LESSON 6 Plotting with Pandas Last Update Mar 3, 2020

To plot with pandas you need to import the matplot library

```
import matplotlib.pyplot as plt
```

pandas uses the **matplotlib** to do all its plotting

To plot with pandas you use the panda **plot** function on your data frame. The pandas plot function actually uses the matplot **plt.plot** function for plotting.

The pandas plot function just makes it much easier to plot by automatically by making the x and y labels and legends for you.

Plotting a DataFrame

It is easy to plot a data frame on pandas, just use the data frame name and call the **plot** function. Everything is done automatically for you.

For our plotting examples we will use the highest, normal and lowest temperature groups for a lake temperature for the year 2019 for the months January to December.

Lake Temperatures for year 2019

Month	Highest	Normal	Lowest
Jan	43	35	32
Feb	36	33	32
Mar	36	32	32
Apr	41	33	32
May	50	40	32
June	66	55	47
July	76	68	60
Aug	78	73	66
Sept	77	72	67
Oct	69	64	59
Nov	60	54	48
Dec	48	43	37

We have made a csv file called temperatures.csv

```
Month,Highest,Normal,Lowest
Jan,  43,   35,   32
Feb,  36,   33,   32
Mar,  36,   32,   32
Apr,  41,   33,   32
May,  50,   40,   32
June, 66,   55,   47
July, 76,   68,   60
Aug,  78,   73,   66
Sept, 77,   72,   67
Oct,  69,   64,   59
Nov,  60,   54,   48
Dec,  48,   43,   37
```

We can now read in the csv file and make a plot. We will use the month column names as an index column, this allows the month column names to be the x-axis.

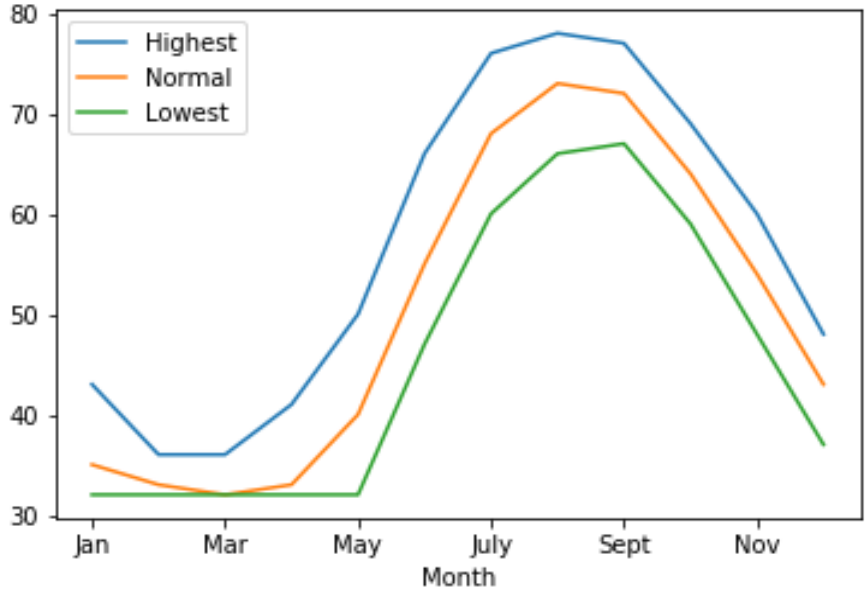
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("temperatures.csv",index_col="Month")
print(df)
```

Our data frame is:

```
       Highest  Normal  Lowest
Month
Jan         43      35       32
Feb         36      33       32
Mar         36      32       32
Apr         41      33       32
May         50      40       32
June        66      55       47
July        76      68       60
Aug         78      73       66
Sept        77      72       67
Oct         69      64       59
Nov         60      54       48
Dec         48      43       37
```

df.plot()
plt.show()

Our plot is:



Customizing pandas plot

We need to add more month ticks on the x axis, label the y axis, supply a title and make the plot wider.

To customize the plot we need to get the **Axes** object from the plot and set the size to a new width and height in inches using the **figsize** parameter. An **Axes** object contains most of the figure elements: Axis, Tick, Line2D, Text, Polygon, etc., and sets the coordinate system.

ax=df.plot(figsize=(10,5))

To add more x axis ticks we make a range of 12 ticks and the call the **set_xticks** functions from **Axes** object.

positions = range(12)

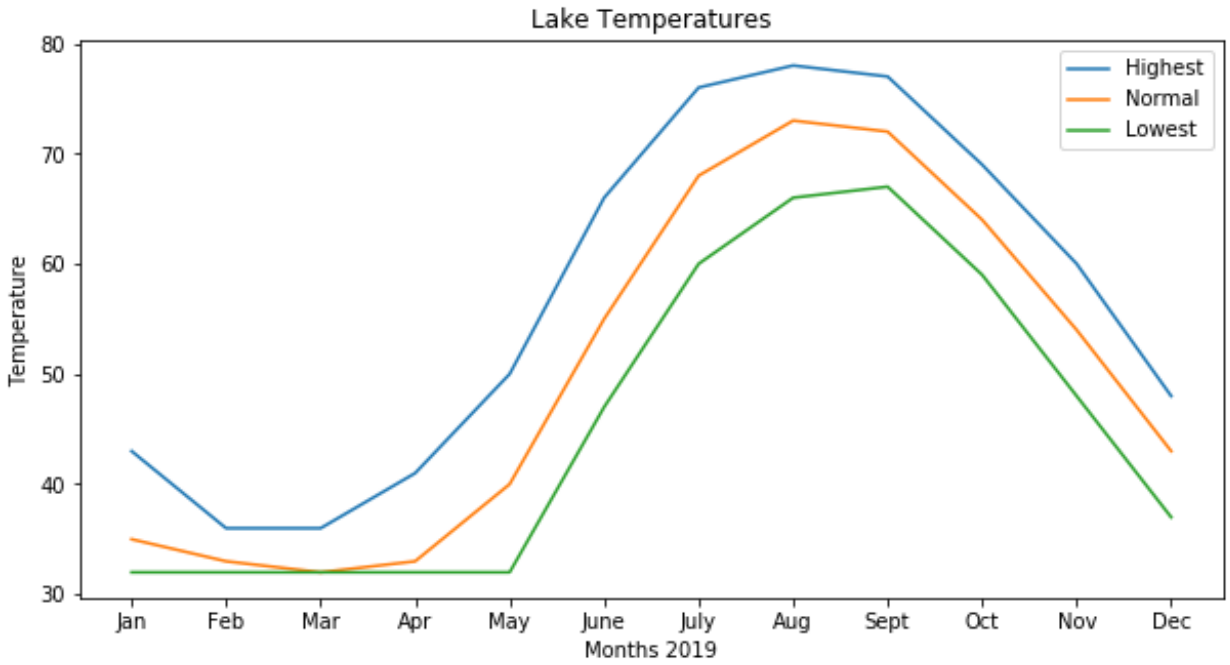
ax.set_xticks(positions)

We set the x labels to the index column of month names.

