SAMSUNG

Together for Tomorrow!
Enabling People
Education for Future Generations

Samsung Innovation Campus

Artificial Intelligence Course

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 3.

Python Libraries

AI Course

Samsung Innovation Campus

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 3.

Python Libraries

UNIT 3.

Visualization

Samsung Innovation Campus

Unit 3. Visualization

What this unit is about:

- This unit is about visualization techniques.
- You will learn when to use the different visualization types.
- You will learn how to create basic visualization types using libraries.
- You will learn how to create more advanced visualization types using libraries.

Expected outcome:

- Ability to select the proper visualization type.
- Ability to express data as a visualization to communicate and enhance the understanding.
- Ability to aid the exploratory data analysis (EDA) with visualization.

How to check your progress:

- Coding Exercises.
- Quiz.

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 3.

Python Libraries

UNIT 1. NumPy Package

1.1. NumPy array basics.

1.2. NumPy array operations.

1.3. Linear algebra: vectors and matrices.

UNIT 2. Pandas Package

2.1. Pandas Series and DataFrame.

2.2. Data summarization and manipulation.

UNIT 3. Visualization

3.1. Introduction to visualization.

3.2. Matplotlib and Pandas visualization.

3.3. Seaborn visualization.

Visualization Principles

Reasons for applying visualization:

- To describe statistical properties of data.
- To show structural relations, correlations, etc. that may exist in the data.
- To summarize large amount of data.
- To compare different theories and hypotheses.
- To validate and support the analysis.
- To communicate our findings to others.

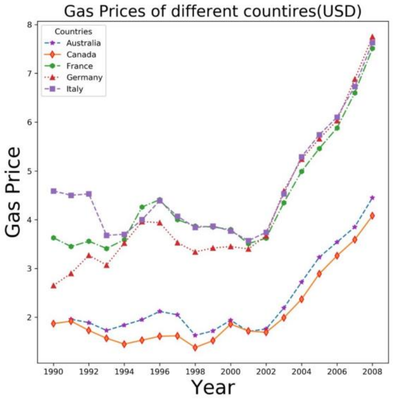
Visualization and EDA

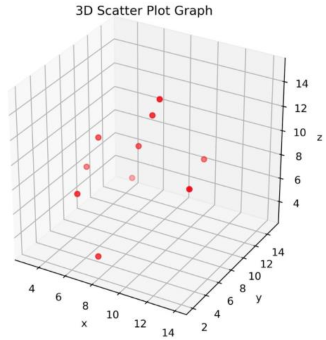
Visualization in Exploratory Data Analysis (EDA):

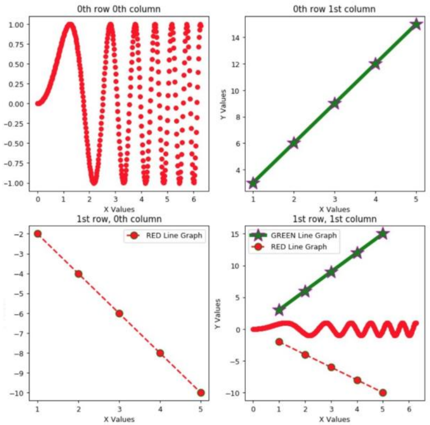
- Visualization is particularly useful for EDA.
- Visualization helps to determine future course of actions.
- You may need to make many “low-quality” graphics to enhance our own understanding of the data.
- You are the end consumers, thus decorative elements are kept to the minimum.

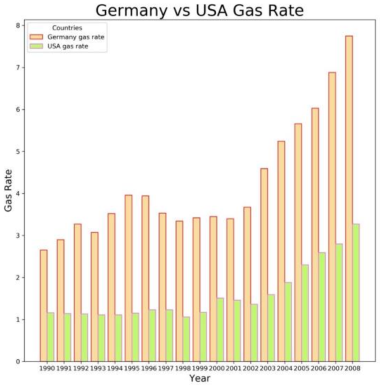
Basic Visualization Types

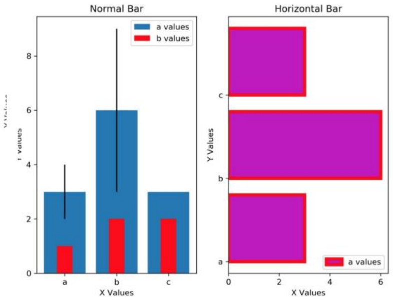
- Matplotlib Plots:

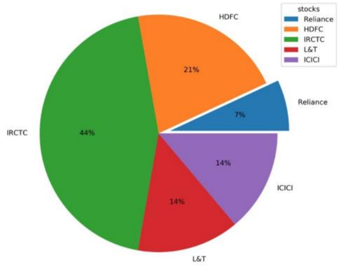








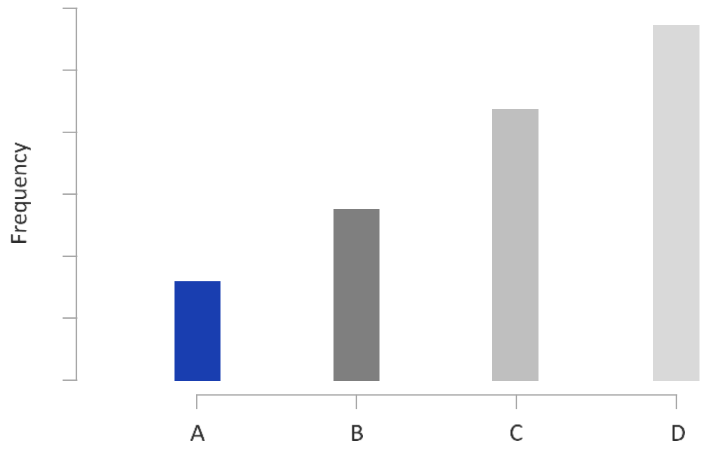




Basic Visualization Types (4/13)

Univariate visualization:

- One categorical variable: **Bar plot**.

