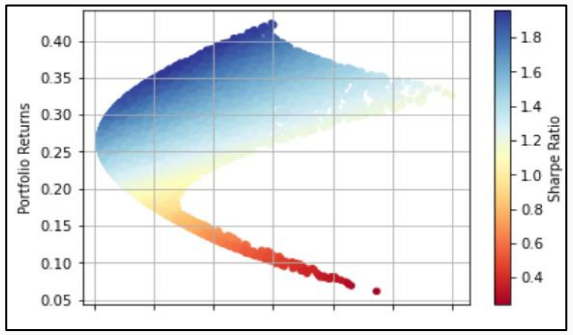
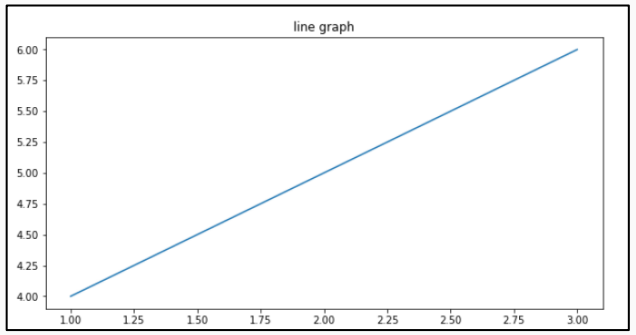
DATA VISUALIZATION USING PYTHON

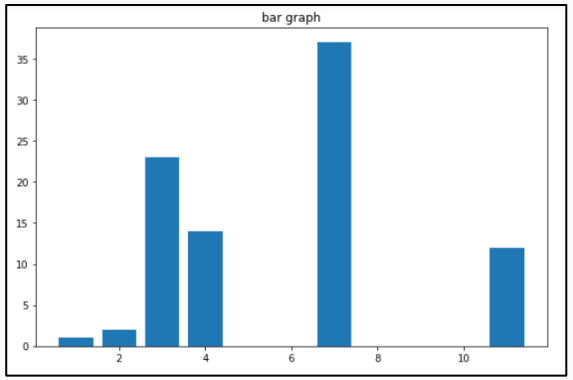
Meta Brains

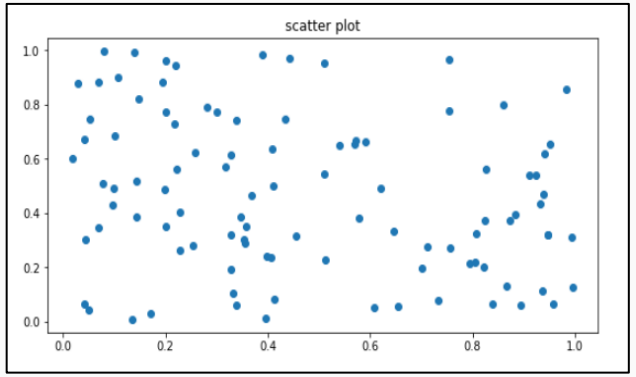
About Matplotlib

- Matplotlib is the most popular python library for plotting different kinds of graphs.
- The Pyplot module inside the Matplotlib makes it work like Matlab.





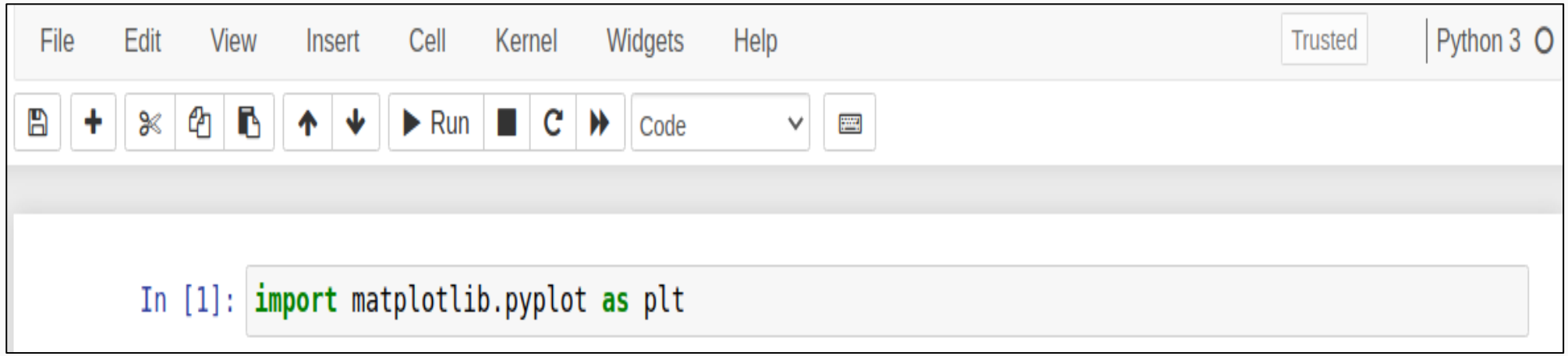




Meta Brains

Importing Matplotlib

- To import matplotlib.pyplot, type ‘import matplotlib.pyplot’ in Jupyter Notebook and run the cell.
- The common abbreviation used for matplotlib.pyplot is plt.

