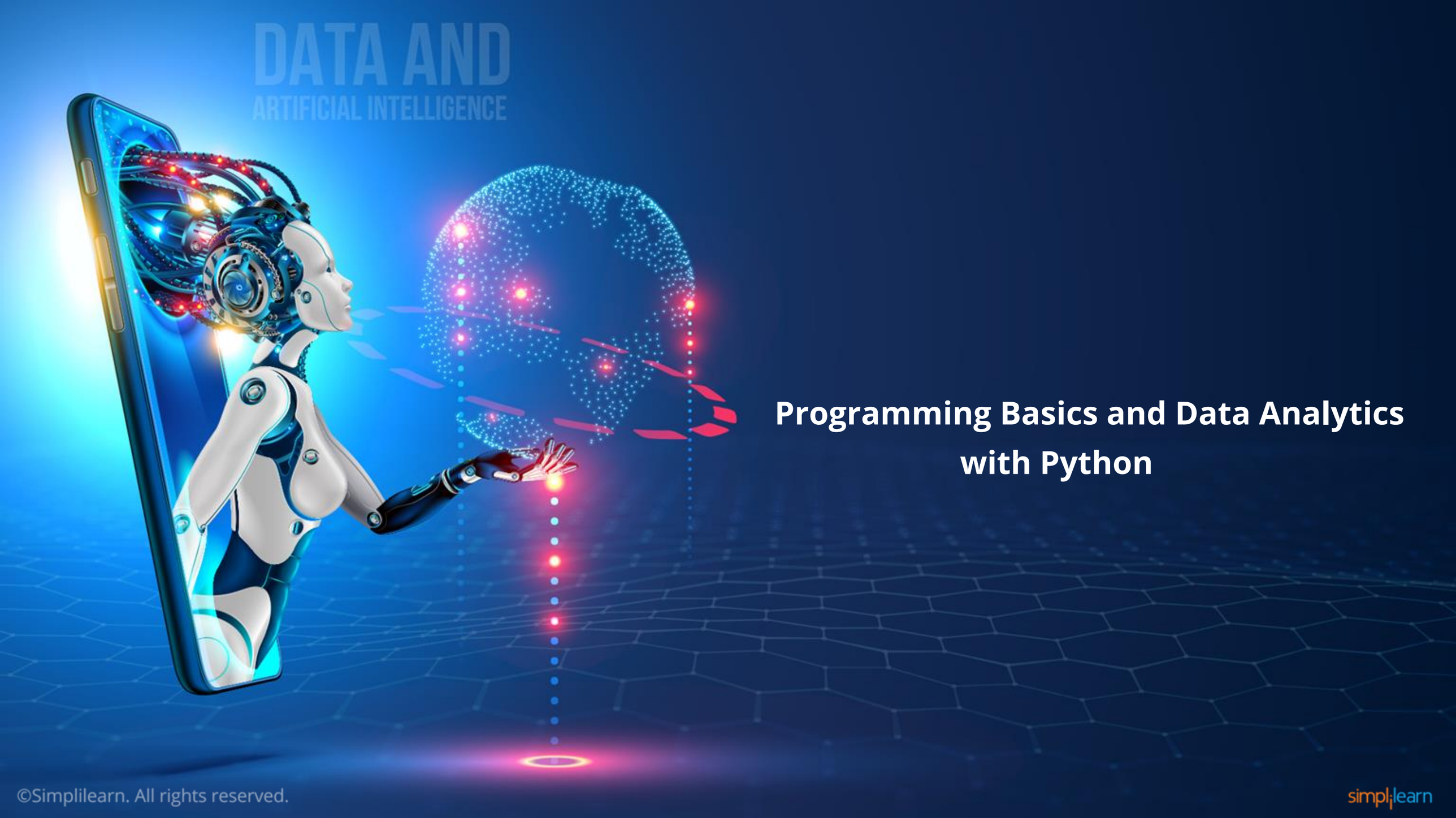


Programming Basics and Data Analytics with Python

simplilearn

Data Visualization with Python

Learning Objectives

By the end of this lesson, you will be able to:

- Explain data visualization and its importance in today's world
- Understand why Python is considered one of the best data visualization tools
- Describe matplotlib and its data visualization features in Python
- List the types of plots and the steps involved in creating these plots

DATA AND
ARTIFICIAL INTELLIGENCE

Data Visualization

Data Visualization

Data visualization is a technique to present the data in a pictorial or graphical format.

Data Visualization

You are a Sales Manager in a leading global organization. The organization plans to study the sales details of each product across all regions and countries. This is to identify the product which has the highest sales in a particular region and up the production. This research will enable the organization to increase the manufacture of that product in that particular region.

Data Visualization

The data involved in this research might be huge and complex. Manual research on this large numeric data is difficult and time-consuming.

Data Visualization

When these numeric data are plotted on a graph or converted to charts, it is easy to identify the patterns and predict the result accurately.

Data Visualization

The main benefits of data visualization are as follows:

Explores new patterns and reveals hidden patterns

Identifies the relationship between data points and variables

Simplifies the complex quantitative information

Analyzes and explores Big Data easily

DATA AND
ARTIFICIAL INTELLIGENCE

Considerations of Data Visualization

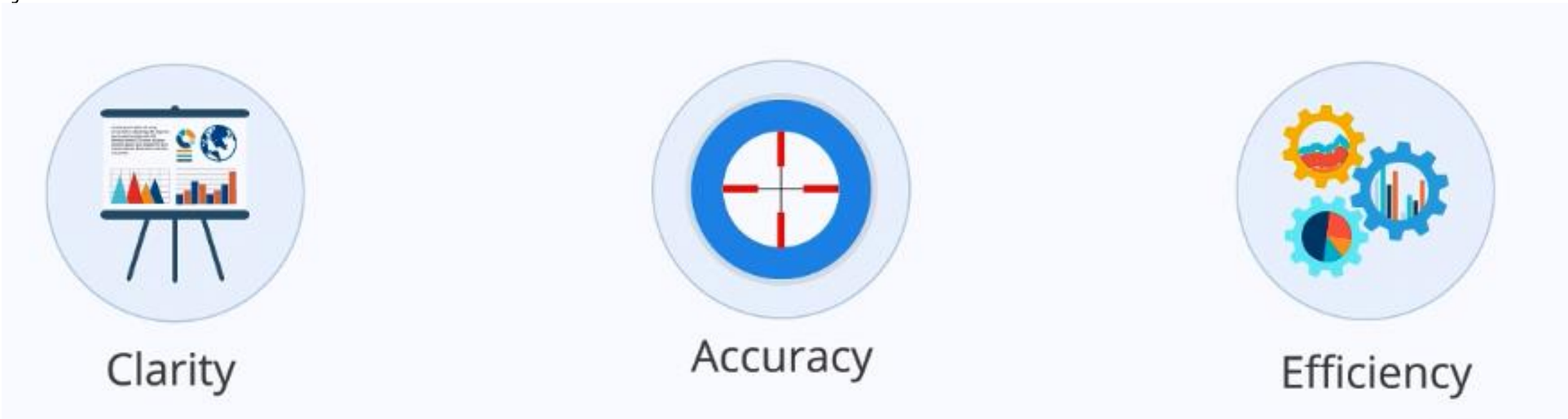
Considerations of Data Visualization

Three major considerations for data visualization are:

Clarity includes ensuring that the dataset is complete and relevant. This enables the Data Scientist to use the new patterns obtained from the data in the relevant places.

Considerations of Data Visualization

Three major considerations for data visualization are:



Accuracy includes ensuring that you use appropriate graphical representation to convey the intended message.

Considerations of Data Visualization

Three major considerations for data visualization are:

Efficiency includes the use of efficient visualization techniques that highlight all the data points.

DATA AND
ARTIFICIAL INTELLIGENCE

Factors of Data Visualization

Factors of Data Visualization

There are some basic factors that one needs to be aware of, before visualizing the data:

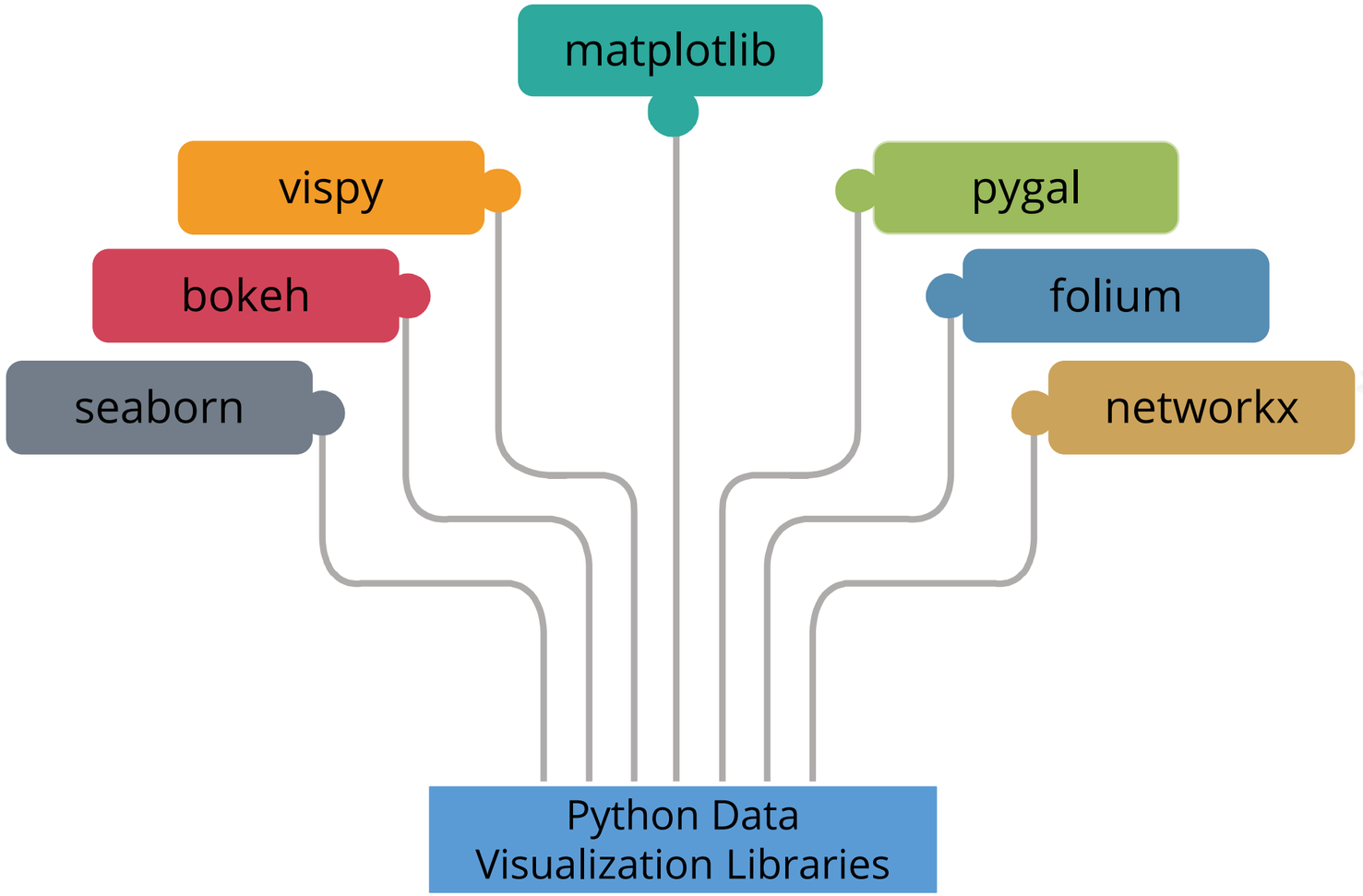
- The visual effect includes the usage of appropriate shapes, colors, and sizes to represent the analyzed data.
- The coordinate system helps organize the data points within the provided coordinates.
- The data types and scale choose the type of data; for example, numeric or categorical.
- The informative interpretation helps create visuals in an effective and easily interpretable manner using labels, titles, legends, and pointers.

DATA AND
ARTIFICIAL INTELLIGENCE

Python Libraries

Python Libraries

Many Python data visualization libraries are being introduced recently.

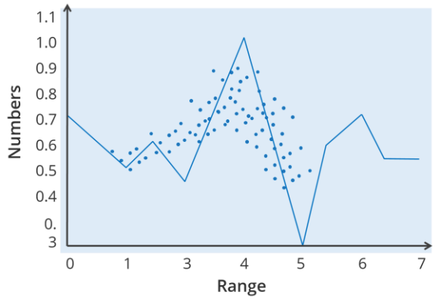


Python's Matplotlib

Using Python's matplotlib, the data visualization of large and complex data becomes easy.

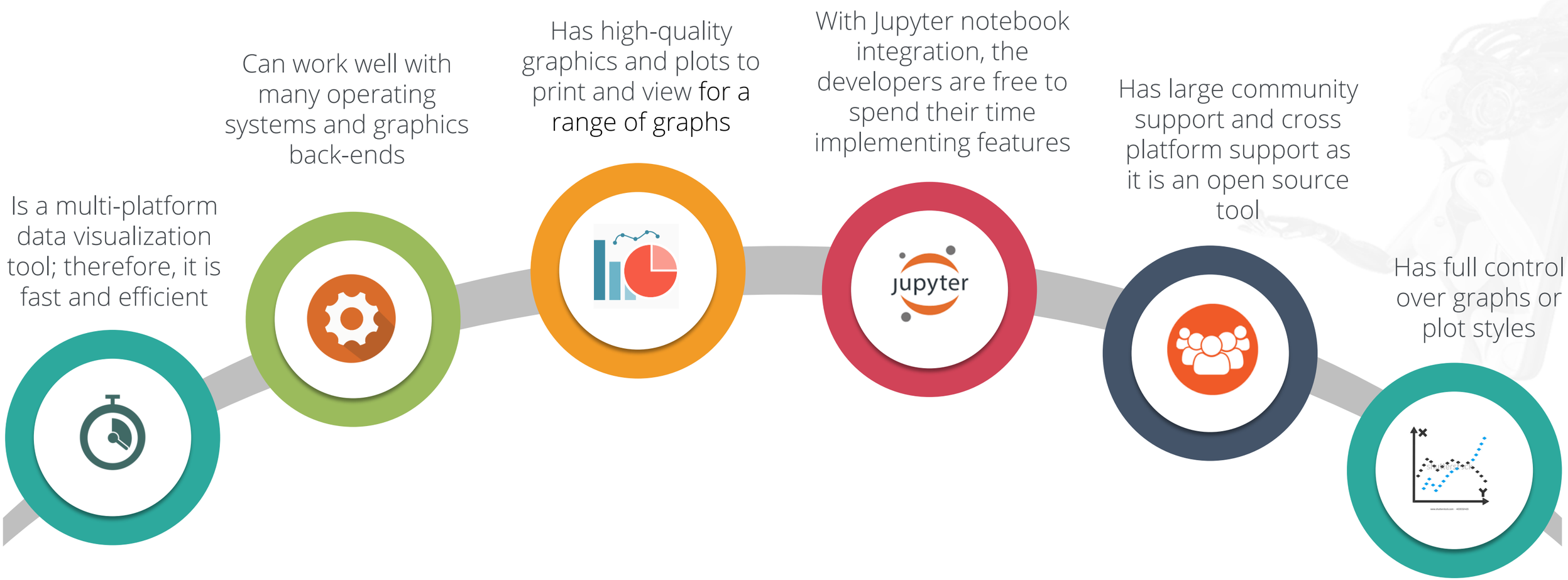
matplotlib

Python 2D plotting library



Python Libraries: Matplotlib

There are several advantages of using matplotlib to visualize data. They are as follows:



Matplotlib Architecture

Scripting Layer
(pyplot)

Artist Layer
(Artist)

Back-End Layer
(FigureCanvas, Renderer, Event)

Matplotlib Architecture

Scripting layer (pyplot)

Comprised mainly of **pyplot**, a scripting interface then is lighter that the **Artist** layer

Artist layer (Artist)

Comprised of one main object:: **Artist**

- Title, lines, tick labels, and images, all correspond to individual **Artist** instances.
- Two types of **Artist** objects:
  1. **Primitive**: Line2D, Rectangle, Circle, and Text
  2. **Composite**: Axis, Tick, Axes, and Figure
- Each *composite* artist may contain other *composite* artists as well as *primitive* artists.

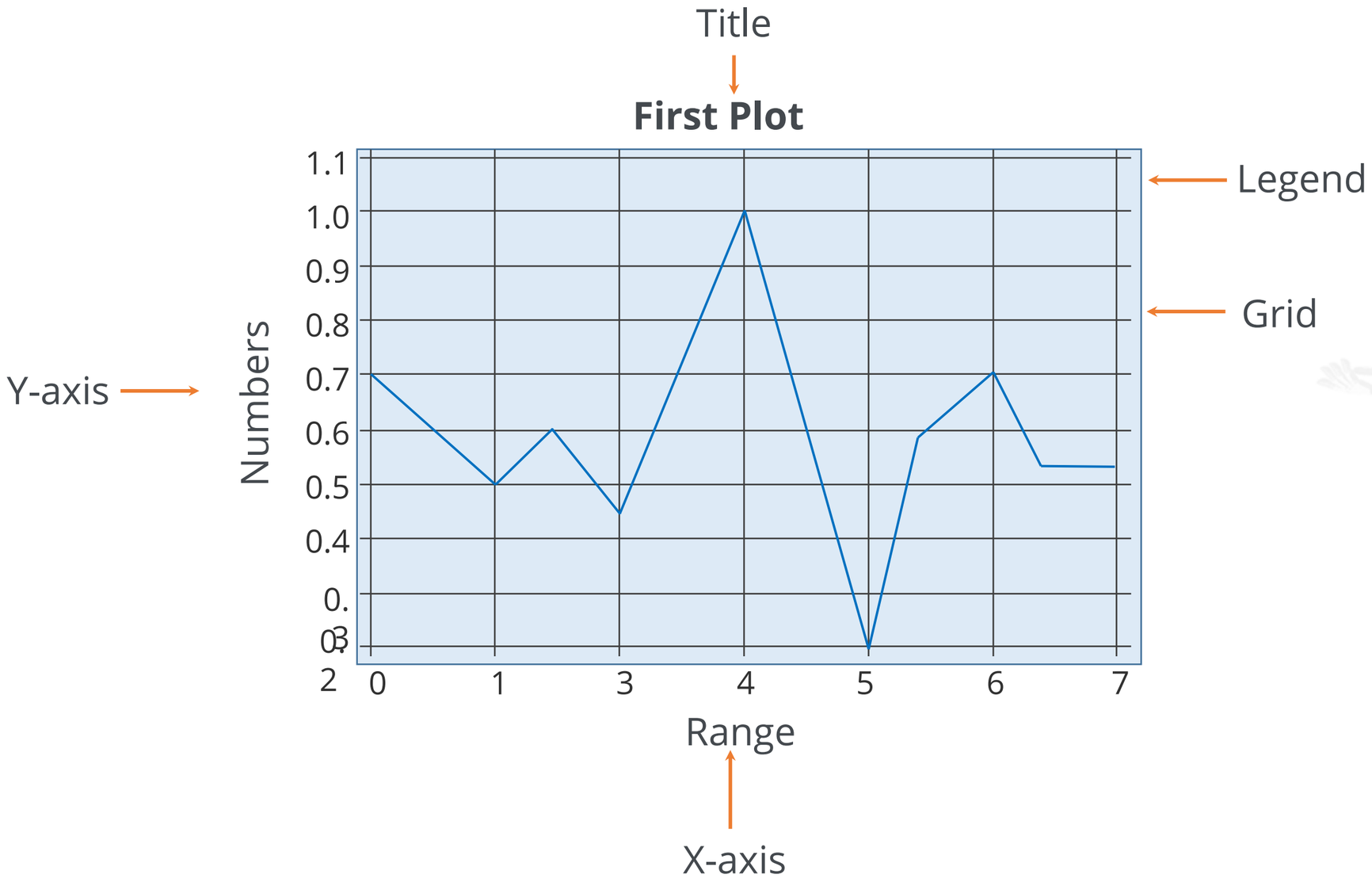
Back-end layer (FigureCanvas, Renderer, Event)

Comprised of three built-in abstract interface classes:

1. FigureCanvas: Encompasses the area onto which the figure is drawn
2. Renderer: Knows how to draw on the FigureCanvas
3. Event: Handles user inputs such as keyboard strokes and mouse clicks

The Plot

A plot is a graphical representation of data which shows the relationship between two variables or the distribution of data.



Steps to Create a Plot

You can create a plot using four simple steps:

Step 01: Import the required libraries

Step 02: Define or import the required dataset

Step 03: Set the plot parameters

Step 04: Display the created plot

Steps to Create Plot: Example

```
In [1]: #import numpy for generating random numbers
        import numpy as np
        #import matplotib library
        import matplotlib.pyplot as plt
        from matplotlib import style
        %matplotlib inline
```

Plot the numbers — pyplot

set the grid style — style

```
In [21]: #generate random numbers (total 10)
         randomNumber = np.random.rand(10)
```

used numpy random method to generate random numbers

```
In [22]: #view them
         print randomNumber
```

view the created random numbers

```
[ 0.71892609  0.49065612  0.61092193  0.43397501  0.94771363  0.31505178
  0.58568599  0.6929941   0.4288734   0.43774794]
```

```
In [23]: #select the style of the plot
         style.use('ggplot')
         #plot the random number
         plt.plot(randomNumber,'g',label='line one',linewidth=2)
         #x axis is number of random numbers (index)
         plt.xlabel('Range')
         #y axis is actual random number
         plt.ylabel('Numbers')
         #Title of the plot
         plt.title('First Plot')
```

ggplot — Set the style

Set the legend

Set line width

Set coordinates labels

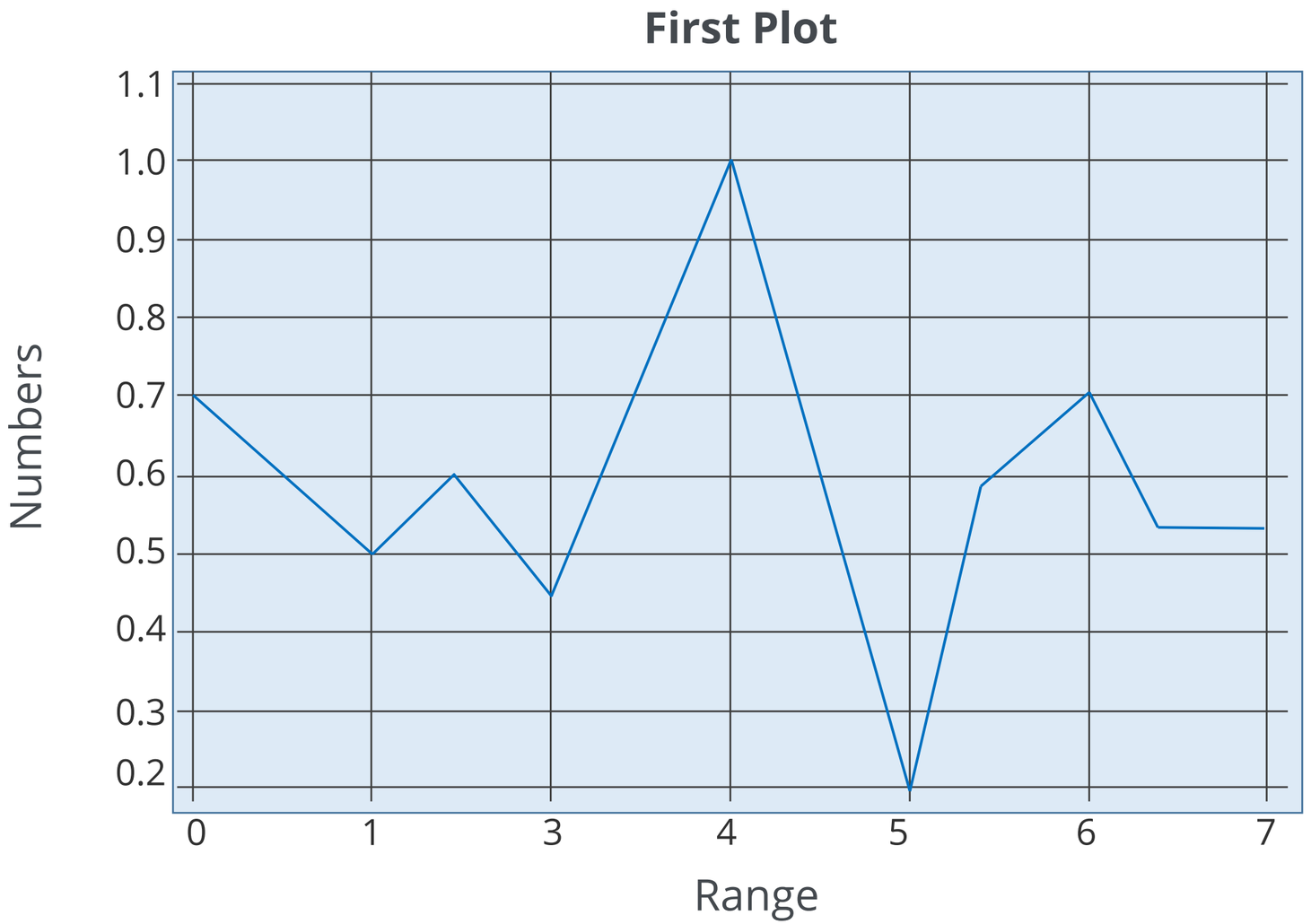
Set the title

```
plt.legend()
plt.show()
```

Plot the graph

Display the created plot

Steps to Create Plot: Example



Create Your First Plot Using Matplotlib

Objective: Use the given **FIFA 19 dataset,** containing the detailed attributes for every player registered in the latest edition of FIFA 19 database, to load the data and create a plot between Name and Potential of 10 players.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

Line Properties

Line Properties

Line Properties

1. alpha
   set the transparency of the line
2. animated
   set the transparency of the line

Plot Graphics

1. linestyle
2. linewidth
3. marker style

matplotlib also offers various line colors.

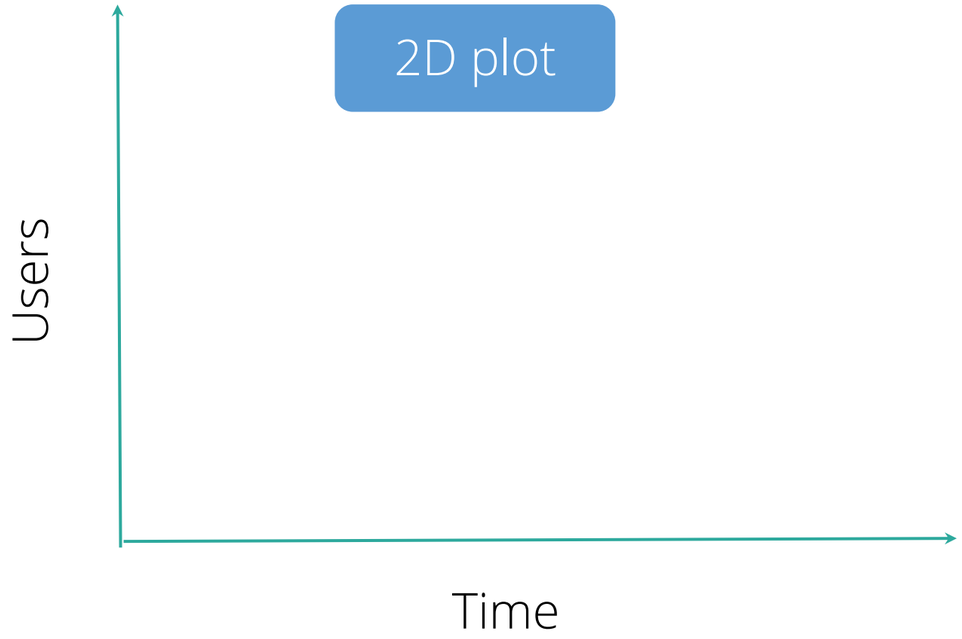
Line Properties

Property	Value Type
alpha	float
animated	[True | False]
antialiased or aa	[True | False]
clip_box	a matplotlib.transform.Bbox instance
clip_on	[True | False]
clip_path	a Path instance and a Transform instance, a Patch
color or c	any matplotlib color
contains	the hit testing function
dash_capstyle	['butt' | 'round' | 'projecting']
linestyle or ls	['-' | '--' | '-.' | ':' | 'steps' | ...]
linewidth or lw	float value in points
marker	['+' | ',' | '.' | '1' | '2' | '3' | '4']

Alias	Color
b	Blue
r	Red
c	Cyan
m	Magenta
g	Green
y	Yellow
k	Black
w	White

Plot with (X,Y)

A leading global organization wants to know how many people visit its website in a particular time. This analysis helps it control and monitor the website traffic.



Plot with (X,Y)

In [1]:
```
#import matplotib library
import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline
```

In [2]:
```
#website traffic data
#number of users/ visitors on the web site
web_customers = [123,645,950,1290,1630,1450,1034,1295,465,205,80 ]
#Time distribution (hourly)
time_hrs = [7,8,9,10,11,12,13,14,15,16,17]
```

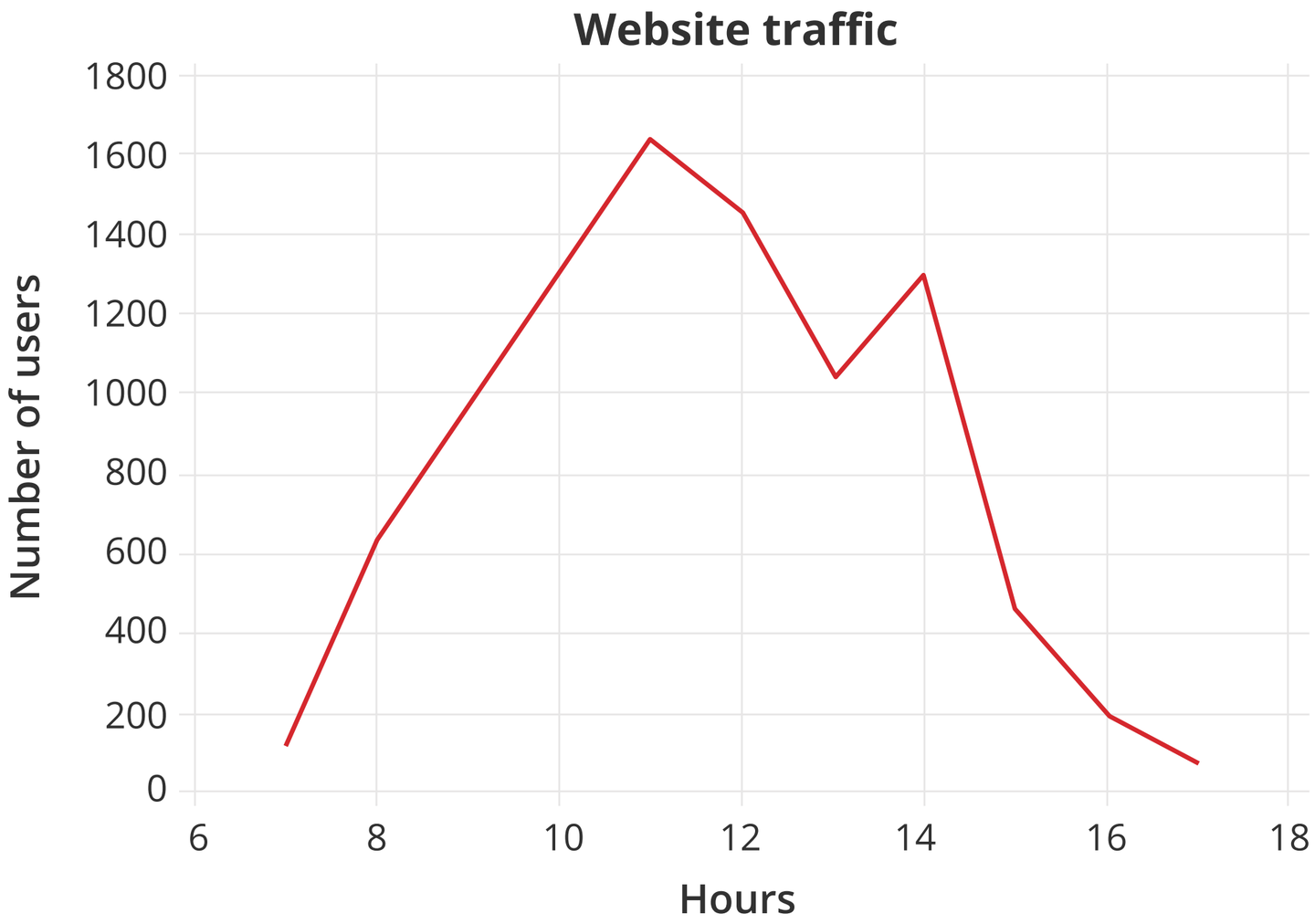
List of users

Time

In [3]:
```
#select the style of the plot
style.use('ggplot')
#plot the web site traffif data (X-axis hrs and Y axis as number of users)
plt.plot(time_hrs,web_customers)
#set the title of the plot
plt.title('Web site traffic')
#set label for x axis
plt.xlabel('Hrs')
#set label for y axis
plt.ylabel('Number of users')
plt.show()
```

Use %matplotlib inline to display or view the plot on Jupyter notebook.

Plot with (X,Y)

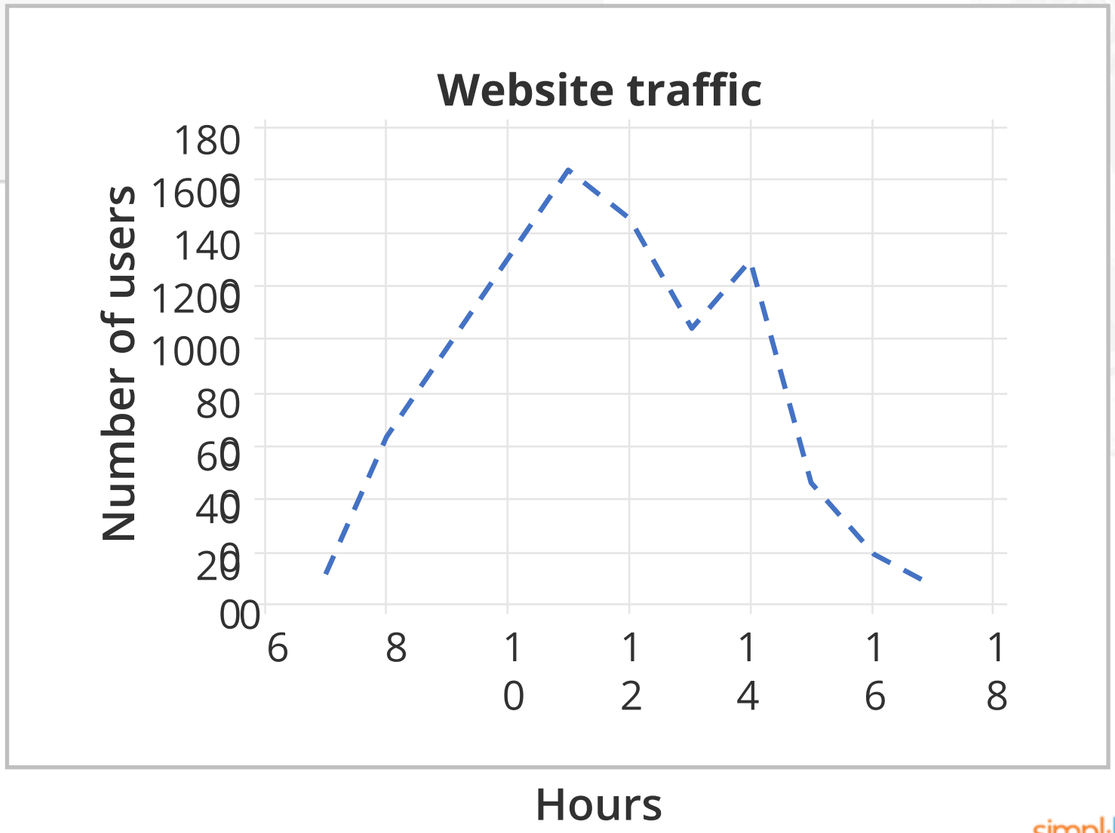


Controlling Line Patterns and Colors

```
#select the style of the plot
style.use('ggplot')
#plot the web stite traffic data (x axis hrs and y asis as number of users)
plt.plot(time_hrs,web_customers,color = 'b',linestyle = '--',linewidth=2.5)
#set the title of the plot
plt.title('Web site traffic')
#set the label for x axis
plt.xlabel('hrs')
#set the label for y axis
plt.ylabel('number of users')
plt.show()
```

Line Color (blue)

Dashed (--)



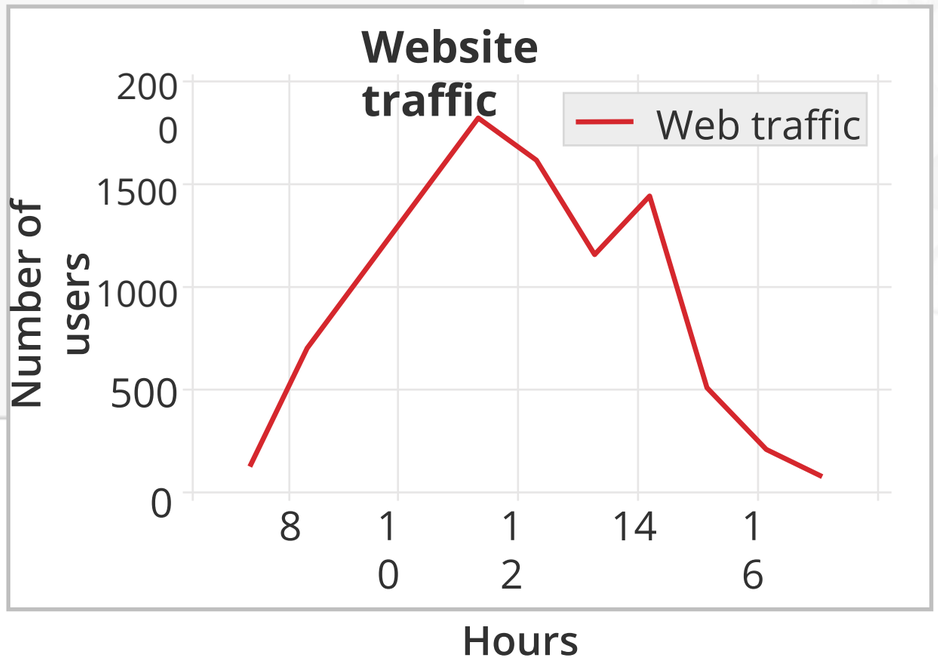
Set Axis, Labels, and Legend Property

Using matplotlib, it is also possible to set the desired axis to interpret the result.

Axis is used to define the range on the x-axis and y-axis.

```
:  #select the style of the plot
   style.use('ggplot')
   #plot the web site traffif data (X-axis hrs and Y axis as number of users)
   plt.plot(time_hrs,web_customers,'r',label='web traffic',linewidth=1.5)
   plt.axis([6.5,17.5,50,2000])
   #set the title of the plot
   plt.title('Web site traffic')
   #set label for x axis
   plt.xlabel('Hrs')
   #set label for y axis
   plt.ylabel('Number of users')
   plt.legend()
   plt.show()
```

Set the axis



Create a Line Plot for Football Analytics

Objective: Use the given **FIFA 19 dataset** to create a line plot between Name and Sliding Tackle of 10 players. Also, set the axis, labels, and legend property of the plot.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

DATA AND
ARTIFICIAL INTELLIGENCE

Multiple Plots and Subplots

Alpha and Annotation

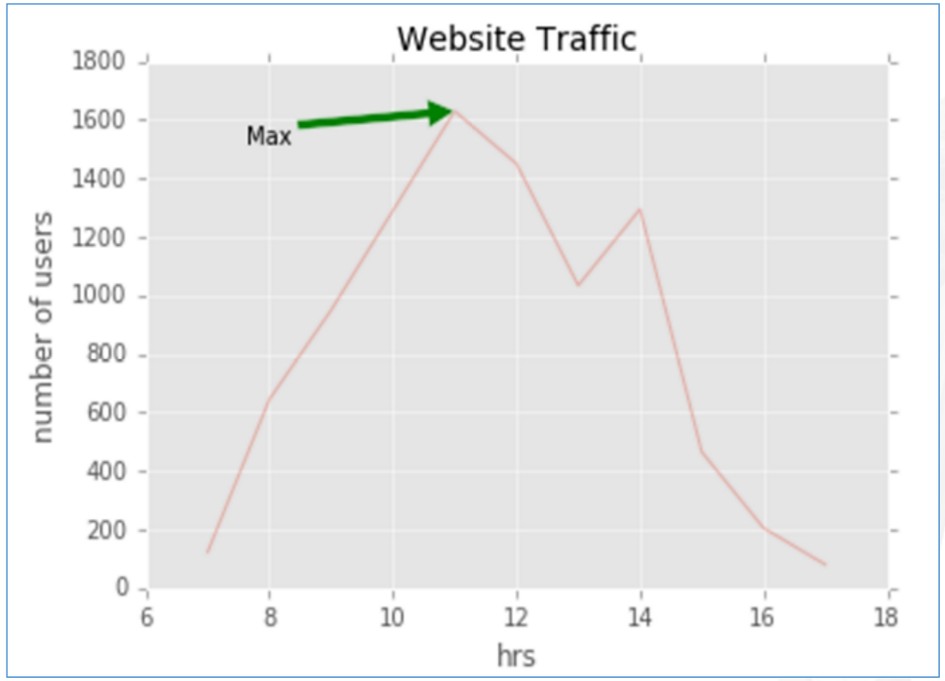
Alpha is an attribute that controls the transparency of the line.
The lower the alpha value, the more transparent the line is.

```
#select the style of the plot
style.use('ggplot')
#plot the web stite traffic data (x axis hrs and y asis as number of users)
#also setting the alpha value for transparency
plt.plot(time_hrs,web_customers,alpha=.4)
#set the title of the plot
plt.title('Website Traffic')
#Annotate
plt.annotate('Max',ha='center',va='bottom',xytext=(8,1500),xy=(11,1630),arrowprops =
             { 'facecolor' : 'green'})
#set the label for x axis
plt.xlabel('hrs')
#set the label for y axis
plt.ylabel('number of users')

plt.show()
```

Alpha and Annotation

Annotate() method is used to annotate the graph. It has several attributes which help annotate the plot.

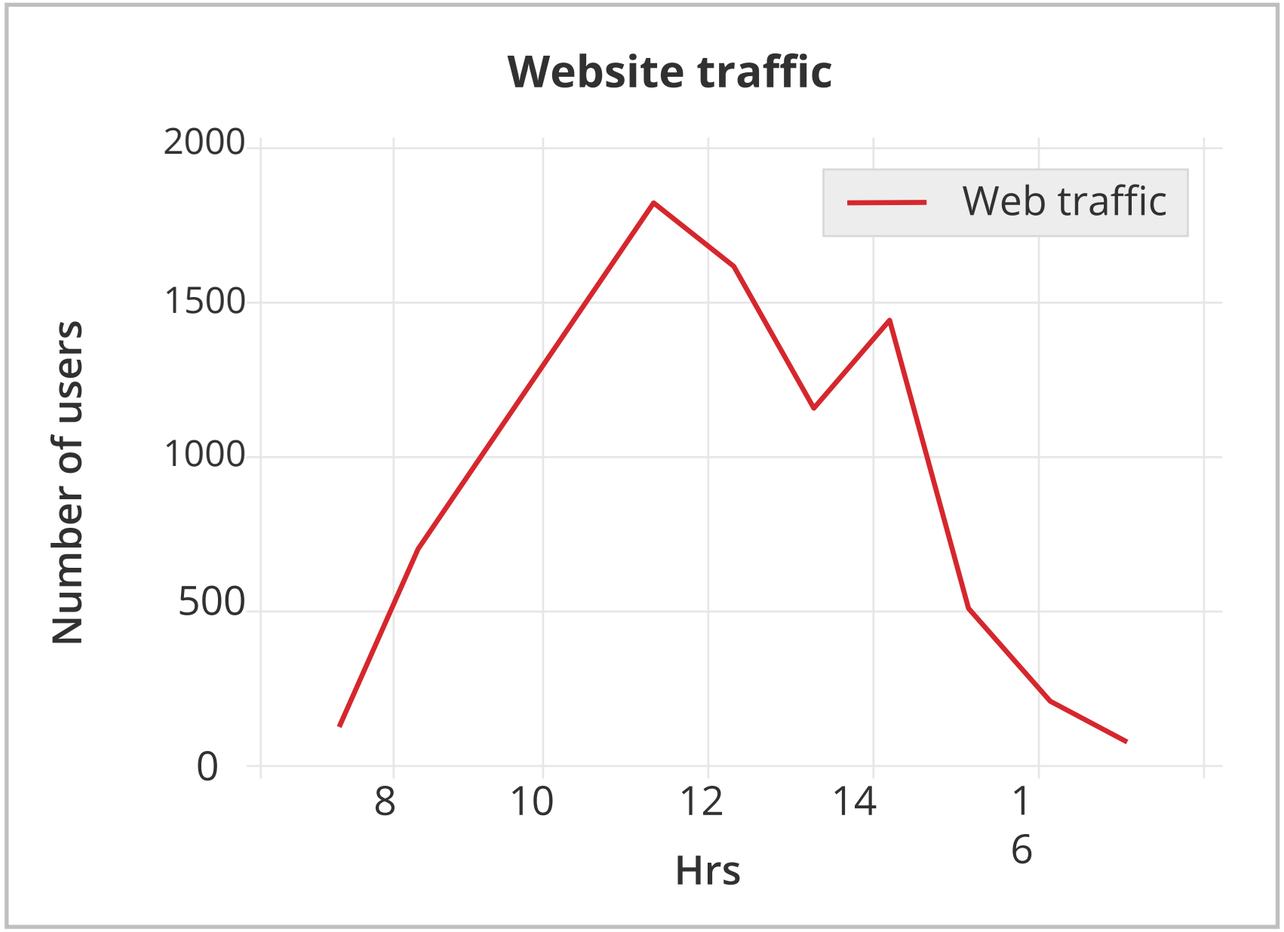
```
#select the style of the plot
style.use('ggplot')
#plot the web stite traffic data (x axis hrs and y asis as number of users)
#also setting the alpha value for transparency
plt.plot(time_hrs,web_customers,alpha=.4)
#set the title of the plot
plt.title('Website Traffic')
#Annotate
plt.annotate('Max',ha='center',va='bottom',xytext=(8,1500),xy=(11,1630),arrowprops =
             { 'facecolor' : 'green'})
#set the label for x axis
plt.xlabel('hrs')
#set the label for y axis
plt.ylabel('number of users')

plt.show()
```



"Max" denotes the annotation text,
"ha" indicates the horizontal alignment,
"va" indicates the vertical alignment,
"xytext" indicates the text position,
"xy" indicates the arrow position, and
"arrowprops" indicates the properties of the arrow.

Multiple Plots

Monday



Multiple Plots

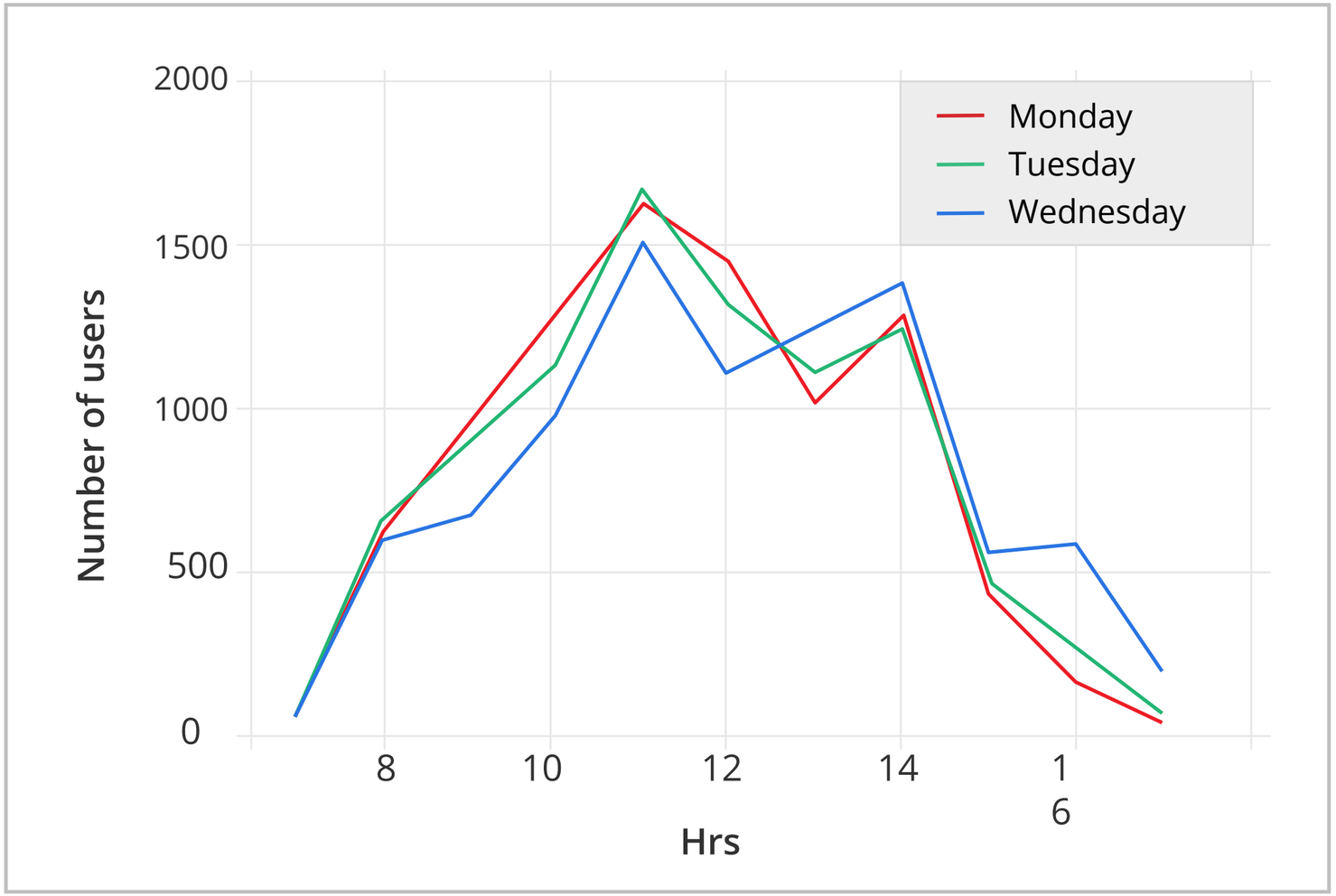
In [4]:
```
#website traffic data
#number of users/ visitors on the web site
#monday web traffic
web_monday = [123,645,950,1290,1630,1450,1034,1295,465,205,80]
#tuesday web traffic
web_tuesday= [95,680,889,1145,1670,1323,1119,1265,510,310,110]
#wednesday web traffic
web_wednesday= [105,630,700,1006,1520,1124,1239,1380,580,610,230]
#Time distribution (hourly)
time_hrs = [7,8,9,10,11,12,13,14,15,16,17]
```

Web traffic data

In [5]:
```
#select the style of the plot
style.use('ggplot')
#plot the web site traffic data (X-axis hrs and Y axis as number of users)
#plot the monday web traffic with red color
plt.plot(time_hrs,web_monday,'r',label='monday',linewidth=1)
#plot the monday web traffic with green color
plt.plot(time_hrs,web_tuesday,'g',label='tuesday',linewidth=1.5)
#plot the monday web traffic with blue color
plt.plot(time_hrs,web_wednesday,'b',label='wednesday',linewidth=2)
plt.axis([6.5,17.5,50,2000])
#set the title of the plot
plt.title('Web site traffic')
#set label for x axis
plt.xlabel('Hrs')
#set label for y axis
plt.ylabel('Number of users')
plt.legend()
plt.show()
```

Set different colors and line widths for different days

Multiple Plots



Create a Plot with Annotation

Objective: Use the given **FIFA 19 dataset** to create a plot of ShotPower of the first ten players. Also, annotate the point of maximum ShotPower.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

Create Multiple Plots to Analyze the Skills of the Players

Objective: Use the given **FIFA 19 dataset** to create multiple plots of skills of 15 players. Use labels, legend, colors, and linewidth to visualize the plot.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

UNASSISTED PRACTICE

Create Multiple Plots to Analyze the Skills of the Players

```
import pandas as pd, numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('fifa-data.csv')
dfx=df.loc[0:14]
```

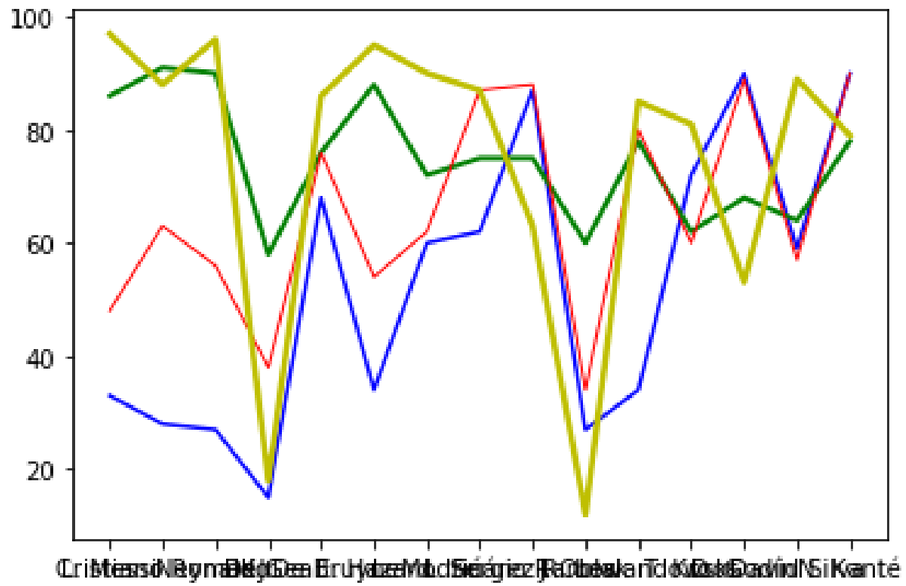
Reading a dataset

Retrieving fifteen columns from the dataframe

```
plt.plot(dfx['Name'],dfx['Marking'],'b',label='Marking',linewidth=1.5)
plt.plot(dfx['Name'],dfx['SprintSpeed'],'g',label='SprintSpeed',linewidth=2.0)
plt.plot(dfx['Name'],dfx['Aggression'],'r',label='Aggression',linewidth=1.0)
plt.plot(dfx['Name'],dfx['Dribbling'],'y',label='Dribbling',linewidth=2.5)
```

Creating plots for the skills like Marking, SprintSpeed, Aggression, and Dribbling

```
[<matplotlib.lines.Line2D at 0x1fdc66c7400>]
```



Output without the legends, xticks values, and labels

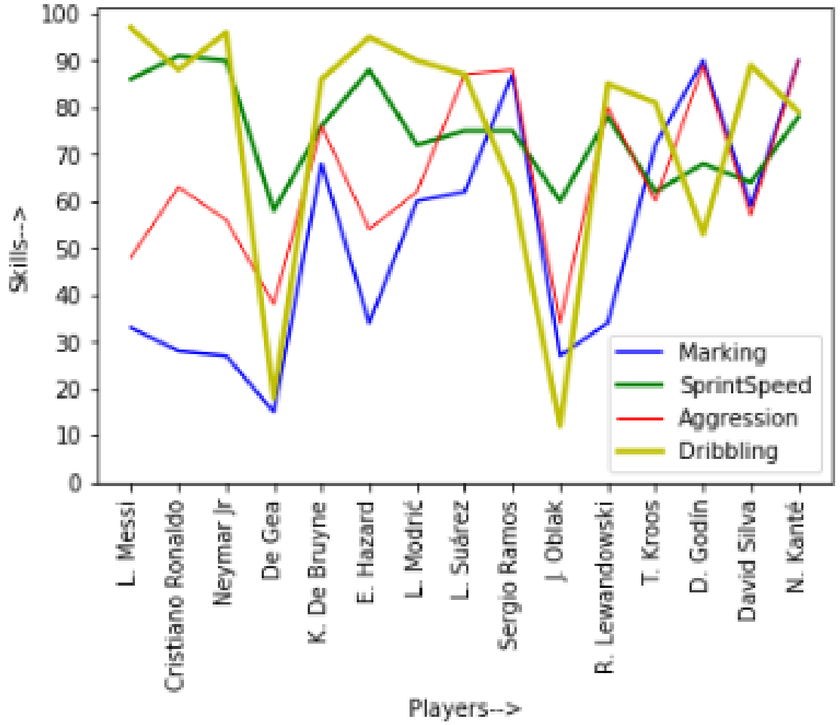
Create Multiple Plots to Analyze the Skills of the Players

```
plt.plot(dfx['Name'],dfx['Marking'],'b',label='Marking',linewidth=1.5)
plt.plot(dfx['Name'],dfx['SprintSpeed'],'g',label='SprintSpeed',linewidth=2.0)
plt.plot(dfx['Name'],dfx['Aggression'],'r',label='Aggression',linewidth=1.0)
plt.plot(dfx['Name'],dfx['Dribbling'],'y',label='Dribbling',linewidth=2.5)
plt.legend()
plt.xticks(np.arange(15), (dfx['Name']),rotation=90)
plt.yticks(np.arange(0,110,10))
plt.xlabel('Players-->')
plt.ylabel('Skills-->')
plt.show()
```

Using legend to indicate the multiple plots

Setting the label names

Setting the x-axis tick values



Final output with the legends, xticks values, and labels

Subplots

Subplots are used to display multiple plots in the same window.

With subplot, you can arrange plots in a regular grid.

The syntax for subplot is

subplot(m,n,p). → It divides the current window into an m-by-n grid and creates an axis for a subplot in the position specified by p.

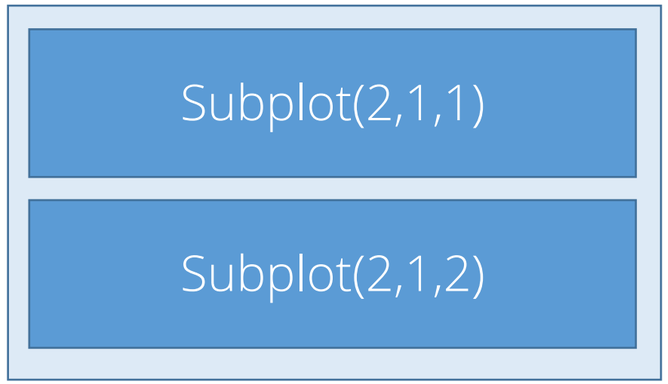
Subplots

For example,

subplot(2,1,2) creates two subplots which are stacked vertically on a grid.

subplot(2,1,4) creates four subplots in one window.

Grid divided into two vertically stacked plots

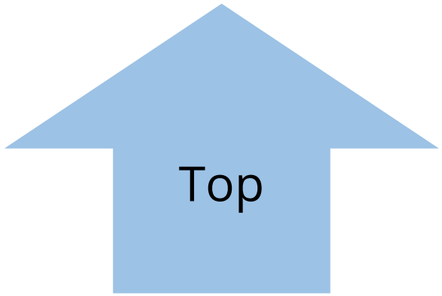


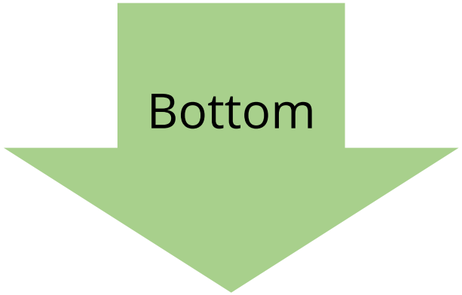
Grid divided into four plots

Layout

Layout and Spacing adjustments are two important factors to be considered while creating subplots.

Use the plt.subplots_adjust() method with the parameters hspace and wspace to adjust the distances between the subplots and to move them around on the grid.





Create Multiple Subplots Using plt.subplots

Objective: Use the given **FIFA 19 dataset** to create four subplots to analyze the skills like ball control, strength, penalties, and interceptions of ten players. Also, add legend for each plot.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

DATA AND
ARTIFICIAL INTELLIGENCE

Types of Plots

Types of Plots: Histogram

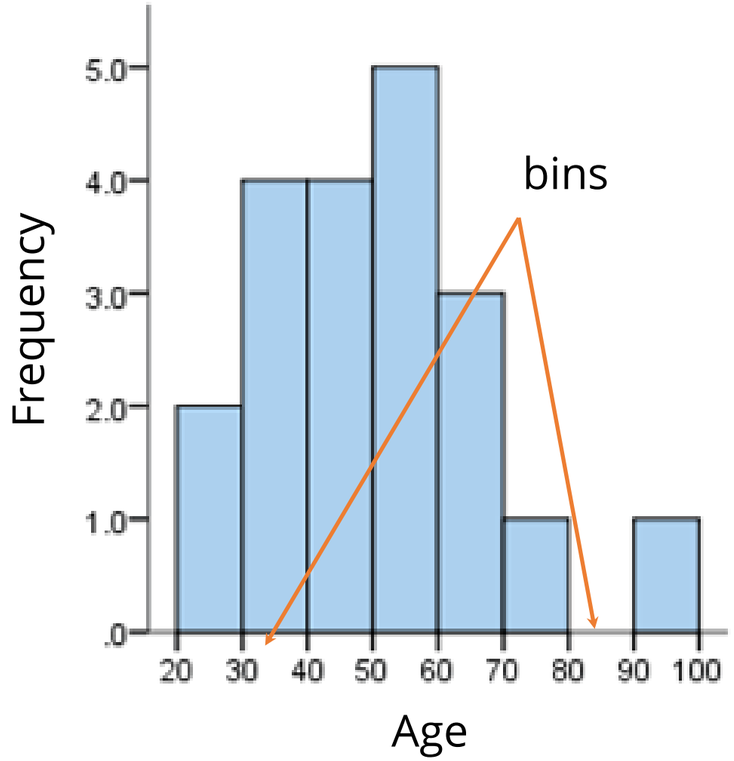
You can create different types of plots using matplotlib:

- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plots
- Word Clouds
- Bar Charts
- Box Plots
- Waffle Charts

Histograms are graphical representations of a probability distribution. A histogram is a kind of a bar chart.
Using matplotlib and its bar chart function, you can create histogram charts.

Advantages of histogram charts:

- Display the number of values within a specified interval
- Are suitable for large datasets as they can be grouped within the intervals



Types of Plots: Histogram

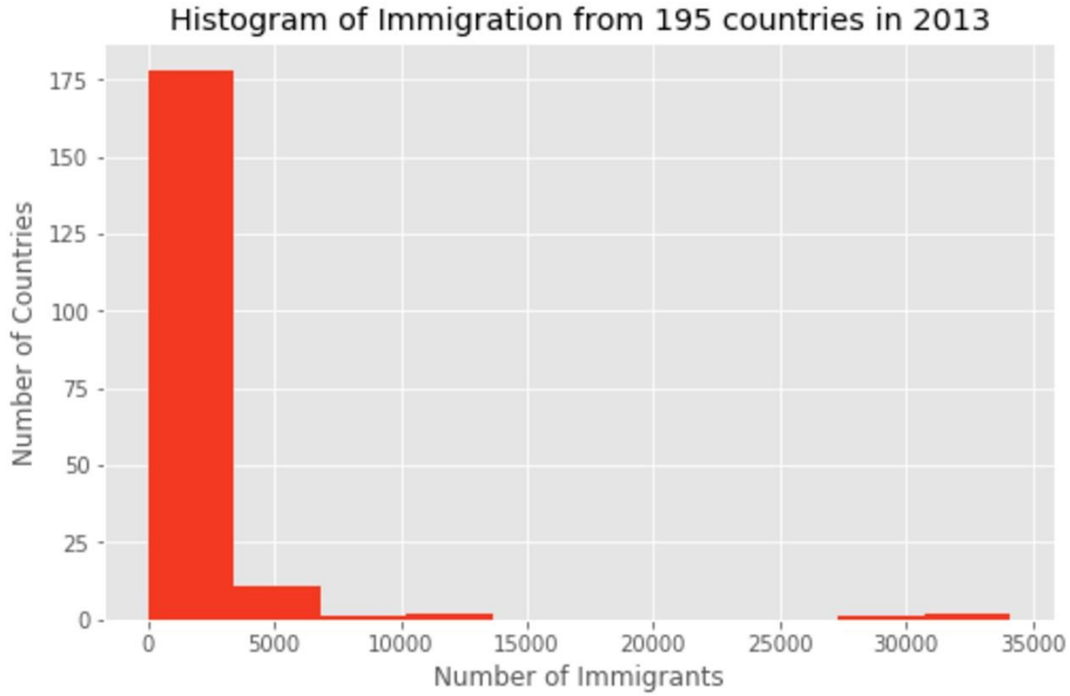
Dataset recap:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	...	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	...	2978	3436	3009	2652	2111	1746	1758	2203	2635	2004
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	...	1450	1223	856	702	560	716	561	539	620	603
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	...	3616	3626	4807	3623	4005	5393	4752	4325	3774	4331
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	...	0	0	1	0	0	0	0	0	0	0
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0	...	0	0	1	1	0	0	0	0	1	1

Types of Plots: Histogram

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

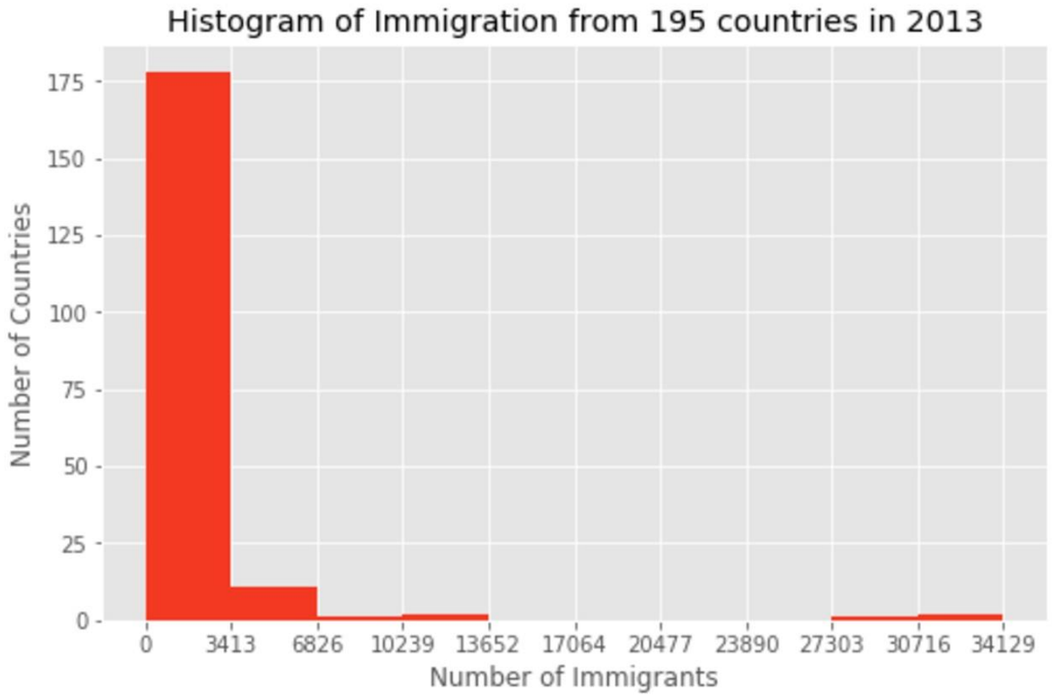
```
df_canada['2013'].plot(kind='hist')

plt.title('Histogram of Immigration from 195 countries in 2013')
plt.ylabel('Number of Countries')
plt.xlabel('Number of Immigrants')

plt.show()
```



Types of Plots: Histogram

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
```

```
count, bin_edges = np.histogram(df_canada['2013'])

df_canada['2013'].plot(kind='hist', xticks = bin_edges)

plt.title('Histogram of Immigration from 195 countries in 2013')
plt.ylabel('Number of Countries')
plt.xlabel('Number of Immigrants')

plt.show()
```



Create a Stacked Histogram

Objective: Use the given **FIFA 19 dataset** to create a stacked histogram plot of the attributes like potential and composure of 10 players. Indicate the potential and composure plot using legend.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

Types of Plots: Scatter Plot

You can create different types of plots using matplotlib:

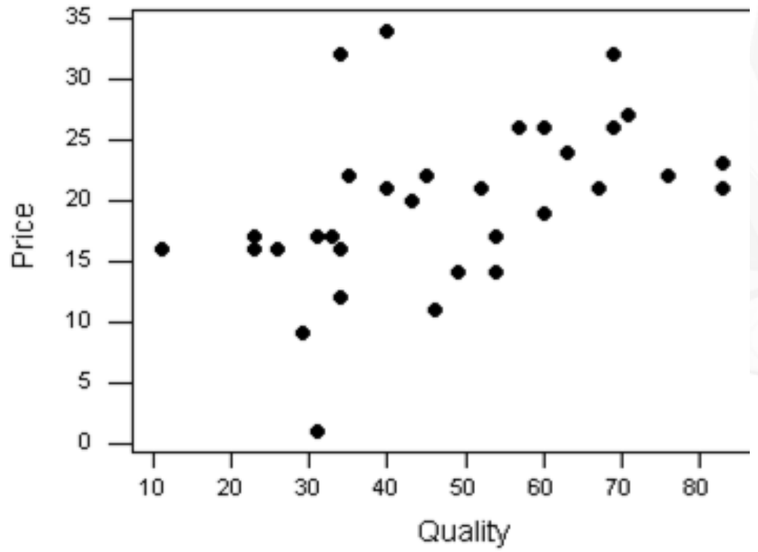
- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plots
- Word Clouds
- Bar Charts
- Box Plots
- Waffle Charts

A scatter plot is used to graphically display the relationships between variables.

Scatter() method is also recommended to control a plot.

Advantages of scatter plot:

- Shows the correlation between variables
- Is suitable for large datasets
- Is easy to find clusters
- Is possible to represent each piece of data as a point on the plot



Types of Plots: Scatter Plot

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

```
df_total.plot(
    kind='scatter',
    x='year',
    y='total',
)

plt.title('Total Immigrant population to Canada from 1980 - 2013')
plt.xlabel ('Year')
plt.ylabel('Number of Immigrants')

plt.show()
```

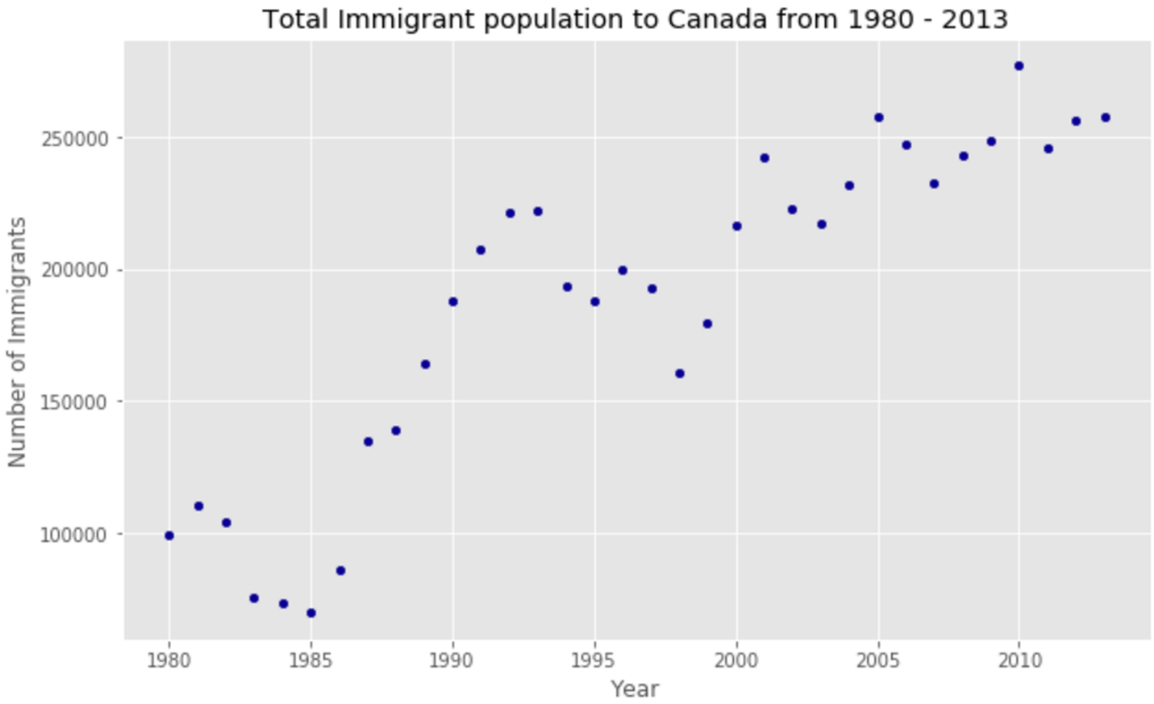
df_total

year	total
1980	99137
1981	110563
1982	104271
1983	75550
1984	73417
.	.
.	.

Types of Plots: Scatter Plot

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

```
df_total.plot(
    kind='scatter',
    x='year',
    y='total',
)

plt.title('Total Immigrant population to Canada from 1980 - 2013')
plt.xlabel ('Year')
plt.ylabel('Number of Immigrants')

plt.show()
```



Create a Scatter Plot of Pretest Scores and Posttest Scores

Objective: Create a dataframe from following data: 'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'], 'last_name': ['Miller', 'Jacobson', 'Ali', 'Milner', 'Cooze'], 'female': [0, 1, 1, 0, 1], 'age': [42, 52, 36, 24, 73], 'preTestScore': [4, 24, 31, 2, 3],
'postTestScore': [25, 94, 57, 62, 70]
Draw a Scatterplot of preTestScore and postTestScore, with the size of each point determined by age.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

Types of Plots: Heat Map

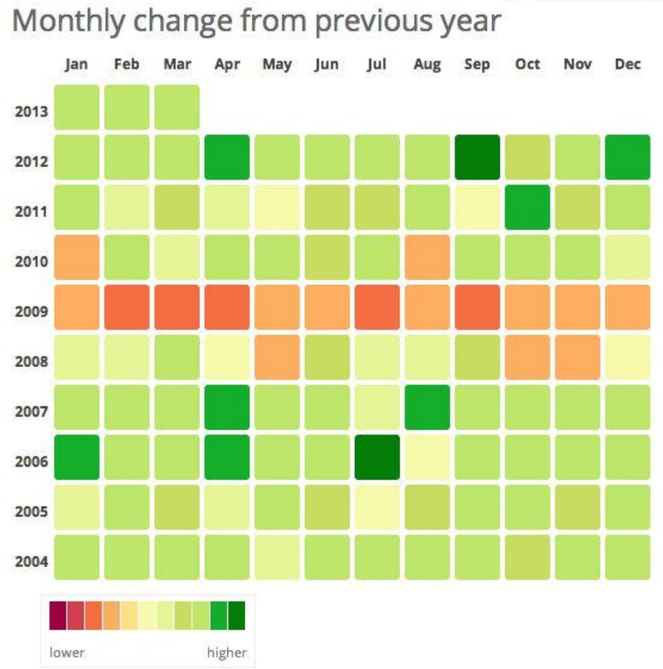
You can create different types of plots using matplotlib:

- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plots
- Word Clouds
- Bar Charts
- Box Plots
- Waffle Charts

A heat map is a way to visualize two-dimensional data. Using heat maps, you can gain deeper and faster insights about data than other types of plots.

Advantages of heat map:

- Draws attention to the risk-prone area
- Uses the entire dataset to draw meaningful insights
- Is used for cluster analysis and can deal with large datasets



Create a Heat Map to Analyze the Sepal Width, Petal Length, and Petal Width of an Iris Dataset

Objective: Use an **iris.csv** to create a heat map to analyze the sepal width, petal length, and petal width. Indicate the plot values using annotations.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

Types of Plots: Pie Chart

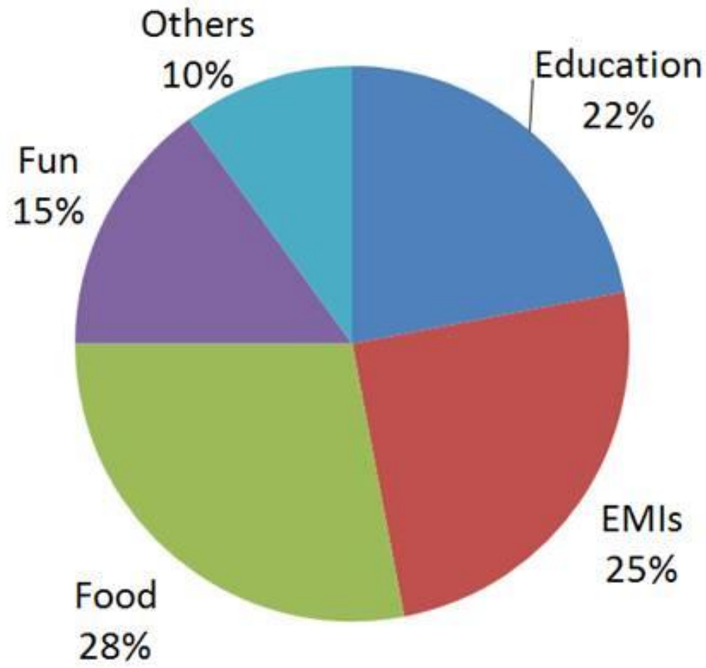
You can create different types of plots using matplotlib:

- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plots
- Word Clouds
- Bar Charts
- Box Plots
- Waffle Charts

Pie charts are used to show percentage or proportional data.
matplotlib provides the pie() method to create pie charts.

Advantages of pie chart:

- Summarizes a large dataset in visual form
- Displays the relative proportions of multiple classes of data
- Size of the circle is made proportional to the total quantity



Types of Plots: Pie Chart

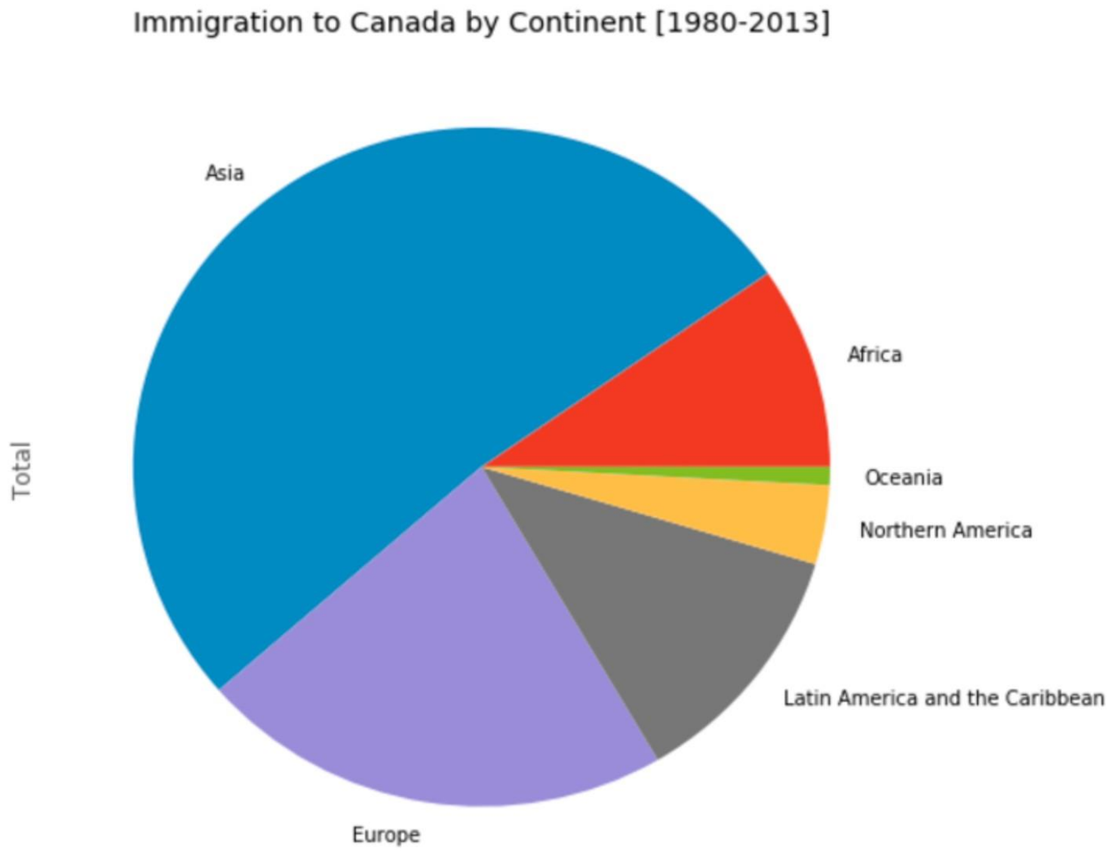
```
df_continents = df_canada.groupby('Continent', axis = 0).sum()
```

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
Continent																					
Africa	3951	4363	3819	2671	2639	2650	3782	7494	7552	9894	...	27523	29188	28284	29890	34534	40892	35441	38083	38543	618948
Asia	31025	34314	30214	24696	27274	23850	28739	43203	47454	60256	...	159253	149054	133459	139894	141434	163845	146894	152218	155075	3317794
Europe	39760	44802	42720	24638	22287	20844	24370	46698	54726	60893	...	35955	33053	33495	34692	35078	33425	26778	29177	28691	1410947
Latin America and the Caribbean	13081	15215	16769	15427	13678	15171	21179	28471	21924	25060	...	24747	24676	26011	26547	26867	28818	27856	27173	24950	765148
Northern America	9378	10030	9074	7100	6661	6543	7074	7705	6469	6790	...	8394	9613	9463	10190	8995	8142	7677	7892	8503	241142
Oceania	1942	1839	1675	1018	878	920	904	1200	1181	1539	...	1585	1473	1693	1834	1860	1834	1548	1679	1775	55174

Types of Plots: Pie Chart

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

```
df_continents['Total'].plot(kind='pie')

plt.title('Immigration to Canada by Continent [1980-2013]')

plt.show()
```



Create a Pie Chart

Objective: Use **BigMartSalesData.csv** to plot a pie chart of the sales of the countries for the year 2011. Find the country which contributes to the highest sales.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

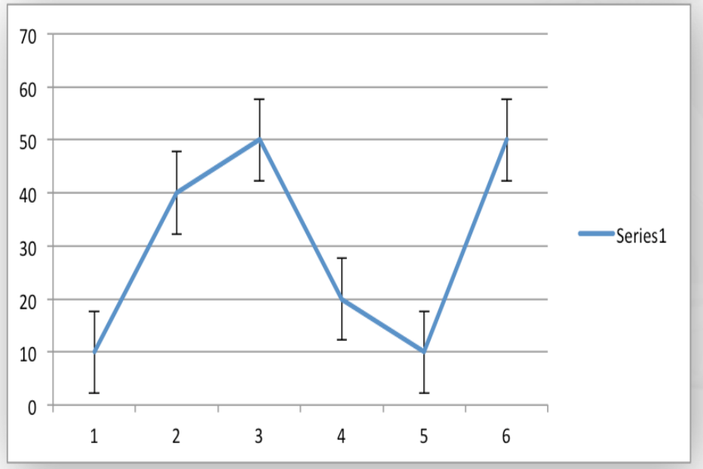
Types of Plots: Error Bar

- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plots
- Word Clouds
- Bar Charts
- Box Plots
- Waffle Charts

An error bar is used to graphically represent the variability of data. It is used mainly to identify errors. It builds confidence about the data analysis by revealing the statistical difference between the two groups of data.

Advantages of error bar:

- Shows the variability in data and indicates the errors
- Depicts precision in the data analysis
- Demonstrates how well a function and model are used in the data analysis
- Describes the underlying data



Create an Error Bar

Objective: Use the given data to plot the error bar:

Let the x-axis data points and y-axis data points be x = [1,2,3,4], y = [20, 21, 20.5, 20.8]

- Draw a simple plot and configure the line and markers in the plot
- Configure the axes, provide title to the graph, and label the x axis and y axis
- Give error bar if y_error = [0.12, 0.13, 0.2, 0.1]
- Define width and height as figsize=(4,5) DPI, adjust plot dpi=100, and change font size to 14

Access: To execute the practice, follow these steps:

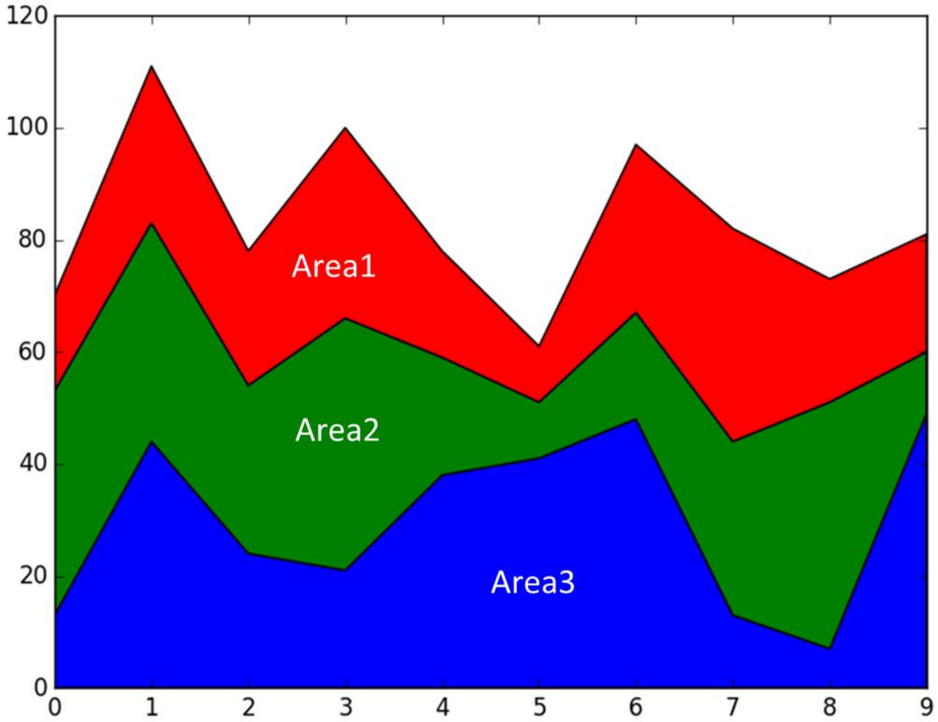
- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

Types of Plots: Area Plot

- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plot
- Word Clouds
- Bar Charts
- Box Plots
- Waffle Charts

Area (also known as area chart or area graph) is based on the line plot. This plot is commonly used to represent cumulated totals using numbers or percentages over time.



Types of Plots: Area Plot

```
df_canada.sort_values(['Total'], ascending = False, axis = 0, inplace = True)
```

	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
Country																					
India	Asia	Southern Asia	Developing regions	8880	8670	8147	7338	5704	4211	7150	...	36210	33848	28742	28261	29456	34235	27509	30933	33087	691904
China	Asia	Eastern Asia	Developing regions	5123	6682	3308	1863	1527	1816	1960	...	42584	33518	27642	30037	29622	30391	28502	33024	34129	659962
United Kingdom of Great Britain and Northern Ireland	Europe	Northern Europe	Developed regions	22045	24796	20620	10015	10170	9564	9470	...	7258	7140	8216	8979	8876	8724	6204	6195	5827	551500
Philippines	Asia	South-Eastern Asia	Developing regions	6051	5921	5249	4562	3801	3150	4166	...	18139	18400	19837	24887	28573	38617	36765	34315	29544	511391
Pakistan	Asia	Southern Asia	Developing regions	978	972	1201	900	668	514	691	...	14314	13127	10124	8994	7217	6811	7468	11227	12603	241600

Types of Plots: Area Plot

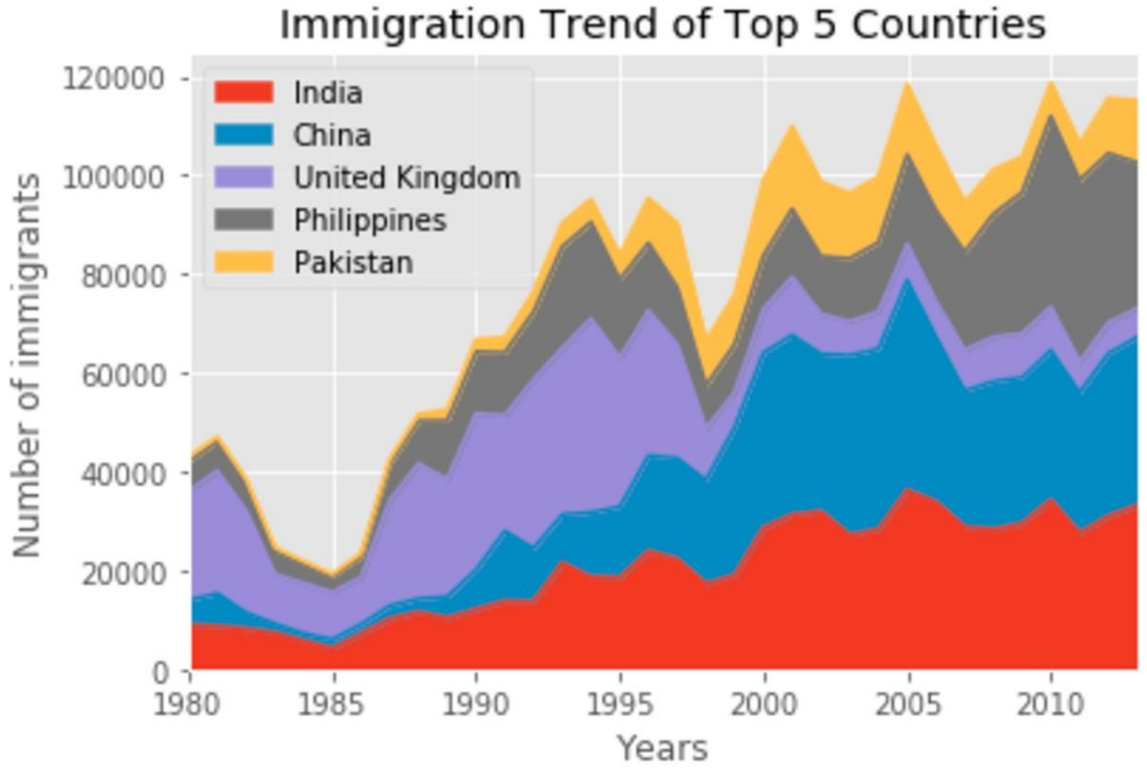
```
years = list(map(str, range(1980, 2014)))

df_canada.sort_values(['Total'], ascending = False, axis = 0, inplace = True)

df_top5 = df_canada.head()
df_top5 = df_top5[years].transpose()
```

Country	India	China	United Kingdom of Great Britain and Northern Ireland	Philippines	Pakistan
1980	8880	5123	22045	6051	978
1981	8670	6682	24796	5921	972
1982	8147	3308	20620	5249	1201
1983	7338	1863	10015	4562	900
1984	5704	1527	10170	3801	668

Types of Plots: Area Plot

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

```
df_top5.plot(kind='area')

plt.title('Immigration trend of top 5 countries')
plt.ylabel('Number of immigrants')
plt.xlabel('Years')

plt.show()
```



Area Chart to Display the Skills of the Players

Objective: Use **fifa-data.csv** dataset to create an area chart of the skills like SlidingTackle and StandingTackle of the players.

Access: To execute the practice, follow these steps:

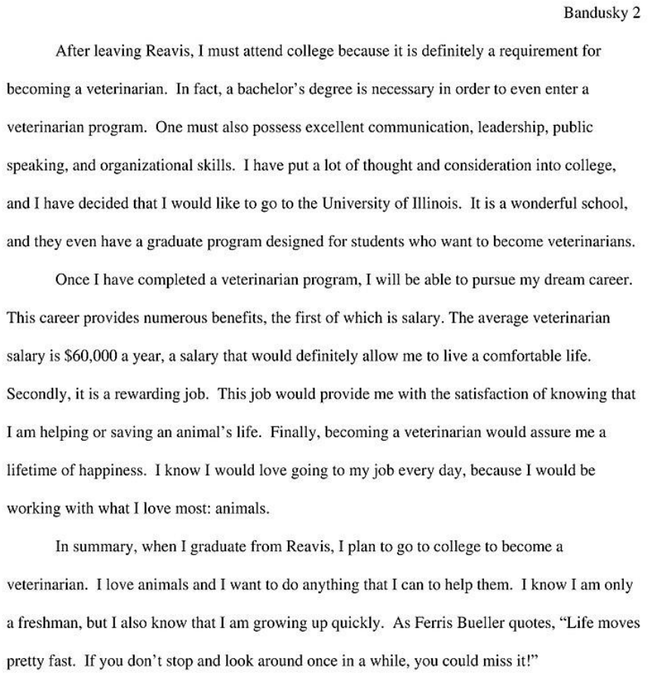
- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

Types of Plots: Word Cloud

- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plots
- Word Cloud
- Bar Charts
- Box Plots
- Waffle Charts

A word cloud is a depiction of the frequency of different words in some textual data.



Bandusky 2

After leaving Reavis, I must attend college because it is definitely a requirement for becoming a veterinarian. In fact, a bachelor's degree is necessary in order to even enter a veterinarian program. One must also possess excellent communication, leadership, public speaking, and organizational skills. I have put a lot of thought and consideration into college, and I have decided that I would like to go to the University of Illinois. It is a wonderful school, and they even have a graduate program designed for students who want to become veterinarians.

Once I have completed a veterinarian program, I will be able to pursue my dream career. This career provides numerous benefits, the first of which is salary. The average veterinarian salary is $60,000 a year, a salary that would definitely allow me to live a comfortable life. Secondly, it is a rewarding job. This job would provide me with the satisfaction of knowing that I am helping or saving an animal's life. Finally, becoming a veterinarian would assure me a lifetime of happiness. I know I would love going to my job every day, because I would be working with what I love most: animals.

In summary, when I graduate from Reavis, I plan to go to college to become a veterinarian. I love animals and I want to do anything that I can to help them. I know I am only a freshman, but I also know that I am growing up quickly. As Ferris Bueller quotes, "Life moves pretty fast. If you don't stop and look around once in a while, you could miss it!"

Create a Word Cloud of a Random Data

Objective: Install word cloud using **pip install wordcloud** and generate a random word cloud.

Access: To execute the practice, follow these steps:

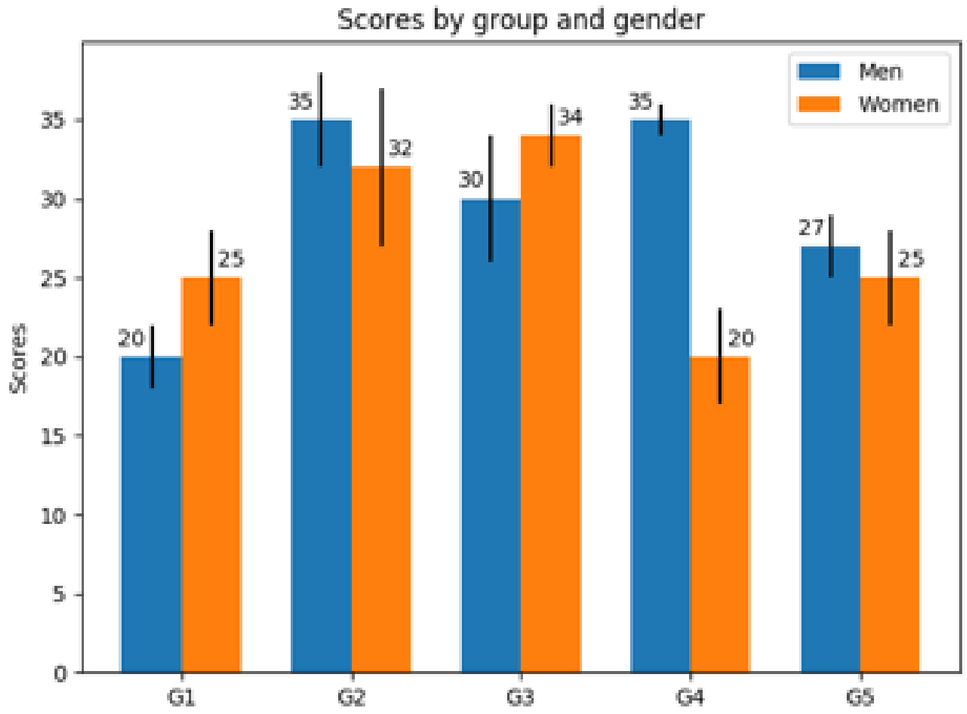
- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

Types of Plots: Bar Chart

- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plots
- Word Clouds
- **Bar Chart**
- Box Plots
- Waffle Charts

Unlike a histogram, a bar chart is commonly used to compare the values of a variable at a given point in time.

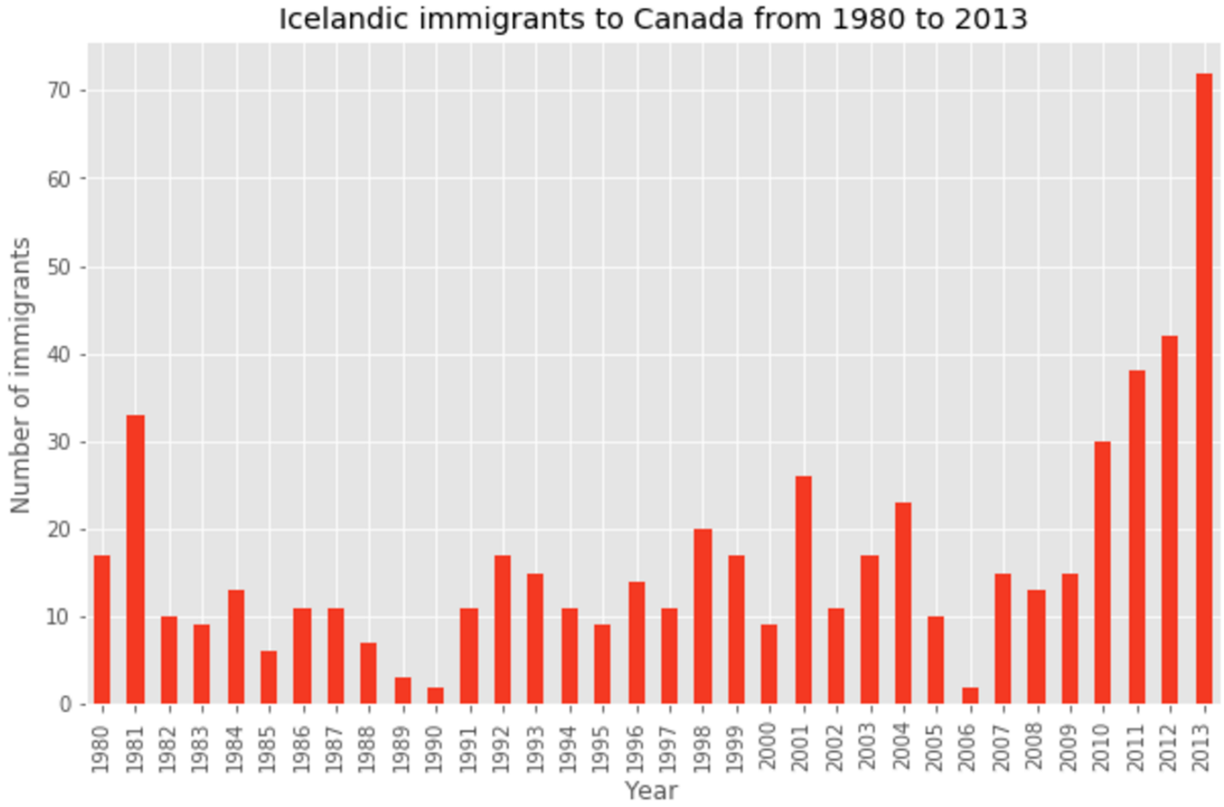


Types of Plots: Bar Chart

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

```
years = list(map(str, range(1980, 2014)))

df_iceland = df_canada.loc['Iceland', years]
```

```
df_iceland.plot(kind='bar')

plt.title('Icelandic immigrants to Canada from 1980 to 2013')
plt.xlabel('Year')
plt.ylabel('Number of immigrants')

plt.show()
```



Create a Bar Chart

Objective: Use **fifa-data.csv** dataset and create a bar chart to analyze the agility skill of any ten players.

Access: To execute the practice, follow these steps:

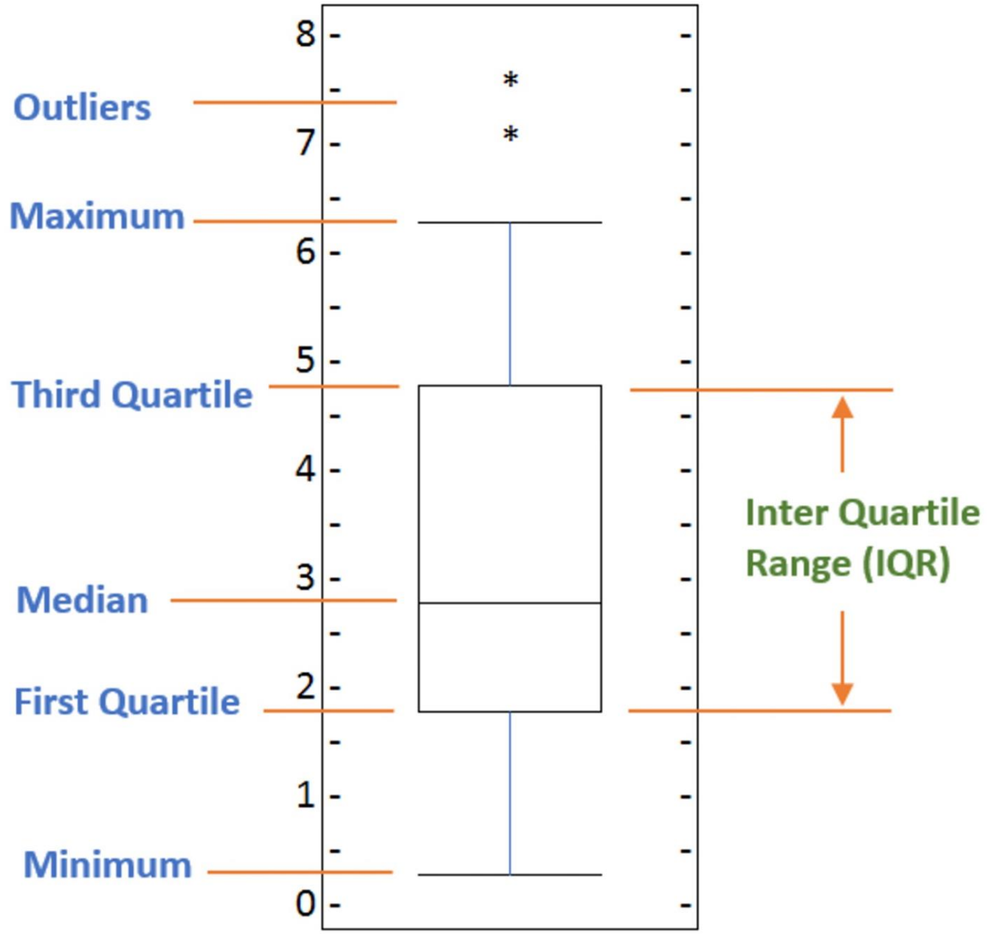
- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

Types of Plots: Box Plot

- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plots
- Word Clouds
- Bar Chart
- Box Plots
- Waffle Charts

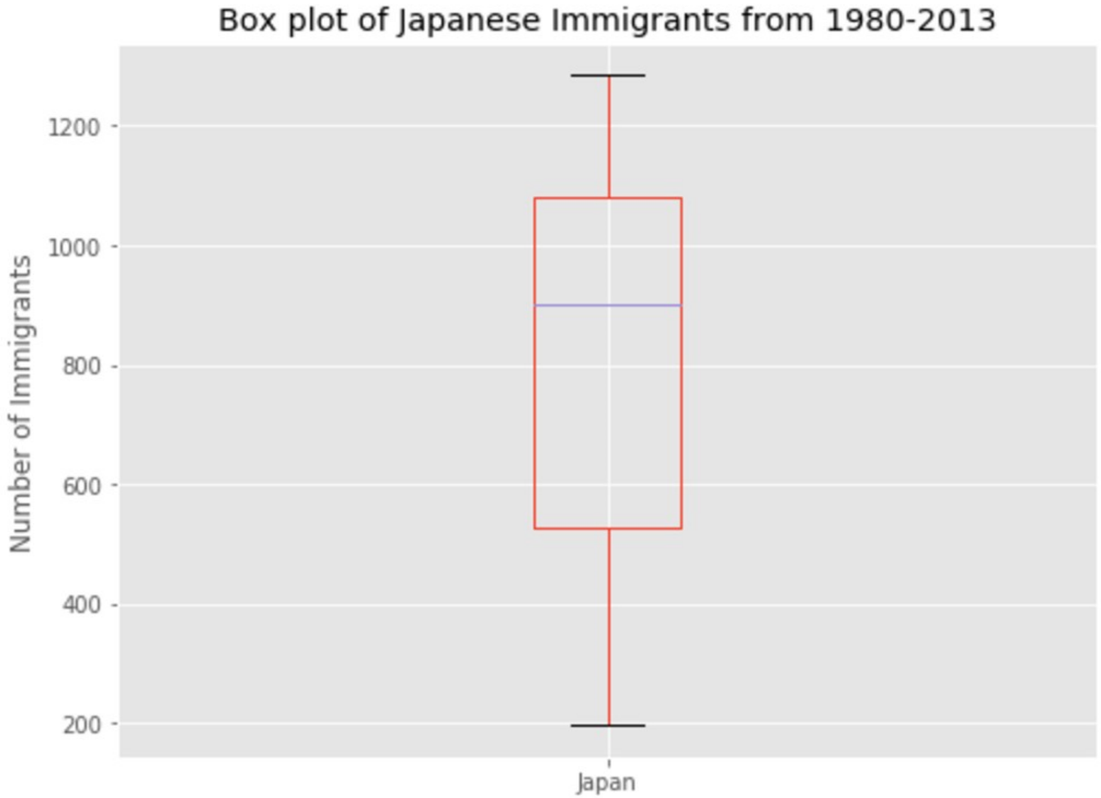
Box plots are used for graphical display of numerical data through their quartiles.



Types of Plots: Box Plot

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

```
df_japan = df_canada.loc[['Japan'], years].transpose()

df_japan.plot(kind='box')

plt.title('Box plot of Japanese Immigrants from 1980-2013')
plt.ylabel("Number of Immigrants")

plt.show()
```



Create Box Plots

Objective: Use **iris.csv** dataset to create box plots using the following inputs:

1. Analyze the petal lengths of all the varieties of flowers
2. Study the distribution of several numerical variables, let's say sepal length and sepal width

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

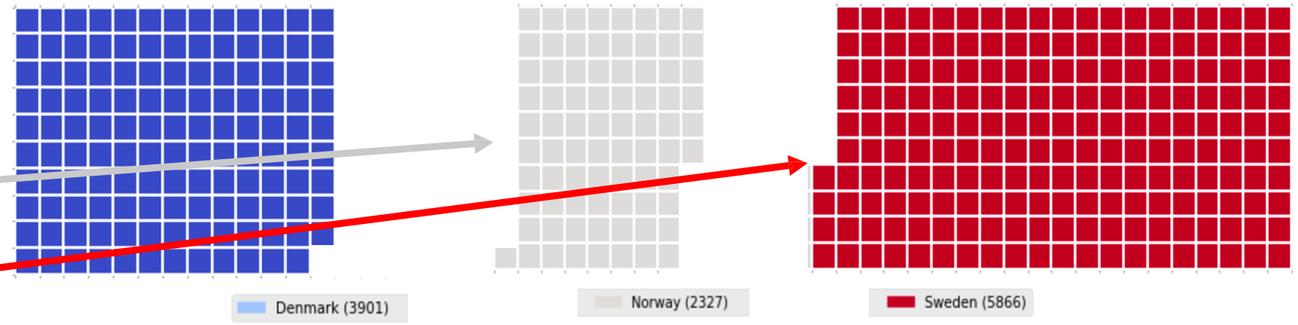
ASSISTED PRACTICE

Types of Plots: Waffle Chart

- Histogram
- Scatter Plot
- Heat Map
- Pie Chart
- Error Bar
- Area plots
- Word Clouds
- Bar Chart
- Box Plots
- Waffle Charts

A waffle chart is an interesting visualization that is normally created to display progress toward goals.

Country	Total Immigrants
Denmark	3901
Norway	2327
Sweden	5866



Create a Waffle Chart

Objective: Use **Immigrants to Canada.csv** dataset to create a waffle chart using REG as a field.

Access: To execute the practice, follow these steps:

- Go to the **PRACTICE LABS** tab on your LMS
- Click the **START LAB** button
- Click the **LAUNCH LAB** button to start the lab

ASSISTED PRACTICE

DATA AND
ARTIFICIAL INTELLIGENCE

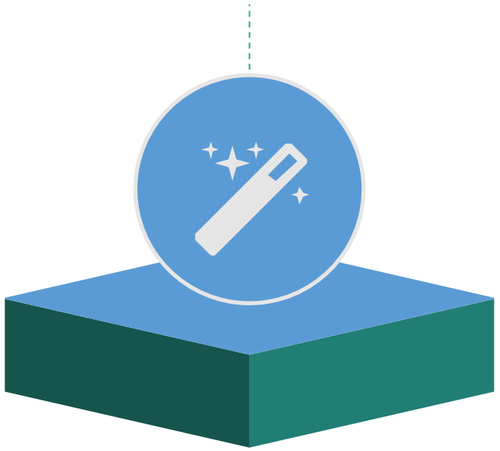
Seaborn and Regression Plots

Seaborn

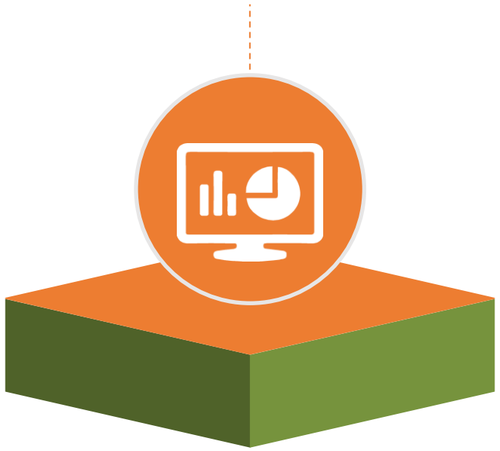
Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface to draw attractive statistical graphics.

Advantages of seaborn:

Possesses built-in themes for better visualizations



Has built-in statistical functions which reveal hidden patterns in the dataset



Has functions to visualize matrices of data

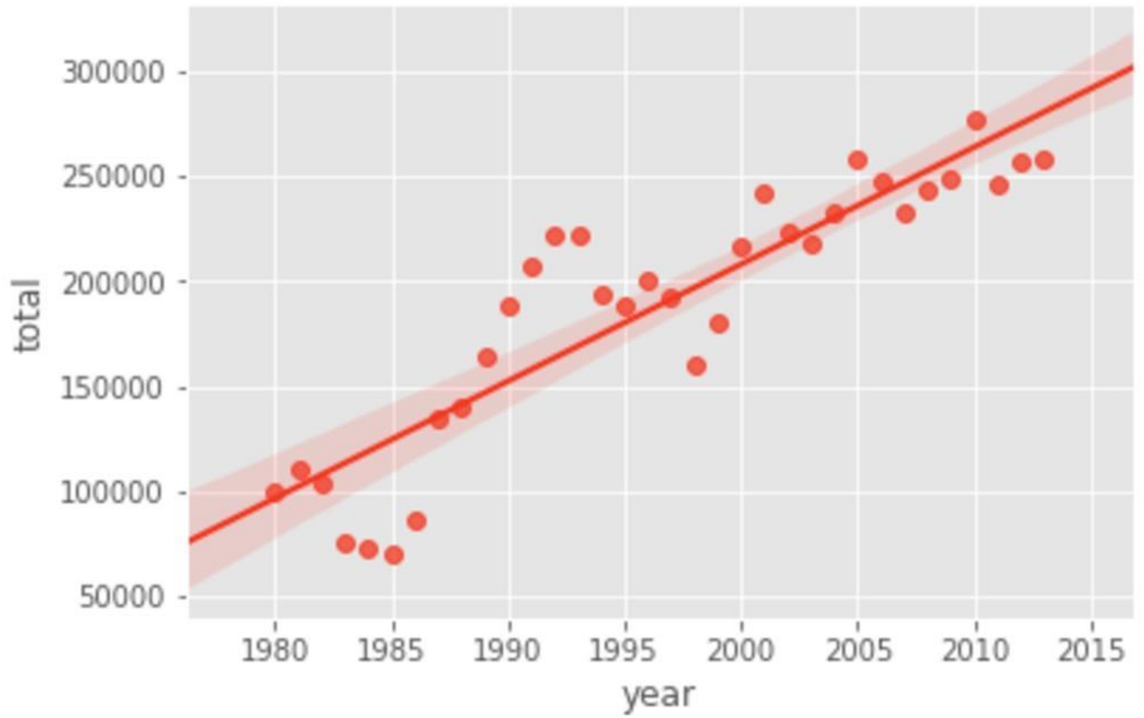
Regression Plots

A plot used to force fit independent variables against a dependent variable is a regression plot.

df_total

year	total
1980	99137
1981	110563
1982	104271
1983	75550
1984	73417
.	.
.	.

```
import seaborn as sns
ax = sns.regplot(x='year', y='total', data=df_tot)
```

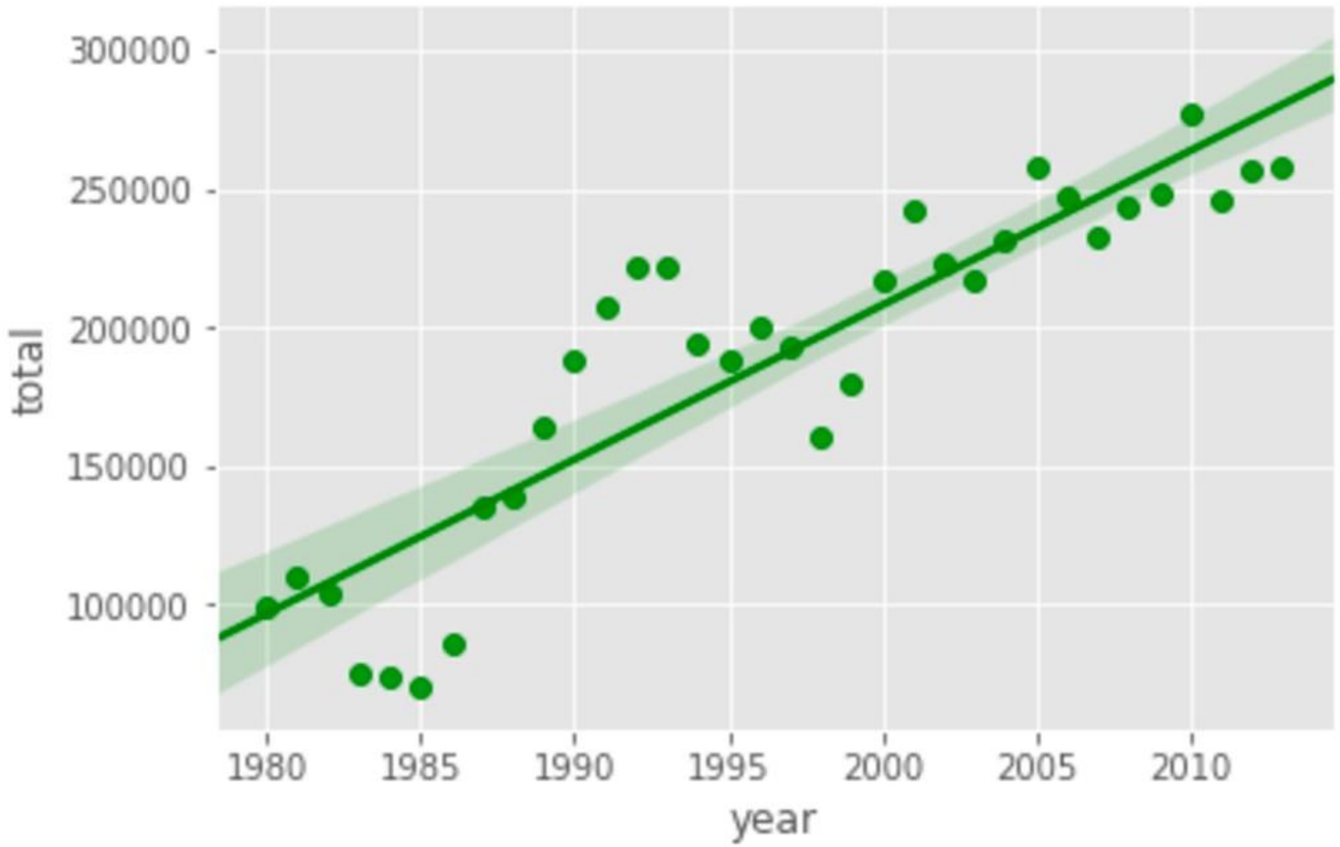


Regression Plots

df_total

year	total
1980	99137
1981	110563
1982	104271
1983	75550
1984	73417
.	.
.	.

```
import seaborn as sns
ax = sns.regplot(x='year', y='total', data=df_tot,
                 color='green')
```

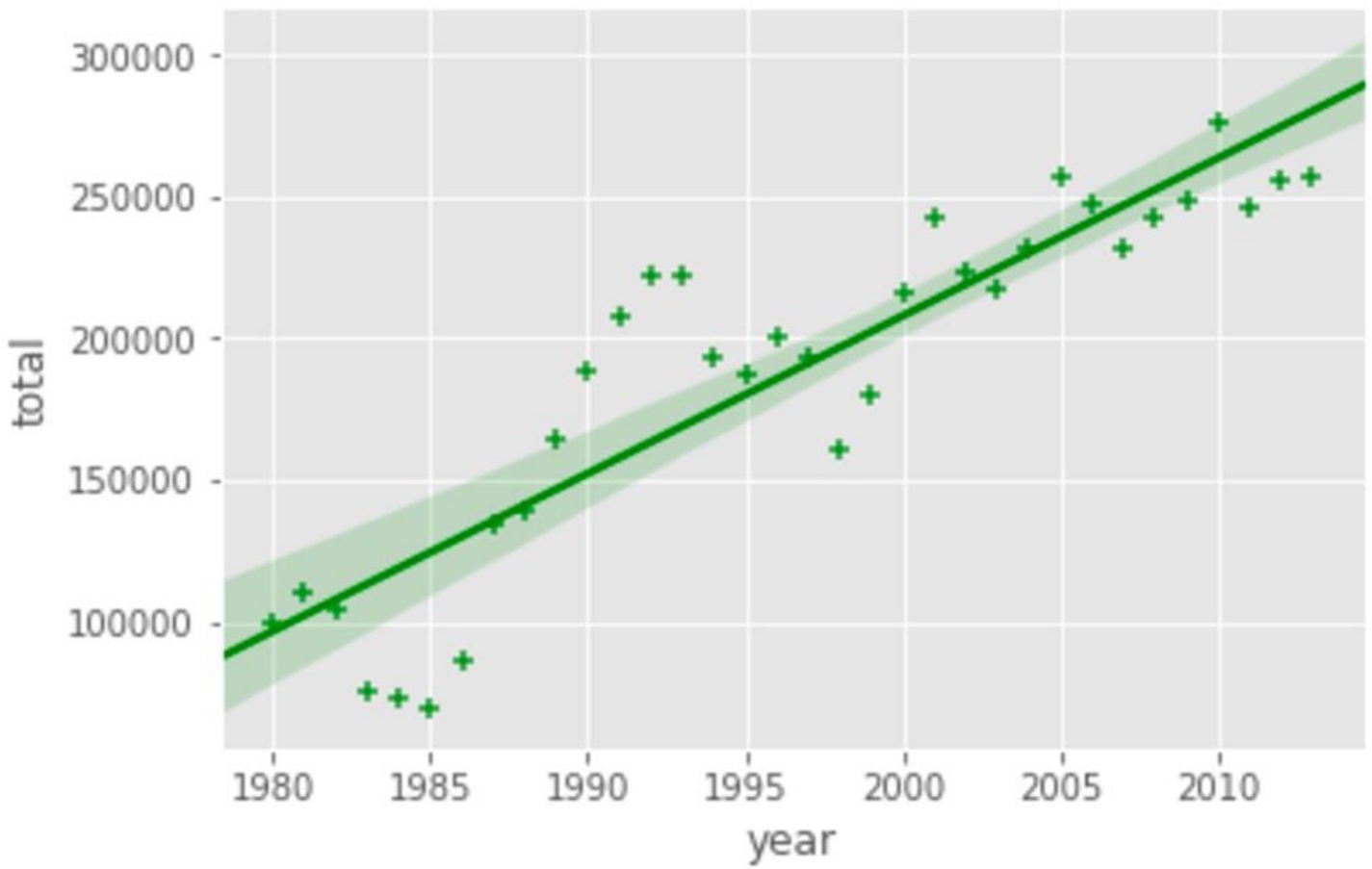


Regression Plots

df_total

year	total
1980	99137
1981	110563
1982	104271
1983	75550
1984	73417
. .	. .

```
import seaborn as sns
ax = sns.regplot(x='year', y='total', data=df_tot,
                 color='green', marker='+')
```



DATA AND
ARTIFICIAL INTELLIGENCE

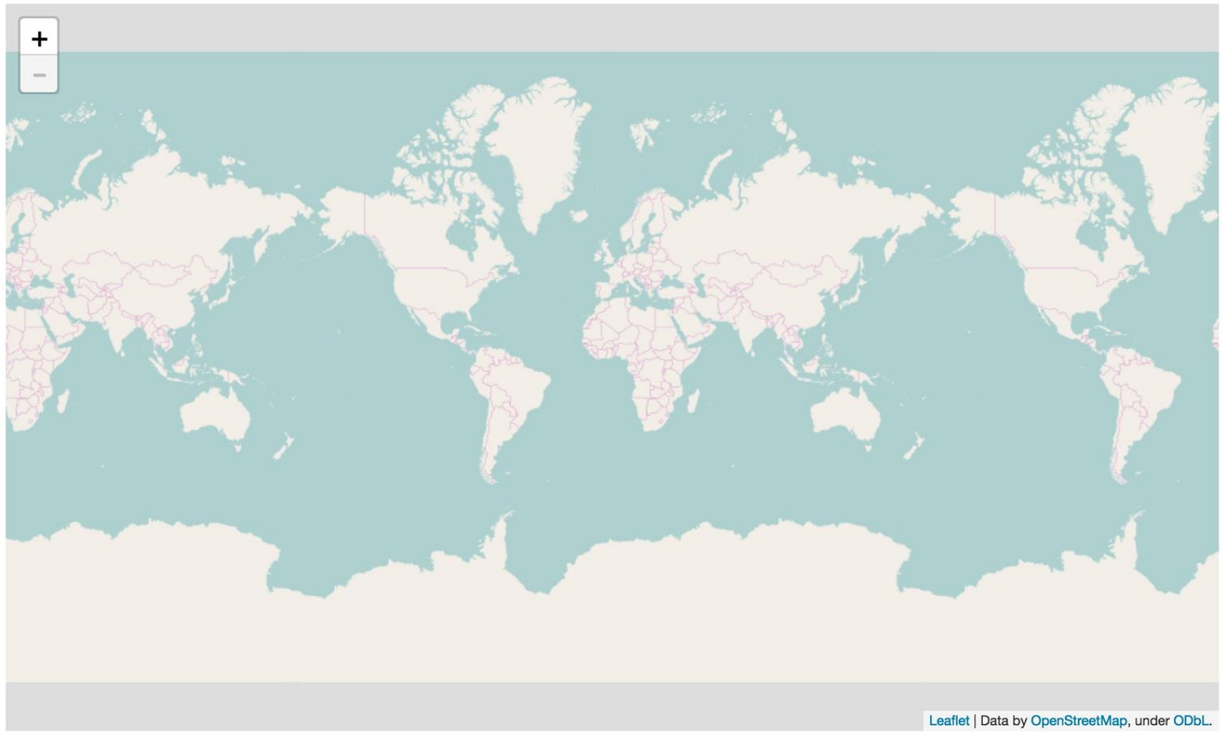
Introduction to Folium

What Is Folium?

- Folium is a powerful Python library that helps you create several types of Leaflet maps.
- It enables both binding of data to a map for choropleth visualizations as well as passing visualizations as markers on the map.
- The library has a number of built-in tilesets from OpenStreetMap, Mapbox, and Stamen and supports custom tilesets with Mapbox API keys.

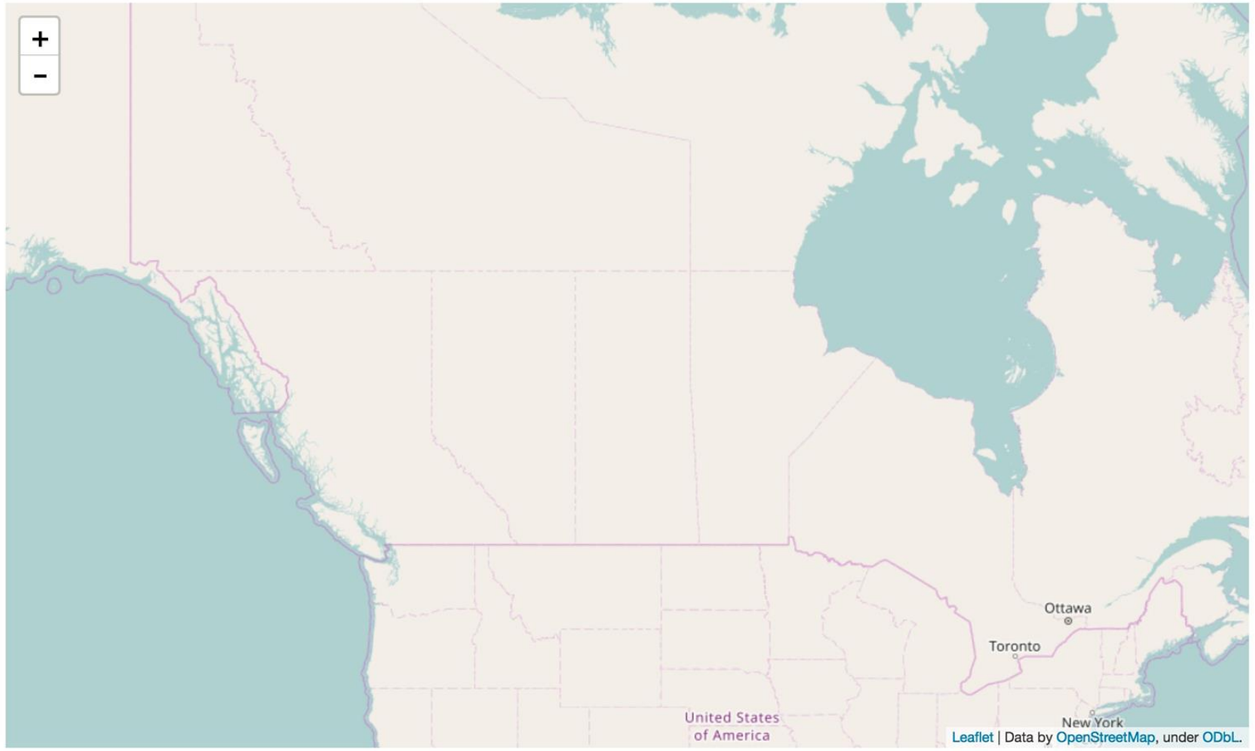
Creating a World Map

```
# define the world map
world_map = folium.Map()

# display world map
world_map
```



Creating a Map of Canada

```
# define the world map centered around
# Canada with a low zoom level
world_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4
)

# display world map
world_map
```



Map Styles: Stamen Toner

```
# create a Stamen Toner map of
# the world centered around Canada
world_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4,
    tiles='Stamen Toner'
)

# display map
world_map
```

Map Styles: Stamen Terrain

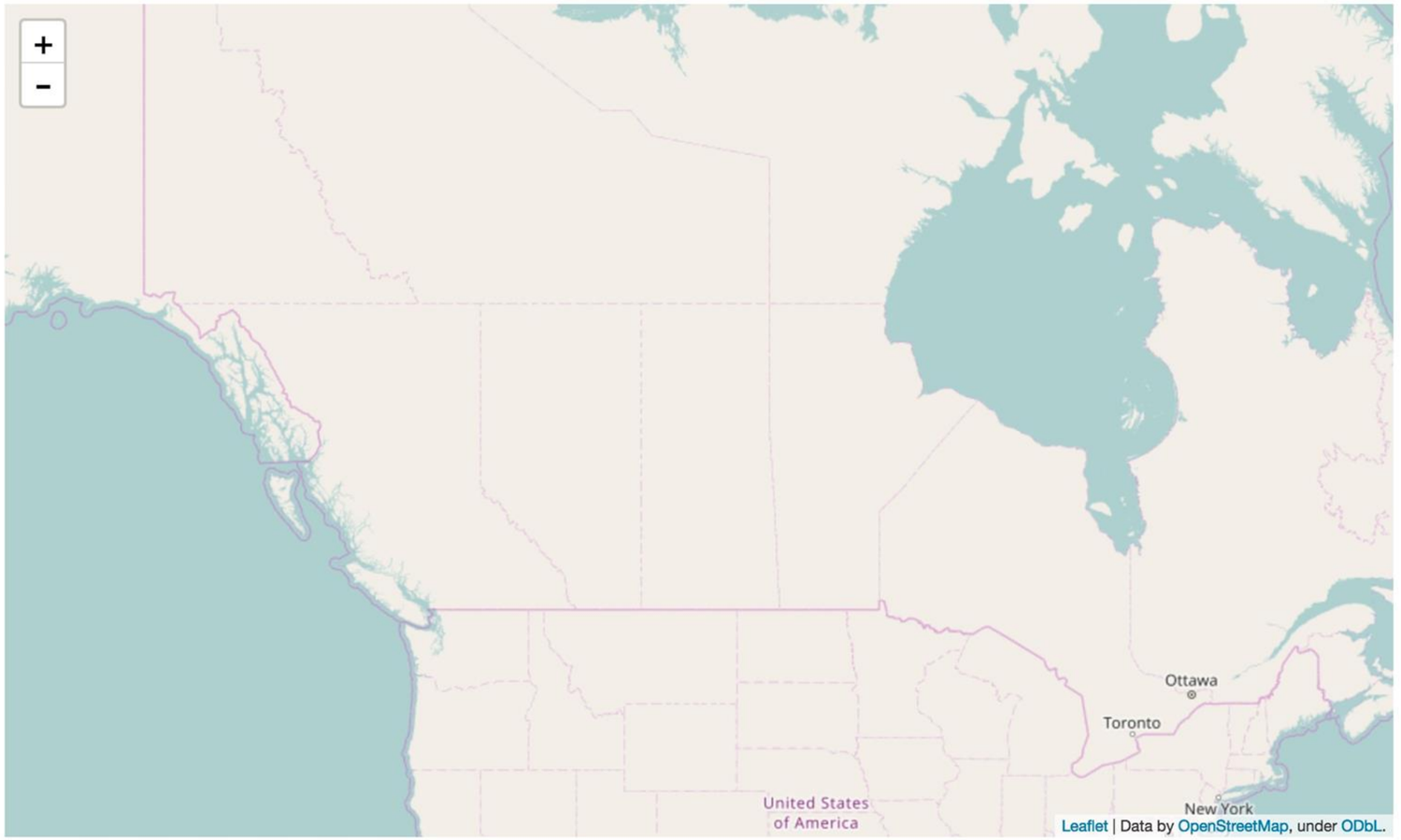
```
# create a Stamen Toner map of
# the world centered around Canada
world_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4,
    tiles='Stamen Terrain'
)

# display map
world_map
```

DATA AND
ARTIFICIAL INTELLIGENCE

Maps with Markers

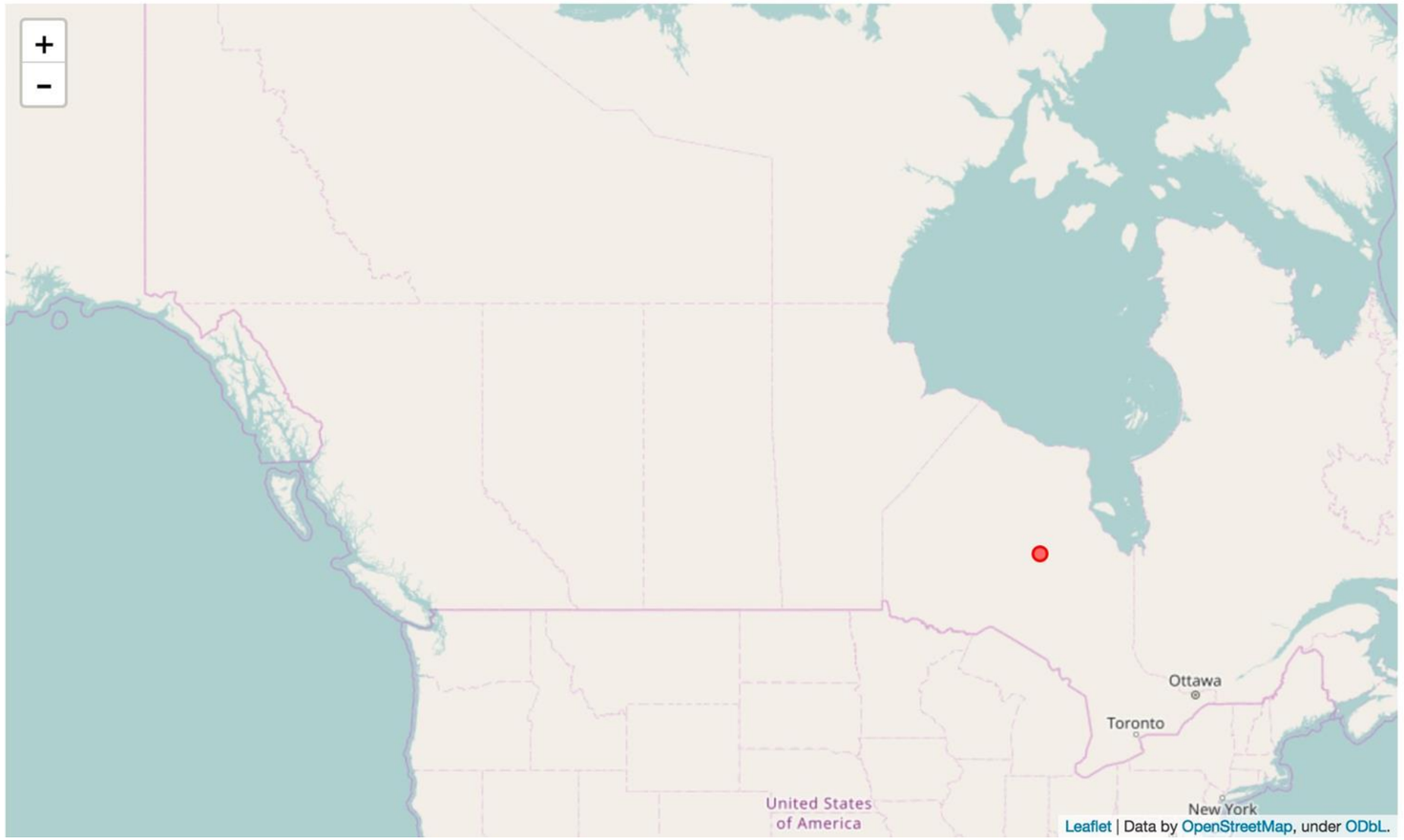
Add a Marker

```
# generate map of Canada
canada_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4
)
```



Add a Marker

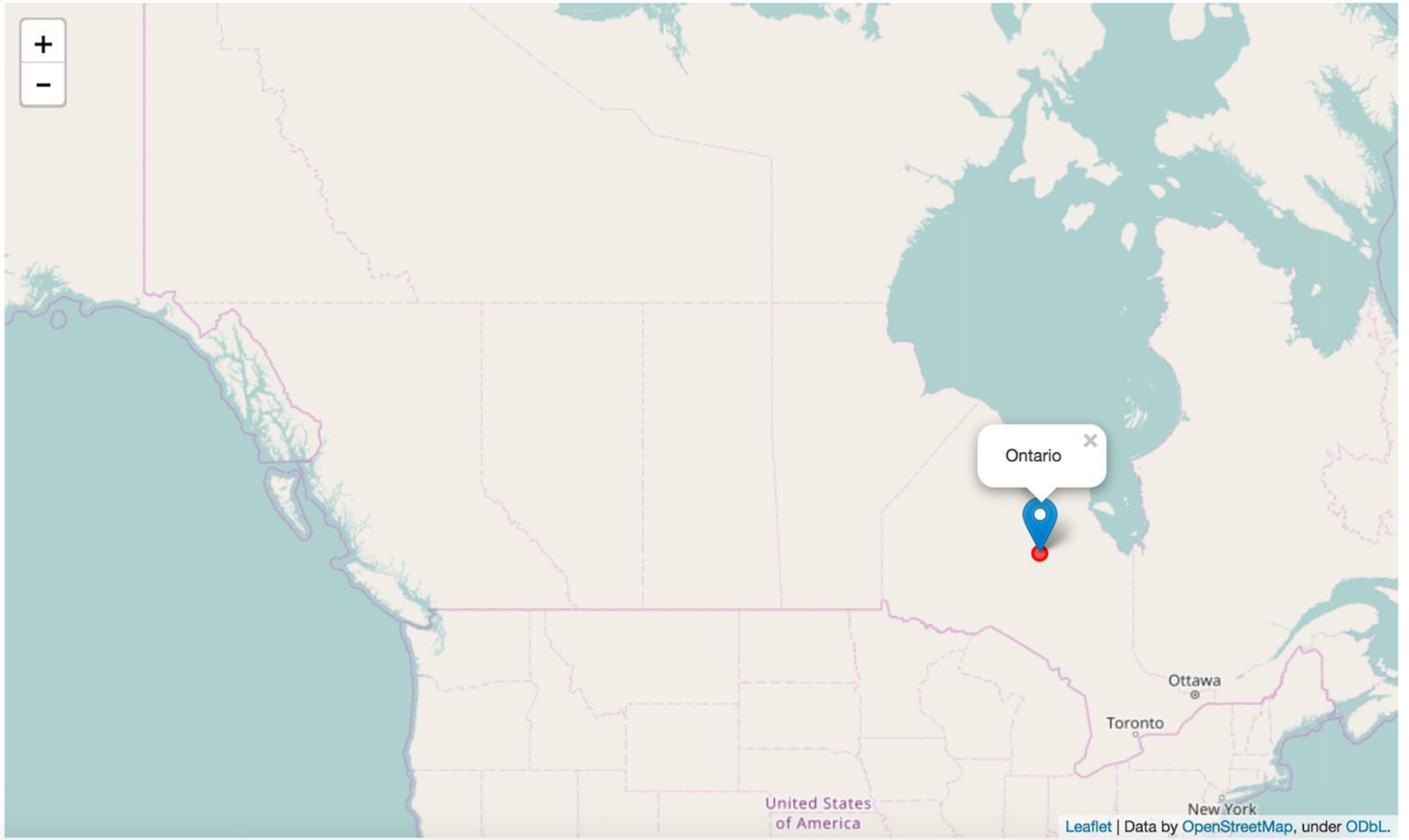
```
# generate map of Canada
canada_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4
)

## add a red marker to Ontario

# create a feature group
ontario = folium.map.FeatureGroup()

# style the feature group
ontario.add_child(
    folium.features.CircleMarker(
    [51.25, -85.32], radius = 5,
    color = "red", fill_color = "Red"
    )
)

# add the feature group to the map
canada_map.add_child(ontario)
```




```
# display map
canada_map
```

Label the Marker

```
# generate map of Canada
canada_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4
)

## add a red marker to Ontario

# create a feature group
ontario = folium.map.FeatureGroup()

# style the feature group
ontario.add_child(
    folium.features.CircleMarker(
    [51.25, -85.32], radius = 5,
    color = "red", fill_color = "Red"
    )
)

# add the feature group to the map
canada_map.add_child(ontario)

# label the marker
folium.Marker([51.25, -85.32],
    popup='Ontario').add_to(canada_map)

# display map
canada_map
```



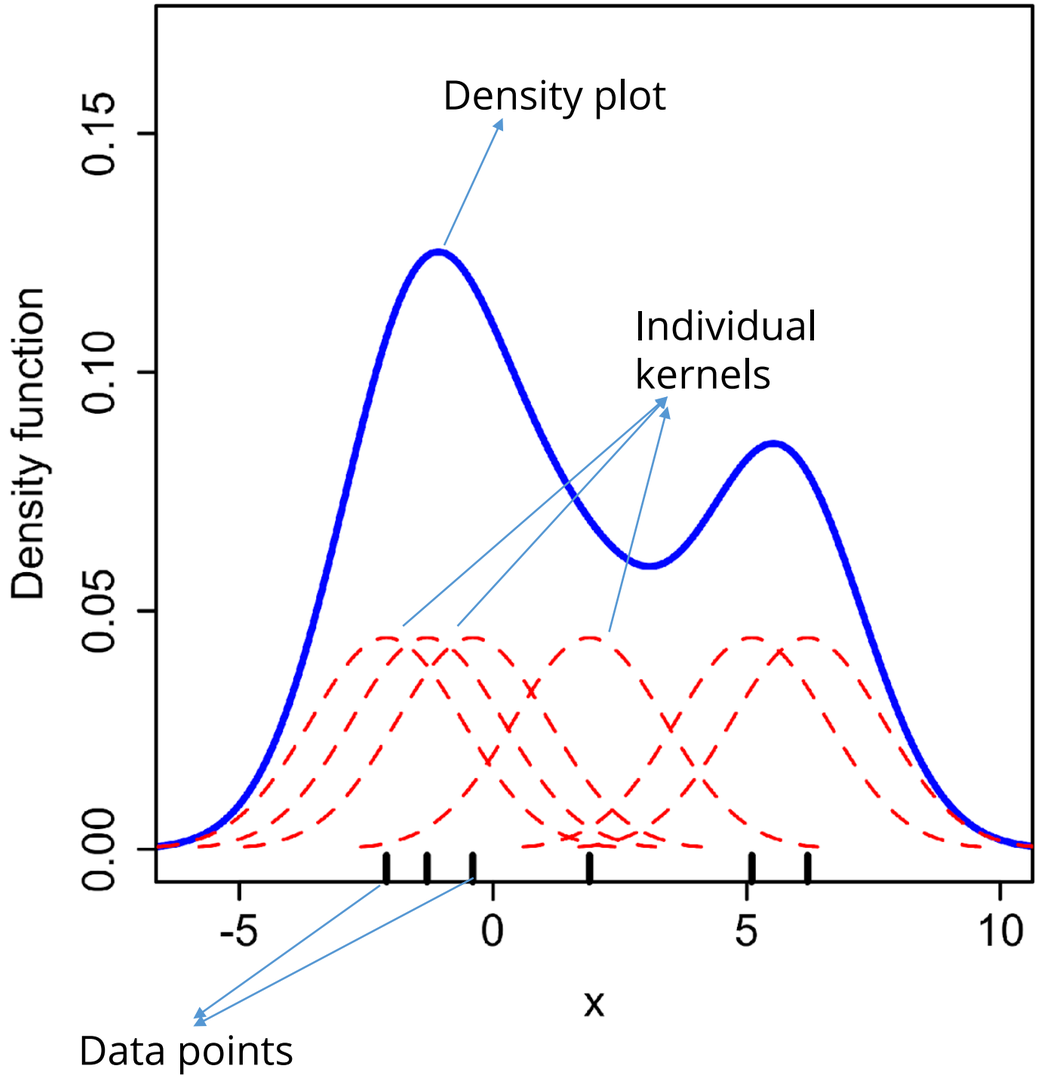
DATA AND
ARTIFICIAL INTELLIGENCE

Kernel Density Estimate Plots

Kernel Density Estimate Plots

A density plot is a smooth and continuous version of a histogram estimated from the data. It shows the distribution of a numerical variable.

A kernel density estimate (KDE) is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different data points in a continuous variable.



KDE with Pandas and Seaborn

A diabetes dataset and KDE plot to visualize the insights of the dataset.

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter
0	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300100	0.147100	0.2419	0.07871	...	17.33	184.60
1	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900	0.070170	0.1812	0.05667	...	23.41	158.80
2	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197400	0.127900	0.2069	0.05999	...	25.53	152.50
3	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241400	0.105200	0.2597	0.09744	...	26.50	98.87
4	20.290	14.34	135.10	1297.0	0.10030	0.13280	0.198000	0.104300	0.1809	0.05883	...	16.67	152.20
5	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157800	0.080890	0.2087	0.07613	...	23.75	103.40
6	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112700	0.074000	0.1794	0.05742	...	27.66	153.20

About the dataset:

Target: malignant(0), benign(1)
Dimensions: 569 rows × 31 columns

One-dimensional KDE Plot

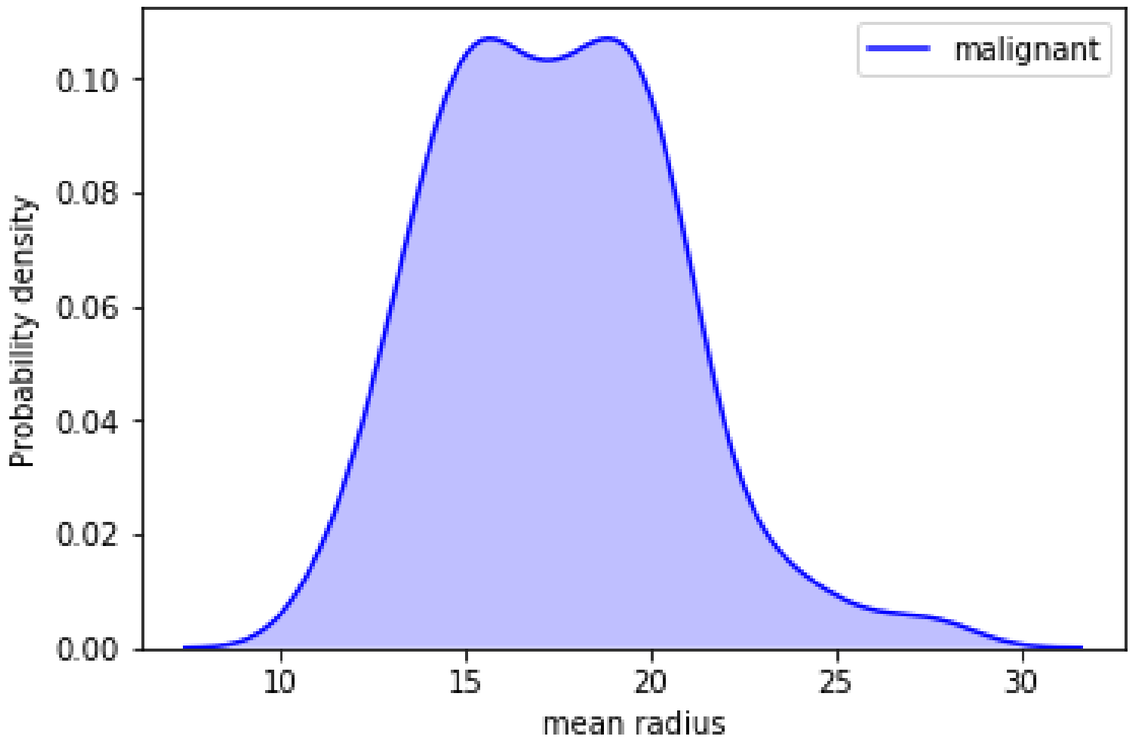
Visualize the probability distribution of a sample against a single continuous attribute.

```
# importing the required libraries
from sklearn import datasets
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# setting up the DataFrame
cancer = datasets.load_breast_cancer()
cancer_df = pd.DataFrame(cancer.data, columns=['mean radius', 'mean
texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'])
cancer_df['Target'] = cancer.target
```

One-dimensional KDE Plot

Visualize the probability distribution of a sample against a single continuous attribute.

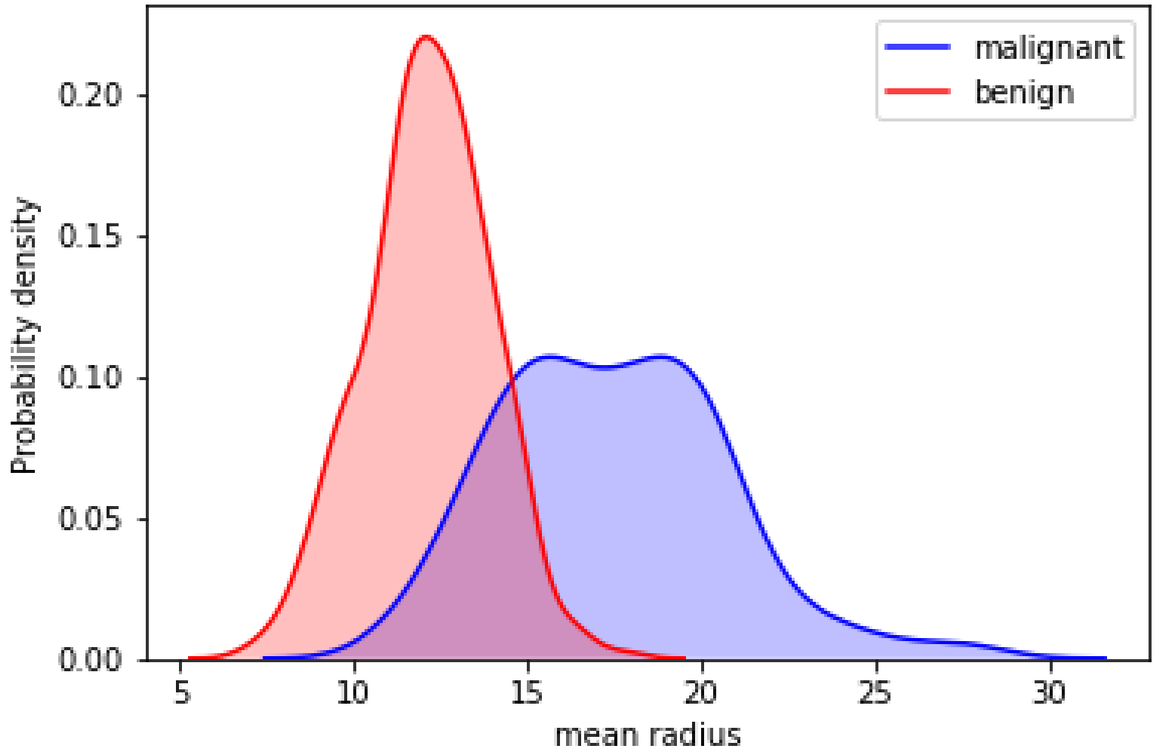
```
cancer_df['Target'].replace([0], 'malignant',
inplace=True)
cancer_df['Target'].replace([1], 'benign',
inplace=True)

#Plotting the KDE plot
sns.kdeplot(cancer_df.loc[(cancer_df['Target']==
'malignant'),
            'mean radius'], color='b',
shade=True, Label='malignant')
plt.xlabel('mean radius')
plt.ylabel('Probability density')
```



One-dimensional KDE Plot

Visualize the probability distribution of multiple samples in a single plot.

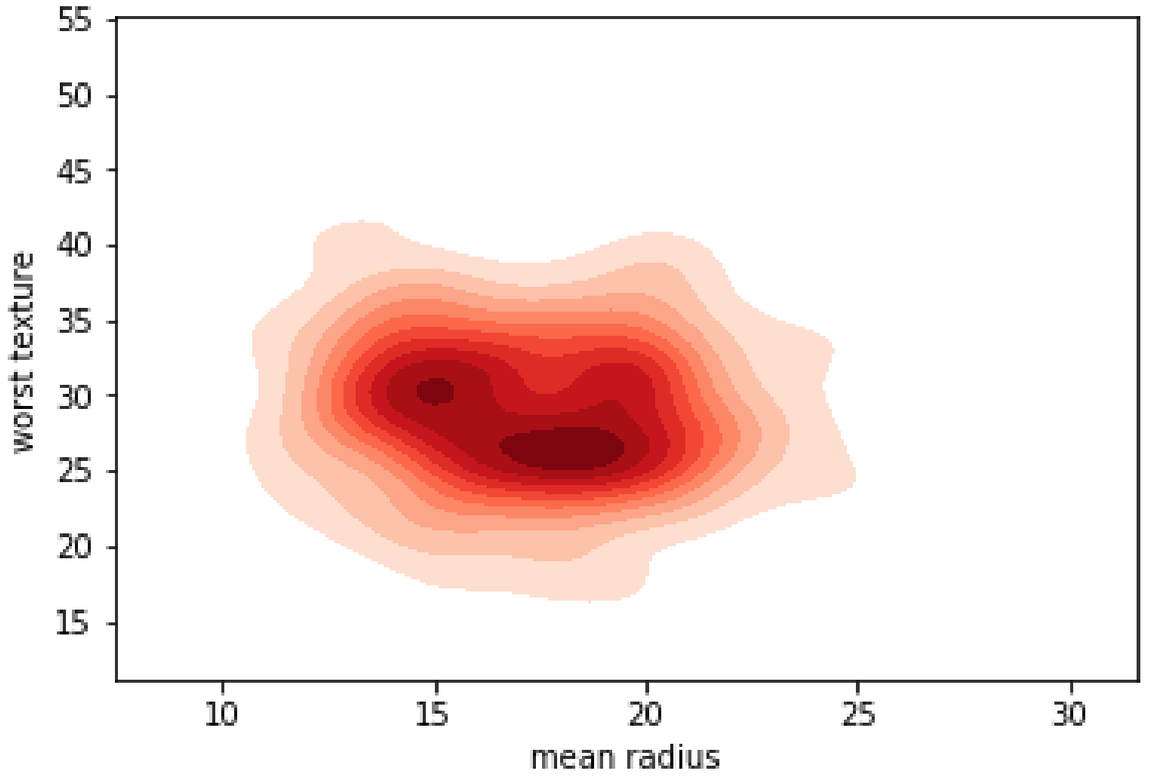
```
#Plotting the KDE plot
sns.kdeplot(cancer_df.loc[(cancer_df['Target']==
'malignant'),
            'mean radius'], color='b',
shade=True, Label='malignant')
sns.kdeplot(cancer_df.loc[(cancer_df['Target']==
'benign'),
            'mean radius'], color='r',
shade=True, Label='benign')
plt.xlabel('mean radius')
plt.ylabel('Probability density')
```



Two-dimensional KDE Plot

Visualize the probability distribution of a sample against multiple continuous attributes.

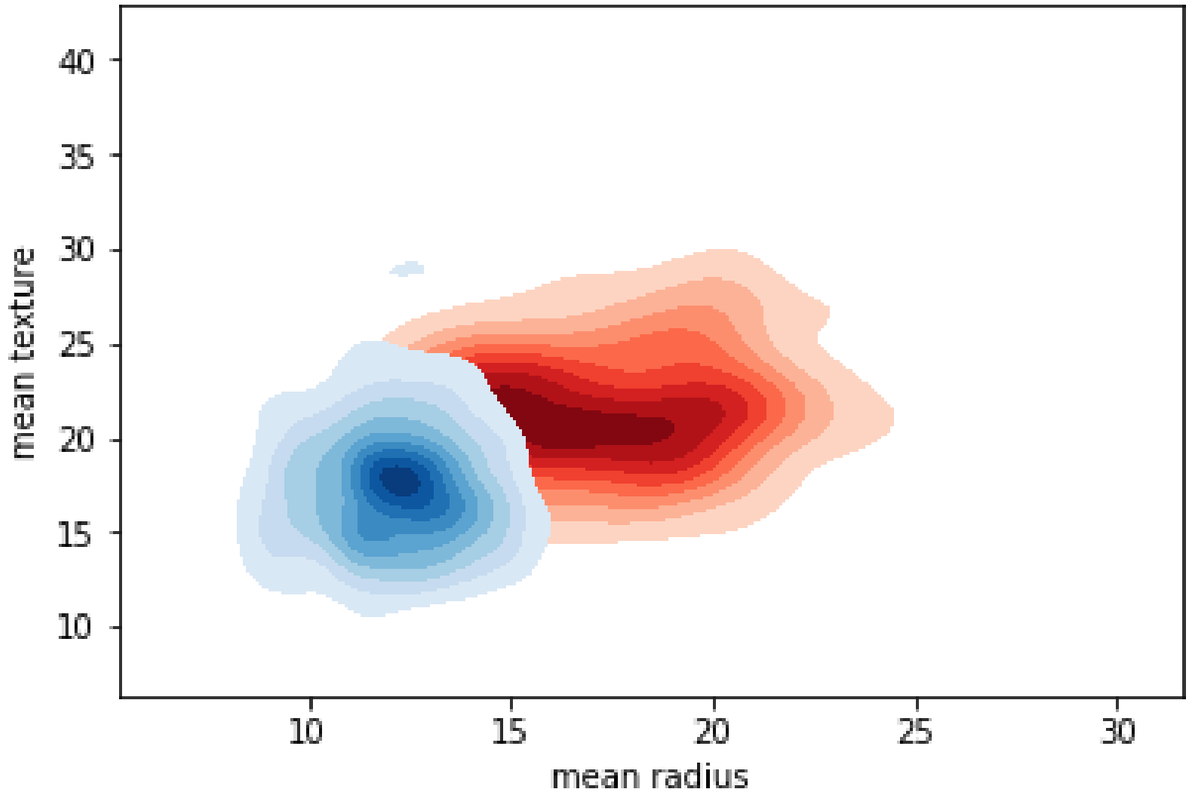
```
# Setting up the samples
malignant =
cancer_df.query("Target=='malignant'")
benign = cancer_df.query("Target=='benign'")

# Plotting the KDE Plot
sns.kdeplot(malignant['mean radius'],
            malignant['mean texture'],
            color='r', shade=True,
Label='malignant',
            cmap="Reds", shade_lowest=False)
```



Two-dimensional KDE Plot

visualize the probability distribution of multiple samples in a single plot.

```
# Plotting the KDE Plot
sns.kdeplot(malignant['mean radius'],
            malignant['mean texture'],
            color='r', shade=True,
Label='malignant',
            cmap="Reds", shade_lowest=False)
# Plotting the KDE Plot
sns.kdeplot(benign['mean radius'],
            benign['mean texture'],
            color='b', shade=True,
Label='benign',
            cmap="Blues", shade_lowest=False)
```



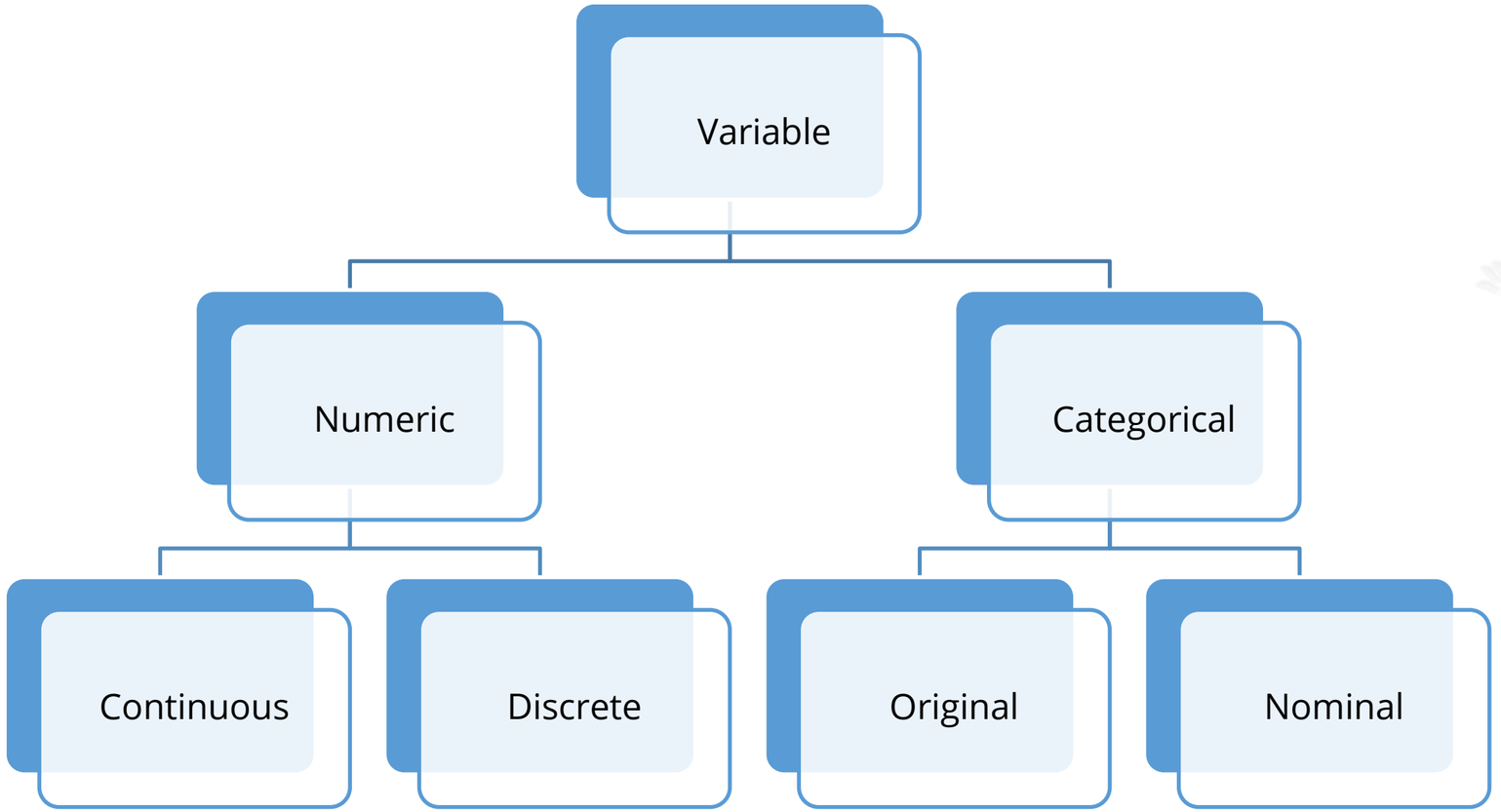
DATA AND
ARTIFICIAL INTELLIGENCE

Analyzing Variables Individually

Types of Variables

There are two types of variables: numerical variables and categorical variables.

Numerical variables: variables for which the values are numbers
Categorical variables: variables for which the values are categories



Analyzing Variables Individually

Redefining the housing DataFrame:

```
In [7]: numerical_vars = ['SalePrice','LotArea', 'OverallQual', 'OverallCond',
                          'YearBuilt', '1stFlrSF', '2ndFlrSF', 'BedroomAbvGr']
        categorical_vars = ['MSZoning', 'LotShape', 'Neighborhood', 'CentralAir', 'SaleCondition', 'MoSold', 'YrSold']

In [8]: housing = housing[numerical_vars+categorical_vars]

In [9]: housing.shape
```

Using the shape attribute to see the size of the new DataFrame

```
Out[9]: (1460, 15)
```

Understanding the Main Variable

Let’s understand the main variable, the SalePrice of the housing dataset.

The first thing to do with a categorical variable is to know their descriptive statistics:

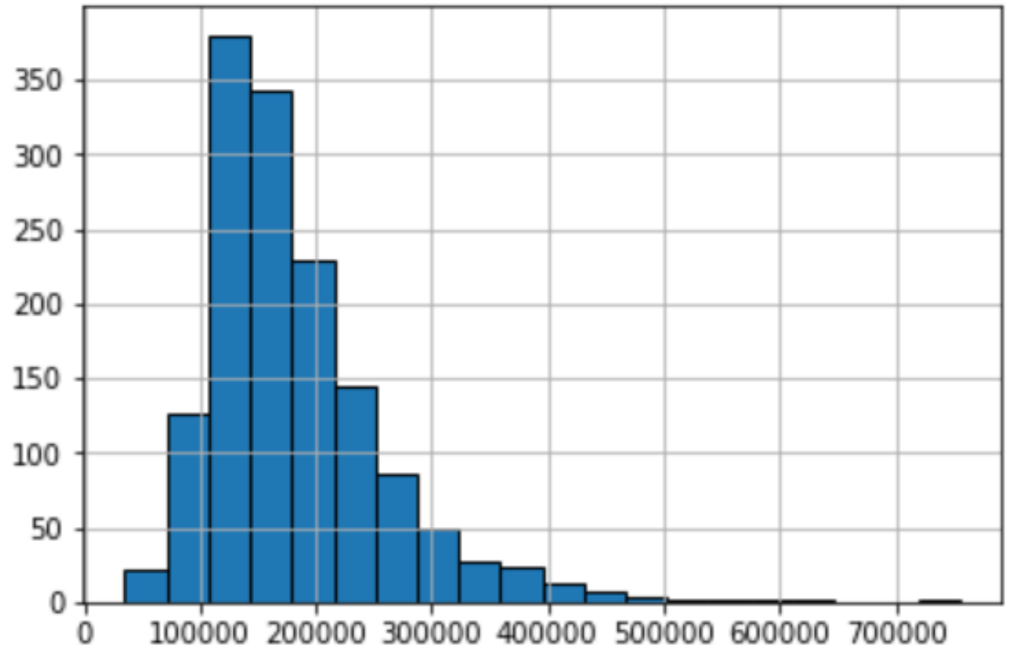
```
In [10]: #descriptive statistics summary
         housing['SalePrice'].describe()

Out[10]: count      1460.000000
         mean     180921.195890
         std       79442.502883
         min       34900.000000
         25%      129975.000000
         50%      163000.000000
         75%      214000.000000
         max      755000.000000
         Name: SalePrice, dtype: float64
```

Range of values for the variable

Understanding the Main Variable

Let's understand the variable visually with a histogram:

In [11]:
```
housing['SalePrice'].hist(edgecolor='black', bins=20);
```



In [12]:
```
#skewness and kurtosis
print("Skewness: {:0.3f}".format(housing['SalePrice'].skew()))
print("Kurtosis: {:0.3f}".format(housing['SalePrice'].kurt()))
```

```
Skewness: 1.883
Kurtosis: 6.536
```

Key Takeaways

You are now able to:

- Explain data visualization and its importance in today's world
- Understand why Python is considered one of the best data visualization tools
- Describe matplotlib and its data visualization features in Python
- List the types of plots and the steps involved in creating these plots

Knowledge Check

Knowledge Check

1

Which of the following methods is used to set the title?

a. Plot()

b. Plt.title()

c. Plot.title()

d. Title()

Knowledge Check

1

Which of the following methods is used to set the title?

a. Plot()

b. Plt.title()

c. Plot.title()

d. Title()

The correct answer is **b**

Plt.title() is used to set the title.

Knowledge Check

2

Which of the following methods is used to adjust the distances between the subplots?

a. plot.subplots_adjust()

b. plt.subplots_adjust()

c. subplots_adjust()

d. plt.subplots.adjust()

Knowledge Check

2

Which of the following methods is used to adjust the distances between the subplots?

a. plot.subplots_adjust()

b. plt.subplots_adjust()

c. subplots_adjust()

d. plt.subplots.adjust()

The correct answer is **b**

plt.subplots_adjust() used to adjust the distances between the subplots.

Knowledge Check

3

Which of the following libraries needs to be imported to display the plot on Jupyter notebook?

a. %matplotlib

b. %matplotlib inline

c. import matplotlib

d. import style

Knowledge Check

3

Which of the following libraries needs to be imported to display the plot on Jupyter notebook?

a. %matplotlib

b. %matplotlib inline

c. import matplotlib

d. import style

The correct answer is **b**

To display the plot on Jupyter notebook “import‘%matplotlib inline.”

Knowledge Check

4

Which of the following keywords is used to decide the transparency of the plot line?

a. Legend

b. Alpha

c. Animated

d. Annotation

Knowledge Check

4

Which of the following keywords is used to decide the transparency of the plot line?

a. Legend

b. Alpha

c. Animated

d. Annotation

The correct answer is **c**

Alpha decides the line transparency in line properties while plotting line plot/ chart.

Knowledge Check

5

Which of the following plots is used to represent data in a two-dimensional manner?

a. Histogram

b. Heat Map

c. Pie Chart

d. Scatter Plot

Knowledge Check

5

Which of the following plots is used to represent data in a two-dimensional manner?

a. Histogram

b. Heat Map

c. Pie Chart

d. Scatter Plot

The correct answer is **b**

Heat Maps are used to represent data in a two-dimensional manner.

Visualize the Sales Data

Problem Statement:

BigMart is one of the biggest retailers of Europe and has operations across multiple countries. You are a Data Analyst in the IT team of BigMart. Invoice and SKU wise sales data for the years 2010 and 2011 is shared with you. You need to prepare meaningful charts to showcase the various sales trends for 2010 and 2011, to the top management.

Instructions to perform the assignment:

Download the dataset “BigMartSalesData.csv”. Use the data provided to create visualizations of the trends.

LESSON-END PROJECT

Visualize the Sales Data

Steps to Perform:

- Plot Total Sales Per Month for the year 2011. How has the total sales increased over the months? Which month has the lowest sales?
- Plot Total Sales Per Month for the year 2011 in a bar chart. Is bar chart better to visualize than a simple plot?
- Plot a pie chart for the year 2010, country wise. Which country contributes the highest and lowest towards sales? Create a pandas series with indexes of the country-wise sales.

LESSON-END PROJECT

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You