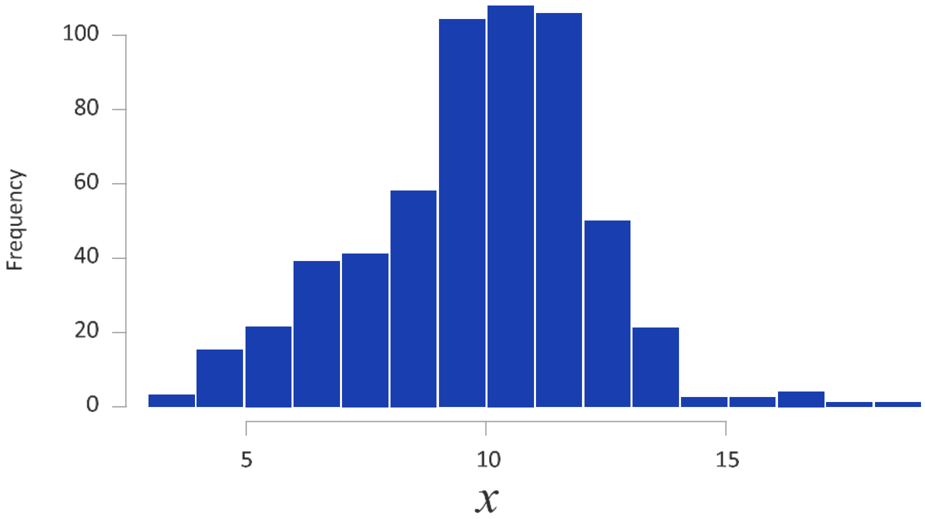


- Shows the absolute or relative frequencies of each category (type).

Basic Visualization Types (2/13)

Univariate visualization:

- One continuous numeric variable: **Histogram**.

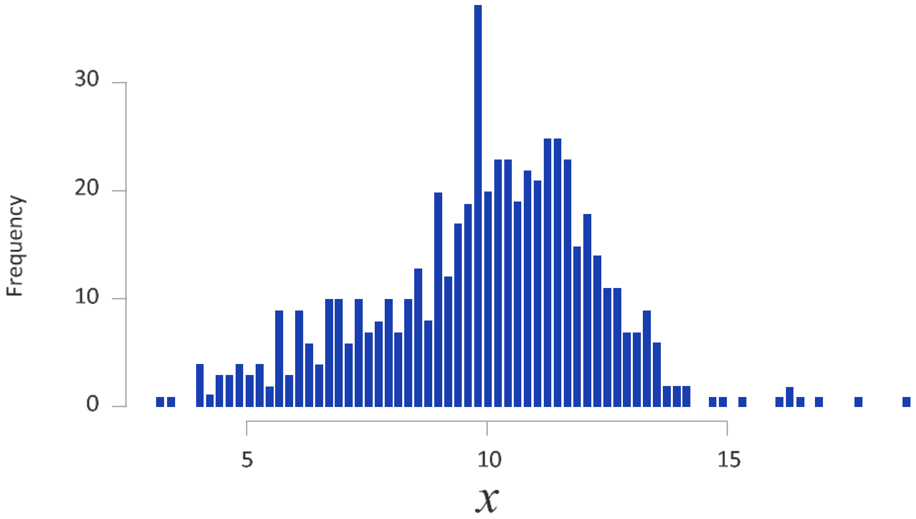


- Shows the absolute or relative frequencies of each interval.

Basic Visualization Types (3/13)

Univariate visualization:

- One continuous numeric variable: **Histogram**.

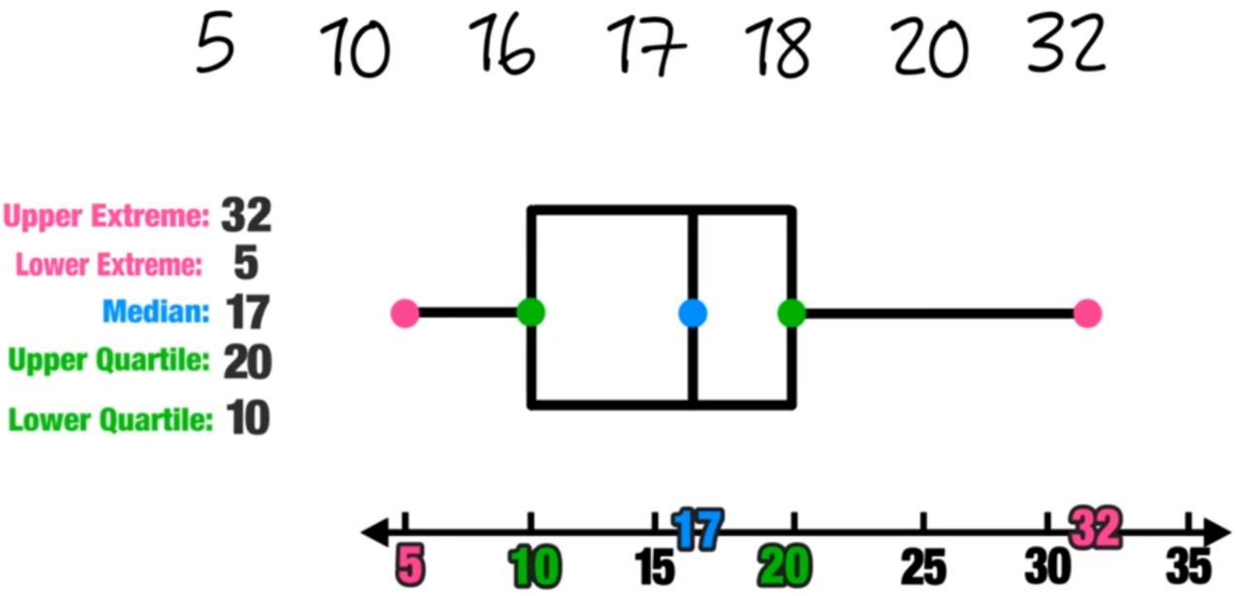


- The interval width can be adjusted.