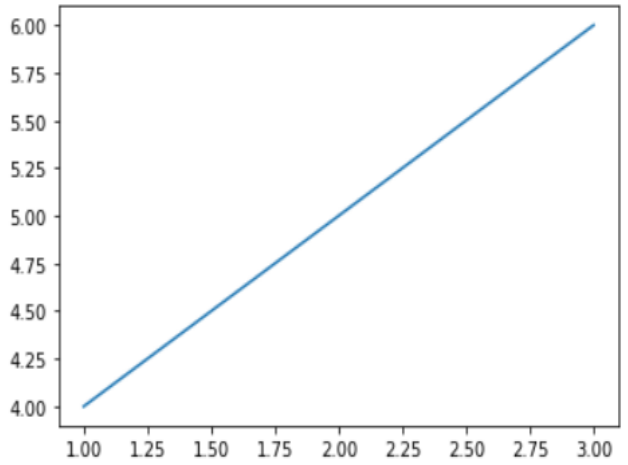


Meta Brains

Plotting Line Plots (1/2)

- We can plot line plots using matplotlib using the .plot() function.
  - The first argument in the .plot() function specifies the x-axis.
  - The second argument in the .plot() function specifies the y-axis.

```
In [3]: x_axis = [1,2,3]
        y_axis = [4,5,6]

        plt.plot(x_axis, y_axis)

Out[3]: [<matplotlib.lines.Line2D at 0x7f186e0faf70>]
```



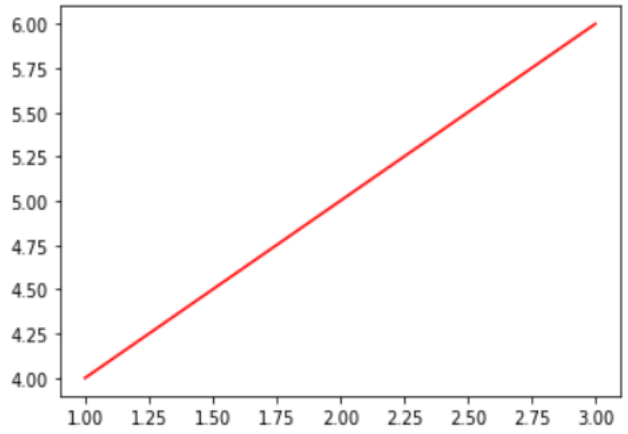
Meta Brains

Plotting Line Plots (2/2)

Changing Color

- We can also change the color of the line by providing the color as third argument in the plot() function.
- A list of the color abbreviations can be found at: https://matplotlib.org/2.1.1/api/_as_gen/matplotlib.pyplot.plot.html

```
In [4]: x_axis = [1,2,3]
        y_axis = [4,5,6]

        plt.plot(x_axis, y_axis, 'r')

Out[4]: [<matplotlib.lines.Line2D at 0x7f186d85eb80>]
```

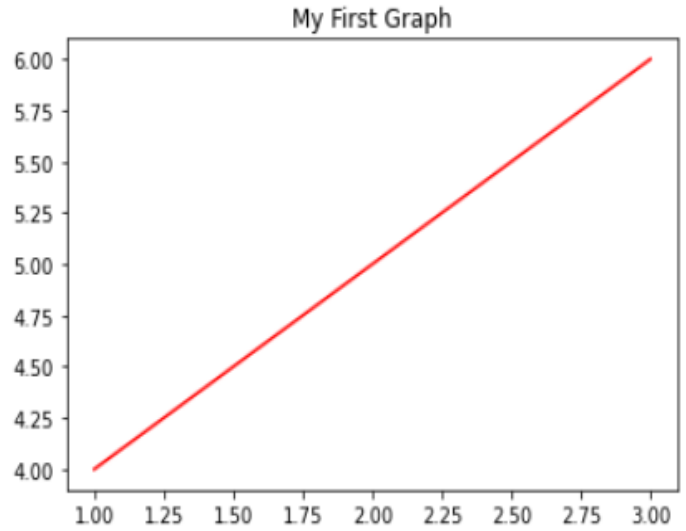


Meta Brains

Title

- To set the title of the plot, use the .title() function.

```
In [4]: x_axis = [1,2,3]
        y_axis = [4,5,6]

        plt.title('My First Graph')
        plt.plot(x_axis, y_axis, 'r')

Out[4]: [<matplotlib.lines.Line2D at 0x7efef8336700>]
```

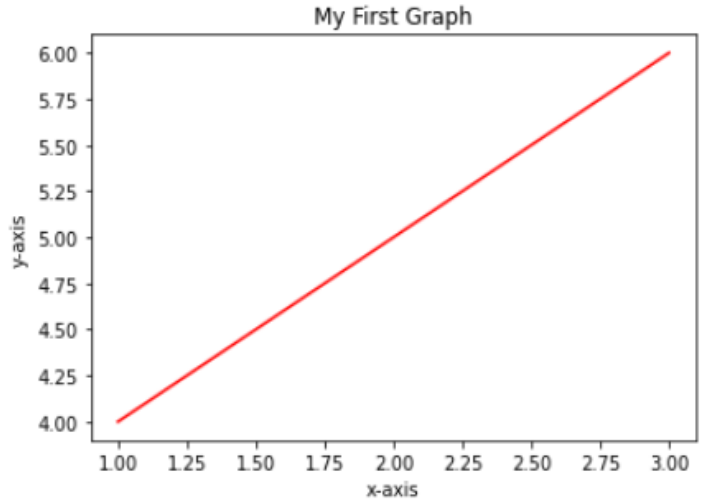


Meta Brains

Labels

- To assign labels to x and y-axis, use .xlabel() and .ylabel() respectively.

```
In [5]: x_axis = [1,2,3]
        y_axis = [4,5,6]

        plt.title('My First Graph')
        plt.xlabel('x-axis')
        plt.ylabel('y-axis')
        plt.plot(x_axis, y_axis, 'r')

Out[5]: [<matplotlib.lines.Line2D at 0x7efef82a4340>]
```

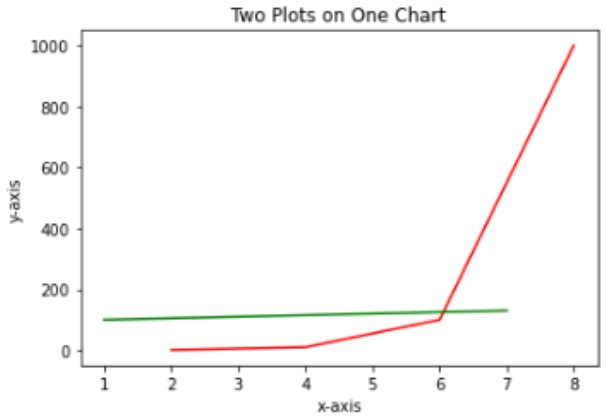


Meta Brains

Legend (1/2)