```
ax.set_xticklabels(df.index.tolist())
```

Finally we set the xlabes, y labels and title using the Axes object

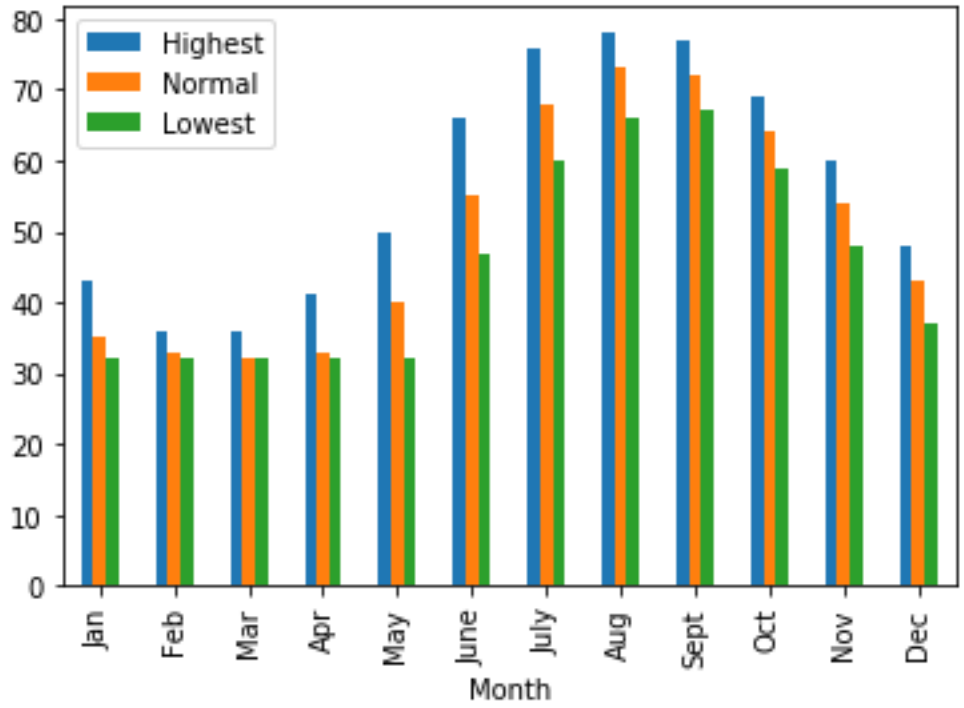
```
ax.set_xlabel('Months 2019')
ax.set_ylabel('Temperature')
ax.set_title("Lake Temperatures")

