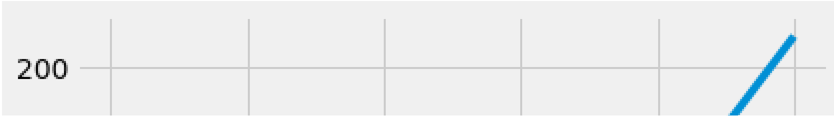
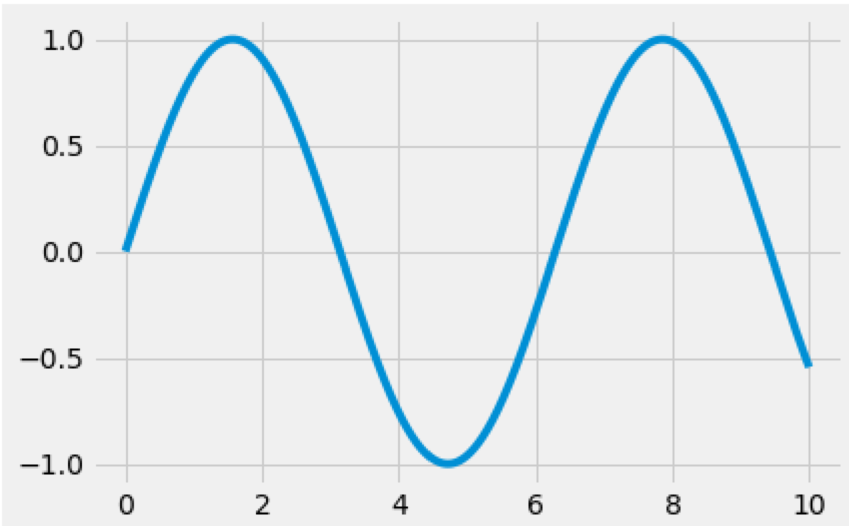
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline

#Graph Styling
# https://tonysyu.github.io/raw_content/matplotlib-style-gallery/gallery.html
plt.style.use('fivethirtyeight')
```

Line Graphs

```
# By default Plot() function will draw a line chart.
x = np.array([1,2,3,4,5,6])
y = np.power(x,3)
plt.plot(x,y)
plt.show() # not needed for jupyter notebook, but required for scipts and shell command
```

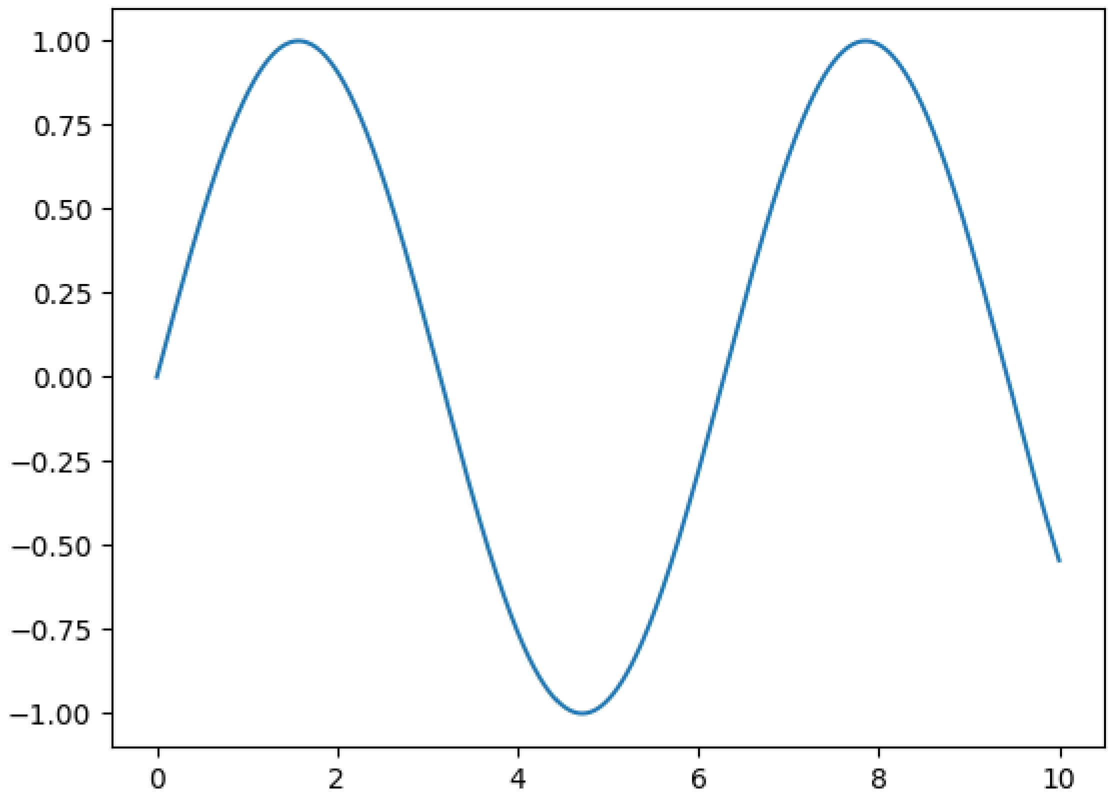



```
x = np.linspace(0, 10, 1000)
y = np.sin(x) # Sine Graph
plt.plot(x,y)
plt.show()
```

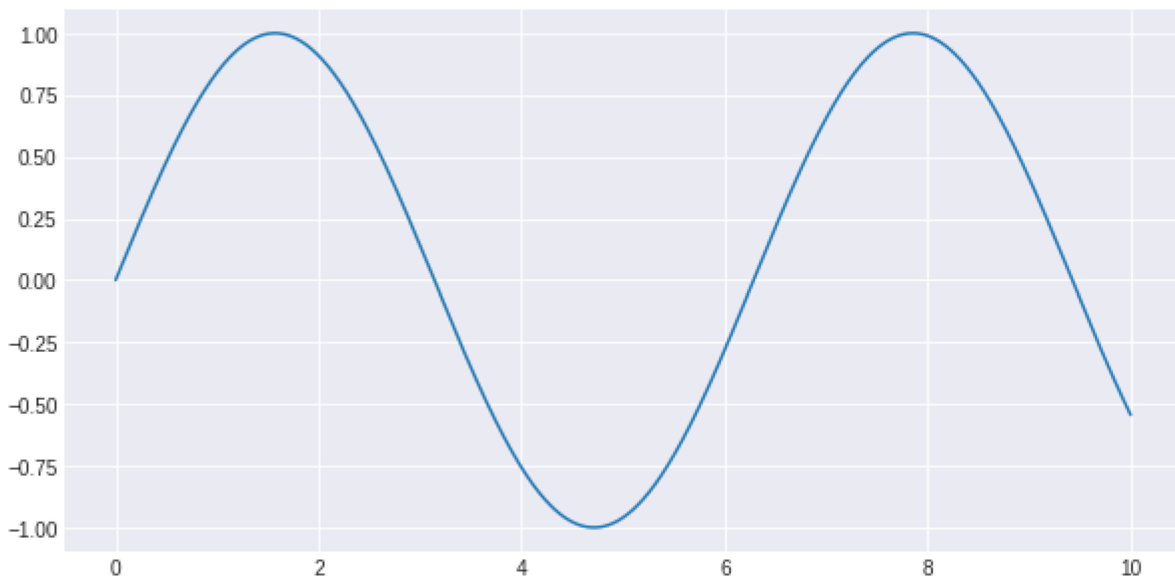



```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)

#print(plt.rcParams) # to examine all values

#print(plt.rcParams.get('figure.figsize'))

x = np.linspace(0, 10, 1000)
y = np.sin(x) # Sine Graph
plt.plot(x,y)
```

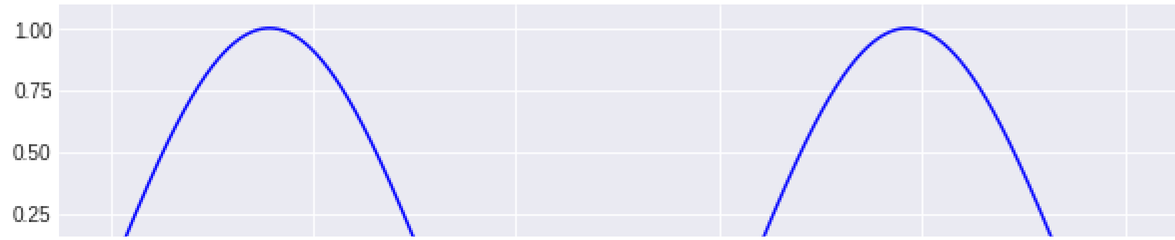
```
plt.plot(x,y)
plt.show()
```



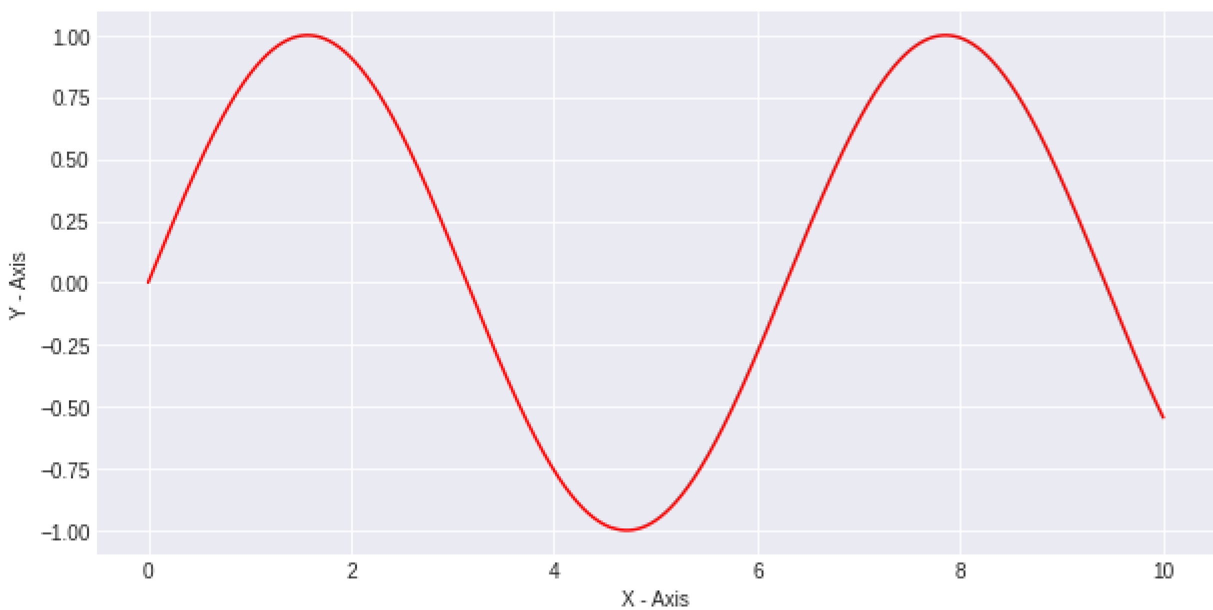

```
plt.style.use('seaborn-darkgrid')
%matplotlib inline

plt.figure(figsize=(10,5))
x = np.linspace(0, 10, 1000)
y = np.sin(x) # Sine Graph
plt.plot(x,y)
plt.show()
```




```
# Solid blue line will be plotted using the argument "b-"
plt.figure(figsize=(10,5))
x = np.linspace(0, 10, 1000)
y = np.sin(x) # Sine Graph
plt.plot(x,y,'b-')
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.show()
```




