Visualization Lab

To really get a good understanding of the data contained in your DataFrame, it is often essential to create plots. Pandas is highly integrated with the plotting library matplotlib, and makes plotting DataFrames very user-friendly! Plotting in a notebook environment usually makes use of the following boilerplate:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
```

matplotlib is the plotting library which pandas' plotting functionality is built upon, and it is usually aliased to `plt`.

`%matplotlib inline` tells the notebook to show plots inline, instead of creating them in a separate window.

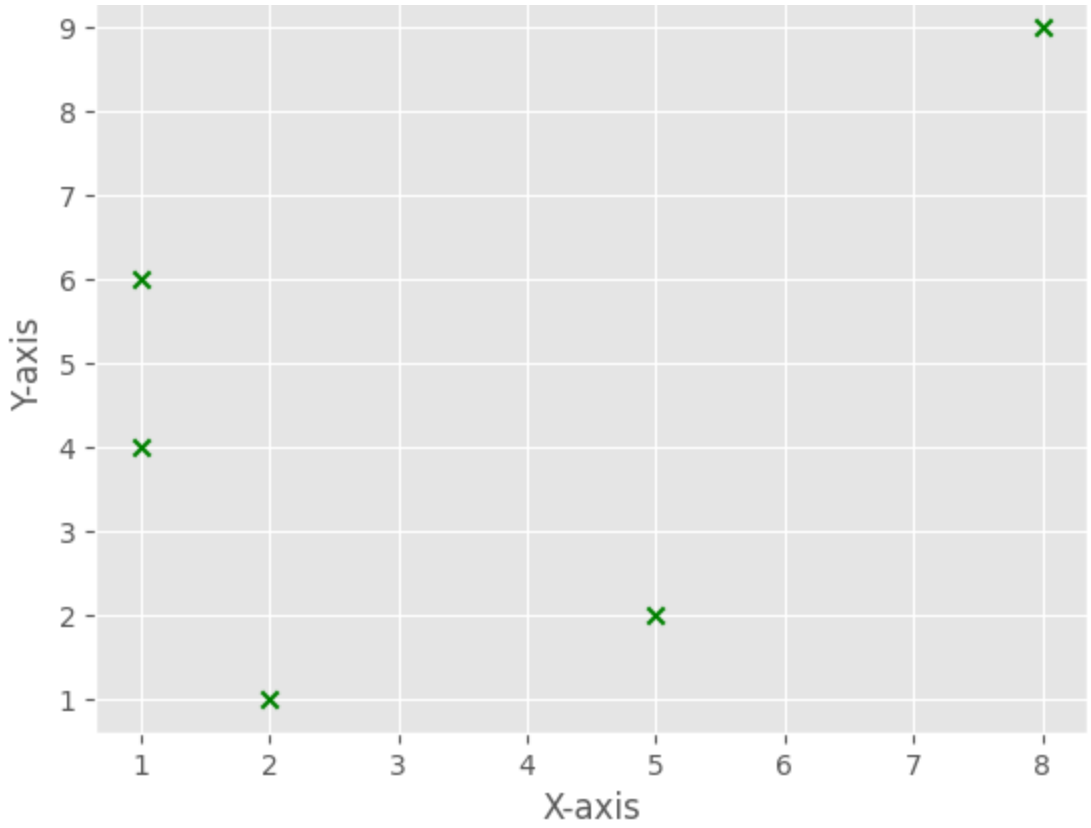
`plt.style.use('ggplot')` is a style theme that most people find agreeable, based upon the styling of R's ggplot package.

1. make a scatter plot of this random data, but use green X's instead of the default markers.

```
df = pd.DataFrame({"xs":[1,5,2,8,1], "ys":[4,2,1,9,6]})
```

```
####Your code here ########
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')

df = pd.DataFrame({"xs":[1,5,2,8,1], "ys":[4,2,1,9,6]})

plt.scatter(df['xs'], df['ys'], marker='x', color='green')

#Setting labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')

