Objective: ***Matplotlib*** is a very useful Python package for visualization. This worksheet introduces the setup of the package and basics of the line plots.

Reference: The flow and the topics of the Matplotlib worksheets are adopted from a wonderful book Python Data Science Handbook by Jake VanderPlas.

- Using the command ***%matplotlib*** the interactivity with the plots can be controlled up to some extent. The option ***inline*** renders the static images while the option ***notebook*** renders the interactive images within the Jupyter notebook. There is a third option for plotting from a Python script which is not explored in this worksheet. Between the two Matplotlib imports the ***plt*** interface will be used more often. Other packages for NumPy and Pandas are also imported as required.

```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

- The aesthetics of the plots can be controlled using the following directive. The usage of different options will be used as and when required in these worksheets.

```
plt.style.use('classic')
```

- If the plot needs a grid the following style can be used:

```
plt.style.use('seaborn-whitegrid')
```

- NumPy provides a useful function ***linspace()*** that provides evenly spaced numbers over an interval. This function is very useful to get the data points to draw a plot. For example, if 25 points are required between 0 to 10, the function can be used in the following way:

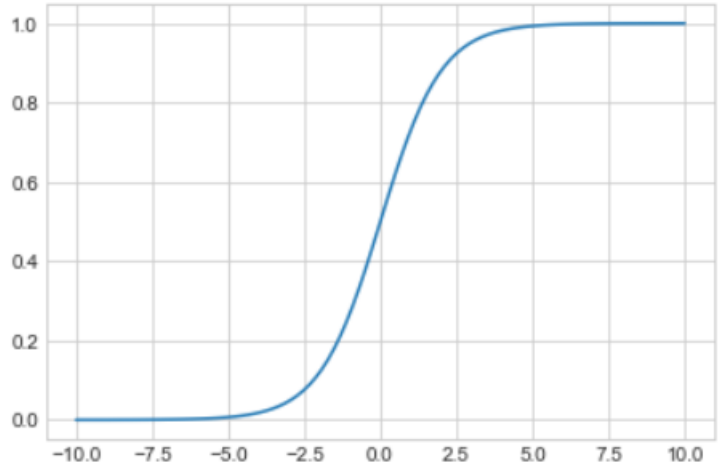
```
x = np.linspace(0, 10, 25)
x
```

```
array([ 0.        ,  0.41666667,  0.83333333,  1.25      ,  1.66666667,
        2.08333333,  2.5       ,  2.91666667,  3.33333333,  3.75      ,
        4.16666667,  4.58333333,  5.        ,  5.41666667,  5.83333333,
        6.25      ,  6.66666667,  7.08333333,  7.5       ,  7.91666667,
        8.33333333,  8.75      ,  9.16666667,  9.58333333, 10.        ])
```

- Using ***type*** and ***shape*** check that the return object x is a NumPy array of 1 dimension (25x1).
- Let us say a function $y = \frac{1}{1+e^{-x}}$ is to be drawn for the values of x from -10 to +10. This function is a famous Sigmoid or Logistic function used extensively in the Deep Learning. The plotting can be done as following using the ***exp()*** function for NumPy for calculating the value of e^{-x}.

```
plt.style.use('seaborn-whitegrid')
fig = plt.figure()
x = np.linspace (-10, 10, 100)
values = 1/(1+np.exp(-x))
plt.plot(x, values)
```

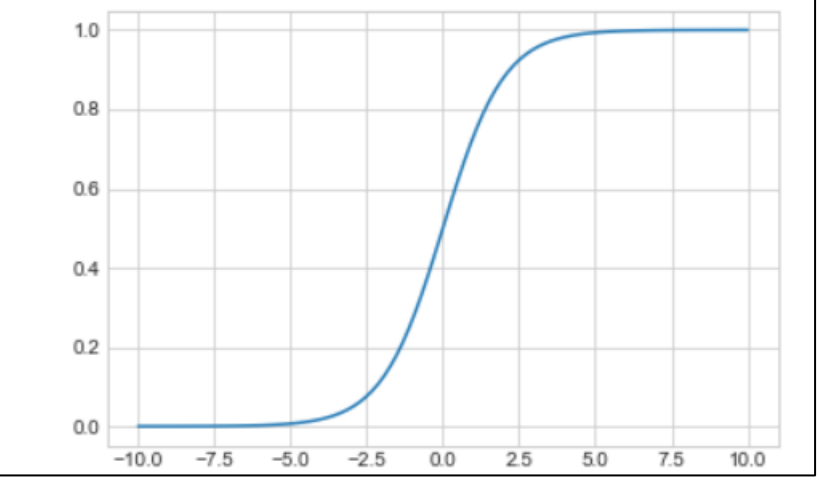
```
[<matplotlib.lines.Line2D at 0x22f32dbfd88>]
```



- Using the ***fig*** variable created in the example above the plot can be saved and retrieved later.