Basic Visualization Types (1/13)

Univariate visualization:

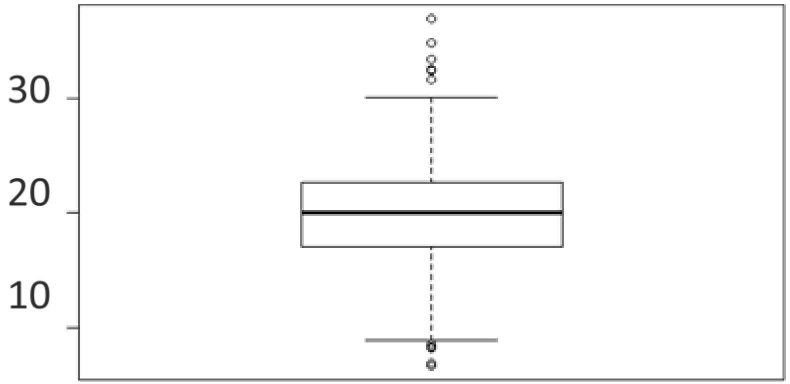
- One continuous numeric variable: **Boxplot**.



Basic Visualization Types (1/13)

Univariate visualization:

- One continuous numeric variable: **Boxplot**.

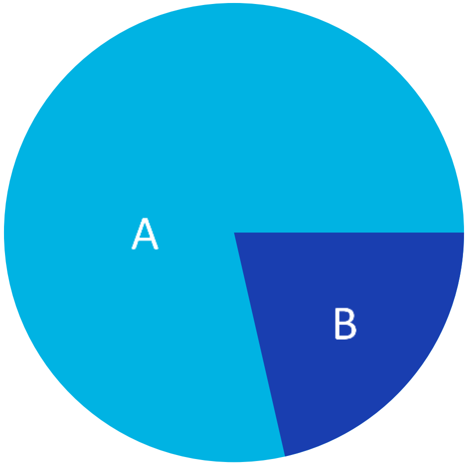


- A boxplot is composed of a box, whiskers, outliers, etc.

Basic Visualization Types (5/13)

Univariate visualization:

- One categorical variable: **Pie chart**.

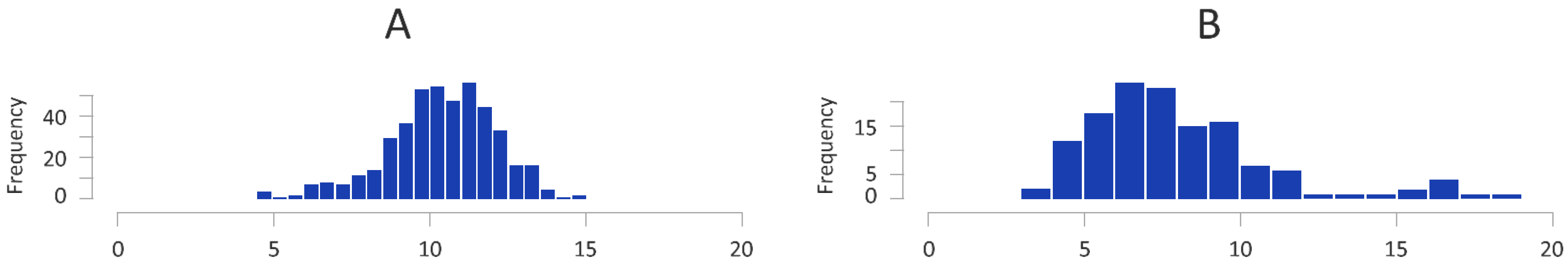


- Shows the proportions of each category (type).

Basic Visualization Types (7/13)

Bivariate visualization:

- One continuous numeric variable & one categorical variable: Multiple Histograms.

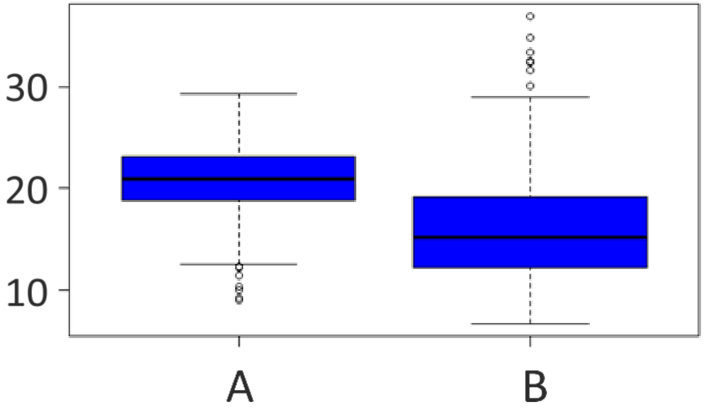


- The number of categories (types) = the number of histograms.
- You should make sure that the axis ranges match for proper comparison.

Basic Visualization Types (6/13)

Bivariate visualization:

- One continuous numeric variable & one categorical variable: **Multiple Boxplots**

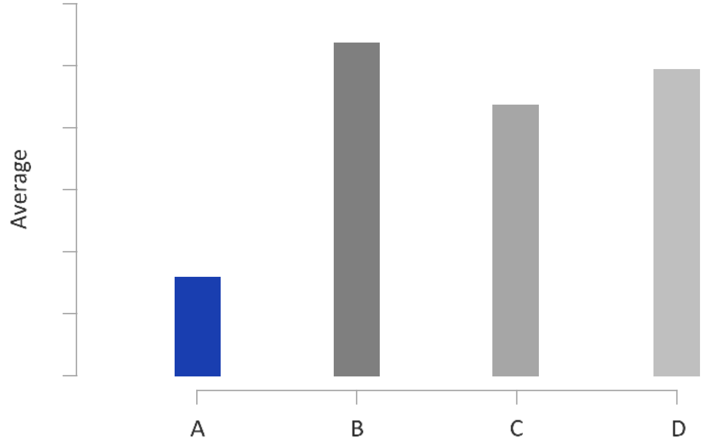


- The number of categories (types) = the number of boxplots.

Basic Visualization Types (8/13)

Bivariate visualization:

- One continuous numeric variable & one categorical variable: **Bar plot**.

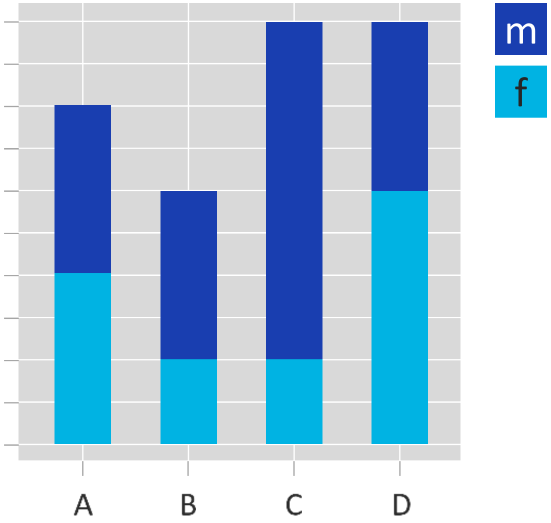


- Shows the summary statistics (average, median, etc.) for each category (type, group).