plt.show()
```



bar plot

The bar plot plots the temperatures for each month as a bar. Since we have 3 sets of temperatures, each temperature set is plotted right beside each other. To plot a bar chart we set **kind** to **'bar'** or use **plot.bar()** function.

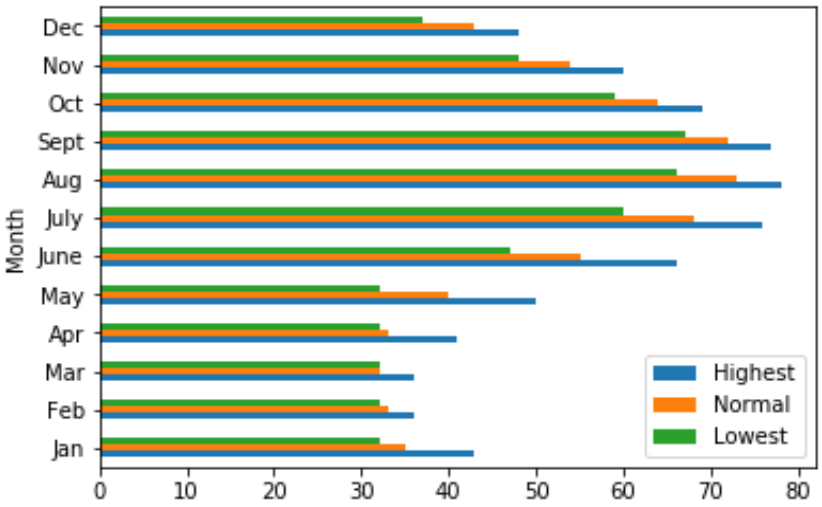
```
df.plot(kind="bar")
# df.plot.bar()
plt.show()
```



Horizontal bar chart

A horizontal bar chart plots all bars horizontally rather than vertically. In this case we set **kind** to **"barh"** or use **plot.bar()** function.

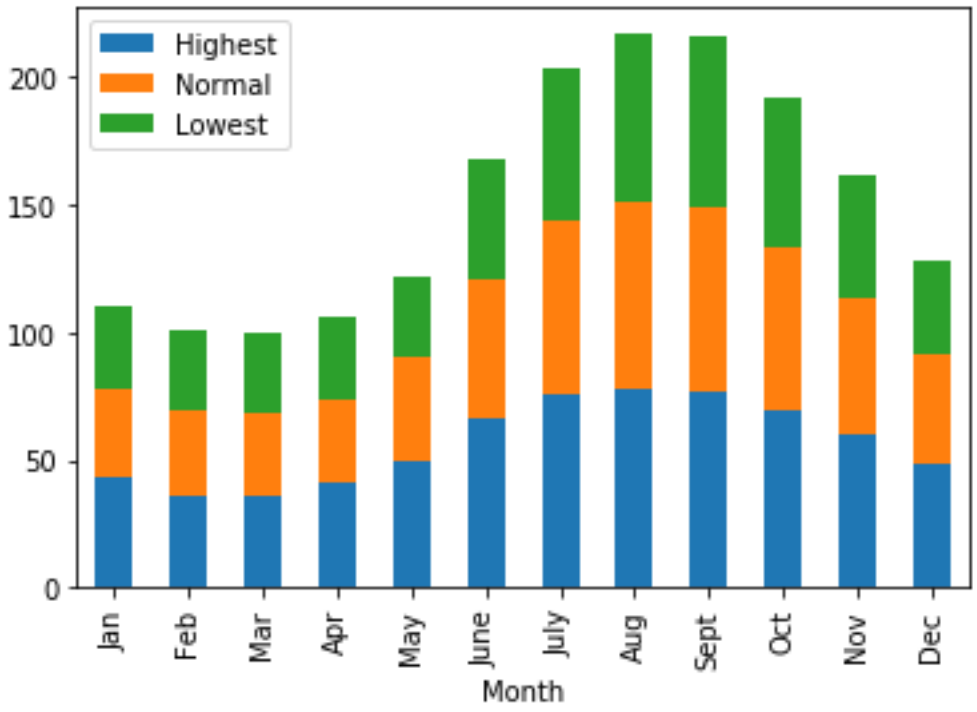
```
df.plot(kind="barh")
#df.plot.barh()
plt.show();
```



Stacked bar chart

A stacked bar chart superimposes bars on each other, so we have only one bar for each month. To distinguish the values for each temperature group, different colors are used, stacked on each other.

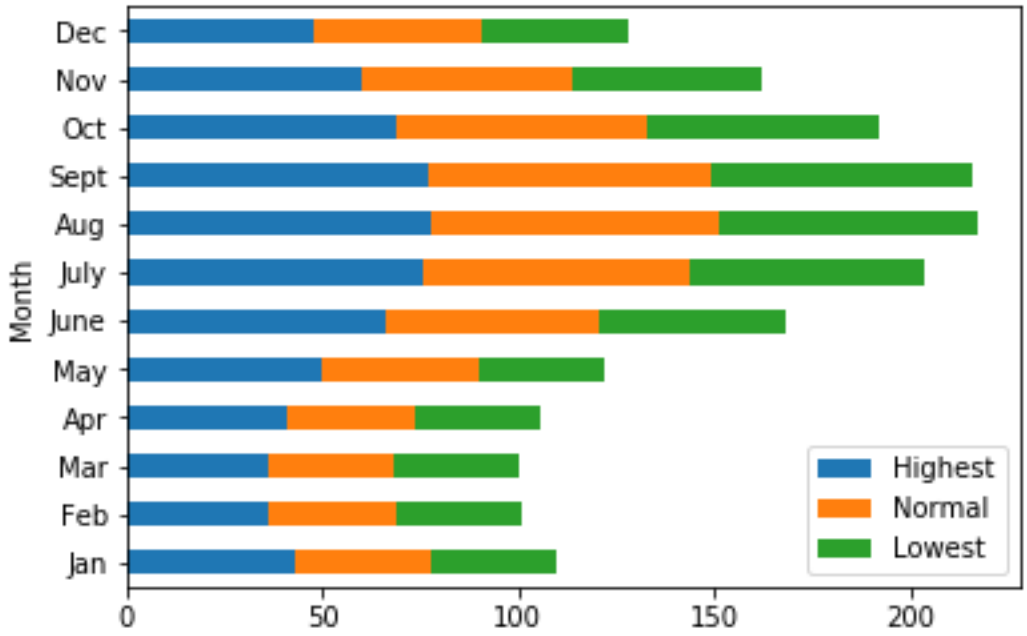
df.plot.bar(stacked=True);
plt.show()



Stacked horizontal bar chart

The stacked horizontal bar chart is identical the stacked bar chart but plotted horizontally rather than vertically.

df.plot.barh(stacked=True)
plt.show()



Histogram