- We can plot multiple plots on the same chart simply by plotting them one by one as in the given example.

```
In [10]: x1_axis = [2, 4, 6, 8]
         y1_axis = [1, 10, 100, 1000]
         x2_axis = [1, 3, 5, 7]
         y2_axis = [100, 110, 120, 130]

         plt.title('Two Plots on One Chart')
         plt.xlabel('x-axis')
         plt.ylabel('y-axis')

         plt.plot(x1_axis, y1_axis, 'r')
         plt.plot(x2_axis, y2_axis, 'g')

Out[10]: [<matplotlib.lines.Line2D at 0x7efef81c7850>]
```

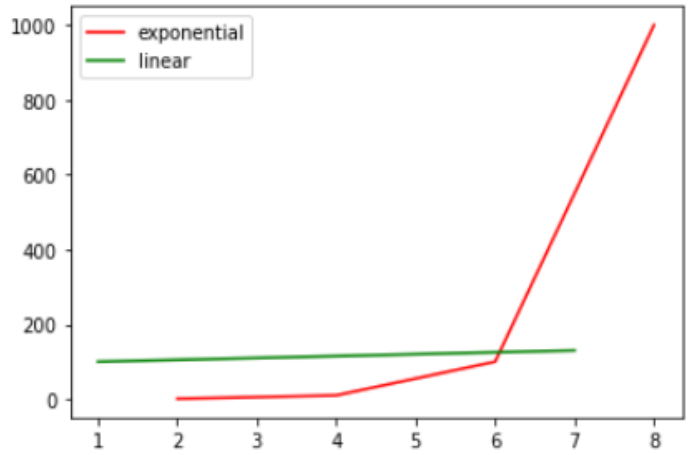


Meta Brains

Legend (2/2)

- If plotting more than one plots, it is a good idea to add a legend in your figure using the .legend() function.

```
In [12]: x1_axis = [2, 4, 6, 8]
         y1_axis = [1, 10, 100, 1000]
         x2_axis = [1, 3, 5, 7]
         y2_axis = [100, 110, 120, 130]

         plt.plot(x1_axis, y1_axis, 'r')
         plt.plot(x2_axis, y2_axis, 'g')

         plt.legend(['exponential', 'linear'])

Out[12]: <matplotlib.legend.Legend at 0x7efef81037c0>
```

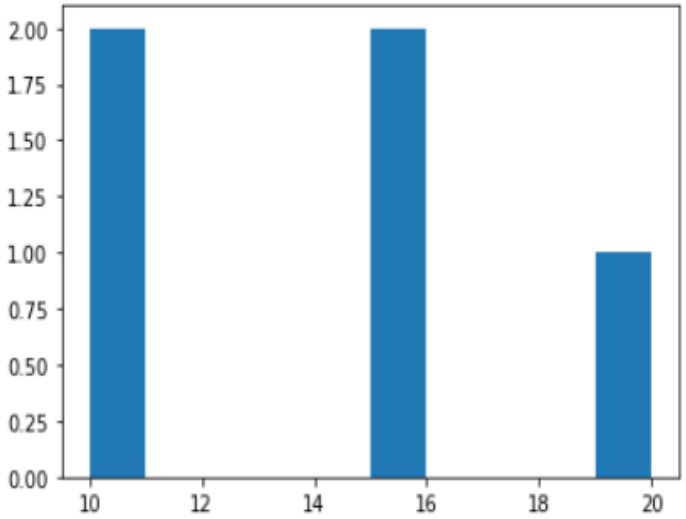


Meta Brains

Plotting Histograms (1/3)

- We can also plot histograms using .hist() function.
- A histogram is generally used to plot frequency which helps identify distribution of data.

```
In [20]: values = [10, 15, 20, 10, 15]
         plt.hist(values)

Out[20]: (array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
          array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
          <BarContainer object of 10 artists>)
```



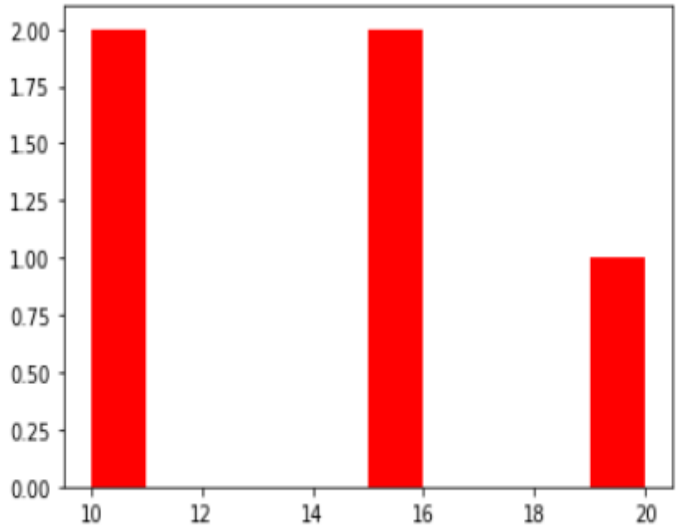
Meta Brains

Plotting Histograms (2/3)

Changing Color

- We can also change color of the bars using the ‘color’ parameter inside the hist() function.

```
In [21]: values = [10, 15, 20, 10, 15]
         plt.hist(values, color='r')

Out[21]: (array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
          array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
          <BarContainer object of 10 artists>)
```



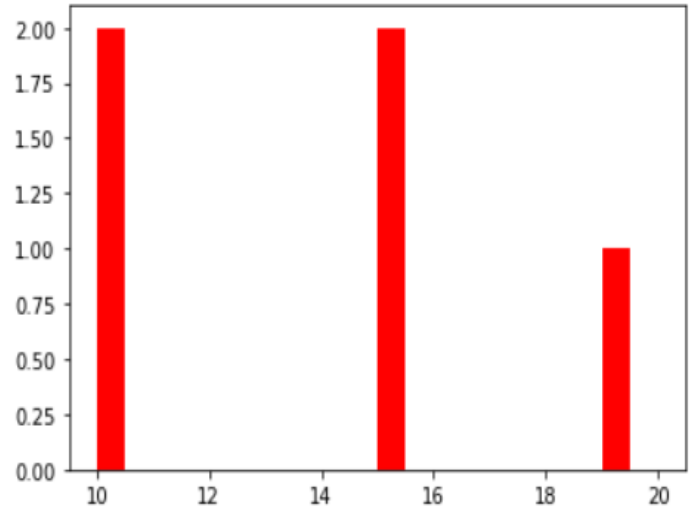
Meta Brains

Plotting Histograms (3/3)