```
# Solid red line will be plotted using the argument "r-"
plt.figure(figsize=(10,5))
x = np.linspace(0, 10, 1000)
y = np.sin(x) # Sine Graph
plt.plot(x,y,'r-')
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.show()
```



For short, you can use the following codes:

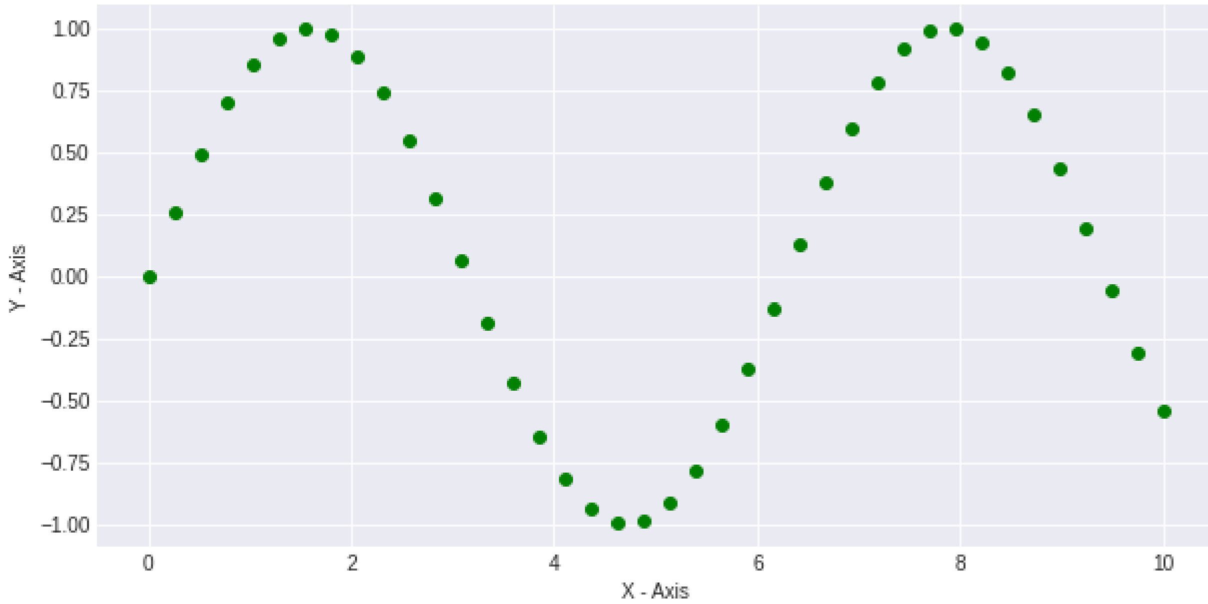
- plt.plot(x, np.sin(x - 0), color='blue') # specify color by name
- plt.plot(x, np.sin(x - 1), color='g') # short color code (rgbcmyk)
- plt.plot(x, np.sin(x - 2), color='0.75') # Grayscale between 0 and 1
- plt.plot(x, np.sin(x - 3), color='#FFDD44') # Hex code (RRGGBB from 00 to FF)
- plt.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3)) # RGB tuple, values 0 to 1
- plt.plot(x, np.sin(x - 5), color='chartreuse'); # all HTML color names supported

- plt.plot(x, x + 0, linestyle='solid')
- plt.plot(x, x + 1, linestyle='dashed')
- plt.plot(x, x + 2, linestyle='dashdot')
- plt.plot(x, x + 3, linestyle='dotted');

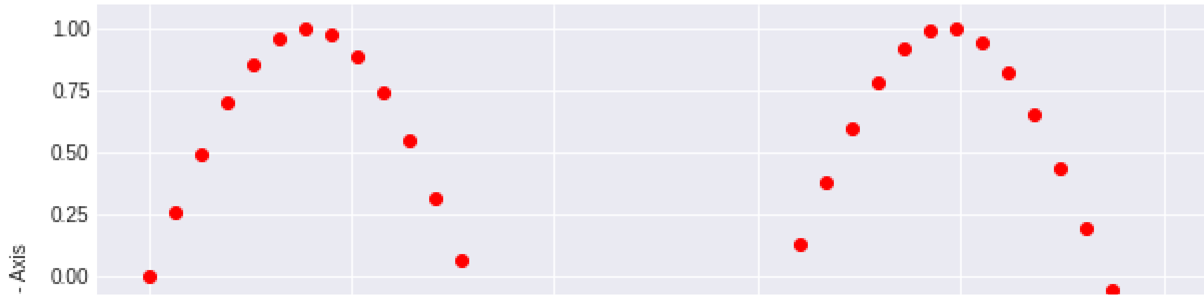
- plt.plot(x, x + 4, linestyle='-') # solid
- plt.plot(x, x + 5, linestyle='--') # dashed
- plt.plot(x, x + 6, linestyle='-.') # dashdot
- plt.plot(x, x + 7, linestyle=':'); # dotted

- plt.plot(x, x + 0, '-g') # solid green
- plt.plot(x, x + 1, '--c') # dashed cyan
- plt.plot(x, x + 2, '-.k') # dashdot black
- plt.plot(x, x + 3, ':r'); # dotted red

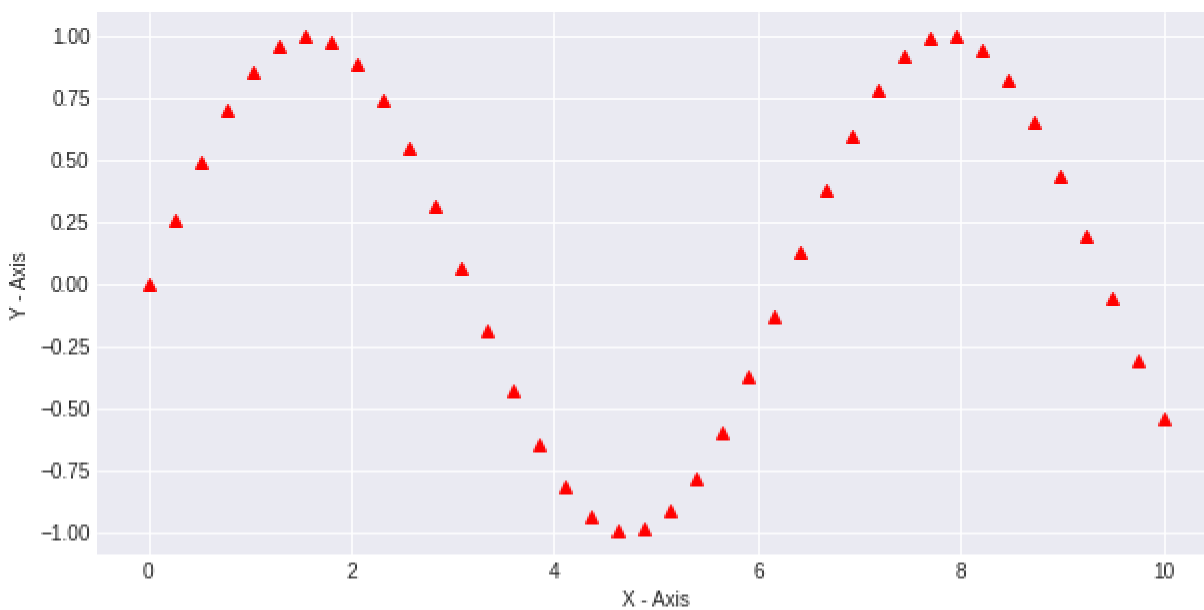
```
# Plot green dots using the argument "go"
plt.figure(figsize=(10,5))
x = np.linspace(0, 10, 40)
y = np.sin(x) # Sine Graph
plt.plot(x,y,'go')
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.show()
```



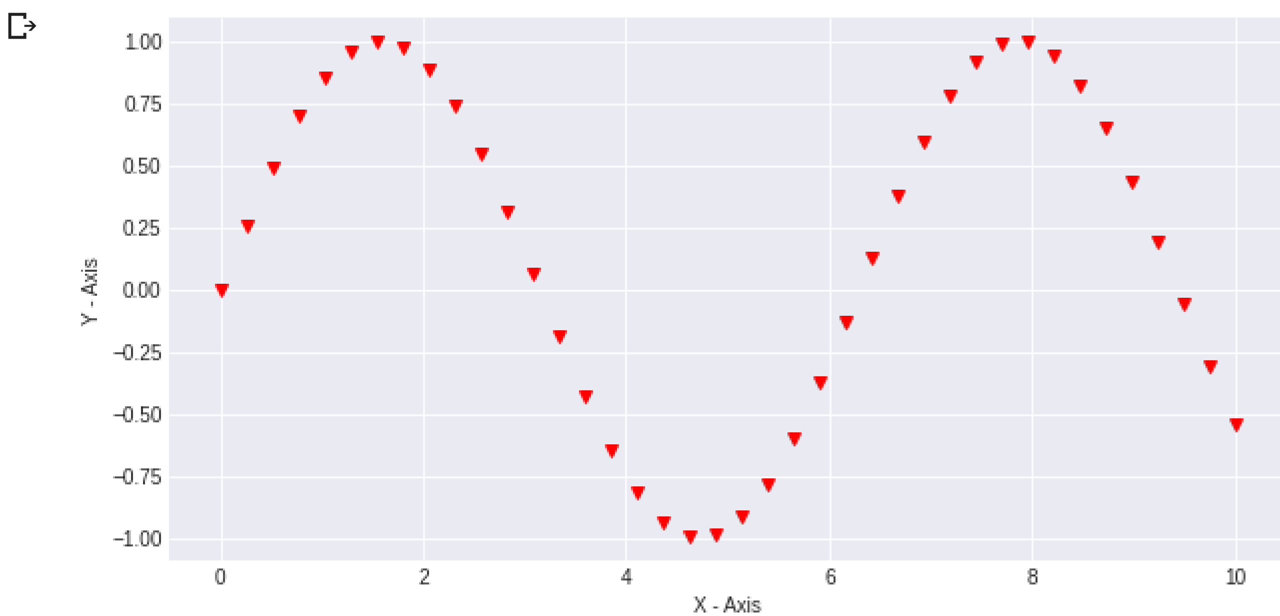

```
# Plotting red dots using the argument "ro"
plt.figure(figsize=(10,5))
x = np.linspace(0, 10, 40)
y = np.sin(x) # Sine Graph
plt.plot(x,y,'ro')
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.show()
```