A histogram is different from a bar chart, it is displaying values vs. the frequency. Frequency is how many temperature entry's there are for a specified certain degree. We may have 2 temperature of 50 degrees.

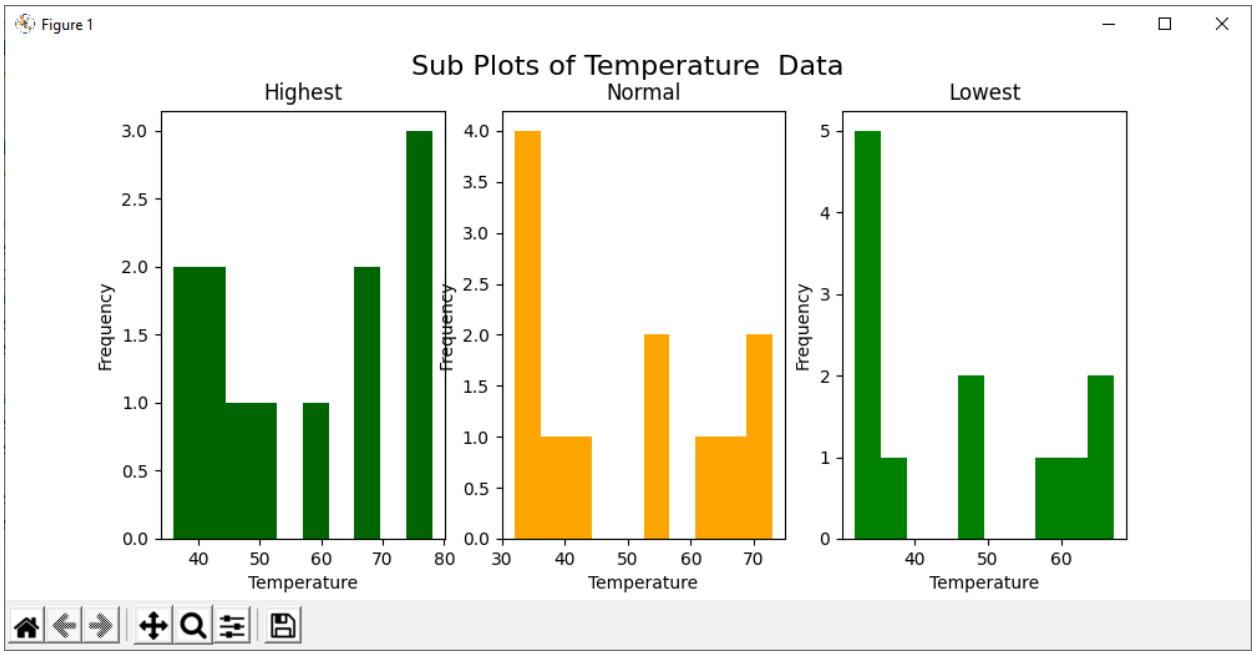
Since we have 3 different groups of temperatures we will make 3 different sub plots: Highest, Normal and Lowest. To make subplots we use the matplotlib **subplots** function. You must state how many rows and columns you want and the subplots will return a **Figure** and **Axes** object. The **Axes** object contains the methods for plotting, as well as most customization options, while the **Figure** object stores all of the figure-level attributes and allow the plot to output as an image. Every **Axes** object has a parent **Figure** object.

You can set the title of a subplot using the **Figure** object and **suptitle** function

fig.suptitle('Sub Plots of Temperature Data', fontsize=16)

The **subplot** function, returns 3 **Axes** object that we use with each subplot to specify the title and x axis label. We also give an axes object to each subplot as to specify column and size as parameters.

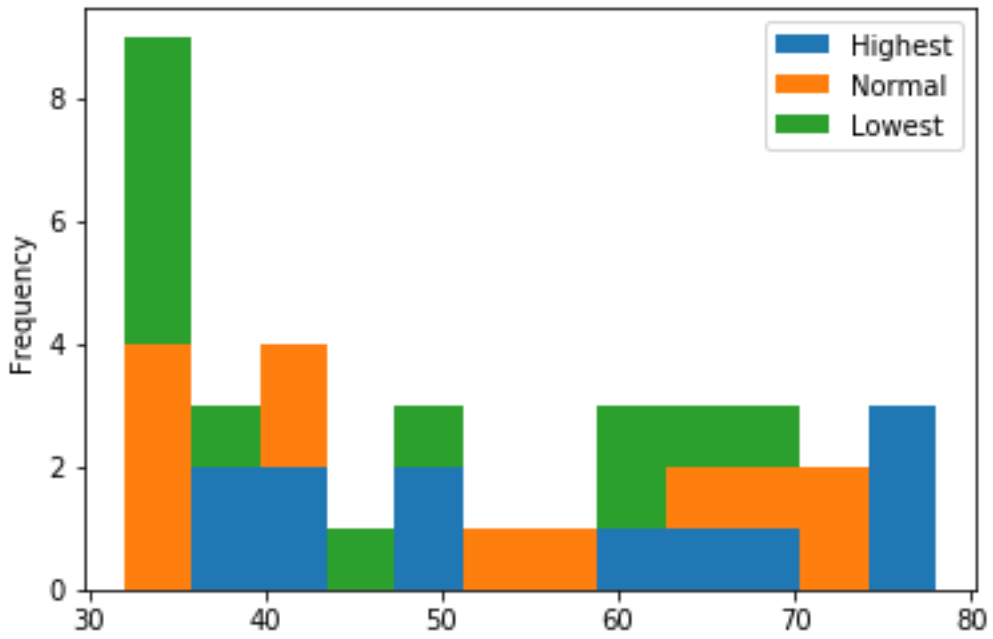
```
fig, axes = plt.subplots(nrows=1, ncols=3)
fig.suptitle('Sub Plots of Temperature  Data', fontsize=16)