Basic Visualization Types (9/13)

Bivariate visualization:

- Two categorical variables: **Bar plot**.

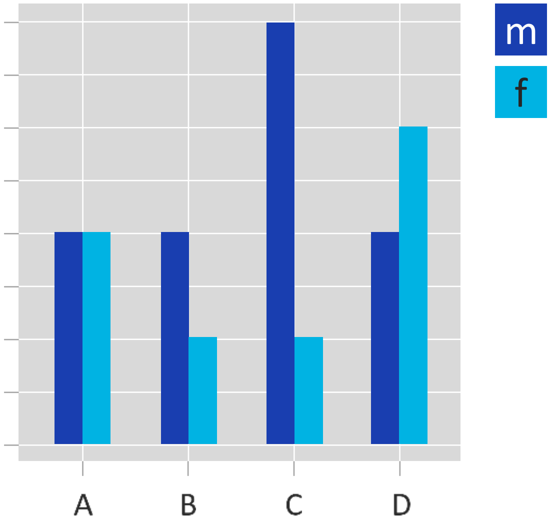


- Use color to distinguish the categories of the secondary variable.

Basic Visualization Types (10/13)

Bivariate visualization:

- Two categorical variables: Bar plot.

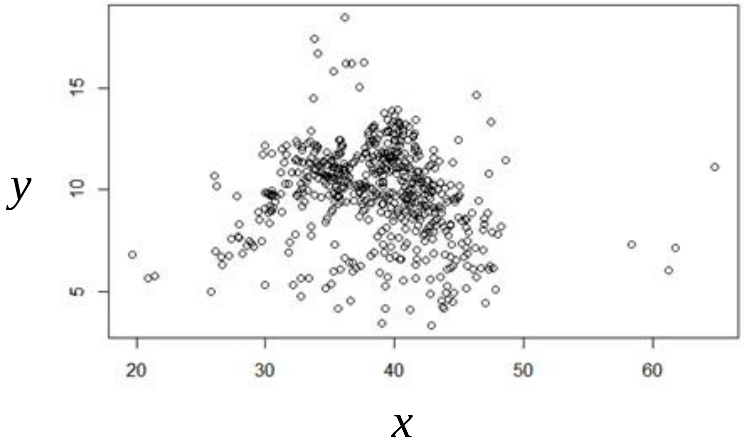


- Use color and dodged bars to distinguish the categories of the secondary variable.

Basic Visualization Types (11/13)

Bivariate visualization:

- Two continuous numeric variables: **Scatter plot**.

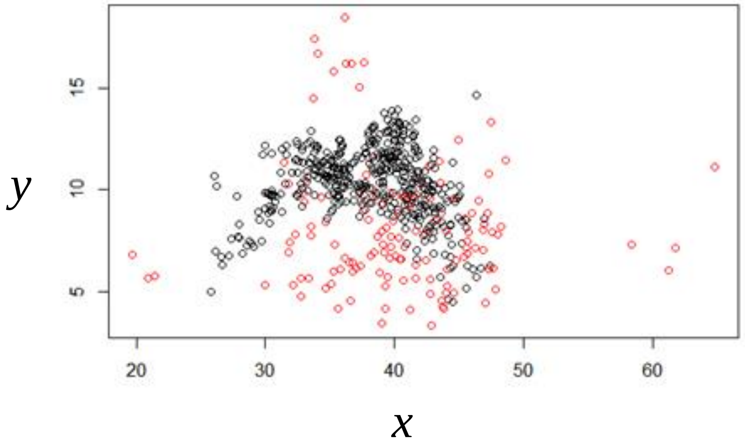


- Identify whether a linear relation exists between the two variables.

Basic Visualization Types (12/13)

Multivariate visualization:

- Two continuous numeric variables and one categorical variable: **Scatter plot**.

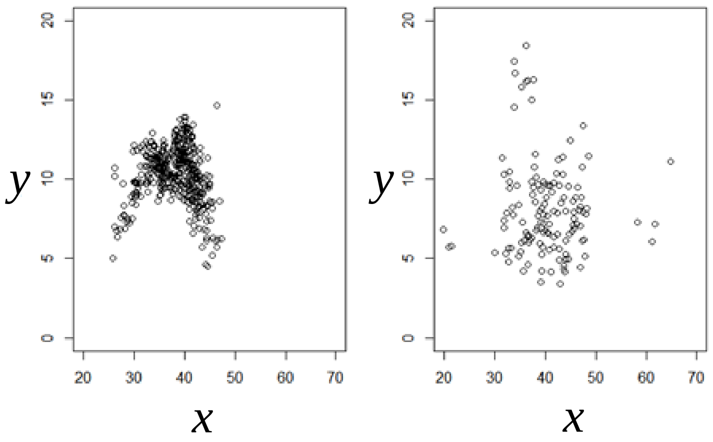


- Different categories can be denoted by different colors or markers (symbols).

Basic Visualization Types (13/13)

Multivariate visualization:

- Two continuous numeric variables and one categorical variable: **Multiple Scatter plots**.



