Python Programming-2
(CE523)
Semester-5
Class: CE/IT

UNIT-4 DATA VISUALIZATION

Section 4.1 Visualizing Information

Instructor: Prof. Snehal Sathwara
RK University, Rajkot

UNIT-4 DATA VISUALIZATION

UNIT 4.1
VISUALIZING INFORMATION

Starting with a Graph

- **Definition: Graph / Chart**

A graph or chart is simply a visual representation of numeric data.

✓ **Library used for Graph / Chart in Data Science.**

- Matplotlib

Matplotlib

Matplotlib: Visualization with Python

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Matplotlib

- Matplotlib makes easy things easy and hard things possible.

Create

- Develop publication quality plots with just a few lines of code
- Use interactive figures that can zoom, pan, update...

Customize

- Take full control of line styles, font properties, axes properties...
- Export and embed to a number of file formats and interactive environments

Extend

- Explore tailored functionality provided by third party packages
- Learn more about Matplotlib through the many external learning resources

- Reference: https://matplotlib.org/index.html

Matplotlib

You can see a gallery of the various graph types that MatPlotLib supports at

- https://matplotlib.org/gallery.html

- Lines, bars, and markers
- Shapes and collections
- Statistical plots
- Images, contours, and fields
- Pie and polar charts
- Color
- Text, labels, and annotations
- Ticks and spines
- Axis scales
- Subplots, axes, and figures

- Style sheets
- Specialty plots
- Showcase
- API
- pylab examples
- mplot3d toolkit
- axes_grid toolkit
- widgets

Defining the Plot

Definition: Plots

- Plots show graphically what you've defined numerically.
- To define a plot, you need some values, the matplotlib.pyplot module, and an idea of what you want to display.

Defining the Plot

- For Example:

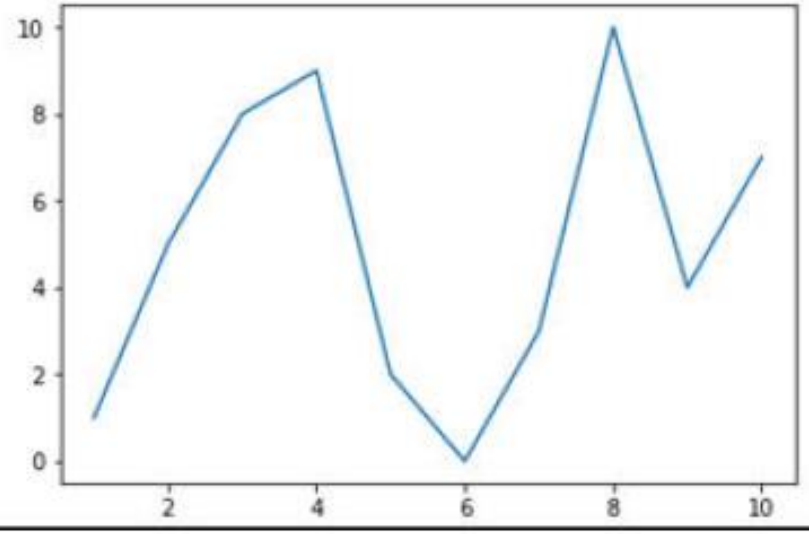
```
import matplotlib.pyplot as plt
%matplotlib inline

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
plt.plot(range(1,11), values)
plt.show()
```

- In this case, the code tells the **plt.plot()** function to create a plot using x-axis values between 1 and 11 and y-axis values as they appear in values.
- Calling **plot.show()** displays the plot in a separate dialog box, as shown in Figure (Next Slide)
- **Notice that the output is a line graph.**

Defining the Plot

- For Example: The output is a line graph.

```
In [1]: import matplotlib.pyplot as plt
        %matplotlib inline

        values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
        plt.plot(range(1,11), values)
        plt.show()
```