df['Highest'].plot.hist(ax=axes[0], color='DarkGreen',figsize=(10,2))
axes[0].set_title('Highest')
axes[0].set_xlabel('Temperature')

df['Normal'].plot.hist(ax=axes[1],color='Orange',figsize=(10,2))
axes[1].set_title('Normal')
axes[1].set_xlabel('Temperature')

df['Lowest'].plot.hist(ax=axes[2], color='Green',figsize=(10,2))
axes[2].set_title('Lowest')
axes[2].set_xlabel('Temperature')
plt.show()
```



Stacked histogram

The stacked histogram will stack each histogram on each other, this can be used for direct comparisons. Every plot tells a story. The **bins** parameter specifies the width of the bins. If **bins** is too small then the bars are too narrow, if too large the bars are too wide. 12 seems to be a good number.

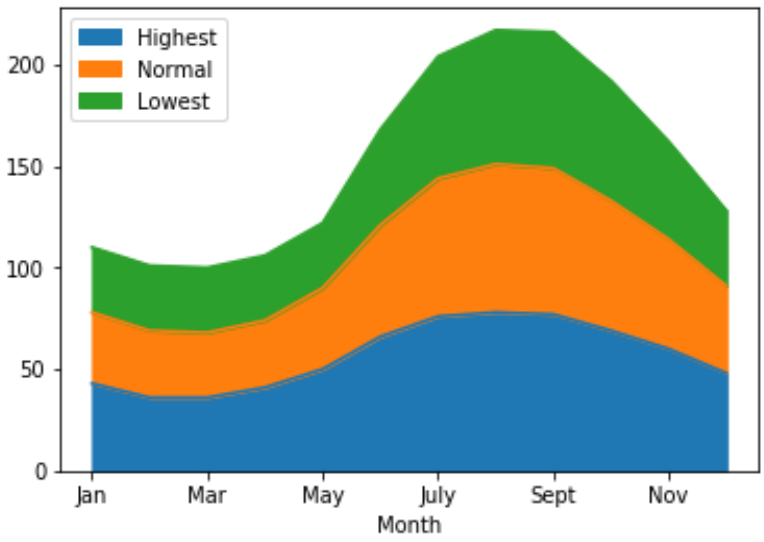
```
df.plot.hist(stacked=True, bins=12)
plt.show()
```



Area plot stacked

The **area plot** is very successful displaying the areas of each temperature group stacked upon each other.

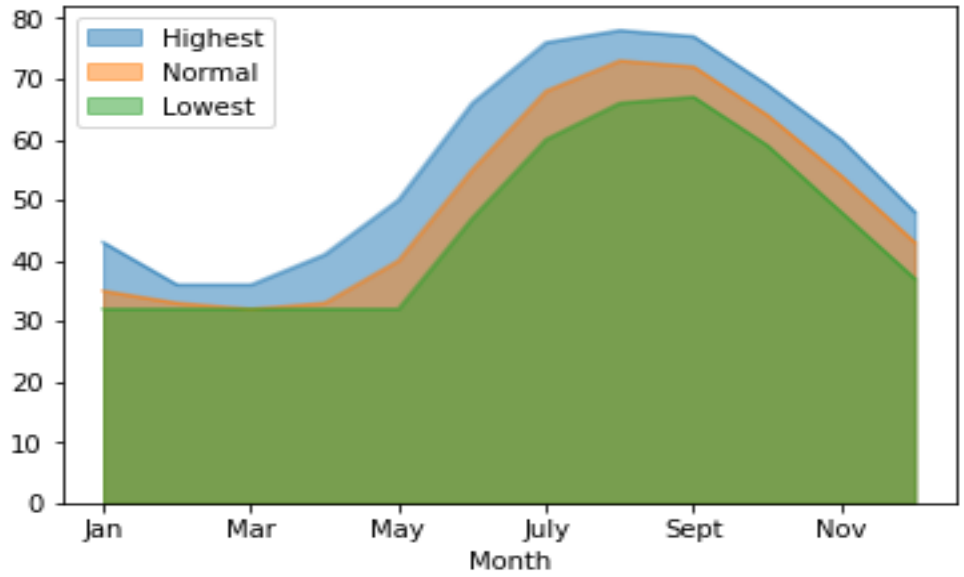
```
df.plot.area()
plt.show()
```



Area plot not stacked

In the not stacked area plot, each temperature group is superimposed on each other. This plot shows the dominance of the normal temperature group.

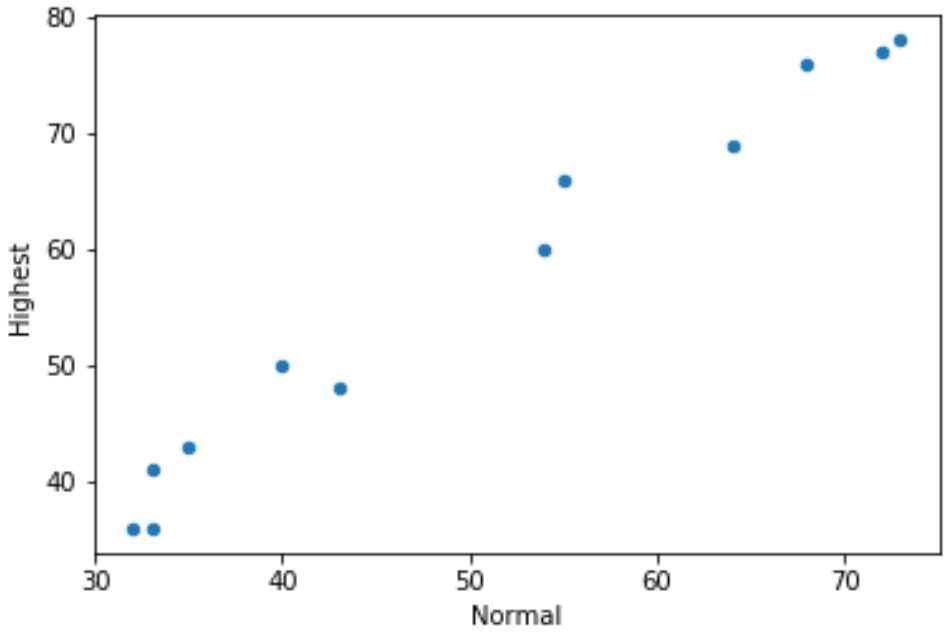
```
df.plot.area(stacked=False)
plt.show()
```



Scatter Plot

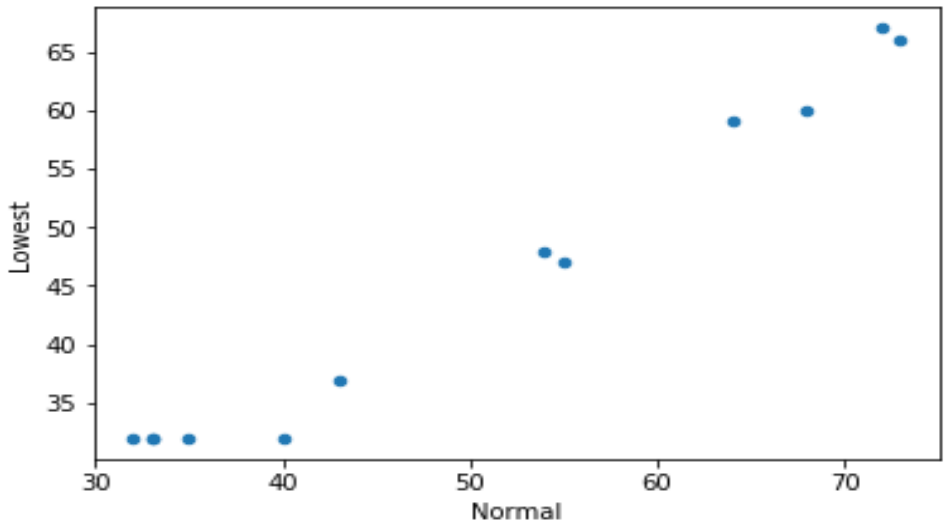
Scatter plots require numbers for x and y values. We can use the scatter plot to compare Highest temperatures to the Normal temperatures

```
df.plot.scatter(x='Normal', y='Highest');
plt.show()
```



We can do the same with the lowest temperature to normal temperature