```
# Plotting traingular dots using the argument "r^"
plt.figure(figsize=(10,5))
x = np.linspace(0, 10, 40)
y = np.sin(x) # Sine Graph
plt.plot(x,y,'r^')
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.show()
```

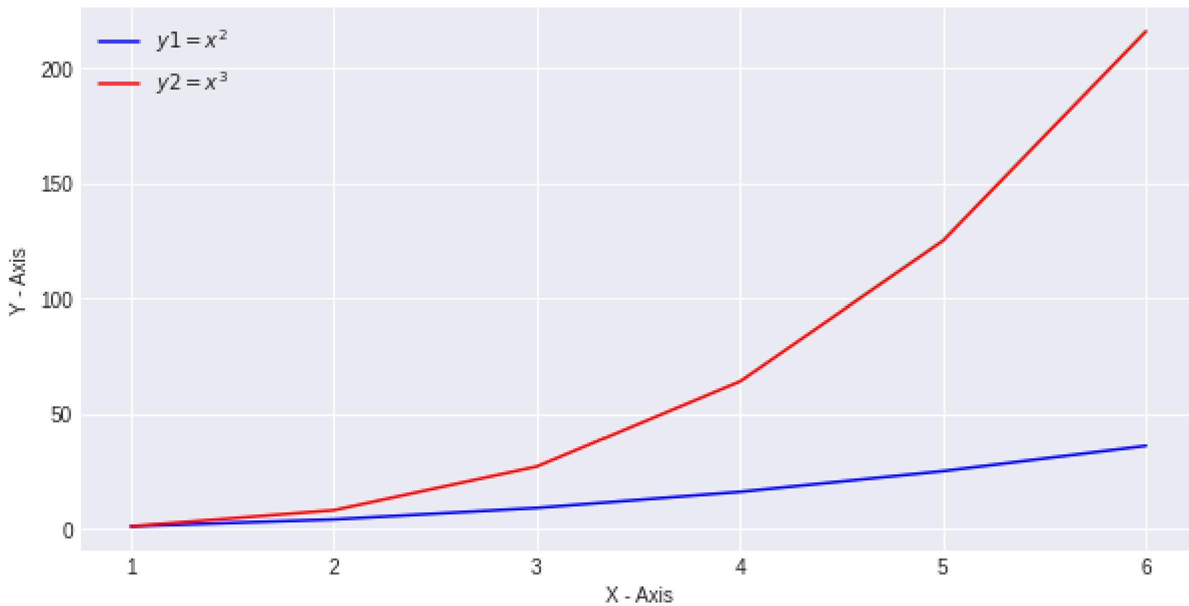


```
# Plotting traingular dots using the argument "rv"
plt.figure(figsize=(10,5))
x = np.linspace(0, 10, 40)
y = np.sin(x) # Sine Graph
plt.plot(x,y,'rv')
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.show()
```

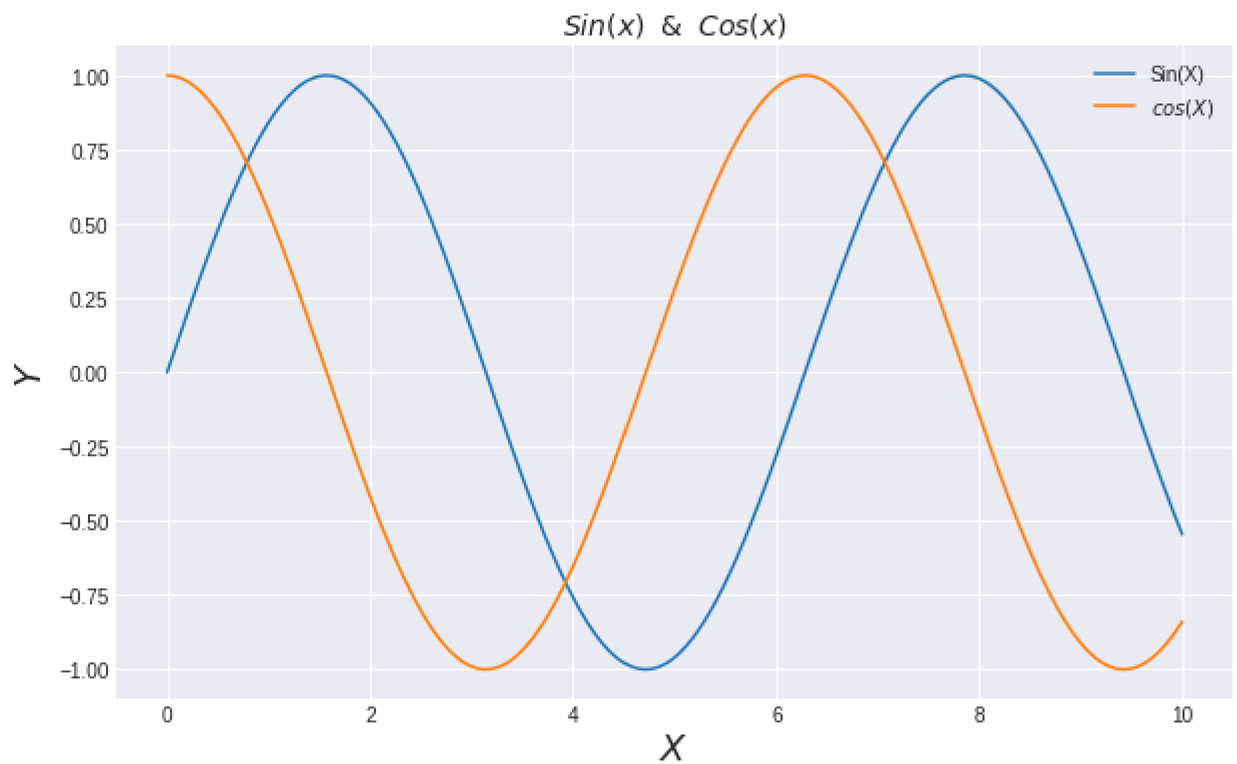



```
#Plotting multiple sets of data
plt.figure(figsize=(10,5))
x = np.array([1,2,3,4,5,6])
y1 = np.power(x,2)
y2 = np.power(x,3)
plt.plot(x,y1, "b-" , label = '$y1 = x^2$') # Setting up legends
plt.plot(x,y2, "r-" ,label ='$y2 = x^3$')   # Setting up legends
plt.xlabel("X - Axis")
```

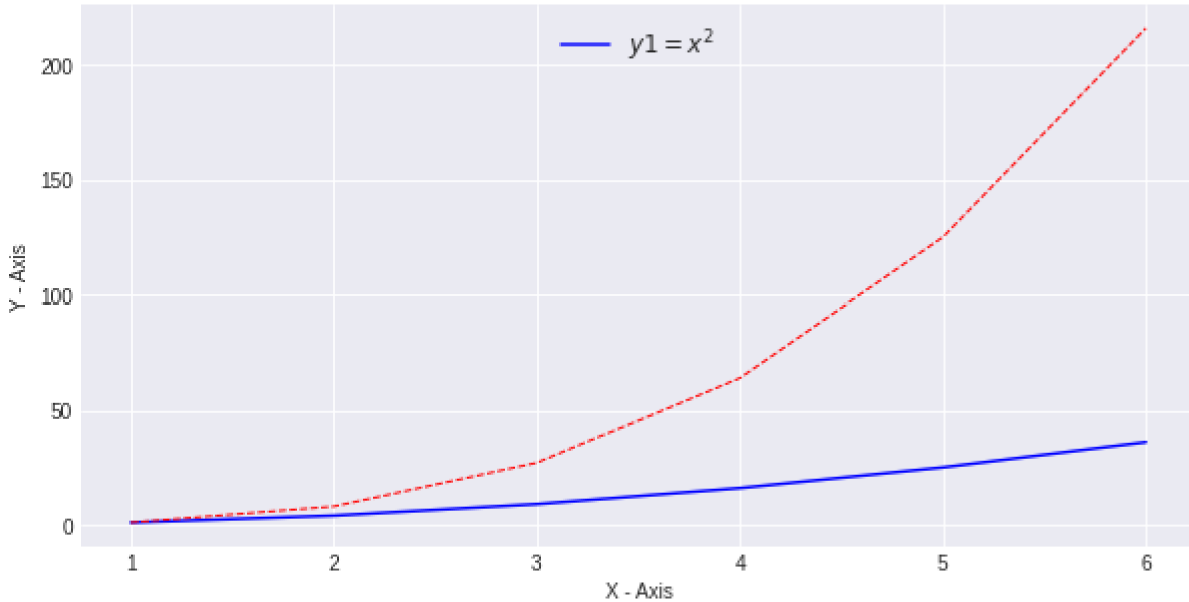
```
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.legend()
#plt.tight_layout() # The tight_layout() function in pyplot module of matplotlib librar
plt.show()
```



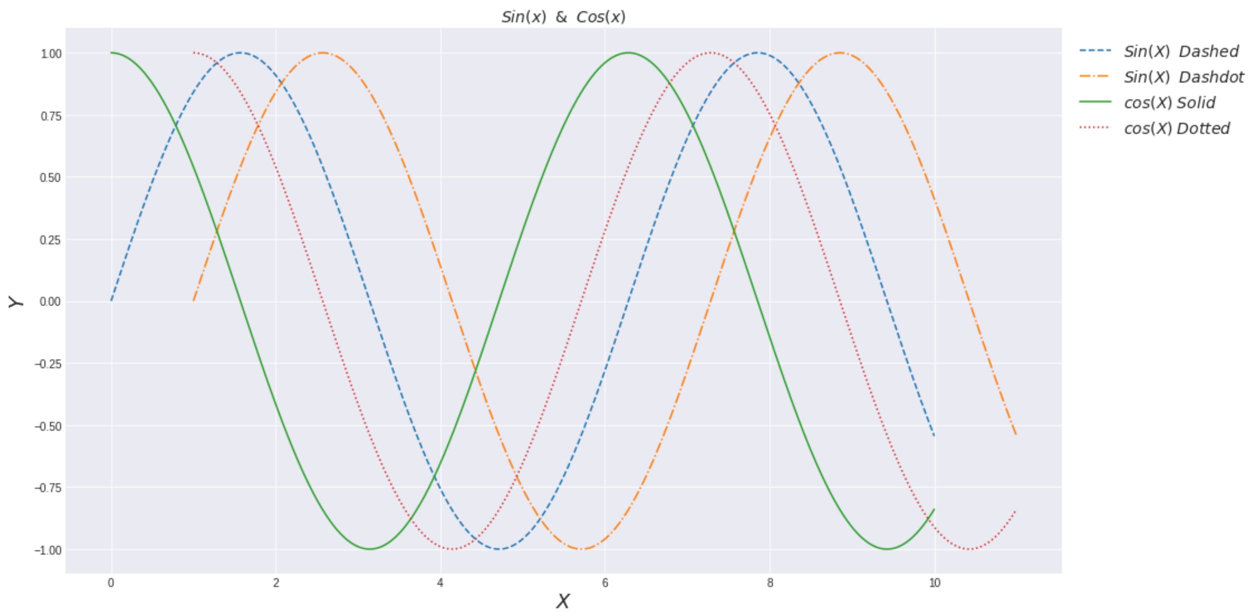