Changing Width

- We can also change width of the bars using the ‘width’ parameter inside the hist() function.

```
In [26]: values = [10, 15, 20, 10, 15]
         plt.hist(values, color='r', width=0.5)

Out[26]: (array([2., 0., 0., 0., 0., 2., 0., 0., 0., 1.]),
          array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20.]),
          <BarContainer object of 10 artists>)
```

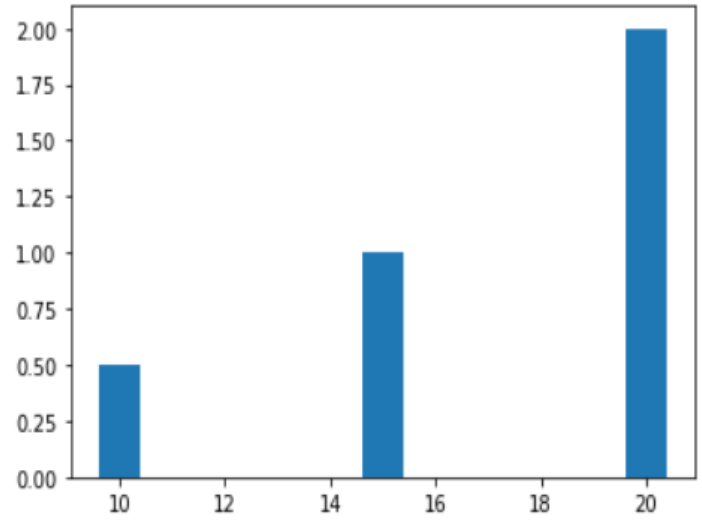


Meta Brains

Plotting Bar Charts (1/2)

- To plot a bar graph, use the .bar() function of the matplotlib.pyplot.
  - First argument in the bar() function is the x-label.
  - Second argument in the bar() function is the height of each bar, which can be a list of values or a single value.

```
In [33]: values = [10, 15, 20]
         plt.bar(values, [0.5, 1, 2])

Out[33]: <BarContainer object of 3 artists>
```



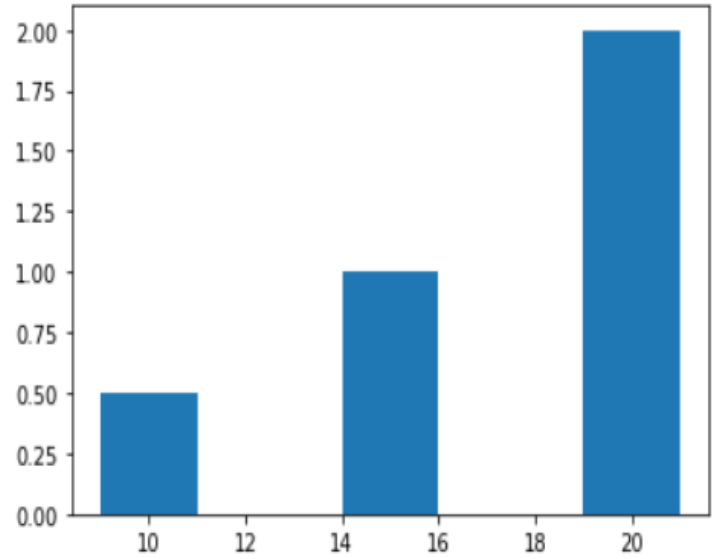
Meta Brains

Plotting Bar Charts (2/2)

Changing Width

- We can also change the width of bars in the bar plot using the ‘width’ parameter.

```
In [34]: values = [10, 15, 20]
         plt.bar(values, [0.5, 1, 2], width=2)

Out[34]: <BarContainer object of 3 artists>
```



Meta Brains

Plotting Pie Charts (1/4)

- .pie() function is used to create a pie chart in matplotlib.pyplot.

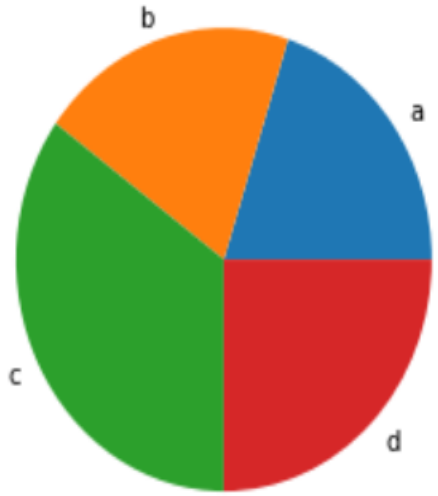
```
In [42]: values = [20, 20, 35, 25]
         plt.pie(values)
```

Plotting Pie Charts (2/4)

Labels

- To add labels in your pie chart, use the ‘labels’ parameter which takes a list of labels.

```
In [43]: values = [20, 20, 35, 25]
         plt.pie(values, labels=['a', 'b', 'c', 'd'])
```



Meta Brains

Plotting Pie Charts (3/4)

Explode

- If you want one or more wedges of your pie chart to stand out, you can use the 'explode' parameter.
  - Provide a list containing distance of each wedge from the center to the 'explode' parameter.

```
In [44]: values = [20, 20, 35, 25]
         plt.pie(values, labels=['a', 'b', 'c', 'd'], explode=[0, 0.2, 0, 0])
```

Meta Brains

Plotting Pie Charts (4/4)

Colors

- We can also change the colors of wedges by providing a list of colors to the ‘colors’ parameter.