```
df.plot.scatter(x='Normal', y='Lowest');
plt.show()
```



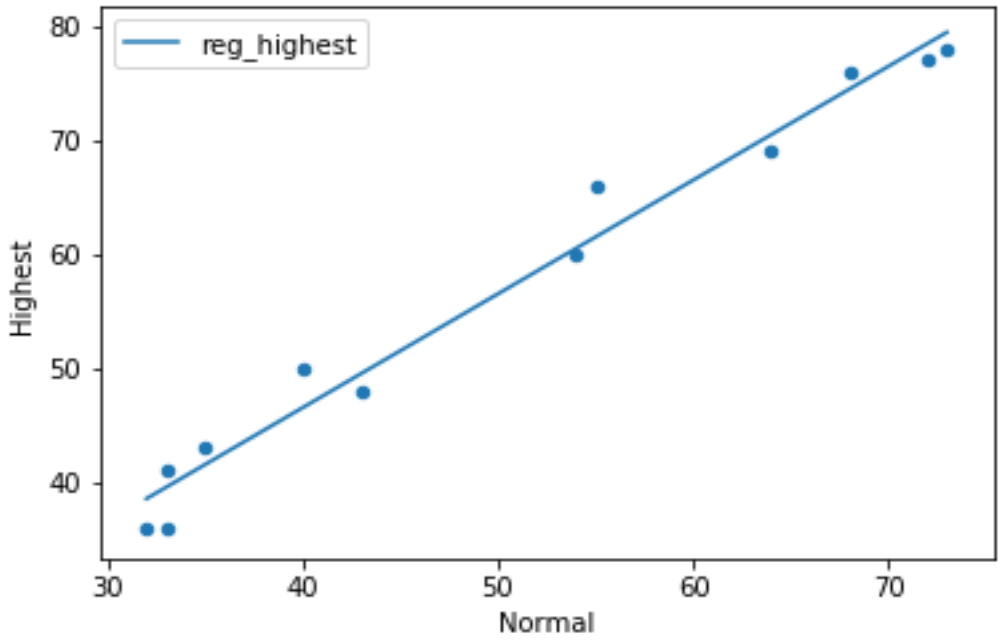
Plotting regression line on a scatter plot

We can also plot a regression line between these points. You cannot plot a regression directly with pandas but you can calculate the regression line slope and b intercept using the numpy **polyfit** function and make two independent columns in our data frame. One column for regression y points and corresponding x value. To use numpy module library you need to import it first

```
import numpy as np

x = df['Normal']
y = df['Highest']
m, b = np.polyfit(x, y, 1)
df['reg_highest'] = sorted(m * x + b)
df['x'] = sorted(x)
```

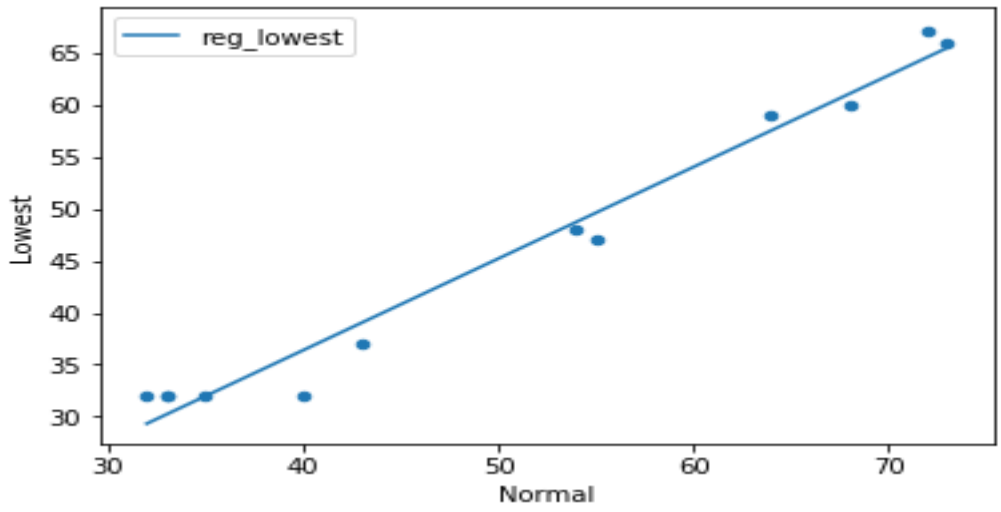
We then plot the regression line along with the scatter plot.

```
ax = df.plot(x='x',y='reg_highest')
df.plot.scatter(x='Normal', y='Highest',ax=ax);
plt.show()
```



We can do the same with lowest temperature to normal