```
#Plotting multiple sets of data
x = np.linspace(0, 10, 2000)
plt.figure(figsize=(10,6))
plt.plot(x,np.sin(x) , label = 'Sin(X)') # see the difference of enclosing $$
plt.plot(x,np.cos(x) , label = '$cos(X)$')
plt.xlabel(r'$X$' , fontsize = 18)
plt.ylabel(r'$Y$' , fontsize = 18)
plt.title("$Sin(x) $  $ & $  $ Cos(x)$" ,fontsize = 14)
plt.legend(loc = 'upper right') # Legend will be placed at upper right position
plt.show()
```




```
#Changing the line style
plt.figure(figsize=(10,5))
x = np.array([1,2,3,4,5,6])
y1 = np.power(x,2)
y2 = np.power(x,3)
plt.plot(x,y1, "b-" , label = '$y1 = x^2$') # Setting up legends
plt.plot(x, y2,color='red',linewidth=1.0,linestyle='--')   # Setting up legends
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.legend(loc='upper center', fontsize='large')
plt.show()
```




```
# Line Styling
x = np.linspace(0, 10, 2000)
plt.figure(figsize=(16, 9))
plt.plot(x,np.sin(x) , label = '$Sin(X) $  $ Dashed $' , linestyle='dashed')
plt.plot(x+1,np.sin(x) , label = '$Sin(X) $  $ Dashdot  $' , linestyle='dashdot')
plt.plot(x,np.cos(x) , label = '$cos(X) $ $ Solid $' , linestyle='solid')
plt.plot(x+1,np.cos(x) , label = '$cos(X)$ $ Dotted $' , linestyle='dotted')
plt.xlabel('$X$' , fontsize = 18)
plt.ylabel('$Y$' , fontsize = 18)
plt.title("$Sin(x) $  $ & $  $ Cos(x)$" ,fontsize = 14)
plt.legend(loc = 'upper right' , fontsize = 14 , bbox_to_anchor=(1.2, 1.0)) # Legend wi
plt.show()
```

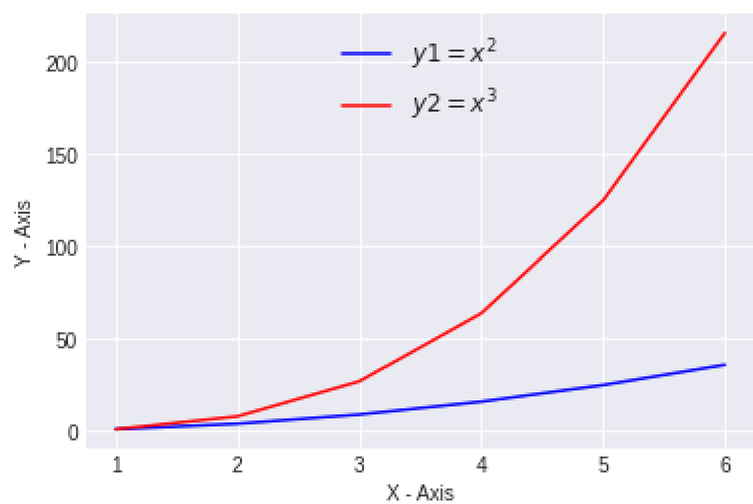



```
# Line Styling
x = np.linspace(0, 10, 2000)
plt.figure(figsize=(16, 9))
plt.plot(x,np.sin(x) , label = '$Sin(X) $  $ Dashed $' , linestyle='--')
plt.plot(x+1,np.sin(x) , label = '$Sin(X) $  $ Dashdot  $' , linestyle='-.')
plt.plot(x,np.cos(x) , label = '$cos(X) $ $ Solid $' , linestyle='-')
plt.plot(x+1,np.cos(x) , label = '$cos(X)$ $ Dotted $' , linestyle=':')
plt.xlabel('$X$' , fontsize = 18)
plt.ylabel('$Y$' , fontsize = 18)
```

```
plt.title(r'$\alpha_i > \beta_i$')  #r followed by mathematical expression
#r'$\alpha_i > \beta_i$'
#r'$\sum_{i=0}^\infty x_i$'