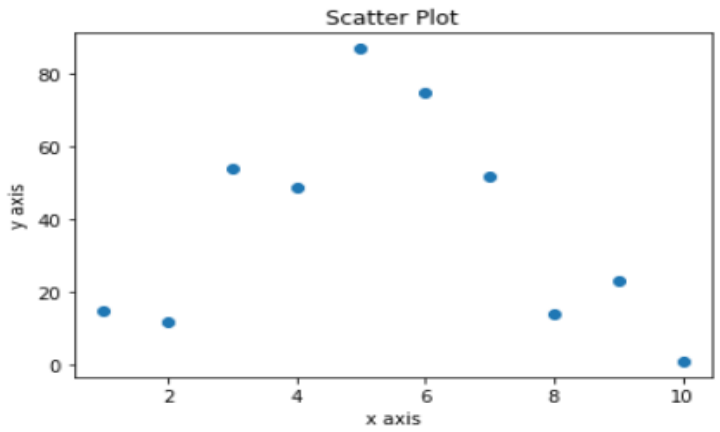
```
In [45]: values = [20, 20, 35, 25]
         plt.pie(values, labels=['a', 'b', 'c', 'd'], colors=['r', 'g', 'b', 'y'])
```

Meta Brains

Plotting Scatter Plot (1/5)

- We can also plot scatter plots using the .scatter() function inside matplotlib.pyplot.
- It takes two lists as arguments;
  - First list specifies the values for the x-axis
  - Second list specifies the values for the y-axis

```
In [48]: x_axis = [1,2,3,4,5,6,7,8,9,10]
         y_axis = [15,12,54,49,87,75,52,14,23,1]

         plt.title('Scatter Plot')
         plt.xlabel('x axis')
         plt.ylabel('y axis')

         plt.scatter(x_axis, y_axis)

Out[48]: <matplotlib.collections.PathCollection at 0x7efef1670a00>
```



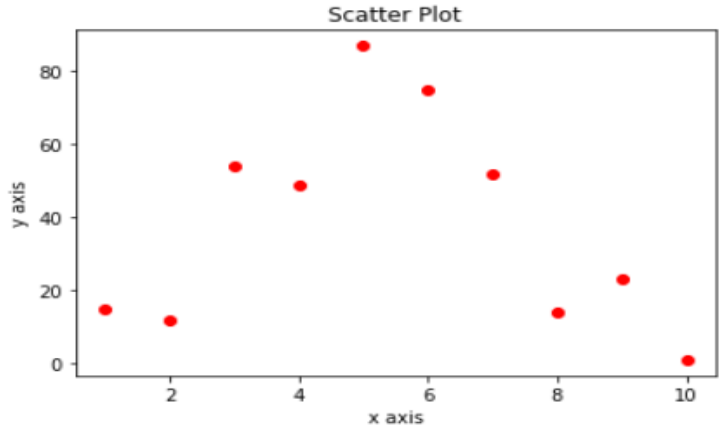
Meta Brains

Plotting Scatter Plot (2/5)

Colors

- We can also change color of the dots using the ‘color’ parameter.

```
In [49]: x_axis = [1,2,3,4,5,6,7,8,9,10]
         y_axis = [15,12,54,49,87,75,52,14,23,1]

         plt.title('Scatter Plot')
         plt.xlabel('x axis')
         plt.ylabel('y axis')

         plt.scatter(x_axis, y_axis, color='r')

Out[49]: <matplotlib.collections.PathCollection at 0x7efef1b2dd00>
```



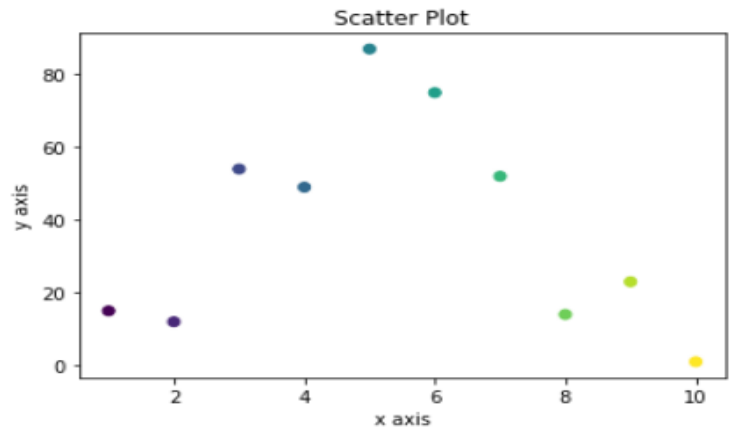
Meta Brains

Plotting Scatter Plot (3/5)

Colormap

- If you would like to give each dot a different color, you can provide a list of colors or integers to the ‘c’ parameter.

```
In [52]: x_axis = [1,2,3,4,5,6,7,8,9,10]
         y_axis = [15,12,54,49,87,75,52,14,23,1]

         plt.title('Scatter Plot')
         plt.xlabel('x axis')
         plt.ylabel('y axis')

         plt.scatter(x_axis, y_axis, c=[1,2,3,4,5,6,7,8,9,10])

Out[52]: <matplotlib.collections.PathCollection at 0x7efef19bf610>
```



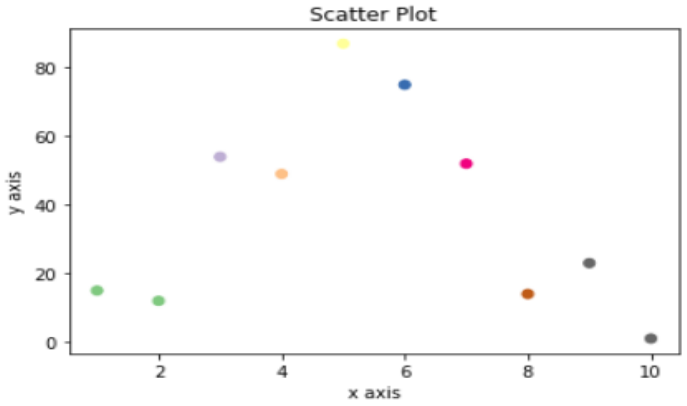
Meta Brains

Plotting Scatter Plot (4/5)

Colormap

- You can also set the colormap using the ‘cmap’ parameter.
- For this, you will need to provide a list of integers that will be mapped to colors.
- We have used ‘Accent’ colormap, which is one of the many built-in colormaps in matplotlib.

```
In [53]: x_axis = [1,2,3,4,5,6,7,8,9,10]
         y_axis = [15,12,54,49,87,75,52,14,23,1]

         plt.title('Scatter Plot')
         plt.xlabel('x axis')
         plt.ylabel('y axis')

         plt.scatter(x_axis, y_axis, c=[1,2,3,4,5,6,7,8,9,10], cmap='Accent')

Out[53]: <matplotlib.collections.PathCollection at 0x7efef188d190>
```



Meta Brains

Plotting Scatter Plot (5/5)

Colormap

- To see the available colormaps in matplotlib, import ‘cm’ module from matplotlib.
- Give the ‘dir(cm)’ command and run the cell.
- A list of all the available colormaps will be displayed.

```
In [54]: from matplotlib import cm

In [55]: dir(cm)
```

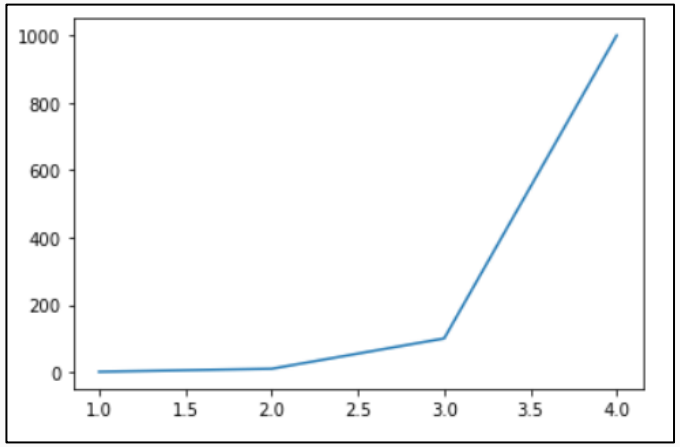
```
'__builtins__',
'__cached__',
'__doc__',
'__file__',
'__loader__',
'__name__',
'__package__',
'__spec__',
'_cmap_registry',
'_gen_cmap_registry',
'_reverser',
'afmhot',
'afmhot_r',
'autumn',
'autumn_r',
'binary',
'binary_r',
'bone',
'bone_r',
```

Meta Brains

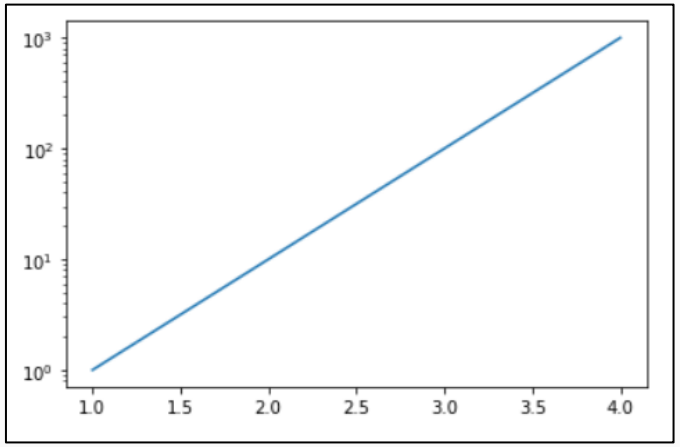
Plotting Log Plots

- We can also plot on logarithmic y-axis scale by using .yscale() function and passing ‘log’ as argument.
- Logarithmic scale is generally used if one or more data points are way bigger than bulk of the data, resulting in a skewed graph.

```
In [60]: x_axis = [1,2,3,4]
         y_axis = [1, 10, 100, 1000]

         plt.yscale('log')
         plt.plot(x_axis, y_axis)
```



Linear Scale



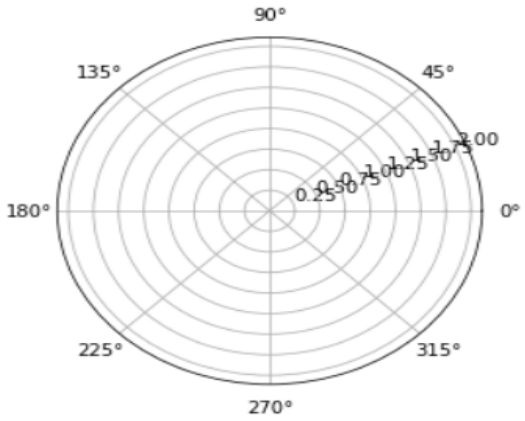
Log Scale

Meta Brains

Plotting Polar Plots

- Matplotlib also allows us to plot a polar plot.
- A point in a polar plot is represented as (r, theta);
  - r is the distance from the origin.
  - theta is the angle along which r is measured.
- Use the .polar() function to plot a polar plot.

```
In [78]: theta = np.arange(0, (2 * np.pi), 0.01)  # generating an array of evenly spaced floats
         r = 2

         for radian in theta:
             plt.polar(radian, r)
```

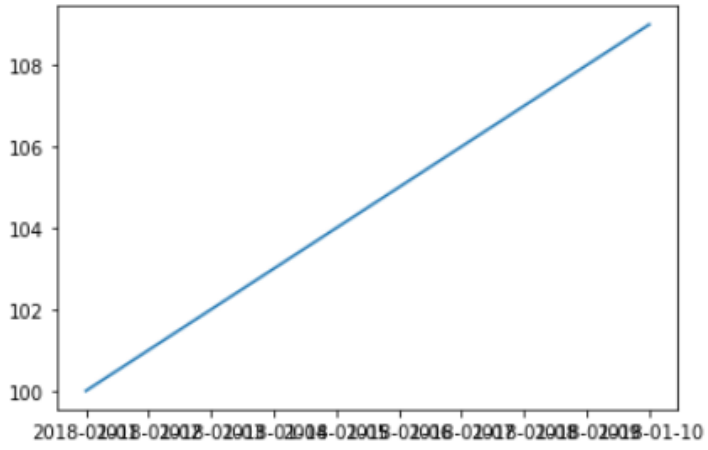


Meta Brains

Handling Dates (1/2)