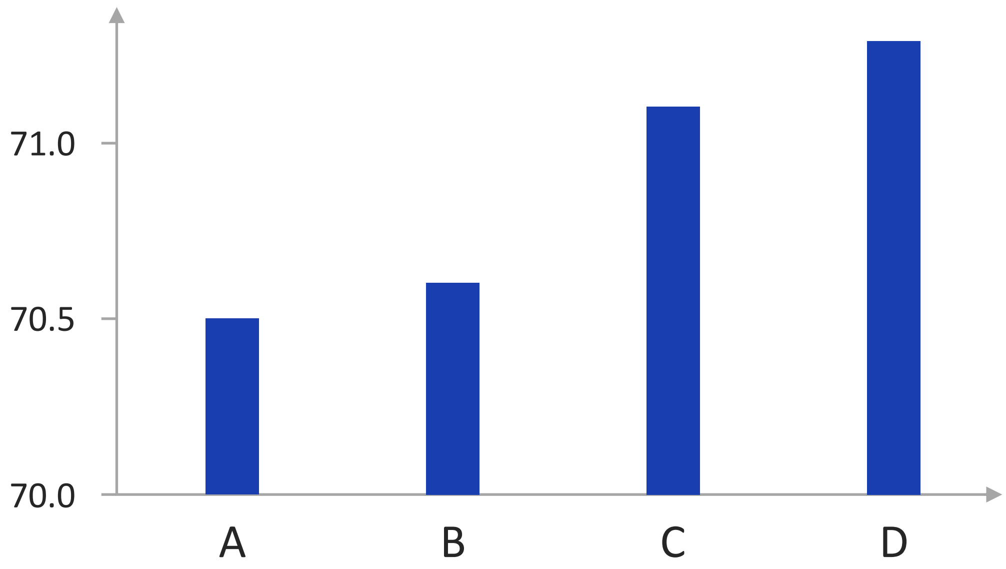
- Different categories can be plotted separately.

- You should make sure that the axis ranges match for proper comparison.

Recommendations (1/3)

In the following bar plot, can you see big difference among the categories?

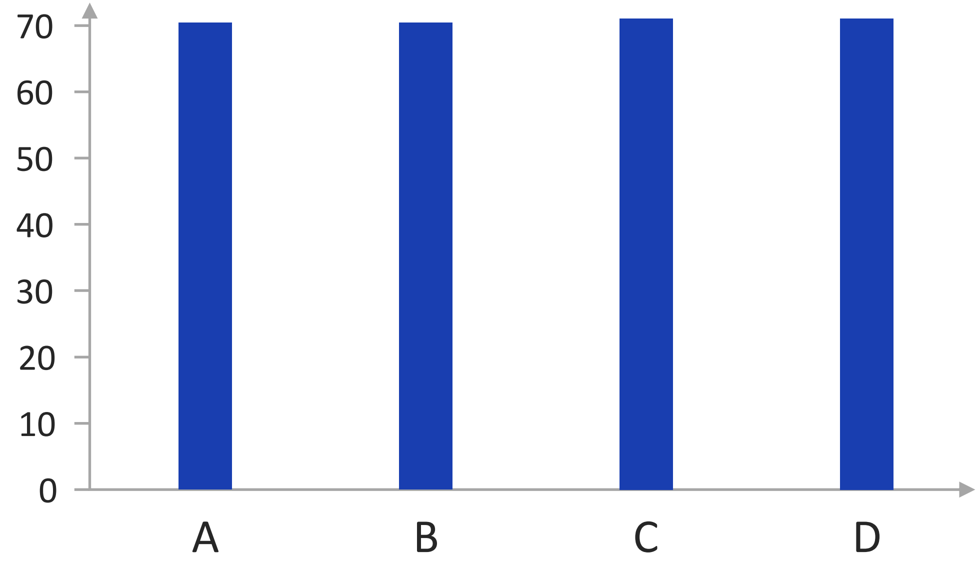
- Apparently, yes?



Recommendations (2/3)

In the following bar plot, can you see big difference between the categories?

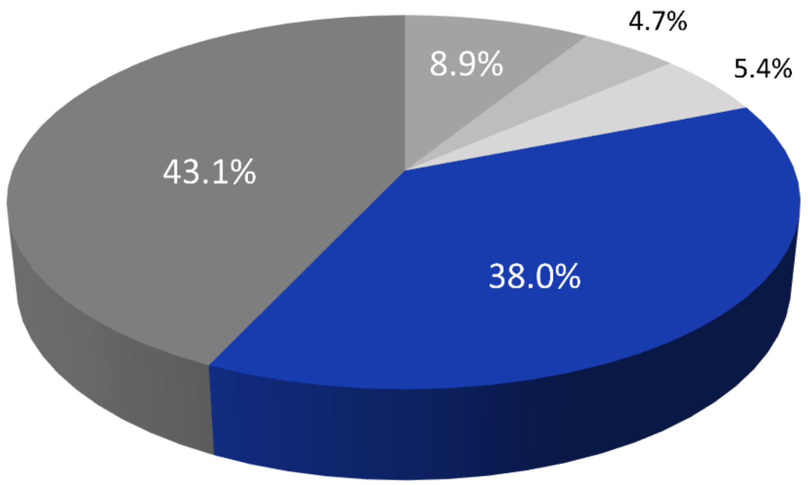
- In this case where the vertical zero is shown, you see little difference.



Recommendations (3/3)

Sometimes 3D effects should be avoided.

- In a 3D pie chart, it is hard to distinguish the relative proportions due to the perspective.



Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 3.

Python Libraries

UNIT 1. NumPy Package

1.1. NumPy array basics.

1.2. NumPy array operations.

1.3. Linear algebra: vectors and matrices.

UNIT 2. Pandas Package

2.1. Pandas Series and DataFrame.

2.2. Data summarization and manipulation.

UNIT 3. Visualization

3.1. Introduction to visualization.

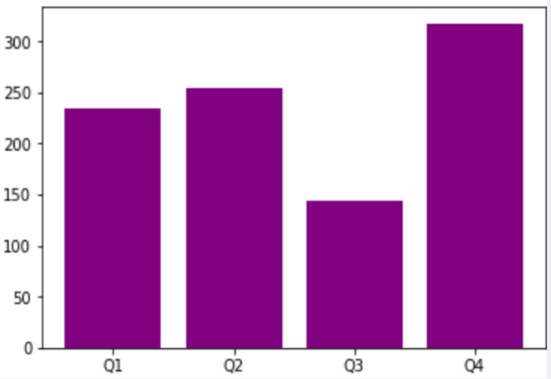
3.2. Matplotlib and Pandas visualization.

3.3. Seaborn visualization.

Basic Matplotlib Visualization (1/8)

Bar plot.