```
x = df['Normal']
y = df['Lowest']
m, b = np.polyfit(x, y, 1)
df['reg_lowest'] = sorted(m * x + b)
df['x'] = sorted(x)
ax = df.plot(x='x',y='reg_lowest')
df.plot.scatter(x='Normal', y='Lowest',ax=ax);
plt.show()
```

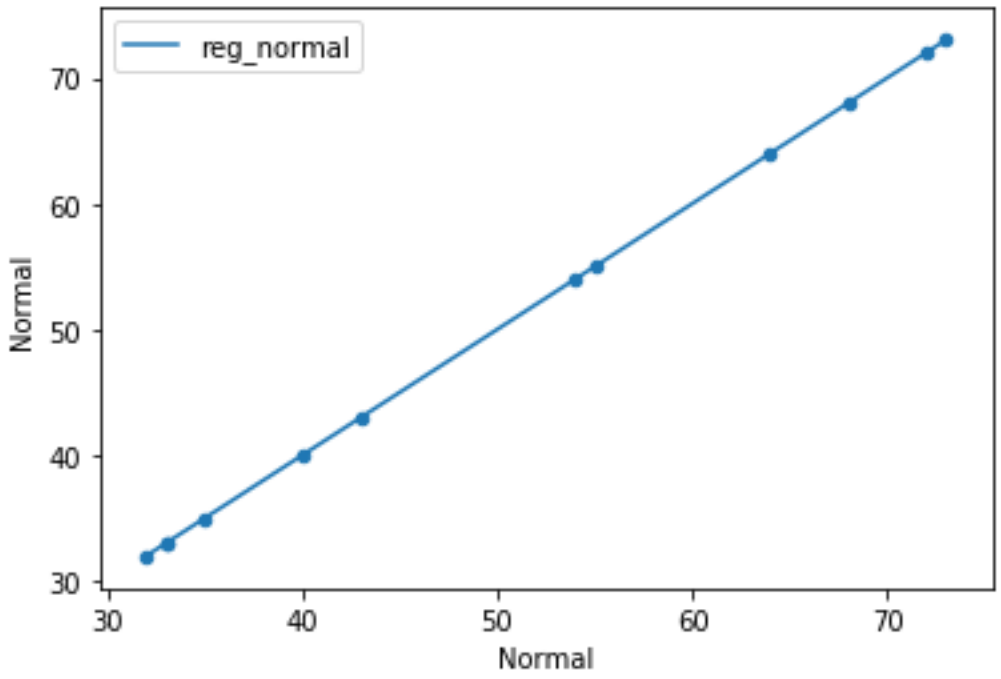


We print out our data frame with the new regression line columns added.

Month	Highest	Normal	Lowest	reg_highest	x	reg_lowest
Jan	43	35	32	38.557375	32	29.323547
Feb	36	33	32	39.554217	33	30.204819
Mar	36	32	32	39.554217	33	30.204819
Apr	41	33	32	41.547900	35	31.967365
May	50	40	32	46.532109	40	36.373728
June	66	55	47	49.522634	43	39.017546
July	76	68	60	60.487893	54	48.711545
Aug	78	73	66	61.484735	55	49.592818
Sept	77	72	67	70.456311	64	57.524272
Oct	69	64	59	74.443678	68	61.049362
Nov	60	54	48	78.431045	72	64.574453
Dec	48	43	37	79.427886	73	65.455726

To do:

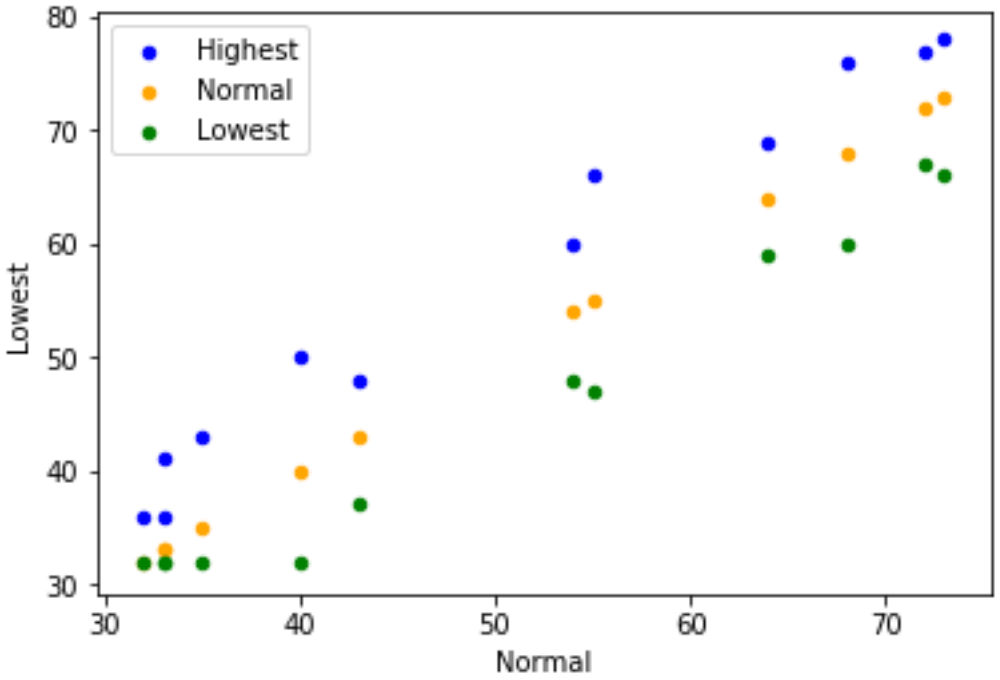
Plot a regression line through a Normal scatter plot. You should get something like this:



Multiple scatter plot

Here we are plotting scatter plots together. We can use different color points or different size points to distinguish temperature group.

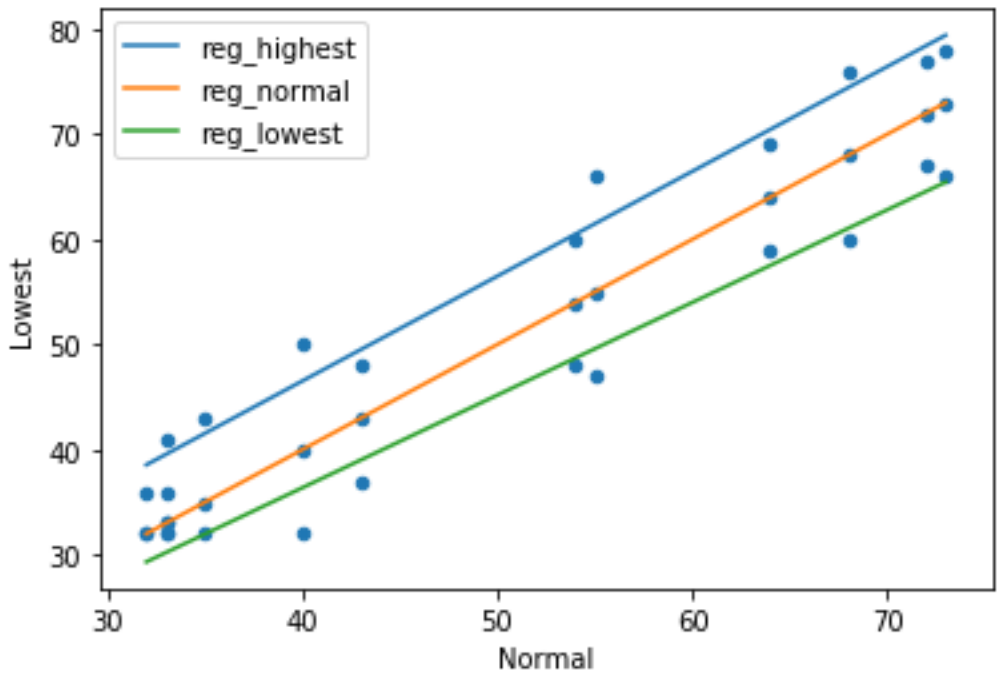
```
ax = df.plot.scatter(x='Normal', y='Highest', color='Blue', label='Highest');
df.plot.scatter(x='Normal', y='Normal', color='Orange', label='Normal', ax=ax);
df.plot.scatter(x='Normal', y='Lowest', color='Green', label='Lowest', ax=ax);
ax.set_xlabel('Temperature')
ax.set_ylabel('Temperature')
plt.show()
```



PLOTTING WITH PANDAS HOMEWORK

Question 1

Plot 3 regression line for each of the temperature groups: highest, normal and lowest. You should get something like this:



PIE Chart

A pie chart shows percentages shaped as a pie, where each slice of the pie is a percentage.

For our temperature example we will group together the following temperature ranges:

>=70
60-70
50-60
40-50
<40

We will just use the Normal Temperature group for our pie chart.

We will use filters to get the count of temperatures per range