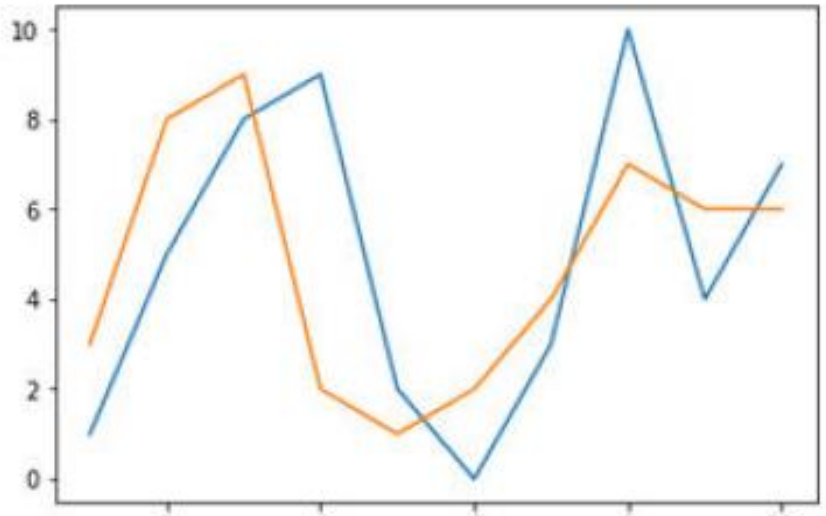
Drawing Multiple Lines & Plots

- Multiple Lines & Plots are used to compare two sets of values.
- To create such plots using MatPlotLib, you simply call **plt.plot()** multiple times — once for each plot line, as shown in the following example(Graph in Next Slide).

```
import matplotlib.pyplot as plt
%matplotlib inline

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
values2 = [3, 8, 9, 2, 1, 2, 4, 7, 6, 6]
plt.plot(range(1,11), values)
plt.plot(range(1,11), values2)
plt.show()
```

Drawing Multiple Lines & Plots

In [2]:
```
import matplotlib.pyplot as plt
%matplotlib inline

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
values2 = [3, 8, 9, 2, 1, 2, 4, 7, 6, 6]
plt.plot(range(1,11), values)
plt.plot(range(1,11), values2)
plt.show()
```



Saving Your Work to Disk

- In the Jupyter Notebook, When you do need to save a copy of your work to disk for later reference or to use it as part of a larger report, you save the graphic programmatically using the **plt.savefig()** function, as shown in the following code:

```
import matplotlib.pyplot as plt
%matplotlib auto

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
plt.plot(range(1,11), values)
plt.ioff()
plt.savefig('MySamplePlot.png', format='png')
```

Saving Your Work to Disk

- In this case (Previous Slide), you must provide a minimum of two inputs.
- The first input is the **filename**.
- You may optionally include a path for saving the file.
- The second input is the **file format**.
- In this case (Below given Code), the example saves the file in Portable Network Graphic (PNG) format,

```
import matplotlib.pyplot as plt
%matplotlib auto

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
plt.plot(range(1,11), values)
plt.ioff()
plt.savefig('MySamplePlot.png', format='png')
```

- **There are other options to save the file:**

✓ Portable Document Format (PDF),

✓ Postscript (PS),

✓ Encapsulated Postscript (EPS),

✓ Scalable Vector Graphics (SVG).

Setting the Axis

- The axes define the x and y plane of the graphic.
- The x axis runs horizontally, and the y axis runs vertically.

```
import matplotlib.pyplot as plt
%matplotlib notebook

values = [0, 5, 8, 9, 2, 0, 3, 10, 4, 7]
ax = plt.axes()
plt.plot(range(1,11), values)
plt.show()
```