```
fig.savefig('sigmoid.png')
```

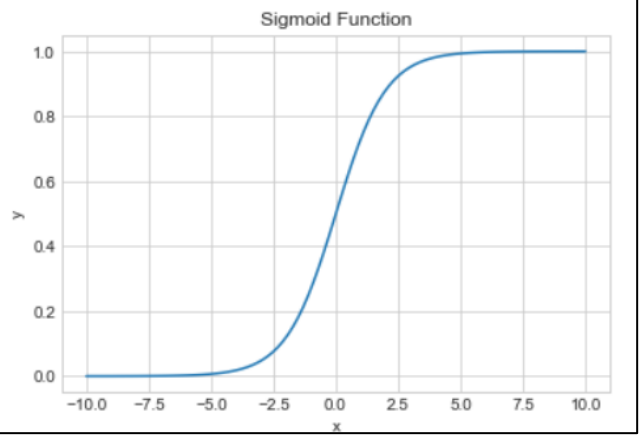
```
from IPython.display import Image
Image('sigmoid.png')
```



- The format of saved file depends on the file extension. Many popular formats (.jpeg, .pdf, .tiff etc. are available).
- The title of the plot, labelling of x and y axes can be also be done using ***title()***, ***xlabel()*** and ***ylabel()*** functions through the ***plt*** interface.

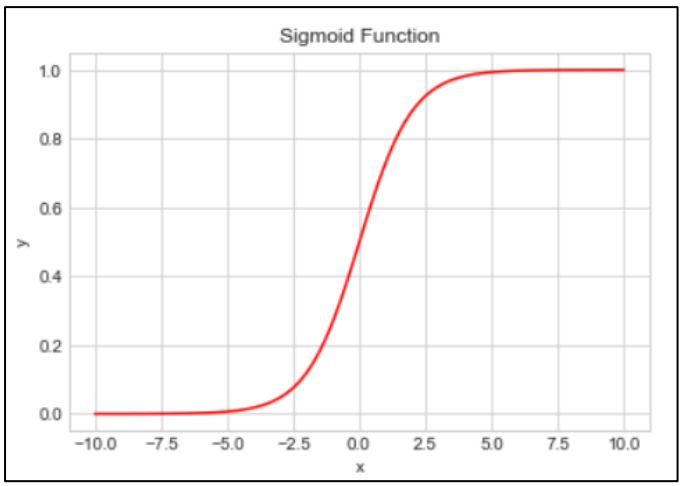
```
plt.plot(x, values)
plt.title ('Sigmoid Function')
plt.xlabel('x')
plt.ylabel('y')
```

```
Text(0, 0.5, 'y')
```



- Along with the ***plot()*** function, the line colour and style can be provided. This is supported in multiple ways. For example, if the plot line is needed in the red colour, anyone of the following will achieve the same result:

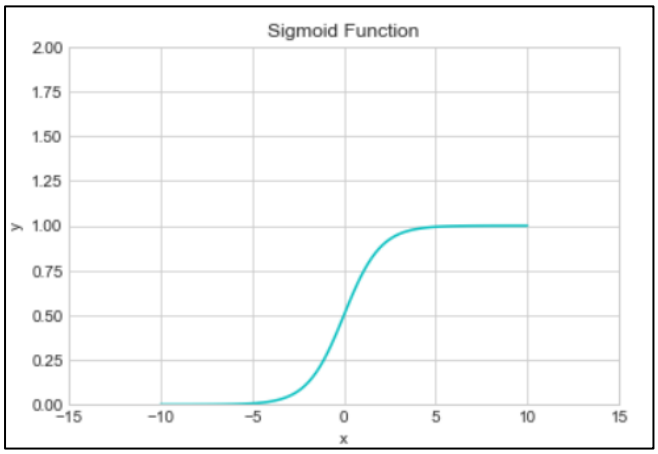
```
plt.plot(x, values, color = 'red')
plt.plot(x, values, color = 'r')       #Lower case
plt.plot(x, values, color= '#FF0000') #RGB Hex code for the red colour
```



- The lower case alphabet for RGBCMYK can be used for Red/Green/Blue/Cyan/Magenta/Yellow/Black colours.
- The limits of the axes can be set using the ***xlim()*** and ***ylim()*** functions. Alternatively ***axis()*** function can also be used to set the axes limits. From the following the first two lines together or the third line achieve the same result.

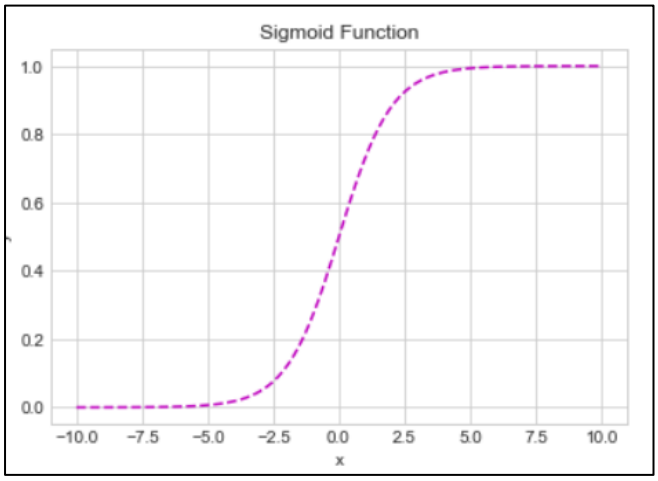
```
plt.xlim (-15, 15)
plt.ylim (0, 2)
```