```
t70 = len(df[df['Normal']>=70])
t60 = (df['Normal']>=60) & (df['Normal']<70)
t60 = len((df[t60]))
t50 = (df['Normal']>=50) & (df['Normal']<60)
t50 = len(df[t50])
t40 = (df['Normal']>=40) & (df['Normal']<50)
t40 = len(df[t40])
t30 = len(df[df['Normal']<40])
```

we then put in a list where **ty** are the counts of each category

```
ty = [t70,t60,t50,t40,t30]
```

we then make a list of labels for display

```
labels = ['>=70','60-70','50-60','40-50','<40']
```

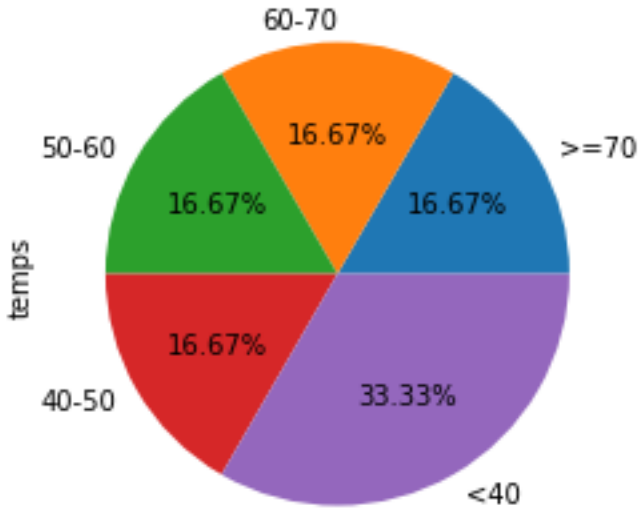
we then make a data frame from the temperature lengths and labels. Our labels become our index and then set columns name to 'temps'

```
df2 = pd.DataFrame(ty,index=labels)
df2.columns=['temps']
print(df2)
```

```
       temps
>=70       2
60-70      2
50-60      2
40-50      2
<40        4
```

From here we can plot the pie chart from the temperature length series. We use **autopct** parameter formatted to display the percentages on the pie chart.

```
df2['temps'].plot.pie(y=ty,autopct='%.2f%%')
plt.show()
```



PLOTTING WITH PANDAS HOMEWORK

Question 2

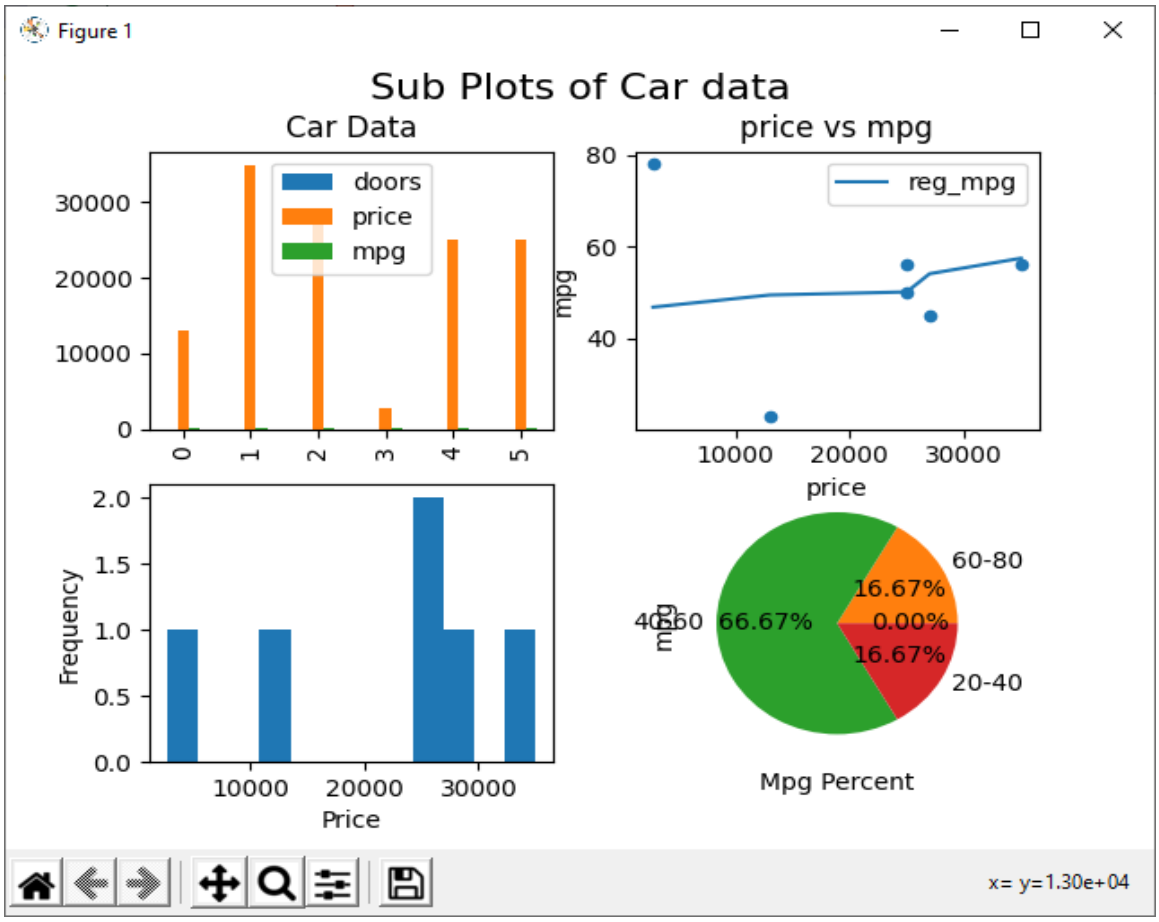
Make a data frame with 3 to 4 columns represent some data that interests you. You can load the dataframe from a csv file, a dictionary or from lists.

An example would be stock prices high low and close for the month or year, unemployment rate or the prices of homes etc.

Make 4 subplots. Put a bar chart into one, a scatter plot with a regression line (s) into another, a histogram in the third one and a pie chart of the percents of some column in the fourth one. Note: a 2 row by 2 column plot returns a 2 dimensional axes array.

Call your homework python file plotpandas_homework.py

You should get something like this:



END