['Long'] = " "

sample\_cities['Max Temp'] = " "

sample\_cities['Wind Speed'] = " "

In [6]:

sample\_cities.head()

Out[6]:

|  | **City** | **Cloudiness** | **Country** | **Date** | **Humidity** | **Lat** | **Long** | **Max Temp** | **Wind Speed** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | saskylakh |  |  |  |  |  |  |  |  |
| **1** | ejutla de crespo |  |  |  |  |  |  |  |  |
| **2** | benalla |  |  |  |  |  |  |  |  |
| **3** | nibbar |  |  |  |  |  |  |  |  |
| **4** | mys shmidta |  |  |  |  |  |  |  |  |

In [7]:

*#sample\_cities.rename(columns={'': 'City'}, inplace=True)*

*#sample\_cities.head()*

**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 [8]:

url = "http://api.openweathermap.org/data/2.5/weather?"

units = 'Imperial'

**for** index, row **in** sample\_cities.iterrows():

city = row['City']

*#target\_url = "http://api.openweathermap.org/data/2.5/weather?units=%s&APPID=%s&q=%s" % (units,api\_key, row['City'])*

target\_url = f"**{url}**appid=**{api\_key}**&q=**{city}**&units=**{units}**"

*#city = row['City']*

response = requests.get(target\_url).json()

*#print(json.dumps(response, indent=4))*

*#sample\_cities.set\_value(index, "Wind Speed", response["wind"]["speed"])*

*#break*

*#*

*# break*

**try**:

*#sample\_cities.loc[index, 'Cloudiness'] = response["clouds"]["all"]*

sample\_cities.loc[index, 'Cloudiness'] = response['clouds']['all']

*# cloud = response['clouds']['all']*

*# print(response)*

*#break*

sample\_cities.loc[index, 'Country'] = response["sys"]["country"]

*# break*

sample\_cities.loc[index, 'Date'] = response["dt"]

sample\_cities.loc[index, 'Humidity'] = response["main"]["humidity"]

sample\_cities.loc[index, 'Lat'] = response["coord"]["lat"]

sample\_cities.loc[index, 'Long'] = response["coord"]["lon"]

sample\_cities.loc[index, 'Max Temp'] = response["main"]["temp\_max"]

sample\_cities.loc[index, 'Wind Speed'] = response["wind"]["speed"]

*#sample\_cities.set\_value(index,"Wind Speed",response.get("wind").get("speed"))*

*# sample\_cities.set\_value(index,"Lat",response.get("coord").get("lat"))*

**except**(**KeyError**):

*#print('Skipping.......')*

sample\_cities.loc[index, 'Cloudiness'] = 0

sample\_cities.loc[index, 'Country'] = 0

sample\_cities.loc[index, 'Date'] = 0

sample\_cities.loc[index, 'Humidity'] = 0

sample\_cities.loc[index, 'Lat'] = 0

sample\_cities.loc[index, 'Long'] = 0

sample\_cities.loc[index, 'Max Temp'] = 0

sample\_cities.loc[index, 'Wind Speed'] = 0

*#response*

*#city\_lat = []*

*#city\_lat = response['coord']['lat']*

*#city\_lat()*

sample\_cities

Out[8]:

|  | **City** | **Cloudiness** | **Country** | **Date** | **Humidity** | **Lat** | **Long** | **Max Temp** | **Wind Speed** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | saskylakh | 90 | RU | 1564247571 | 63 | 71.97 | 114.09 | 47.39 | 11.23 |
| **1** | ejutla de crespo | 75 | MX | 1564247711 | 46 | 16.57 | -96.73 | 73.4 | 7.05 |
| **2** | benalla | 90 | AU | 1564247711 | 100 | -36.55 | 145.99 | 48.99 | 4.65 |
| **3** | nibbar | 20 | TN | 1564247711 | 19 | 36.29 | 8.77 | 96.8 | 12.75 |
| **4** | mys shmidta | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | chagda | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6** | belushya guba | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7** | nizhnyaya tavda | 100 | RU | 1564247712 | 90 | 57.67 | 66.17 | 60.53 | 3.27 |
| **8** | loreto | 0 | PY | 1564247712 | 45 | -23.27 | -57.32 | 68.09 | 5.93 |
| **9** | limoeiro do ajuru | 19 | BR | 1564247470 | 47 | -1.9 | -49.38 | 91.31 | 3.04 |
| **10** | san jose | 34 | HN | 1564247643 | 47 | 14.25 | -87.96 | 80.33 | 6.02 |
| **11** | roros | 43 | NO | 1564247712 | 34 | 62.57 | 11.38 | 82.4 | 6.93 |
| **12** | sabang | 99 | PH | 1564247712 | 97 | 13.8 | 121.11 | 75 | 0.78 |
| **13** | karratha | 87 | AU | 1564247713 | 36 | -20.74 | 116.85 | 72.77 | 15.5 |
| **14** | henties bay | 1 | NA | 1564247713 | 94 | -22.12 | 14.28 | 53.01 | 1.99 |
| **15** | storforshei | 9 | NO | 1564247713 | 70 | 66.4 | 14.52 | 81 | 1.63 |
| **16** | yankton | 1 | US | 1564247460 | 58 | 42.87 | -97.4 | 84.99 | 8.05 |
| **17** | nishihara | 20 | JP | 1564247713 | 78 | 35.74 | 139.53 | 82 | 4.7 |
| **18** | cochrane | 53 | CL | 1564247713 | 94 | -47.25 | -72.57 | 33.17 | 4.05 |
| **19** | acu | 20 | BR | 1564247714 | 62 | -3.35 | -60.45 | 87.8 | 4.7 |
| **20** | carroll | 80 | AU | 1564247714 | 86 | -30.99 | 150.44 | 39.2 | 5.82 |
| **21** | lodja | 18 | CD | 1564247714 | 21 | -3.52 | 23.6 | 77.45 | 0.89 |
| **22** | gazanjyk | 27 | TM | 1564247714 | 44 | 39.24 | 55.52 | 91.13 | 13.35 |
| **23** | kamenka | 51 | MD | 1564247714 | 67 | 48.03 | 28.7 | 78.01 | 6.33 |
| **24** | nouadhibou | 20 | MR | 1564247715 | 73 | 20.93 | -17.03 | 73.4 | 18.34 |
| **25** | narsaq | 20 | GL | 1564247620 | 48 | 60.91 | -46.05 | 62.6 | 8.05 |
| **26** | new norfolk | 40 | AU | 1564247575 | 70 | -42.78 | 147.06 | 46.4 | 6.93 |
| **27** | vaitupu | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **28** | caloundra | 40 | AU | 1564247715 | 93 | -26.8 | 153.14 | 57.99 | 9.17 |
| **29** | lorengau | 96 | PG | 1564247617 | 83 | -2.02 | 147.27 | 78.35 | 9.01 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **470** | kuche | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **471** | polonne | 80 | UA | 1564247802 | 84 | 50.12 | 27.51 | 68.27 | 7.83 |
| **472** | batemans bay | 40 | AU | 1564247803 | 99 | -35.71 | 150.18 | 46 | 1.01 |
| **473** | grimshaw | 90 | CA | 1564247803 | 93 | 56.19 | -117.61 | 59 | 8.05 |
| **474** | upernavik | 61 | GL | 1564247803 | 83 | 72.79 | -56.15 | 42.71 | 4.99 |
| **475** | katsuura | 75 | JP | 1564247803 | 94 | 33.93 | 134.5 | 80.01 | 3.36 |
| **476** | dikson | 100 | RU | 1564247803 | 98 | 73.51 | 80.55 | 34.25 | 8.08 |
| **477** | the valley | 40 | AI | 1564247804 | 66 | 18.22 | -63.06 | 89.6 | 14.99 |
| **478** | anar darreh | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **479** | ouegoa | 0 | NC | 1564247804 | 75 | -20.34 | 164.44 | 69.35 | 5.19 |
| **480** | husavik | 1 | CA | 1564247804 | 48 | 50.56 | -96.99 | 79 | 7 |
| **481** | moussoro | 64 | TD | 1564247804 | 43 | 13.64 | 16.49 | 89.15 | 10.96 |
| **482** | butaritari | 50 | KI | 1564247804 | 70 | 3.07 | 172.79 | 84.65 | 8.25 |
| **483** | lata | 100 | IN | 1564247805 | 91 | 30.78 | 78.62 | 47.39 | 0.74 |
| **484** | wahiawa | 20 | US | 1564247805 | 88 | 21.5 | -158.02 | 80.01 | 4.7 |
| **485** | pevek | 100 | RU | 1564247805 | 98 | 69.7 | 170.27 | 32.99 | 5.97 |
| **486** | wulanhaote | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **487** | fortuna | 0 | ES | 1564247806 | 20 | 38.18 | -1.13 | 95 | 19.46 |
| **488** | yebaishou | 2 | CN | 1564247806 | 87 | 41.4 | 119.64 | 69.89 | 5.82 |
| **489** | manjacaze | 0 | MZ | 1564247806 | 87 | -24.71 | 33.88 | 59.63 | 2.44 |
| **490** | tecolutla | 56 | MX | 1564247807 | 60 | 17.31 | -98.74 | 73.67 | 6.33 |
| **491** | mercedes | 3 | UY | 1564247807 | 70 | -33.25 | -58.03 | 59 | 5.01 |
| **492** | nome | 20 | US | 1564247807 | 70 | 30.04 | -94.42 | 91 | 8.05 |
| **493** | hobyo | 41 | SO | 1564247807 | 73 | 5.35 | 48.53 | 75.65 | 23.91 |
| **494** | vardo | 20 | US | 1564247807 | 54 | 39.62 | -77.74 | 89.01 | 4.7 |
| **495** | dubenskiy | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **496** | san clemente | 1 | US | 1564247790 | 65 | 33.43 | -117.61 | 91.99 | 6.93 |
| **497** | hambantota | 77 | LK | 1564247808 | 82 | 6.12 | 81.12 | 80.69 | 19.84 |
| **498** | baranain | 40 | ES | 1564247808 | 82 | 42.81 | -1.68 | 66 | 10.29 |
| **499** | laguna | 75 | BZ | 1564247808 | 79 | 16.17 | -88.94 | 86 | 4.7 |