- Sometimes, there are too many values on the x-axis and it becomes difficult to distinguish them in the graph.
- This also happens when you have dates on the x-axis and they overlap as shown in the figure.

```
In [94]: dates = ['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04', '2018-01-05',
                  '2018-01-06', '2018-01-07', '2018-01-08', '2018-01-09', '2018-01-10']
         values = [100, 101, 102, 103, 104, 105, 106, 107, 108, 109]

         plt.plot(dates, values)

Out[94]: [<matplotlib.lines.Line2D at 0x7efef13c0d60>]
```

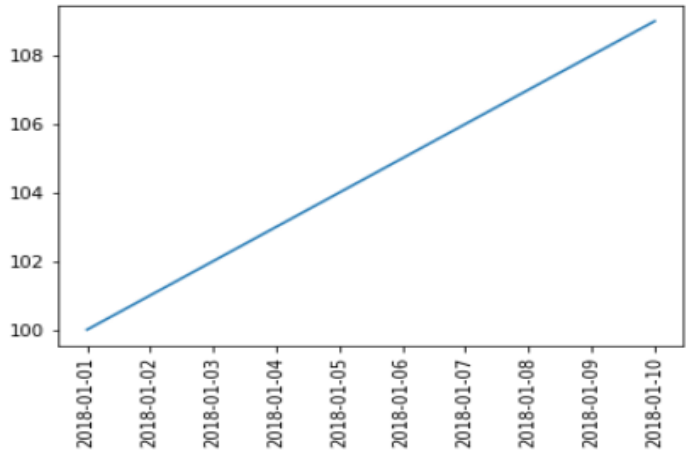


Meta Brains

Handling Dates (2/2)

- We can avoid this by changing the orientation of the values on the x-axis using the .xticks() function.
  - .xticks() has a rotation parameter which can be used to rotate the values on the x-axis by a suitable angle.

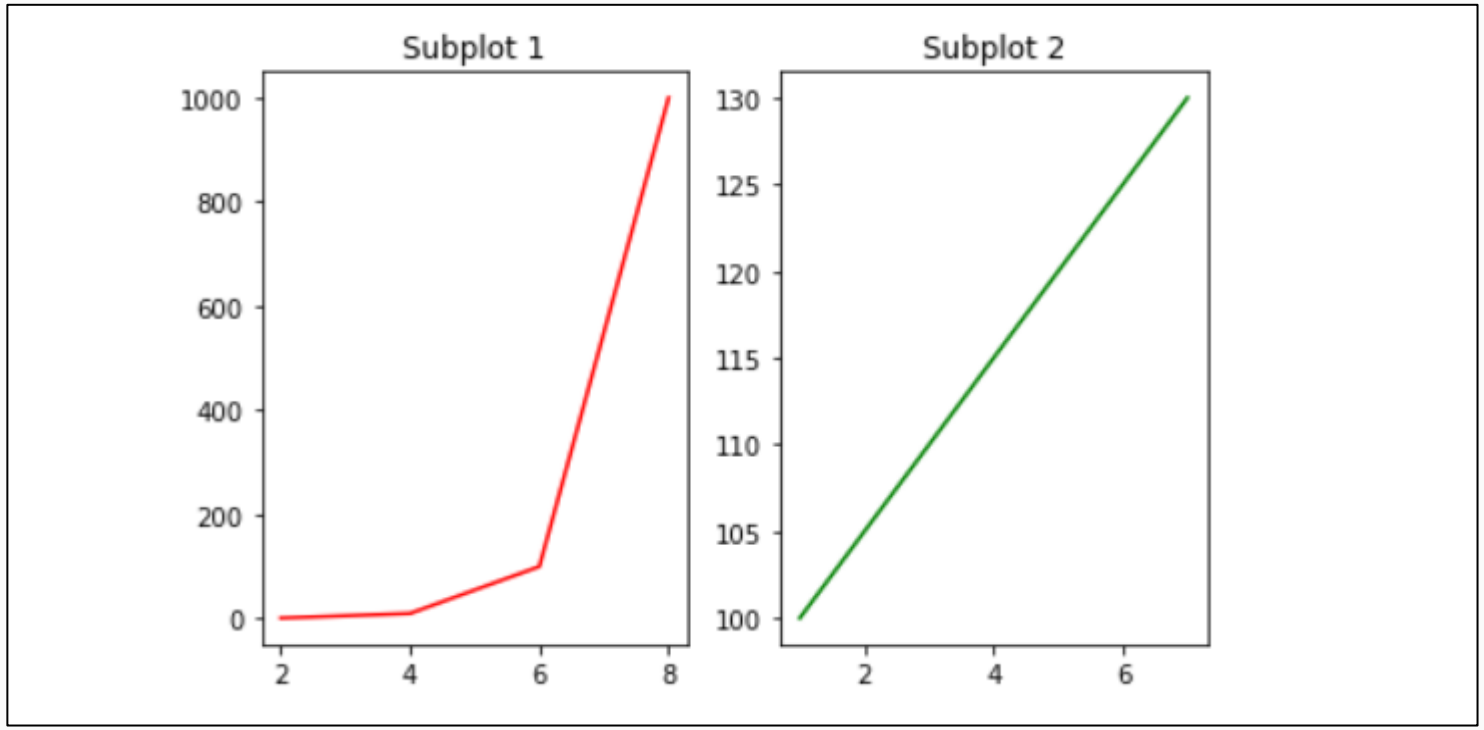
```
In [96]: dates = ['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04', '2018-01-05',
                  '2018-01-06', '2018-01-07', '2018-01-08', '2018-01-09', '2018-01-10']
         values = [100, 101, 102, 103, 104, 105, 106, 107, 108, 109]

         plt.xticks(rotation=90)
         plt.plot(dates, values)

Out[96]: [<matplotlib.lines.Line2D at 0x7efef0dae430>]
```



Meta Brains

Creating Multiple Subplots in One Figure (1/3)

- We saw how we can plot multiple plots on the same chart, but sometimes we would like to have different charts for each plot.
- What we want to have actually is multiple subplots in the same figure, as shown in the given figure.

