6.3.1 Plot Latitude vs. Temperature

After exporting your CSV file, you closed your laptop and headed home, confident in a good day's work.

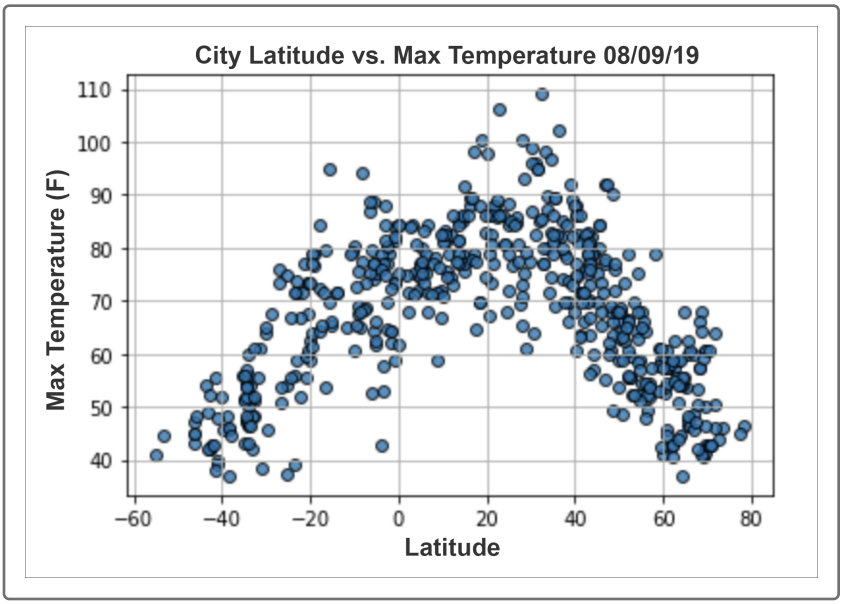
And, when you come into work the next day, you hear that someone else has been impressed with your work as well—Jack! In fact, Jack is so impressed that he wants to give you the opportunity to take the day to dedicate some time to a different project: creating a community outreach website for middle school STEM students.

This is an exciting project. These community-focused opportunities don't come around as often as you would like, and they are a great chance to get kids involved in STEM. Additionally, you work with other tech companies in the area, each taking on one aspect of the project.

Your company is going to focus on climate change, and since you know how to use Matplotlib, you decide to create some visualizations that showcase the weather parameters you retrieved with changing latitude for the 500-plus cities from all over the world. The students will then be able to use these visualizations to explore how weather parameters change based on latitude.

We are going to create a series of scatter plots for each weather parameter against the latitude for all the cities. The students will use these scatter plots to write a summary report on how different weather parameters change based on the latitude.

We'll create scatter plots for latitude vs. maximum temperature, humidity, cloudiness, and wind speed. The first, latitude vs. maximum temperature, should look like the following scatter plot.



REWIND

Recall that to create a scatter plot, we collect then add x- and y-axis data to `plt.scatter()`.

Get Data for Plotting

First, we'll retrieve the data we need to create our scatter plots. We will need latitude, maximum temperature, humidity, cloudiness, and wind speed from all the cities. Add the following code to a new cell and run the cell.

```
# Extract relevant fields from the DataFrame for plotting.
lats = city_data_df["Lat"]
max_temps = city_data_df["Max Temp"]
humidity = city_data_df["Humidity"]
cloudiness = city_data_df["Cloudiness"]
wind_speed = city_data_df["Wind Speed"]
```

Your final scatter plot will need the current date in the title. To add the current date, we will need to import the `time` module, rather than the `datetime` module that we used to convert the date. The `time` module is a standard Python library, so there is no need to install it.

In a new cell in our `API_practice.ipynb` file, we will import the time module and some code to practice how to use this module. Add the following

code in a new cell and run the cell.

```
# Import the time module.
import time
# Get today's date in seconds.
today = time.time()
today
```

When we call the `time()` function with the time module, we get the output of today's time in seconds since January 1, 1970, as a floating-point decimal number.

```
# Get today's date in seconds.
today = time.time()
today
```

```
1565377180.5886168
```

NOTE

> Your time will be different when you run the code.

The format for time appears like the datetime stamp for the JSON weather data. We can convert this using the string format method, `strftime()` and pass the formatting parameters for our date in parentheses. To get the format for today, we can add `%x` inside the parentheses.

In the `API_practice` file, add `strftime("%x")` to the time module for our `today` variable and run the cell. The output will be today's date.

```
today = time.strftime("%x")
today
```

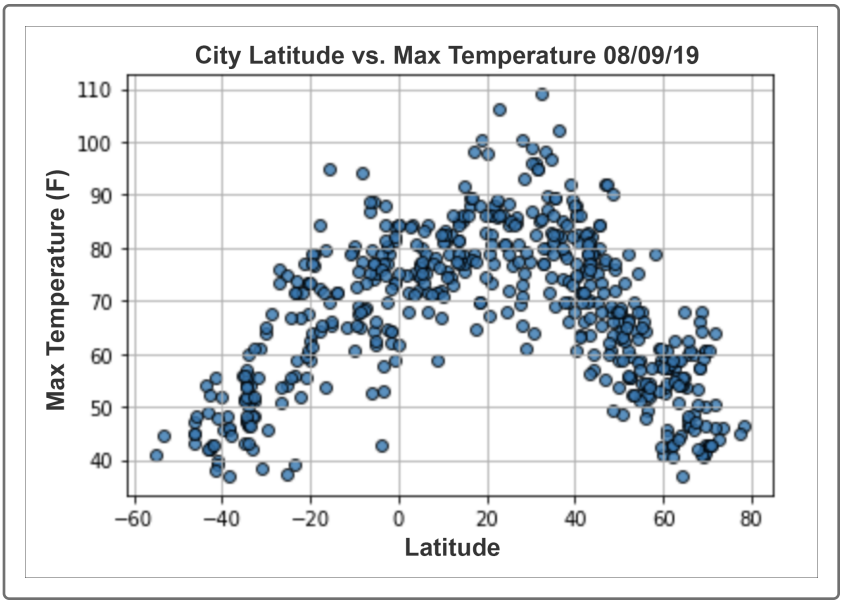
```
'08/09/19'
```

Now, we can add `time.strftime("%x")` to our `plt.title()` function in our scatter plot.

In a new cell in `WeatherPy`, add the following code to create a scatter plot for the latitude vs. maximum temperature and run the cell.

```
# Import time module
import time

# Build the scatter plot for latitude vs. max temperature.
plt.scatter(lats,
            max_temps,
            edgecolor="black", linewidths=1, marker="o",
            alpha=0.8, label="Cities")

# Incorporate the other graph properties.
plt.title(f"City Latitude vs. Max Temperature "+ time.strftime("%x"))
plt.ylabel("Max Temperature (F)")
plt.xlabel("Latitude")
plt.grid(True)

# Save the figure.
plt.savefig("weather_data/Fig1.png")

# Show plot.
plt.show()
```

Our scatter plot will look like the following.



The balance of the scatter plots will share the same format. All we need to do is change the y-axis variable for each weather parameter. Let's create the scatter plots quickly by copying the code and changing the y-axis variable.

NOTE

> For more information, see the **documentation on the time module (https://docs.python.org/3.6/library/time.html#functions)** .