500 rows × 9 columns

In [ ]:

**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 [9]:

plt.scatter(sample\_cities['Lat'], sample\_cities['Max Temp'],marker='o', facecolors='blue', edgecolors='red')

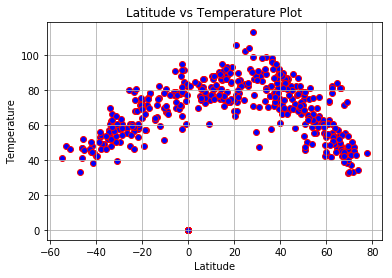
plt.grid()

plt.title('Latitude vs Temperature Plot')

plt.xlabel('Latitude')

plt.ylabel('Temperature')

plt.show()



**Latitude vs. Humidity Plot**

In [10]:

plt.scatter(sample\_cities['Lat'], sample\_cities['Humidity'], marker='o', facecolors='blue', edgecolors='red')

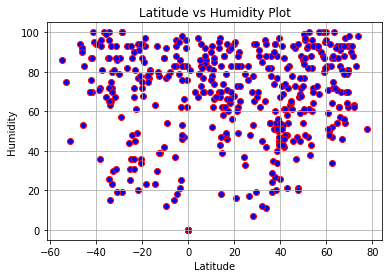
plt.title('Latitude vs Humidity Plot')

plt.xlabel('Latitude')

plt.ylabel('Humidity')

plt.grid()

plt.show()



**Latitude vs. Cloudiness Plot**

In [11]:

plt.scatter(sample\_cities['Lat'], sample\_cities['Cloudiness'], marker='o', facecolors='blue', edgecolors='red')

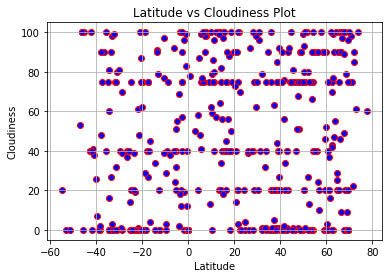
plt.title('Latitude vs Cloudiness Plot')

plt.xlabel('Latitude')

plt.ylabel('Cloudiness')

plt.grid()

plt.show()



**Latitude vs. Wind Speed Plot**

In [12]:

plt.scatter(sample\_cities['Lat'], sample\_cities['Wind Speed'], marker='o', facecolors='blue', edgecolors='red')

plt.title('Latitude vs Wind Speed Plot')

plt.xlabel('Latitude')

plt.ylabel('Wind Speed')

plt.grid()

In [ ]: