Data-X Spring 2019: Homework 04

Name: Isha Mangal

SID: 3031911156

In this homework, you will do some exercises with plotting.

REMEMBER TO DISPLAY ALL OUTPUTS. If the question asks you to do something, make sure to print your results.

1.

Data:

Data Source: Data file is uploaded to bCourses and is named: **Energy.csv**

The dataset was created by Angeliki Xifara (Civil/Structural Engineer) and was processed by Athanasios Tsanas, Oxford Centre for Industrial and Applied Mathematics, University of Oxford, UK).

Data Description:

The dataset contains eight attributes of a building (or features, denoted by X1...X8) and response being the heating load on the building, y1.

- X1 Relative Compactness
- X2 Surface Area
- X3 Wall Area
- X4 Roof Area
- X5 Overall Height
- X6 Orientation
- X7 Glazing Area
- X8 Glazing Area Distribution
- y1 Heating Load

Q1.1

Read the data file in python. Check if there are any NaN values, and print the results.

In [72]:
```
# import needed libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [73]:
```
# load data
energy_data = pd.read_csv("Energy.csv")
```

```
In [74]: # check for any missing (NaN) values
         energy_data.isnull().values.any()

Out[74]: False
```

Q 1.2

Describe (using python function) data features in terms of type, distribution range (max and min), and mean values.

```
In [75]: energy_data.dtypes

Out[75]: X1    float64
         X2    float64
         X3    float64
         X4    float64
         X5    float64
         X6      int64
         X7    float64
         X8      int64
         Y1    float64
         dtype: object
```

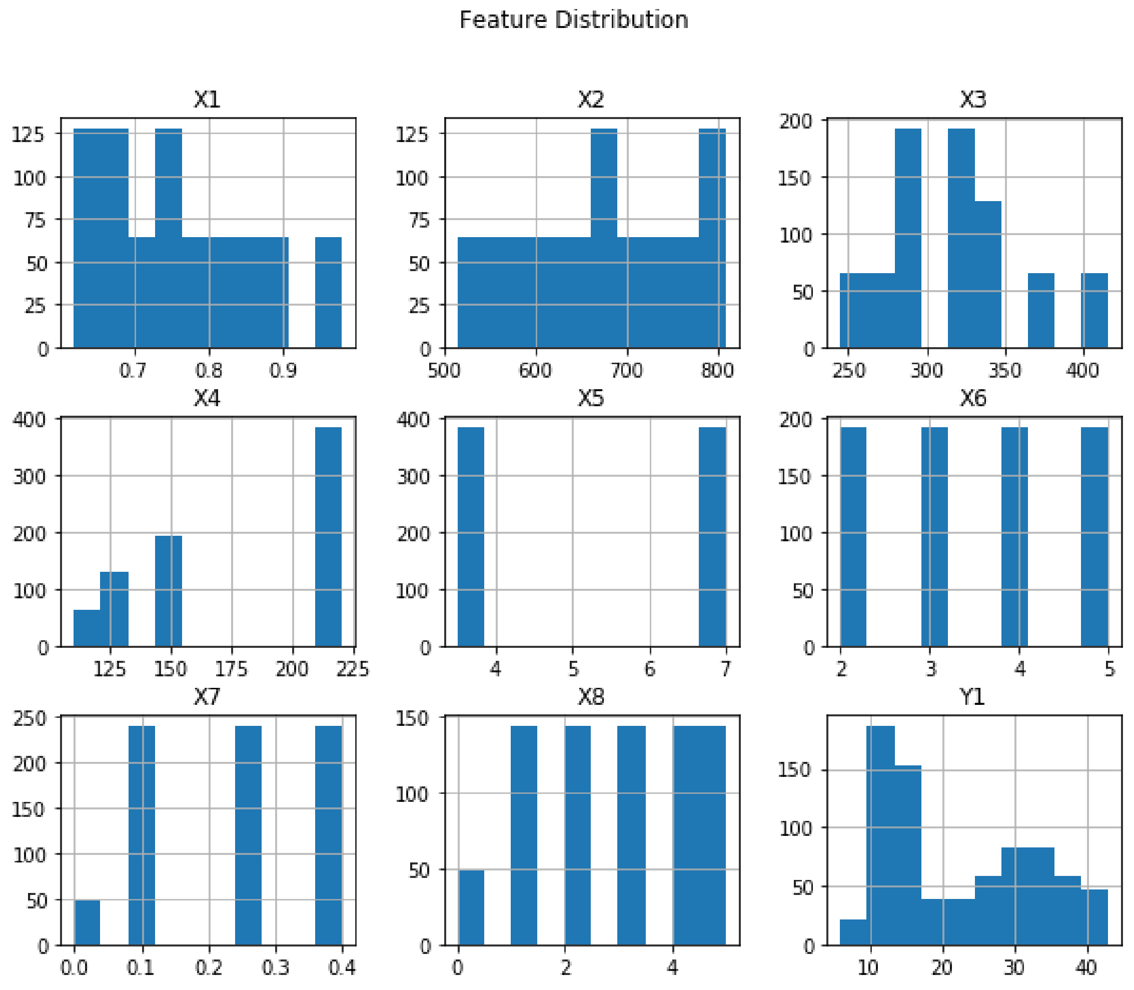
```
In [76]: energy_data.describe()

Out[76]:
```

	X1	X2	X3	X4	X5	X6	X7	X8	
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.00000	76
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.81250	2
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.55096	1
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000	0.00000	
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000	1.75000	1
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000	3.00000	1
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000	4.00000	3
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000	5.00000	4

Q 1.3

Plot feature distributions for all the attributes in the dataset (Hint - Histograms are one way to plot data distributions). This step should give you clues about data sufficiency.

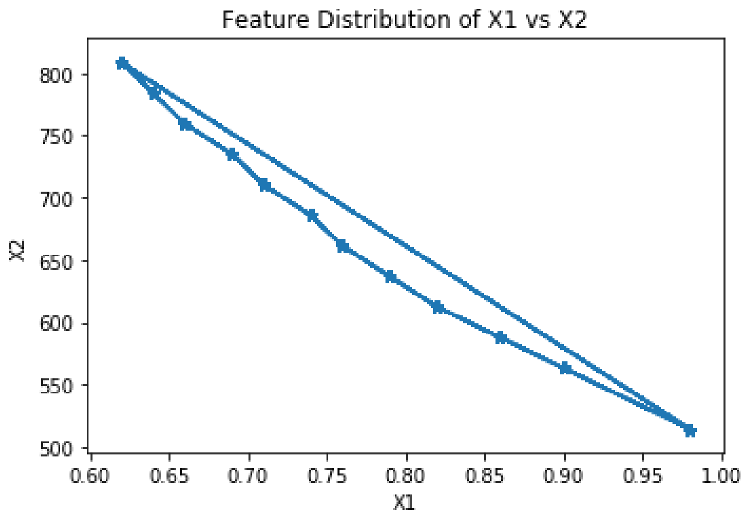
In [77]:
```
energy_data.hist(figsize=(10,8))
plt.suptitle("Feature Distribution")
plt.show()
```



Q1.4

Create a combined line and scatter plot for attributes 'X1' and 'X2' with a marker (*). You can choose either of the attributes as x & y. Label your axes and give a title to your plot.

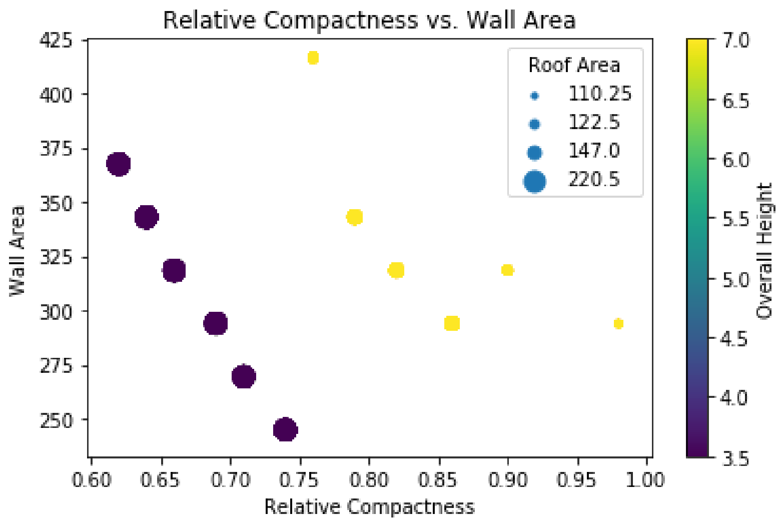
In [78]:
```
plt.plot(energy_data['X1'],energy_data['X2'])
plt.scatter(energy_data['X1'],energy_data['X2'],marker='*')
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Feature Distribution of X1 vs X2')
plt.show()
```



Q1.5

Create a scatter plot for how 'Wall Area' changes with 'Relative Compactness'. Give different colors for different 'Overall Height' and different bubble sizes by 'Roof Area'. Label the axes and give a title. Add a legend to your plot.

In [79]:
```
for size, group in energy_data.groupby("X4"):
    plt.scatter(group["X1"], group["X3"],c=group["X5"],
                s=group["X4"]-100,
                label = size,vmin=3.5,
                vmax=7)
plt.xlabel("Relative Compactness")
plt.colorbar(label="Overall Height")
plt.ylabel("Wall Area")
plt.title("Relative Compactness vs. Wall Area")
plt.tight_layout()
plt.legend(title = "Roof Area")
plt.show()
```



2.

Q 2.1a.

Create a dataframe called `icecream` that has column `Flavor` with entries `Strawberry`, `Vanilla`, and `Chocolate` and another column with `Price` with entries `3.50`, `3.00`, and `4.25`. Print the dataframe.

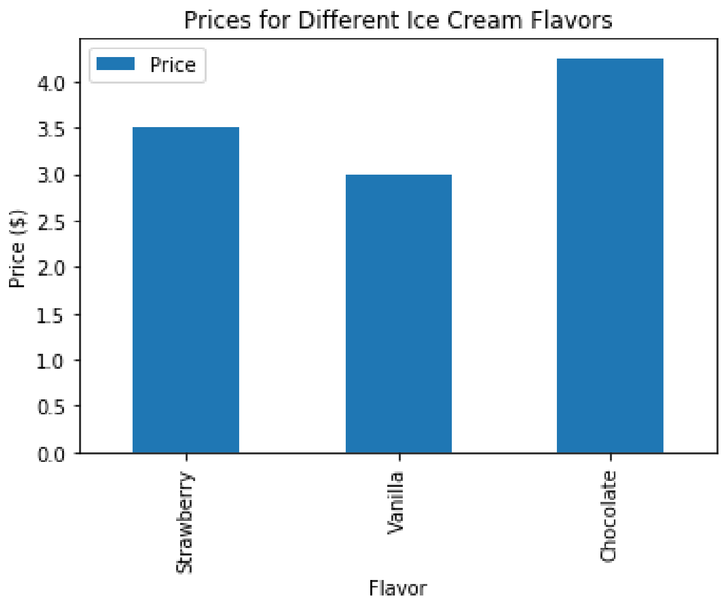
In [80]:
```
icecream = pd.DataFrame()
icecream['Flavor'] = ['Strawberry', 'Vanilla', 'Chocolate']
icecream['Price'] = [3.5, 3.00, 4.25]
print(icecream)
```

```
       Flavor  Price
0  Strawberry   3.50
1     Vanilla   3.00
2   Chocolate   4.25
```

Q 2.1b

Create a bar chart representing the three flavors and their associated prices. Label the axes and give a title.

In [81]:
```
icecream.plot.bar(x='Flavor')

plt.xlabel('Flavor')
plt.ylabel('Price ($)')
plt.title('Prices for Different Ice Cream Flavors')

plt.show()
```



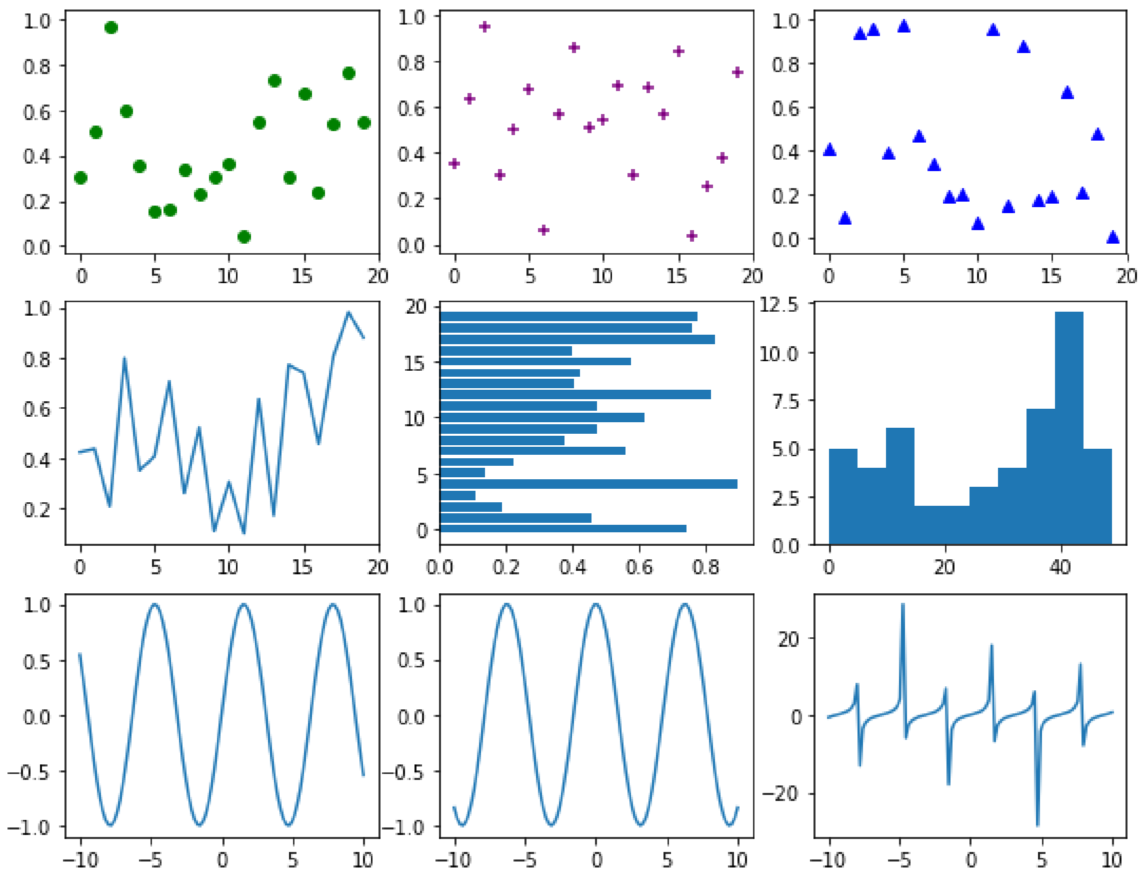
Q 2.2

Create 9 random plots in a figure (Hint: There is a numpy function for generating random data).

The top three should be scatter plots (one with green dots, one with purple crosses, and one with blue triangles. The middle three graphs should be a line graph, a horizontal bar chart, and a histogram. The bottom three graphs should be trignometric functions (one sin, one cosine, one tangent). Keep in mind the range and conditions for the trignometric functions.

All these plots should be on the same figure and not 9 independent figures.

In [82]:
```python
plots = plt.figure(figsize=(10,8))
for i in np.arange(1,6):
    points = 20
    x, y = np.arange(points), np.random.rand(points)
    plt.subplot(3,3,i)
    if i == 1:
        plt.scatter(x, y, c="g")
    elif i == 2:
        plt.scatter(x, y, c="purple", marker="+")
    elif i == 3:
        plt.scatter(x, y, c="b", marker="^")
    elif i == 4:
        plt.plot(x, y)
    elif i == 5:
        plt.barh(x, y)
for i in np.arange(6,10):
    plt.subplot(3,3,i)
    x = np.linspace(-10,10,100)
    if i == 6:
        plt.hist(np.random.choice(np.arange(50),50))
    elif i == 7:
        plt.plot(x, np.sin(x))
    elif i == 8:
        plt.plot(x, np.cos(x))
    elif i == 9:
        plt.plot(x, np.tan(x))
plt.show()
```

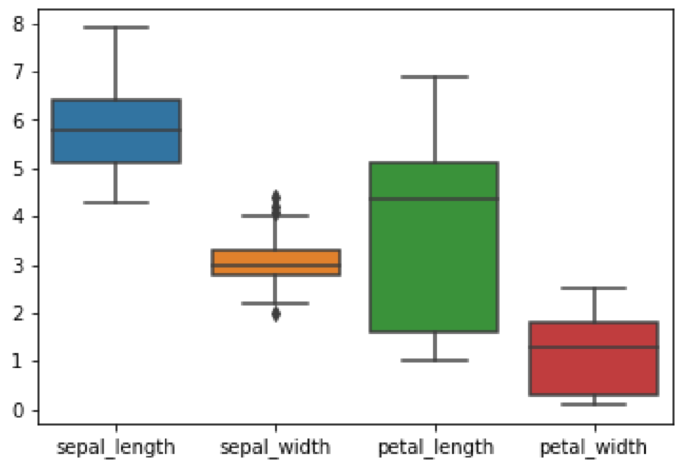


3.

Q 3.1

Load the 'Iris' dataset using seaborn. Create a box plot for the attributes 'sepal_length', sepal_width', 'petal_length' and 'petal_width' in the Iris dataset.

```
In [83]: iris_data = sns.load_dataset('iris')
         sns.boxplot(data=iris_data)
         plt.show()
```



Q 3.2

In a few sentences explain what can you interpret from the above box plot.

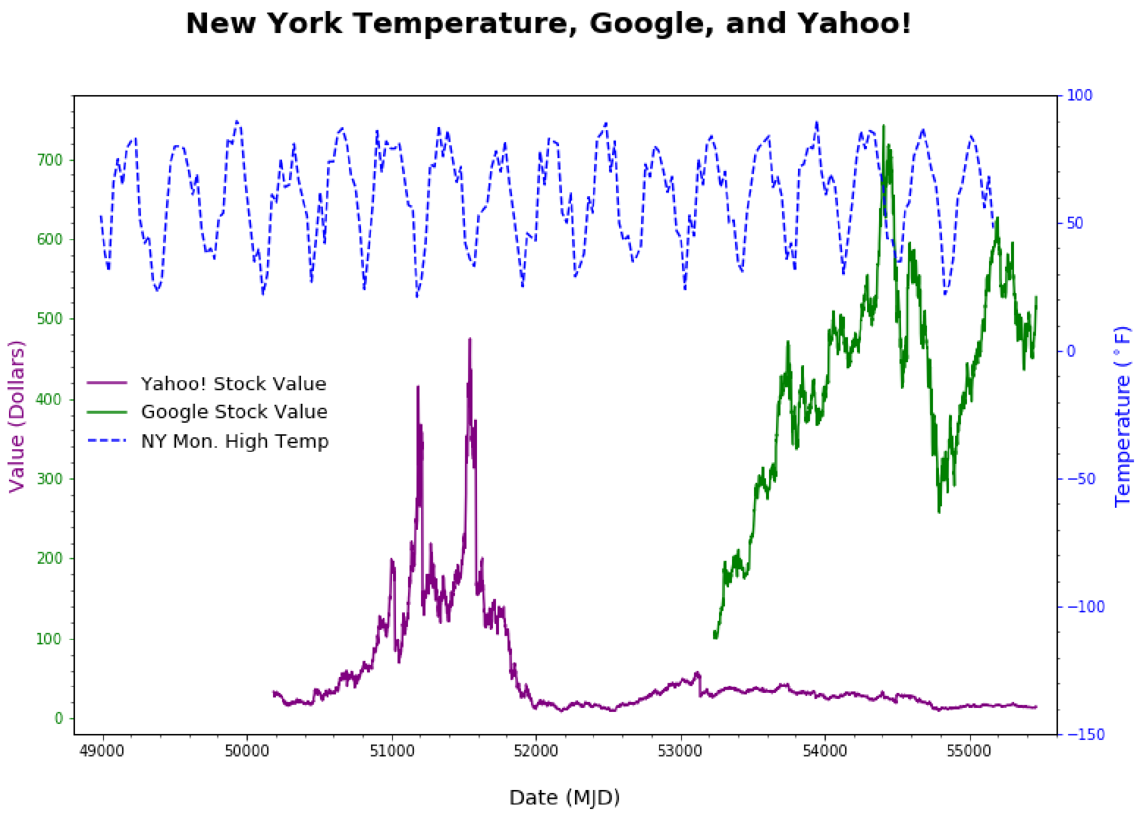
Based on the boxplot above, the data is relatively symmetrical for each attribute. However, for petal_length the data is skewed left and there appear to be a few outliers for petal_width. In comparison to all the attributes, petal_length has the largest range.

Q 4.

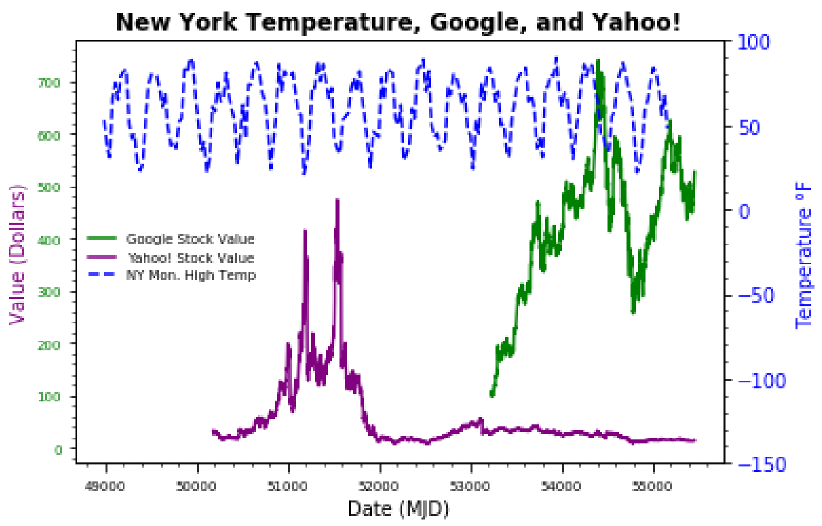
The data files needed:

`google_data.txt, ny_temps.txt & yahoo_data.txt`

Use your knowledge with `Python, NumPy, pandas and matplotlib` to reproduce the plot below:


New York Temperature, Google, and Yahoo!
Value (Dollars)
Temperature (° F)
Date (MJD)
Yahoo! Stock Value
Google Stock Value
NY Mon. High Temp
0
100
200
300
400
500
600
700
100
50
0
−50
−100
−150
49000
50000
51000
52000
53000
54000
55000

In [84]:
```python
google_data = pd.read_table("google_data.txt")
ny_data = pd.read_table("ny_temps.txt")
yahoo_data = pd.read_table("yahoo_data.txt")