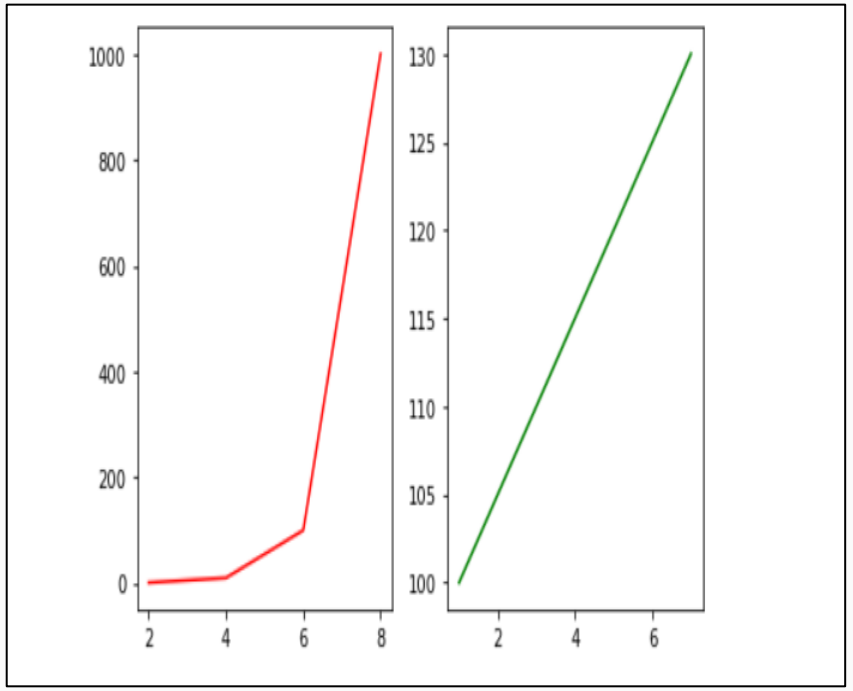


Meta Brains

Creating Multiple Subplots in One Figure (2/3)

- To create subplots, we use the .subplot() function before plotting each graph.
- .subplot() takes 3 parameters;
  - First parameter is the number of rows you want to have in your figure.
  - Second parameter is the number of columns you want to have in your figure.
  - Third parameter is the id/position of the plot in the figure.

```
In [101]: x1_axis = [2, 4, 6, 8]
          y1_axis = [1, 10, 100, 1000]
          x2_axis = [1, 3, 5, 7]
          y2_axis = [100, 110, 120, 130]

          plt.subplot(1, 2, 1)
          plt.plot(x1_axis, y1_axis, 'r')

          plt.subplot(1, 2, 2)
          plt.plot(x2_axis, y2_axis, 'g')

          plt.tight_layout()
```

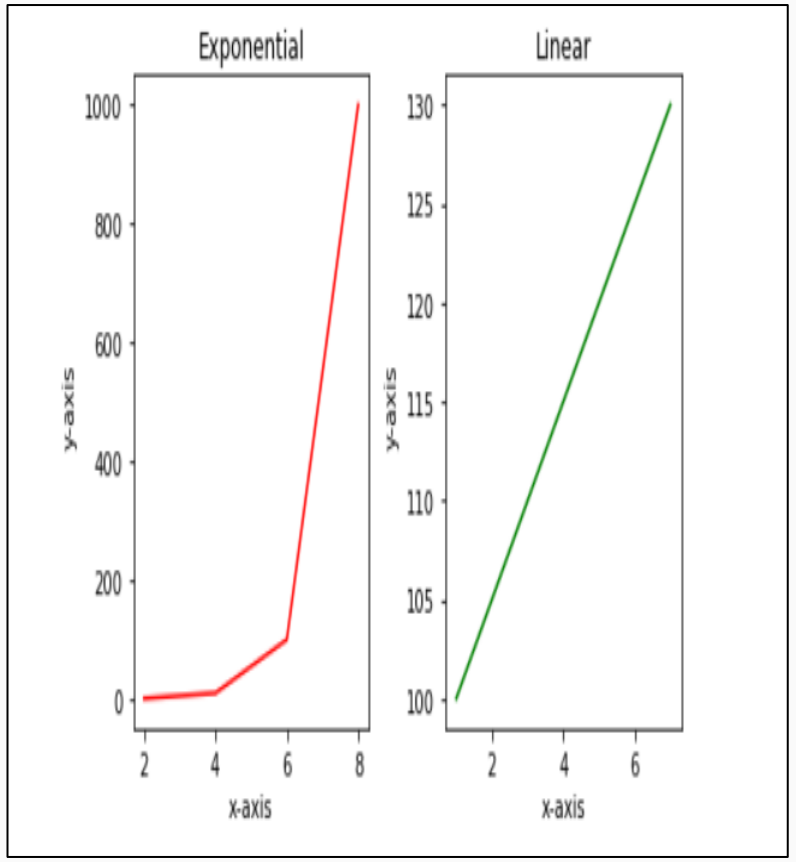


Meta Brains

Creating Multiple Subplots in One Figure (3/3)

- We can have separate x and y labels as well as title for each subplot in the figure.

```
In [103]: x1_axis = [2, 4, 6, 8]
          y1_axis = [1, 10, 100, 1000]
          x2_axis = [1, 3, 5, 7]
          y2_axis = [100, 110, 120, 130]

          plt.subplot(1, 2, 1)
          plt.title('Exponential')
          plt.xlabel('x-axis')
          plt.ylabel('y-axis')
          plt.plot(x1_axis, y1_axis, 'r')

          plt.subplot(1, 2, 2)
          plt.title('Linear')
          plt.xlabel('x-axis')
          plt.ylabel('y-axis')
          plt.plot(x2_axis, y2_axis, 'g')

          plt.tight_layout()
```



Meta Brains

Resources

- https://www.w3schools.com/python/matplotlib_intro.asp

Meta Brains