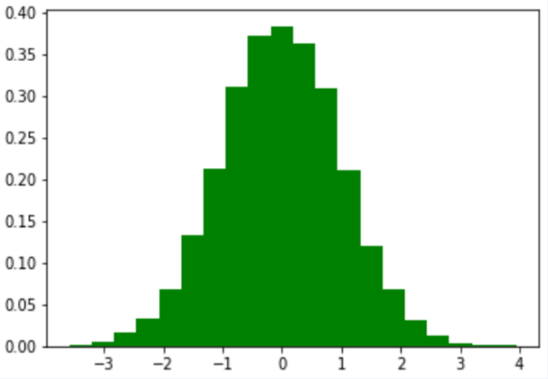
```
In[1] :  import matplotlib.pyplot as plt
         import numpy as np
         x = np.array(['Q1', 'Q2', 'Q3','Q4'])
         y = np.array([ 234.0, 254.7, 144.6, 317.6])
         plt.bar(x, y, color = 'purple')
         plt.show()
```



Basic Matplotlib Visualization (2/8)

Histogram.

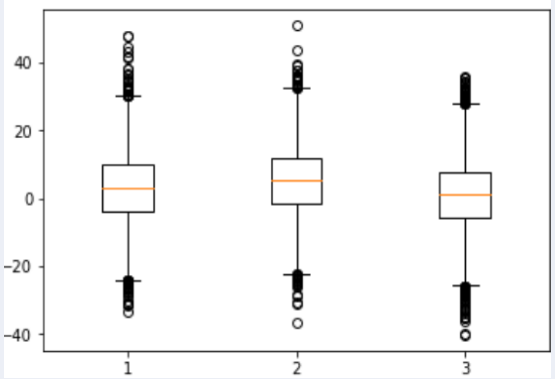
```
In[1] :  x = np.random.randn(10000)                              # NumPy array of random normal values.
          plt.hist(x ,bins=20, color='green', density=True)
          plt.show()
```



Basic Matplotlib Visualization (3/8)

Multiple Boxplots.

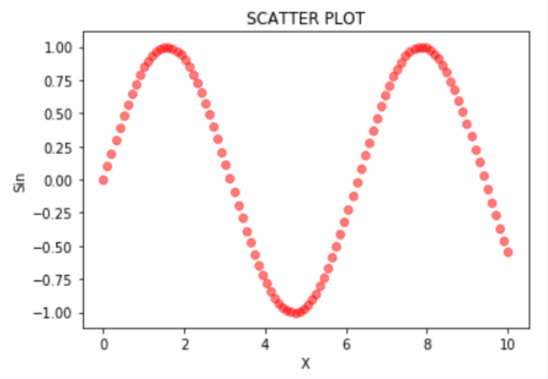
```
In[1] : x = np.random.randn(10000)*10+3                    # x, y, z = NumPy arrays of random normal values.
        y = np.random.randn(10000)*10+5
        z = np.random.randn(10000)*10+1
        plt.boxplot([x, y, z], 0)
        plt.show()
```



Basic Matplotlib Visualization (4/8)

Scatter plot.

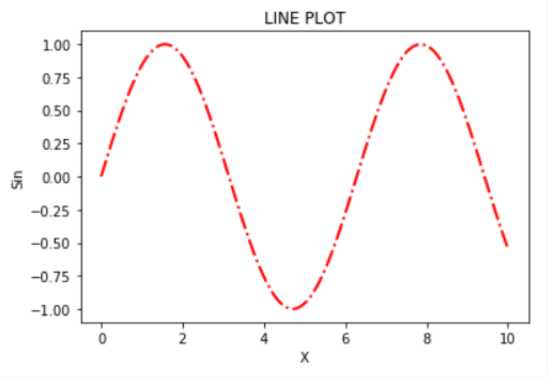
```
In[1]  : x = np.linspace(0,10,100)                                    # 100 equally spaced values between 0 and 10.
         y = np.sin(x)
         plt.scatter(x, y, c='red', marker='o', alpha=0.5)
         plt.xlabel('X')
         plt.ylabel('Sin')
         plt.title('SCATTER PLOT')
         plt.show()
```



Basic Matplotlib Visualization (5/8)

Line plot.

```
In[1] : x = np.linspace(0,10,100)                                   # 100 equally spaced values between 0 and 10.
        y = np.sin(x)
        plt.plot(x, y, color='red', linestyle='-.', linewidth=2)
        plt.xlabel('X')
        plt.ylabel('Sin')
        plt.title('LINE PLOT')
        plt.show()
```



Basic Matplotlib Visualization (6/8)

Arguments of the plot() function:

Argument	Explanation
color	Color.
alpha	Transparency.
linewidth	Line width.
linestyle	Line style.
marker	Marker type.
markersize	Marker size.
markerfacecolor	Marker color inside.
markeredgecolor	Color of the marker edge.
markeredgewidth	Width of the marker edge.

More information can be found at https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.plot.html

Basic Matplotlib Visualization (7/8)

Values of the **linestyle** argument:

linestyle	Explanation
'none'	No line.
':'	Dotted line.
'--'	Dashed line.
'-.'	Dash dot.
'-'	Continuous line.
'steps'	In steps.

More information can be found at https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.plot.html

Basic Matplotlib Visualization (8/8)

- Values of the **marker** argument:

marker	Explanation
'.'	Point.
','	Pixel.
'o'	Circle.
'^'	Triangle up.
'v'	Triangle down.
's'	Square.
'*'	Star.
'+'	Plus sign.
'x'	X character.
'D'	**Diamond.**
'p'	Pentagon.

- More information can be found at https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.plot.html

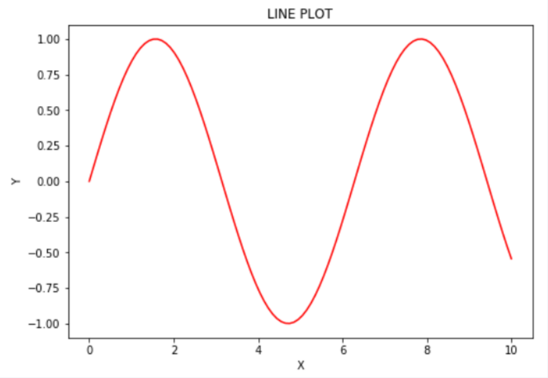
Coding Exercise #0209

Follow practice steps on 'ex_0209.ipynb' file

Matplotlib Visualization with Objects (1/5)