#plt.title("$Sin(x) $  $ & $  $ Cos(x)$" ,fontsize = 14)
plt.legend(loc = 'upper right' , fontsize = 14 , bbox_to_anchor=(1.2, 1.0)) # Legend wi
plt.show()
```

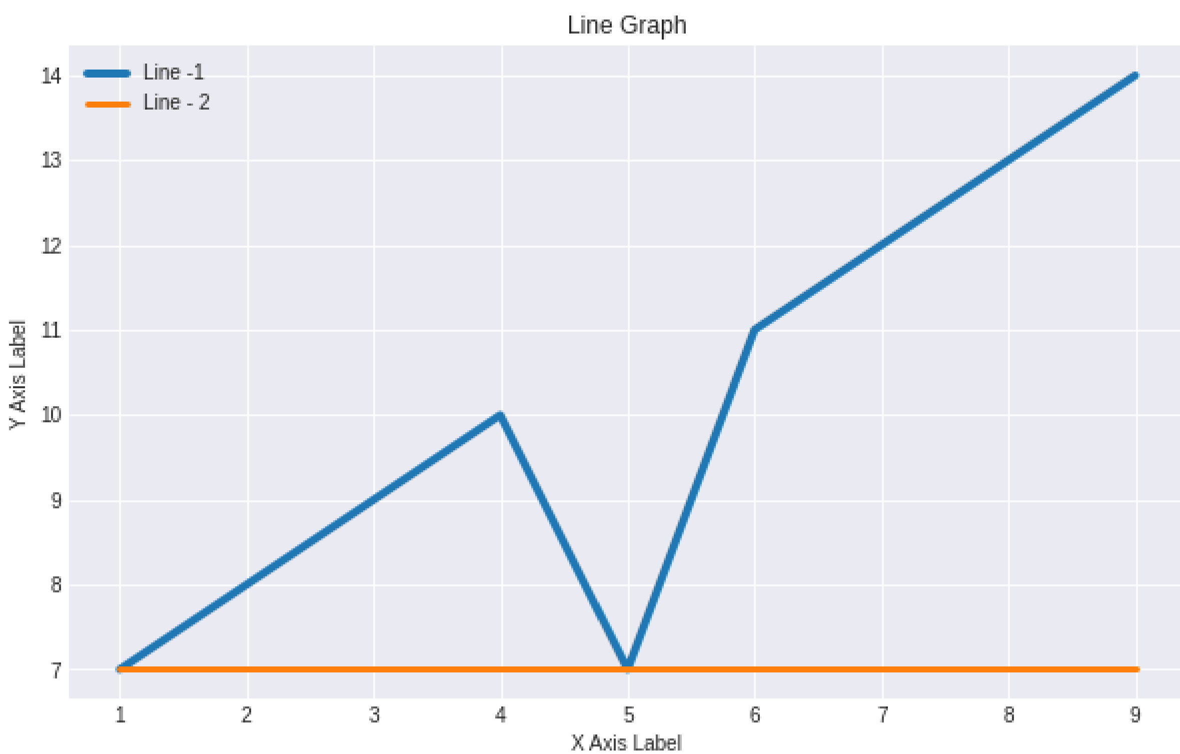

```
# Shading Regions with fill_between() function
x = np.linspace(0, 10, 2000)
plt.figure(figsize=(10,6))
plt.plot(x,np.sin(x) , label = '$Sin(X)$')
plt.plot(x,np.cos(x) , label = '$cos(X)$')
plt.fill_between(x,0,np.sin(x))
plt.fill_between(x,0,np.cos(x))
plt.xlabel(r'$X$' , fontsize = 18)
plt.ylabel(r'$Y$' , fontsize = 18)
plt.title("$Sin(x) $  $ & $  $ Cos(x)$" ,fontsize = 14)
plt.legend(loc = 'lower left') # Legend will be placed at lower left position
plt.show()
```


```
#Changing Legend position & font
x = np.array([1,2,3,4,5,6])
y1 = np.power(x,2)
y2 = np.power(x,3)
plt.plot(x,y1, "b-" , label = '$y1 = x^2$') # Setting up legends
plt.plot(x,y2, "r-" ,label ='$y2 = x^3$')   # Setting up legends
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.legend(loc='upper center', fontsize='large')
plt.show()
```



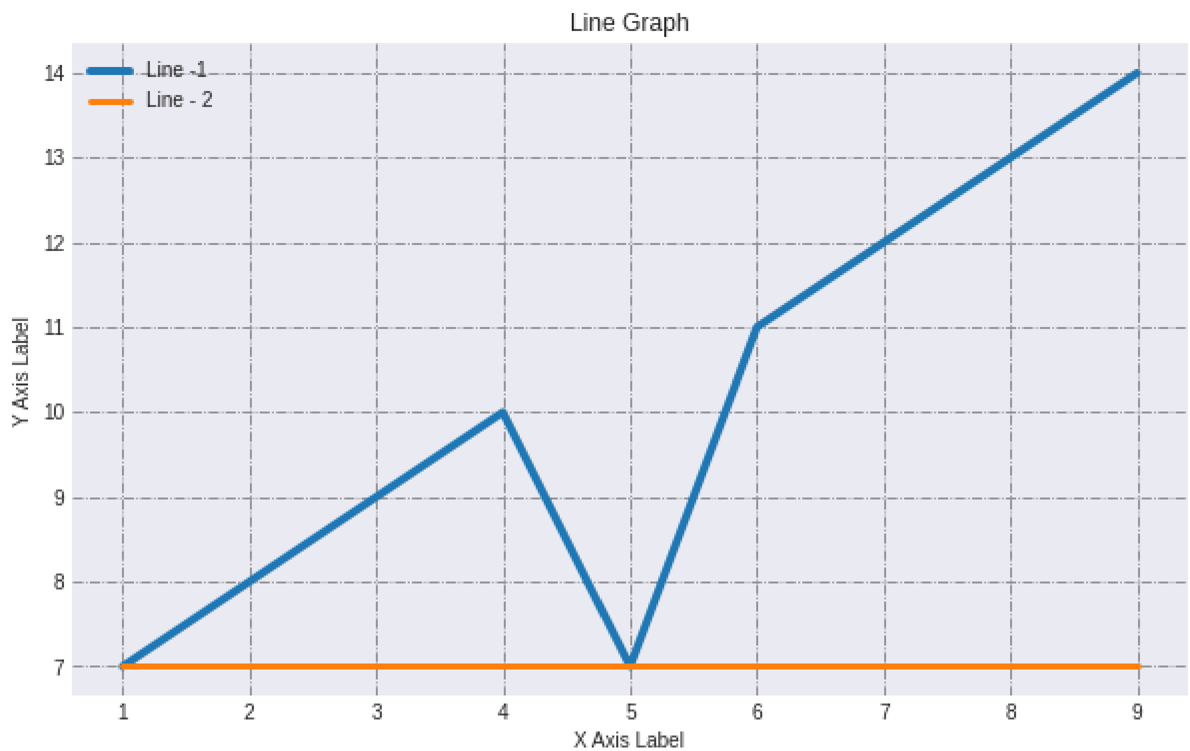

```
# Changing line width
plt.figure(figsize=(10,6))
x= [1,2,3,4,5,6,7,8,9]
y= [7,8,9,10,7,11,12,13,14]
y2 = [7,7,7,7,7,7,7,7,7]
```

```
plt.plot(x , y, linewidth = 4 ,label = 'Line -1')  # Changing line width
plt.plot(x , y2, linewidth = 3,label = 'Line - 2')
plt.xlabel('X Axis Label')
plt.ylabel('Y Axis Label')
plt.title ('Line Graph')
plt.legend()
plt.show()
```



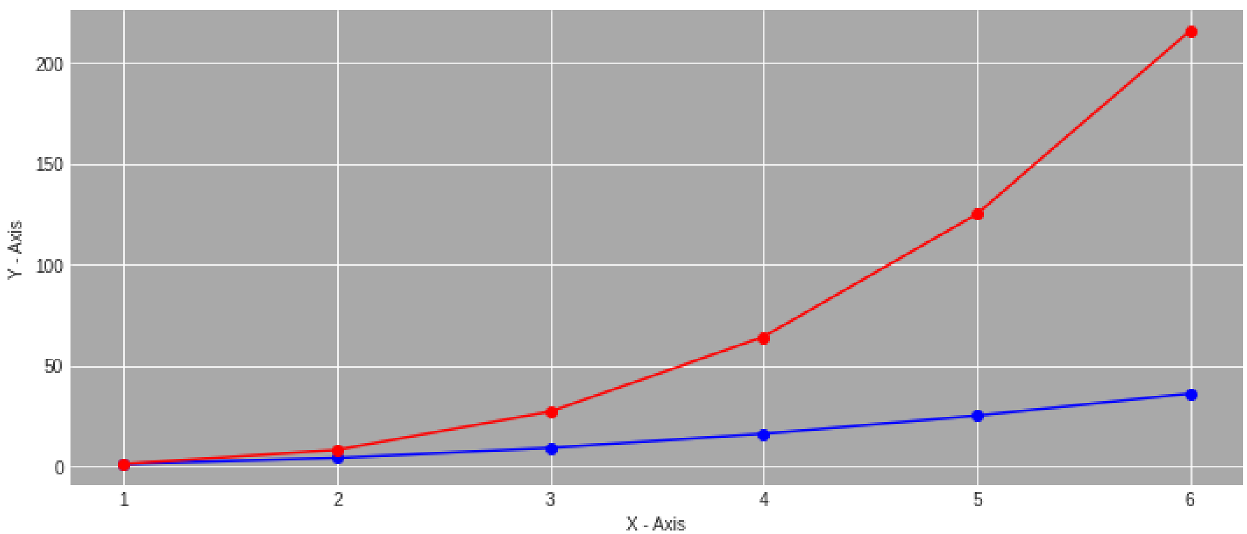