```
plt.axis ([-15, 15, 0, 2])     # the argument is in a list
```



- The argument ***'tight'*** with the ***axis()*** function tightens the bounds only around the plot.
- The line style of the plot can be specified using the ***linestyle*** argument. The available line styles are solid (-), dashed (--), dashdot (-.) and dotted (:). For example:

```
plt.plot(x, values, color = 'm', linestyle = '--')
```

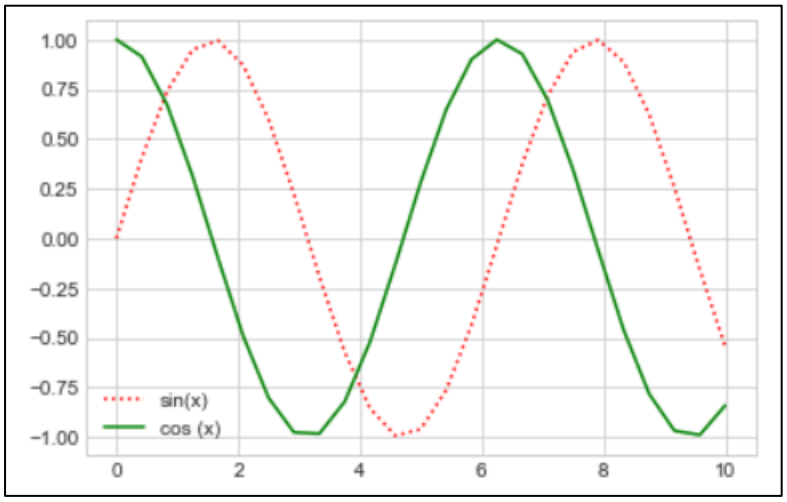


- Colour and linestyle can be provided together in a shorthand. For example dashed green line can be specified as:

```
plt.plot(x, values,'--g')
```

- Many times a single plot shows more than one function. For example, sin(x) and cos(x) on the same plot. A key has to be provided to distinguish which plot is what. This can be achieved using ***legend()*** function. Observe the example below for details of the ***legend()*** function and ***label*** attribute that is provided as an argument with the ***plot()*** function.

```
plt.plot(x, np.sin(x), 'r:', label = 'sin(x)')
plt.plot(x, np.cos(x), 'g-', label = 'cos (x)')
plt.legend()
```



- The location of the legend key can be specified using the ***loc*** argument of the ***legend ()*** function. Possible values are 'upper left', 'upper right', 'lower left', and 'lower right'.

Exercise:

1. Plot tanh(x) function in cyan colour with dots. Take the values of x from –5 to +5. The plot should be properly titled and axes labelled. The function tanh (x) is defined as $y = \frac{1-e^{-2x}}{1+e^{-2x}}$.
2. Plot Sigmoid and tanh(x) function on the same plot and provide legend to distinguish them.
3. Draw a parabola as $y = x^2$ for the values of x from -4 to +4. Use ***np.array ()*** function to provide the values of x and y. The array y should be prepared using the broadcast on x.