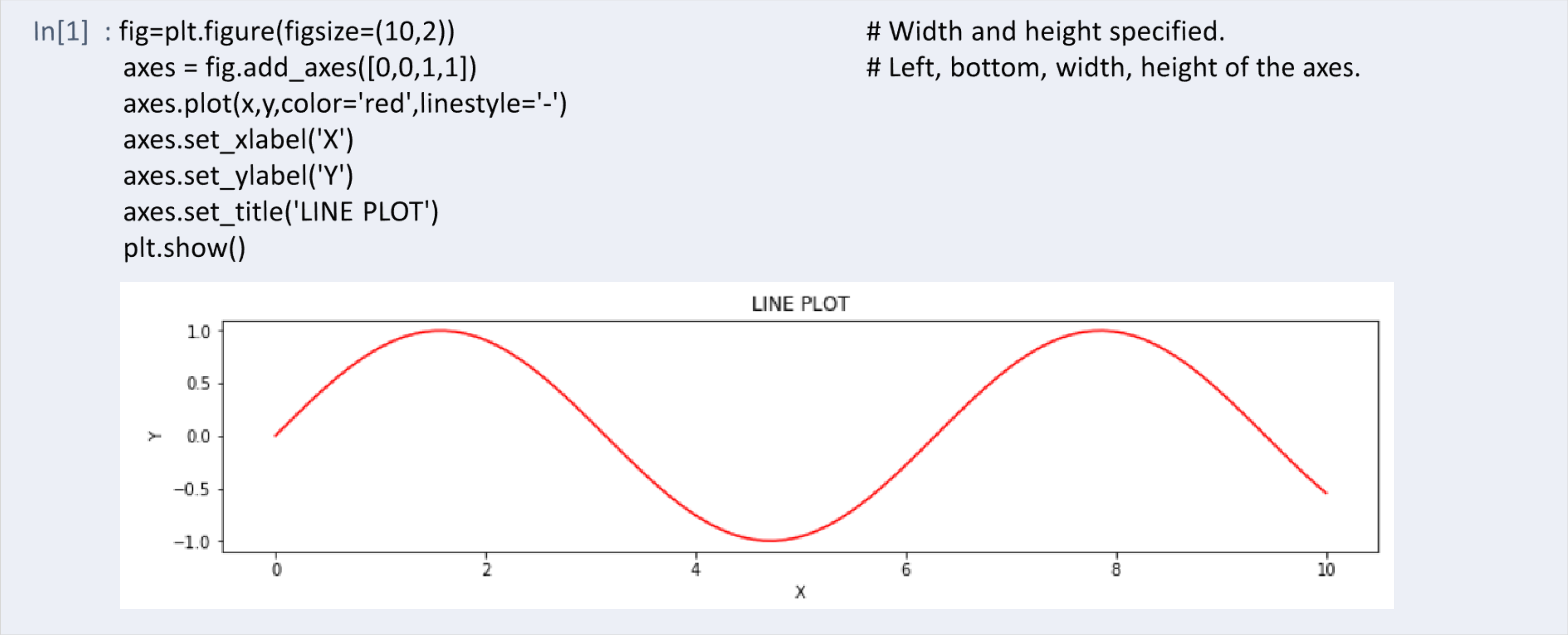
Visualization with a figure object:

```
In[1]  : fig=plt.figure()                                        # Create a figure object.
         axes = fig.add_axes([0,0,1,1])                      # Left, bottom, width, height of the axes.
         axes.plot(x,y,color='red',linestyle='-')
         axes.set_xlabel('X')
         axes.set_ylabel('Y')
         axes.set_title('LINE PLOT')
         plt.show()
```



Matplotlib Visualization with Objects (2/5)

Visualization with a figure object:

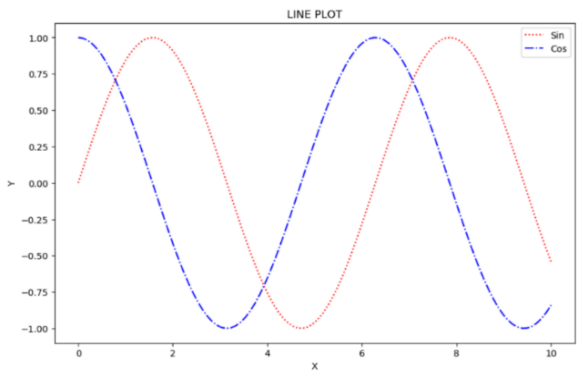


```
In[1]  : fig=plt.figure(figsize=(10,2))                    # Width and height specified.
         axes = fig.add_axes([0,0,1,1])                    # Left, bottom, width, height of the axes.
         axes.plot(x,y,color='red',linestyle='-')
         axes.set_xlabel('X')
         axes.set_ylabel('Y')
         axes.set_title('LINE PLOT')
         plt.show()
```

Matplotlib Visualization with Objects (3/5)

Multiple plots within the same axes:

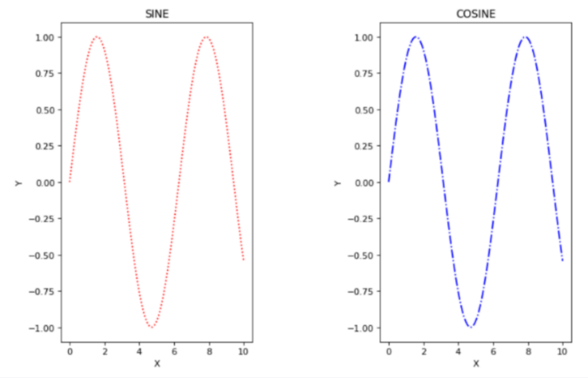
```
In[1]  : fig=plt.figure(figsize=(8,5), dpi=100)                    # Width, height and DPI setting.
         axes = fig.add_axes([0,0,1,1])                            # Left, bottom, width, height of the axes.
         axes.plot(x,y,color='red',linestyle=':', label='Sin')     # Label for the legend.
         axes.plot(x,z,color='blue',linestyle='-.', label='Cos')   # Label for the legend.
         axes.legend(loc=0)                                        # Legend at the top-right corner.
         axes.set_xlabel('X')
         axes.set_ylabel('Y')
         axes.set_title('LINE PLOT')
         plt.show()
```



Matplotlib Visualization with Objects (4/5)

Multiple plots in separate axes:

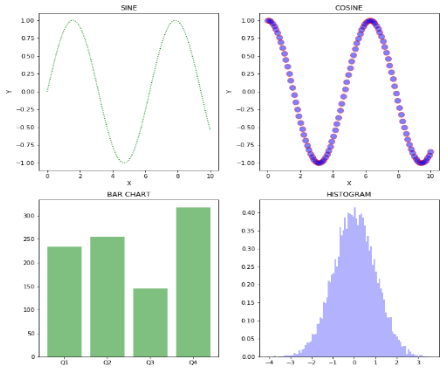
```
In[1]  : fig=plt.figure(figsize=(10,5), dpi=80)                    # Width, height and DPI setting.
         axes1 = fig.add_axes([0,0,0.3,1])                # Left, bottom, width, height of the axes1.
         axes2 = fig.add_axes([0.5,0,0.3,1])                  # Left, bottom, width, height of the axes2.
         axes1.plot(x,y,color='red',linestyle=':')
         axes2.plot(x,y,color='blue',linestyle='-.')
         axes1.set_xlabel('X')
         axes1.set_ylabel('Y')
         axes1.set_title('SINE')
         axes2.set_xlabel('X')
         axes2.set_ylabel('Y')
         axes2.set_title('COSINE')
         plt.show()
```



Matplotlib Visualization with Objects (5/5)

Multiple plots in an array of axes:

```
In[1] : fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10,10))          # A 2×2 array of axes.
         # (0,0) <= The top-left axes.
         axes[0,0].plot(x,y,color='green',linestyle=':')
         axes[0,0].set_xlabel('X')
         axes[0,0].set_ylabel('Y')
         axes[0,0].set_title('SINE')
         # (0,1) <= Continue with the top-right axes.
           ⋮
           ⋮
           ⋮
           ⋮
         plt.tight_layout()                                           # Avoid overlapping.
         plt.show()
```



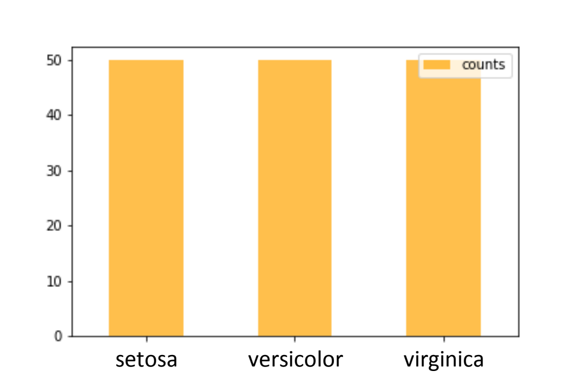
Coding Exercise #0210

Follow practice steps on 'ex_0210.ipynb' file

Pandas Visualization (1/4)

Visualize directly from Series and DataFrames:

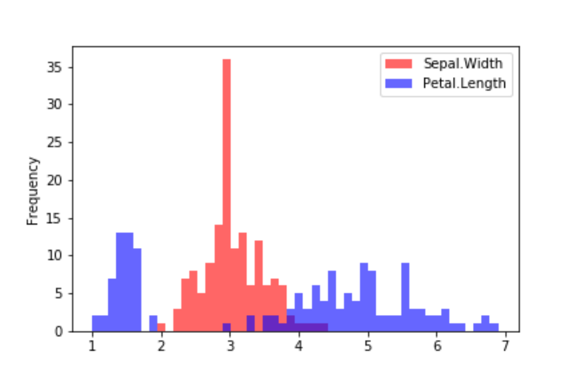
```
In[1] : frequencies = df.Species.value_counts()
        my_counts = list(frequencies.values)
        my_labels = list(frequencies.index)
        df2 = pd.DataFrame( {'counts':my_counts}, index = my_labels)
        df2.plot.bar(color='orange', alpha=0.7)
        plt.show()
```



Pandas Visualization (2/4)

Visualize directly from Series and DataFrames:

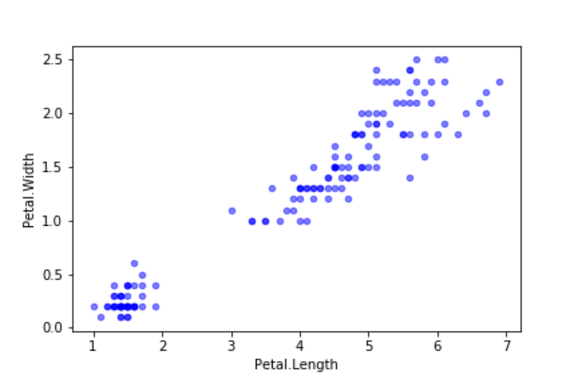
```
In[1] : df.loc[:,['Sepal.Width','Petal.Length']].plot.hist(bins=50, color=['red','blue'], alpha=0.6)
        plt.show()
```



Pandas Visualization (3/4)

Visualize directly from Series and DataFrames:

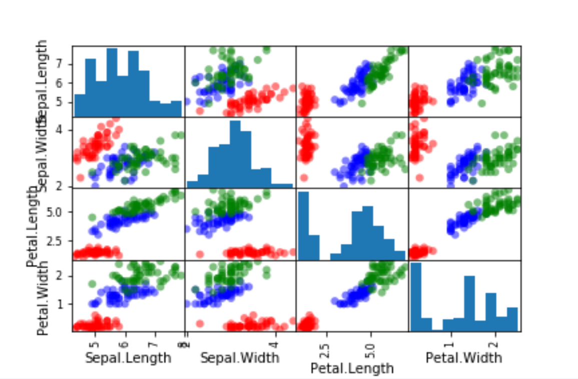
```
In[1]  : df.plot.scatter(x='Petal.Length', y='Petal.Width', color='blue', alpha=0.5,marker='o', s=20)
         plt.show()
```



Pandas Visualization (4/4)

Pandas provides specialized visualization functions:

```
In[1]  : my_cols_dict = {'setosa':'red', 'virginica':'green', 'versicolor':'blue’}
         my_cols = df['Species'].apply(lambda x: my_cols_dict[x])                    # Convert species into colors.
         pd.plotting.scatter_matrix(df, c=my_cols, marker='o', alpha=0.5)
         plt.show()
```



Coding Exercise #0211

Follow practice steps on 'ex_0211.ipynb' file

Coding Exercise #0212

Follow practice steps on 'ex_0212.ipynb' file