fig, ax1 = plt.subplots()

ax1.plot(google_data["Modified Julian Date"],
         google_data["Stock Value"],
         c="g",
         label="Google Stock Value")
ax1.plot(yahoo_data["Modified Julian Date"],
         yahoo_data["Stock Value"],
         c="purple",
         label="Yahoo! Stock Value")
ax1.tick_params(axis='y', labelcolor="g",labelsize=7)

plt.xlabel("Date (MJD)")
plt.ylabel("Value (Dollars)", color="purple")
plt.tick_params(axis='x',labelsize=7, color="black", length=5)
plt.minorticks_on()
plt.grid(False)

ax2 = ax1.twinx()
ax2.set_ylim([-150,100])
ax2.plot(ny_data["Modified Julian Date"],
         ny_data["Max Temperature"],
         "--",
         c="b",
         label="NY Mon. High Temp")
ax2.tick_params(axis='y', labelcolor="blue")
ax2.grid(False)

fig.legend(loc = "center left",
           fontsize="x-small",
           frameon = False,
           bbox_to_anchor=(0.1, 0.5))

plt.title("New York Temperature, Google, and Yahoo!",fontweight='bold')
plt.tight_layout()
plt.minorticks_on()
plt.ylabel("Temperature °F", color="blue")

plt.show()
```



```
In [ ]:
```