#Displaying the plot
plt.show()
```

✓ 1s completed at 4:18 PM



2. Columns in your DataFrame can also be used to modify colors and sizes. Bill has been keeping track of his performance at work over time, as well as how good he was feeling that day, and whether he had a cup of coffee in the morning. Make a plot which incorporates all four features of this DataFrame.

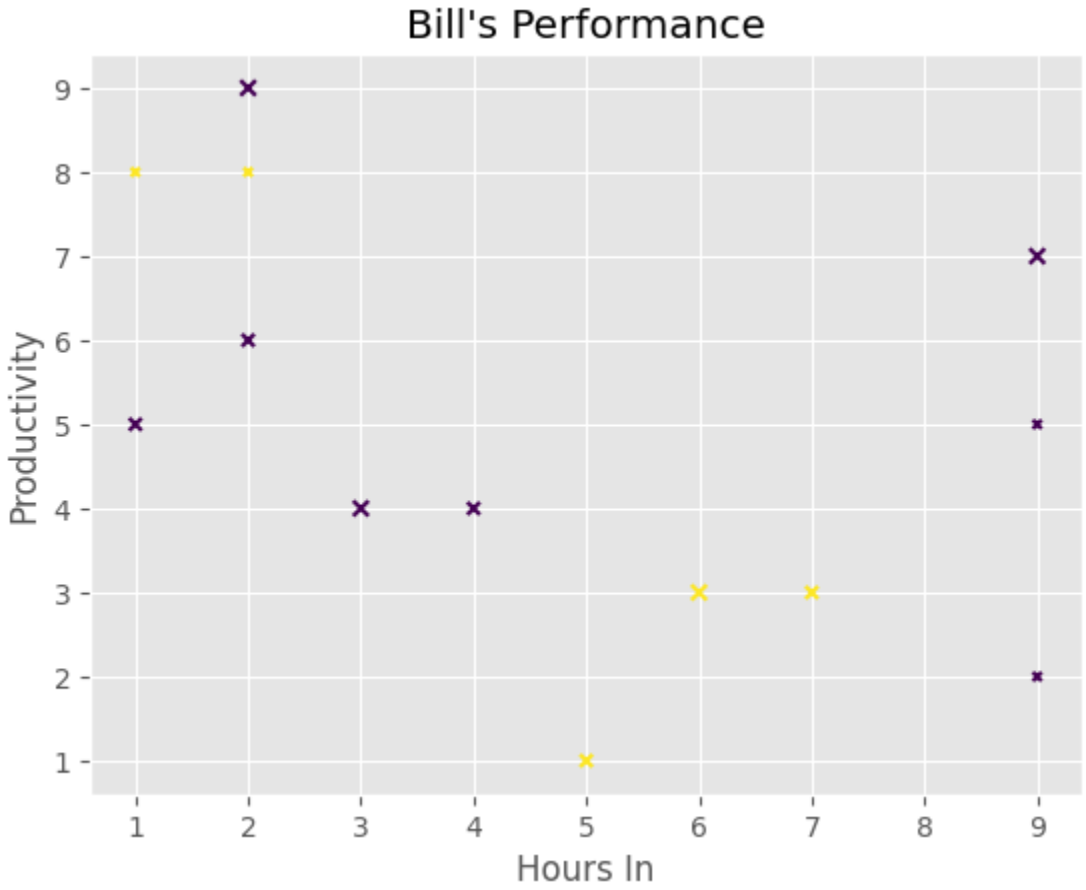
(Hint: If you're having trouble seeing the plot, try multiplying the Series which you choose to represent size by 10 or more)

The chart doesn't have to be pretty: this isn't a course in data viz!

```
 df = pd.DataFrame({"productivity":[5,2,3,1,4,5,6,7,8,3,4,8,9],
                    "hours_in"    :[1,9,6,5,3,9,2,9,1,7,4,2,2],
                    "happiness"   :[2,1,3,2,3,1,2,3,1,2,2,1,3],
                    "caffienated" :[0,0,1,1,0,0,0,0,1,1,0,1,0]})
```

```
#######Your code here ########
df = pd.DataFrame({"productivity":[5,2,3,1,4,5,6,7,8,3,4,8,9],
                   "hours_in"    :[1,9,6,5,3,9,2,9,1,7,4,2,2],
                   "happiness"   :[2,1,3,2,3,1,2,3,1,2,2,1,3],
                   "caffienated" :[0,0,1,1,0,0,0,0,1,1,0,1,0]})
```

```
plt.scatter(df['hours_in'],
            df['productivity'],
            s=df['happiness']*10,
            c=df['caffienated'],
            marker='x')

#Setting labels and title
plt.xlabel('Hours In')
plt.ylabel('Productivity')
plt.title('Bill\'s Performance')

#Displaying the plot
plt.show()
```



3. What if we want to plot multiple things? Pandas allows you to pass in a matplotlib *Axis* object for plots, and plots will also return an Axis object.

Make a bar plot of monthly revenue with a line plot of monthly advertising spending (numbers in millions)

```
df = pd.DataFrame({"revenue":[57,68,63,71,72,90,80,62,59,51,47,52],
                   "advertising":[2.1,1.9,2.7,3.0,3.6,3.2,2.7,2.4,1.8,1.6,1.3,1.9],
                   "month":range(12)
```

```
                    })


#######Your code here
df = pd.DataFrame({"revenue":[57,68,63,71,72,90,80,62,59,51,47,52],
                   "advertising":[2.1,1.9,2.7,3.0,3.6,3.2,2.7,2.4,1.8,1.6,1.3,1.9],
                   "month":range(12)
                  })

fig, ax = plt.subplots()

#Monthly revenue bar plot
df.plot(x='month',
        y='revenue',
        kind='bar',
        ax=ax,
        color='blue',
        alpha=0.7,
        legend=False)

#Monthly advertising spending line plot
df.plot(x='month',
        y='advertising',
        kind='line',
        ax=ax,
        marker='o',
        color='red',
        legend=False)

#Setting labels and title
plt.xlabel('Month')
plt.ylabel('Amount (Millions)')
plt.title('Monthly Revenue and Advertising Spending')

#Displaying the plot
plt.show()
```