```
# Plot with Grid Lines
plt.figure(figsize=(10,6))
x= [1,2,3,4,5,6,7,8,9]
y= [7,8,9,10,7,11,12,13,14]
y2 = [7,7,7,7,7,7,7,7,7]
plt.plot(x , y, linewidth = 4 ,label = 'Line -1')  # Changing line width
```

```
plt.plot(x , y, linewidth = 4 ,label = 'Line -1') # Changing line width
plt.plot(x , y2, linewidth = 3,label = 'Line - 2')
plt.xlabel('X Axis Label')
plt.ylabel('Y Axis Label')
plt.title ('Line Graph')
plt.legend()
plt.grid(b=True , linestyle = '-.' , which = 'major' , color = 'grey') # Grid Lines
plt.show()
```



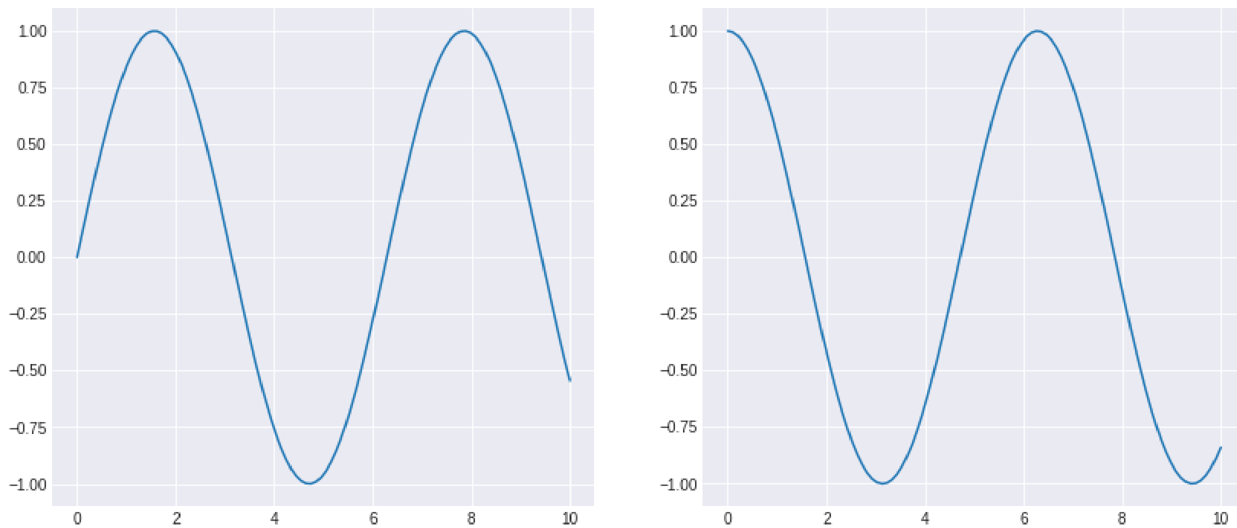

```
# Setting the background color
x = np.array([1,2,3,4,5,6])
y1 = np.power(x,2)
y2 = np.power(x,3)
plt.figure(figsize=(12,5)) # Setting the figure size
```

```
ax = plt.axes()
ax.set_facecolor("darkgrey") # Setting the background color by using Hex code
plt.plot(x,y1,"bo-", x,y2, "ro-")
plt.xlabel("X - Axis")
plt.ylabel("Y - Axis")
plt.show()
```



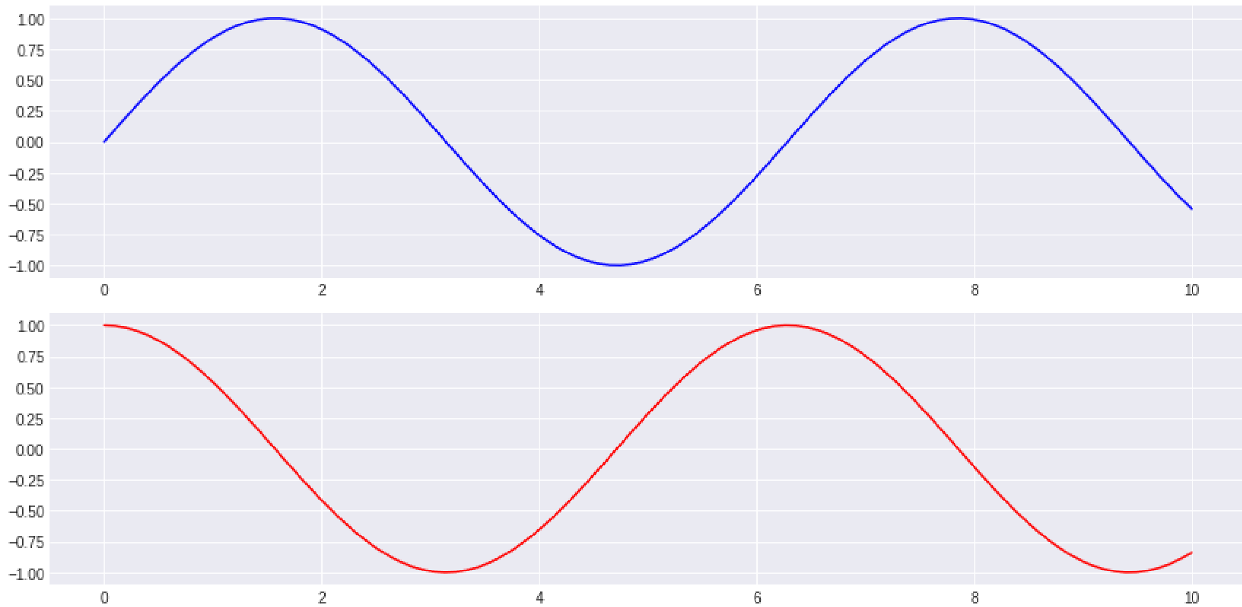

```
# Display multiple plots in one figure (1 row & 2 columns)
plt.figure(figsize=(14,6))
x = np.linspace(0, 10, 100)
y1 = np.sin(x) # Sine Graph
y2 = np.cos(x) # cosine graph
plt.subplot(1,2,1)
plt.plot(x,y1)
plt.subplot(1,2,2)
plt.plot(x,y2)
```