Setting the Ticks

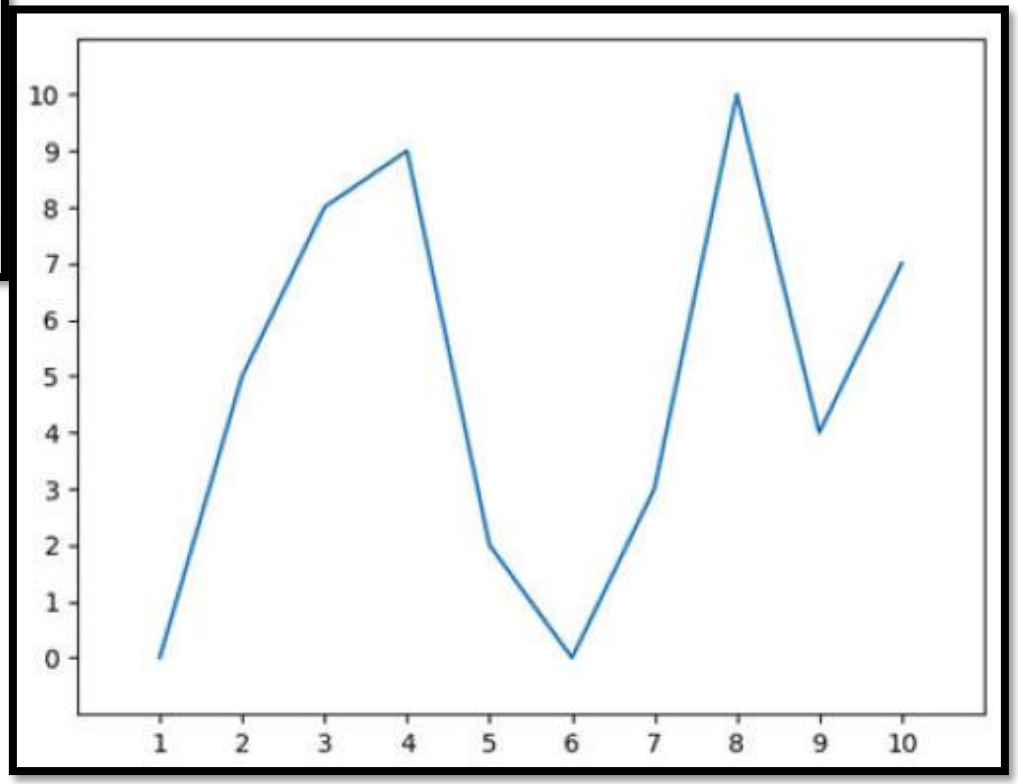
- For Example:

```
import matplotlib.pyplot as plt
%matplotlib notebook

values = [0, 5, 8, 9, 2, 0, 3, 10, 4, 7]
ax = plt.axes()
ax.set_xlim([0, 11])
ax.set_ylim([-1, 11])
ax.set_xticks([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
ax.set_yticks([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
plt.plot(range(1,11), values)
plt.show()
```

- In this case, the set_xlim() and set_ylim() calls change the axes limits — the length of each axis.
- The set_xticks() and set_yticks() calls change the ticks used to display data.

Setting the Ticks

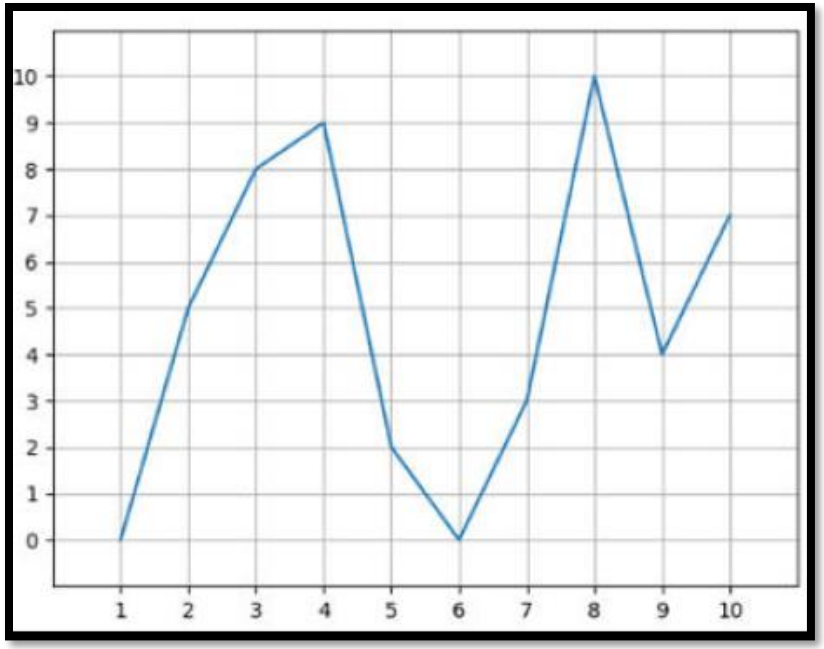
```
import matplotlib.pyplot as plt
%matplotlib notebook

values = [0, 5, 8, 9, 2, 0, 3, 10, 4, 7]
ax = plt.axes()
ax.set_xlim([0, 11])
ax.set_ylim([-1, 11])
ax.set_xticks([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
ax.set_yticks([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
plt.plot(range(1,11), values)
plt.show()
```



Setting the Grids

- Grid lines enable you to see the precise value of each element of a graph.
- You can more quickly determine both the x and y coordinates, which allow you to perform comparisons of individual points with greater ease.
- The following code shows how to add a grid to the graph in the previous section:

```
import matplotlib.pyplot as plt
%matplotlib notebook

values = [0, 5, 8, 9, 2, 0, 3, 10, 4, 7]
ax = plt.axes()
ax.set_xlim([0, 11])
ax.set_ylim([-1, 11])
ax.set_xticks([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
ax.set_yticks([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
ax.grid()
plt.plot(range(1,11), values)
plt.show()
```

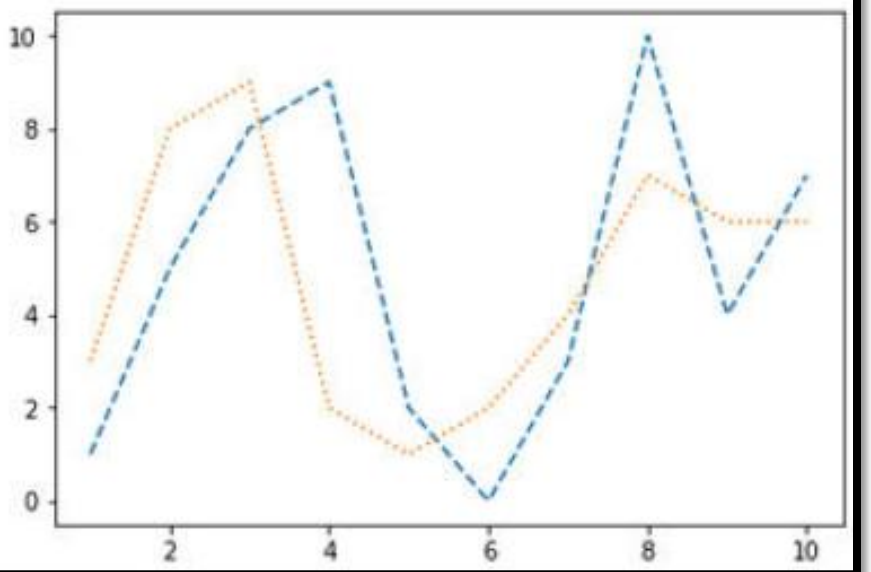


Defining the Line Appearance

- Matplotlib Line Style

Character	*Line Style*
'-'	Solid line
'--'	Dashed line
'-.'	Dash-dot line
':'	Dotted line

```
import matplotlib.pyplot as plt
%matplotlib inline

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
values2 = [3, 8, 9, 2, 1, 2, 4, 7, 6, 6]
plt.plot(range(1,11), values, '--')
plt.plot(range(1,11), values2, ':')
plt.show()
```



Defining the Line Appearance

- Matplotlib Colours

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

```
import matplotlib.pyplot as plt
%matplotlib inline

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
values2 = [3, 8, 9, 2, 1, 2, 4, 7, 6, 6]
plt.plot(range(1,11), values, 'r')
plt.plot(range(1,11), values2, 'm')
plt.show()
```

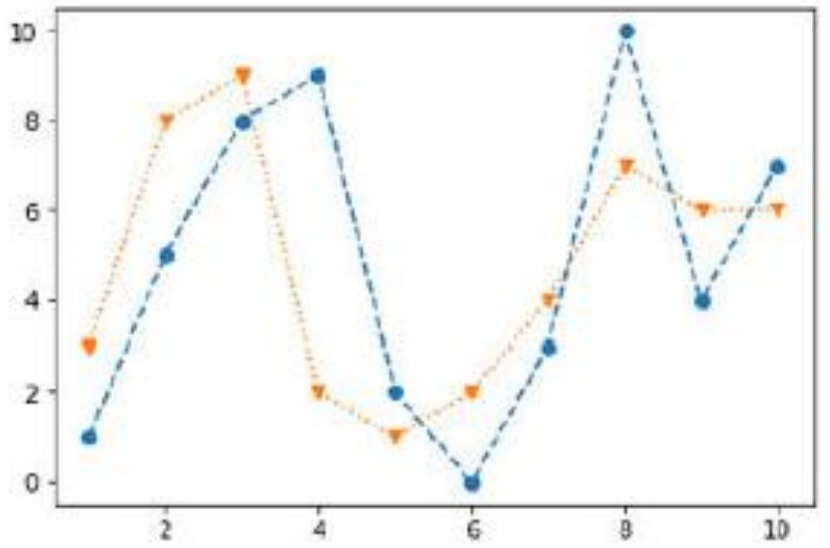
Defining the Line Appearance

- Adding Markers

Character	Marker Type
'.'	Point
','	Pixel
'o'	Circle
'v'	Triangle 1 down
'^'	Triangle 1 up
'<'	Triangle 1 left
'>'	Triangle 1 right
'1'	Triangle 2 down
'2'	Triangle 2 up
'3'	Triangle 2 left
'4'	Triangle 2 right
's'	Square

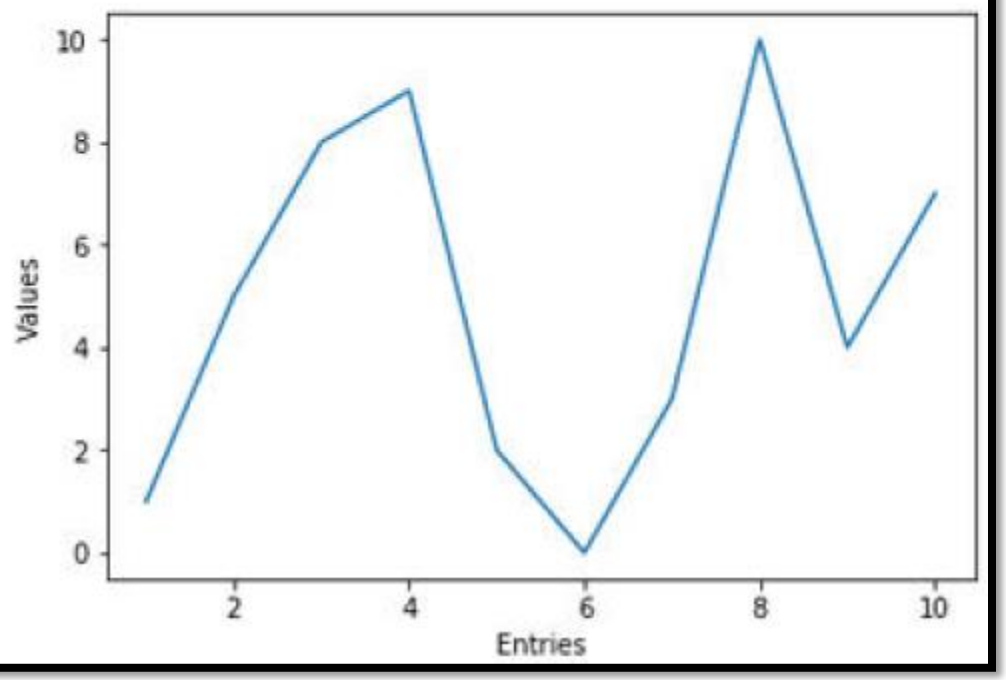
'p'	Pentagon
'*'	Star
'h'	Hexagon style 1
'H'	Hexagon style 2
'+'	Plus
'x'	X
'D'	Diamond
'd'	Thin diamond
'|'	Vertical line
'_'	Horizontal line

```
import matplotlib.pyplot as plt
%matplotlib inline

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
values2 = [3, 8, 9, 2, 1, 2, 4, 7, 6, 6]
plt.plot(range(1,11), values, 'o--')
plt.plot(range(1,11), values2, 'v:')
plt.show()
```



Using lables, Annotations and Legends

```
import matplotlib.pyplot as plt
%matplotlib inline

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
plt.xlabel('Entries')
plt.ylabel('Values')
plt.plot(range(1,11), values)
plt.show()
```

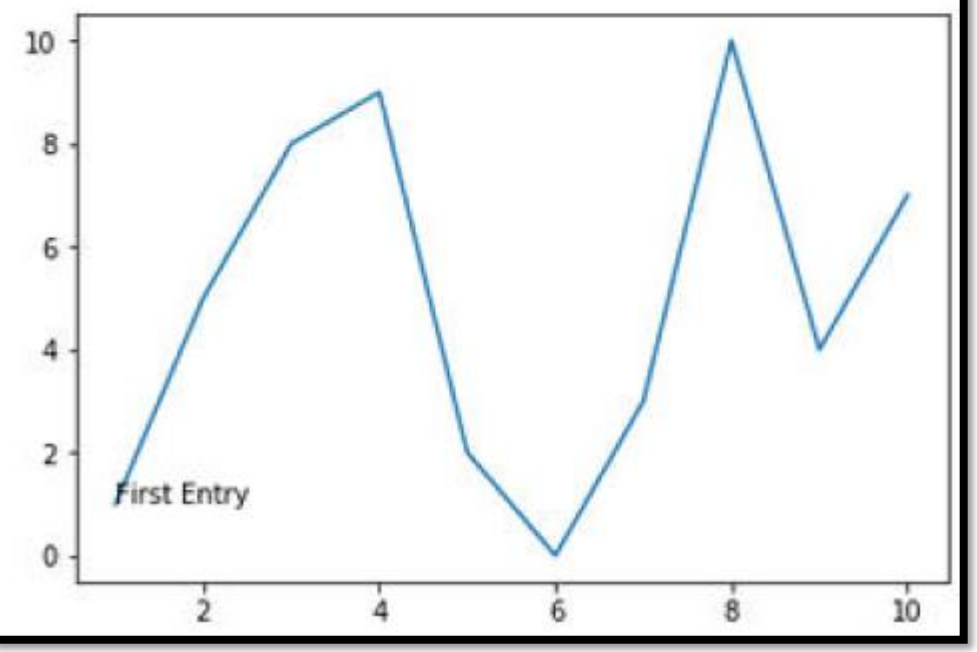


- Label

Label: Provides positive identification of a particular data element or grouping. The purpose is to make it easy for the viewer to know the name or kind of data illustrated.

Using lables, Annotations and Legends

```
import matplotlib.pyplot as plt
%matplotlib inline

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
plt.annotate(xy=[1,1], s='First Entry')
plt.plot(range(1,11), values)
plt.show()
```

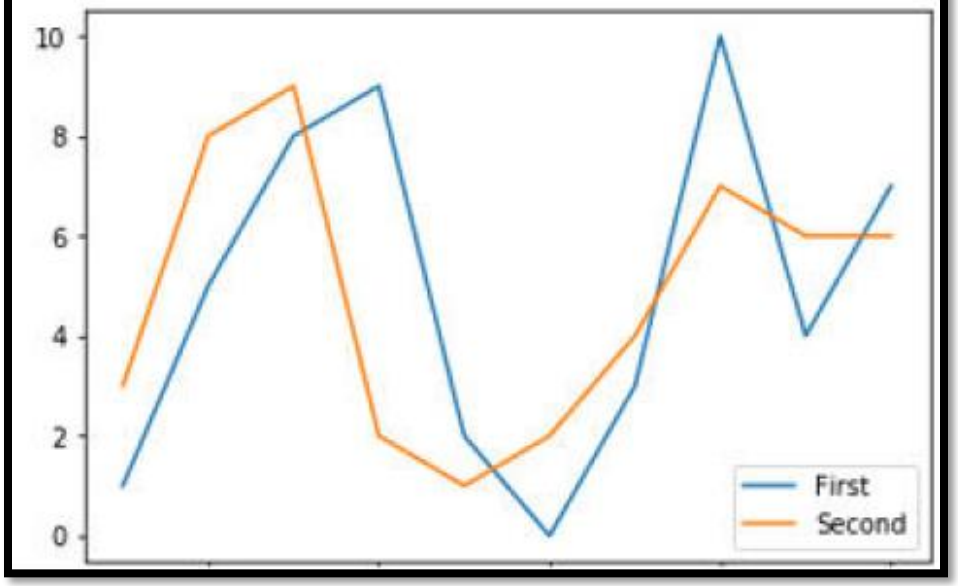


- Annotation

Annotation: Augments the information the viewer can immediately see about the data with notes, sources, or other useful information. In contrast to a label, the purpose of annotation is to help extend the viewer's knowledge of the data rather than simply identify it.

Using lables, Annotations and Legends

```
import matplotlib.pyplot as plt
%matplotlib inline

values = [1, 5, 8, 9, 2, 0, 3, 10, 4, 7]
values2 = [3, 8, 9, 2, 1, 2, 4, 7, 6, 6]
line1 = plt.plot(range(1,11), values)
line2 = plt.plot(range(1,11), values2)
plt.legend(['First', 'Second'], loc=4)
plt.show()
```



- Legends

Legend: Presents a listing of the data groups within the graph and often provides cues (such as line type or color) to make identification of the data group easier. For example, all the red points may belong to group A, while all the blue points may belong to group B.