```
plt.show()
```




```
# Display multiple plots in one figure (2 row & 1 columns)

plt.figure(figsize=(12,6))
x = np.linspace(0, 10, 100)
y1 = np.sin(x) # Sine Graph
y2 = np.cos(x) # cosine graph

plt.subplot(2,1,1)
plt.plot(x,y1, "b-")

plt.subplot(2,1,2)
plt.plot(x,y2, "r-")
plt.tight_layout()
```

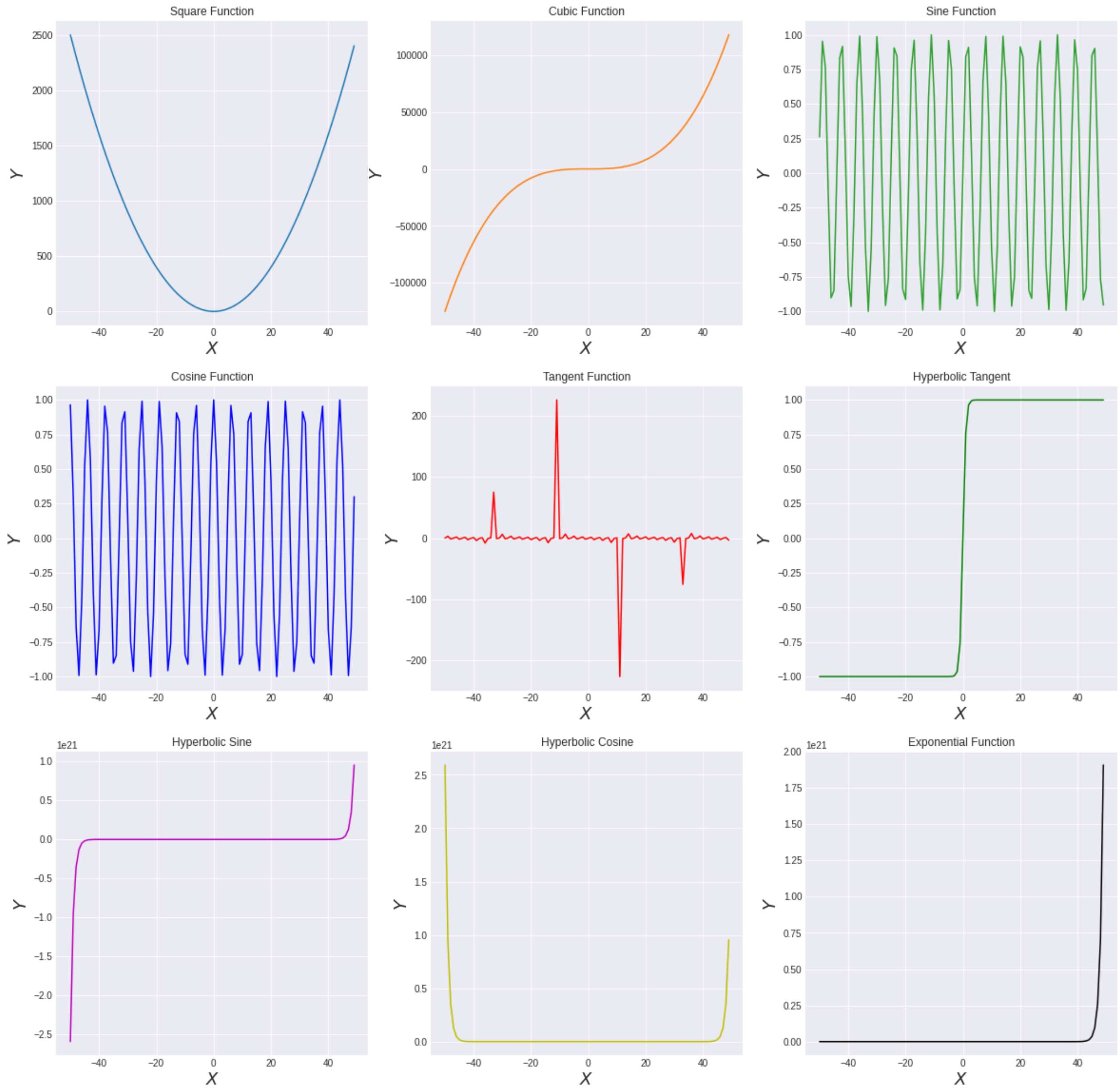
```
plt.tight_layout()
plt.show()
```




```
# # Display multiple plots in one figure using subplots()
x = np.arange(-50,50)
y1 = np.power(x,2)
y2 = np.power(x,3)
y3 = np.sin(x)
y4 = np.cos(x)
y5 = np.tan(x)
y6 = np.tanh(x)
y7 = np.sinh(x)
```

```
y8 = np.cosh(x)
y9 = np.exp(x)

fig1 , ax1 = plt.subplots(nrows=3,ncols=3 , figsize = (20,20)) # Create a figure and su
ax1[0,0].plot(x,y1,"tab:blue") # set the color of the line chart
ax1[0,0].set_title("Square Function") # setting title of subplot
ax1[0,0].set_xlabel(r'$X$' , fontsize = 18) #Set the label for the x-axis
ax1[0,0].set_ylabel(r'$Y$' , fontsize = 18) #Set the label for the y-axis


ax1[0,1].plot(x,y2,"tab:orange")
ax1[0,1].set_title("Cubic Function")
ax1[0,1].set_xlabel(r'$X$' , fontsize = 18)
ax1[0,1].set_ylabel(r'$Y$' , fontsize = 18)

ax1[0,2].plot(x,y3,"tab:green")
ax1[0,2].set_title("Sine Function")
ax1[0,2].set_xlabel(r'$X$' , fontsize = 18)
ax1[0,2].set_ylabel(r'$Y$' , fontsize = 18)


ax1[1,0].plot(x,y4,"b-")
ax1[1,0].set_title("Cosine Function")
ax1[1,0].set_xlabel(r'$X$' , fontsize = 18)
ax1[1,0].set_ylabel(r'$Y$' , fontsize = 18)


ax1[1,1].plot(x,y5,"r-")
ax1[1,1].set_title("Tangent Function")
ax1[1,1].set_xlabel(r'$X$' , fontsize = 18)
```

```
ax1[1,1].set_xlabel(r'$X$' , fontsize = 18)
ax1[1,1].set_ylabel(r'$Y$' , fontsize = 18)

ax1[1,2].plot(x,y6,"g-")
ax1[1,2].set_title("Hyperbolic Tangent")
ax1[1,2].set_xlabel(r'$X$' , fontsize = 18)
ax1[1,2].set_ylabel(r'$Y$' , fontsize = 18)

ax1[2,0].plot(x,y7,"m-")
ax1[2,0].set_title("Hyperbolic Sine")
ax1[2,0].set_xlabel(r'$X$' , fontsize = 18)
ax1[2,0].set_ylabel(r'$Y$' , fontsize = 18)

ax1[2,1].plot(x,y8,"y-")
ax1[2,1].set_title("Hyperbolic Cosine")
ax1[2,1].set_xlabel(r'$X$' , fontsize = 18)
ax1[2,1].set_ylabel(r'$Y$' , fontsize = 18)

ax1[2,2].plot(x,y9,"k-")
ax1[2,2].set_title("Exponential Function")
ax1[2,2].set_xlabel(r'$X$' , fontsize = 18)
ax1[2,2].set_ylabel(r'$Y$' , fontsize = 18)

plt.show()
```


Square Function
Cubic Function
Sine Function
Cosine Function
Tangent Function
Hyperbolic Tangent
Hyperbolic Sine
Hyperbolic Cosine
Exponential Function
1e21
X
Y

```
y = [[1,2,3,4,5] , [10,20,30,40,50],[60,70,80,90,100] ]
cnt =0
plt.figure(figsize=(10,6))
for i in y:
    x1 = [10,20,30,40,50]
    cnt +=1
    print ('iteration Number :- {}'.format(cnt))
    print ('X1 Value :- {}'.format(x1))
    print('Y value (i) :- {}'.format(i))
    plt.plot(x1,i)
plt.show()
```

```
iteration Number :- 1
X1 Value :- [10, 20, 30, 40, 50]
Y value (i) :- [1, 2, 3, 4, 5]
iteration Number :- 2
X1 Value :- [10, 20, 30, 40, 50]
Y value (i) :- [10, 20, 30, 40, 50]
iteration Number :- 3
X1 Value :- [10, 20, 30, 40, 50]
Y value (i) :- [60, 70, 80, 90, 100]
```