**Prepared by Asif Bhat**

# Data Visualization With Python (Matplotlib - Part 1)

In [74]:

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

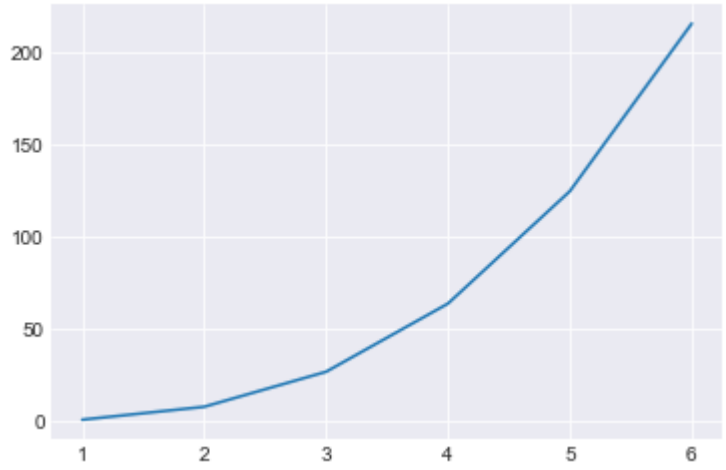
In [75]:

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

# Line Graphs

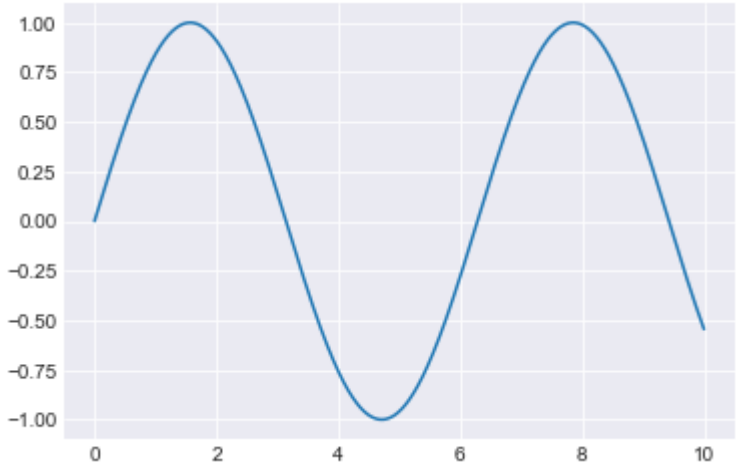
In [76]:

```
# 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()
```



In [77]:

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

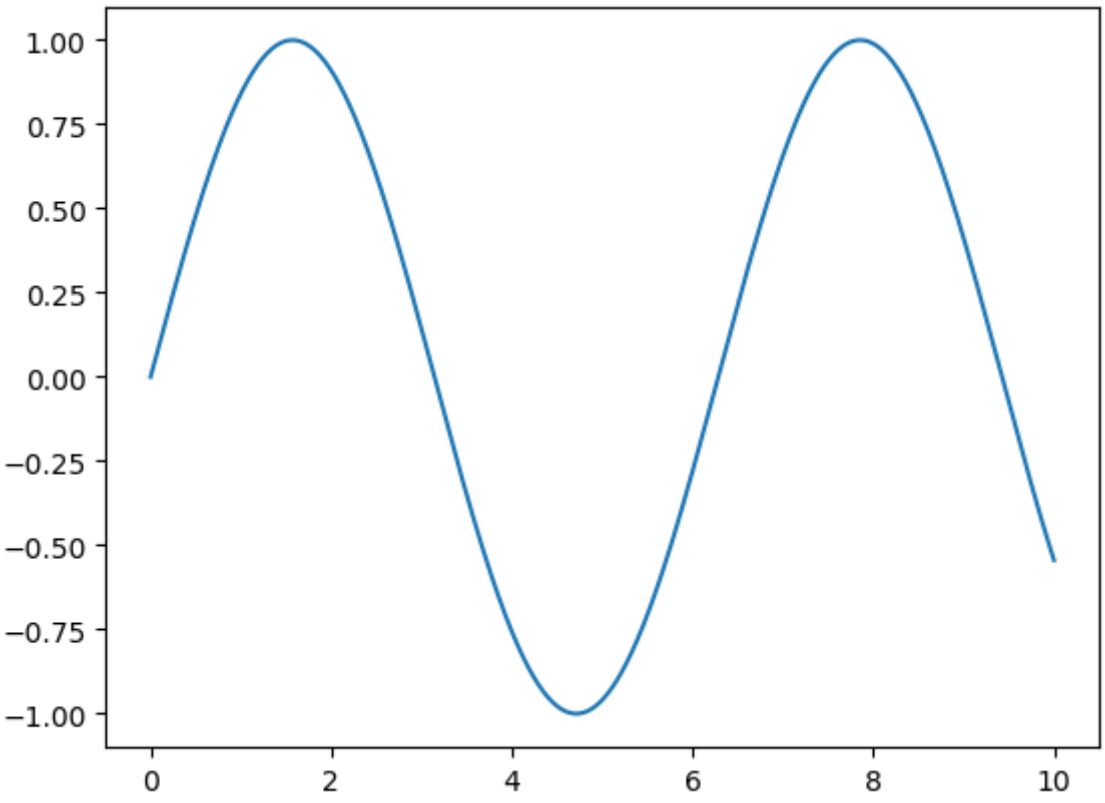



In [78]:

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

In [79]:

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

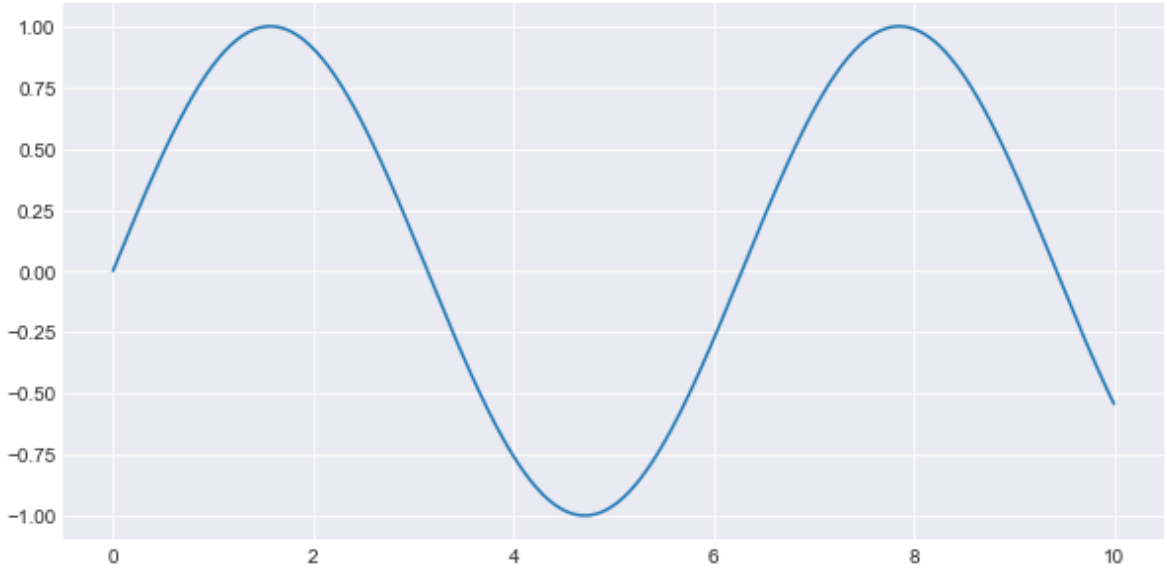


In [80]:

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

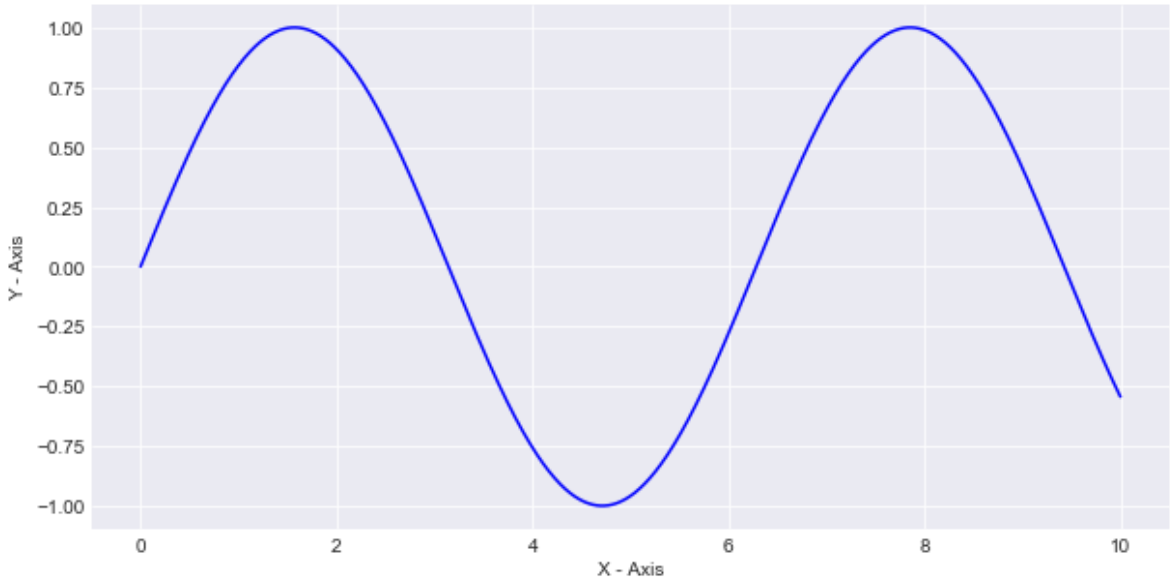
In [81]:

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



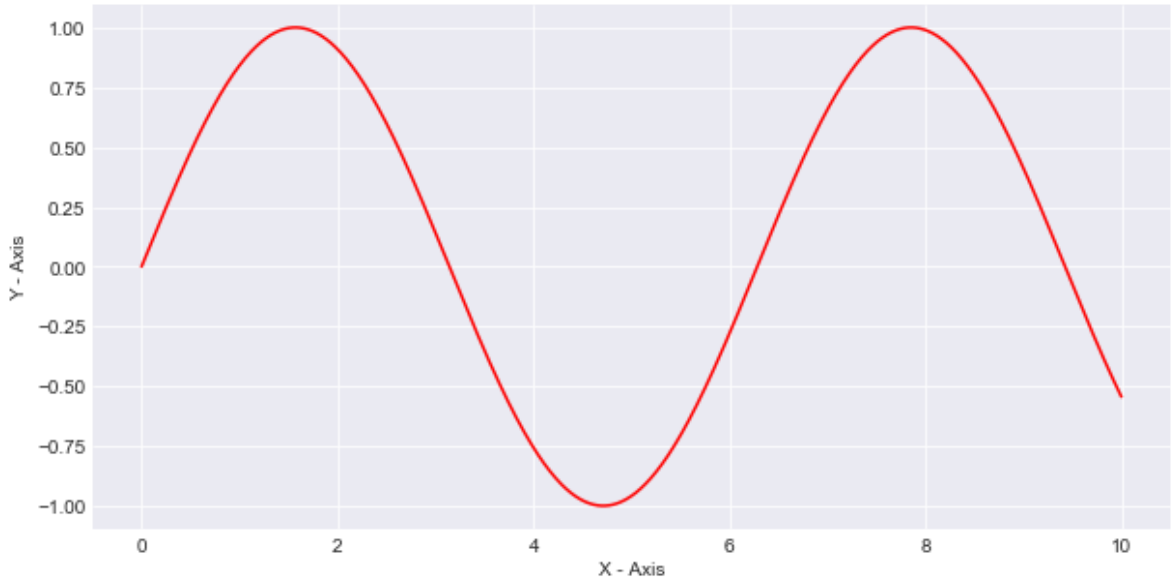
In [82]:

```
# 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()
```



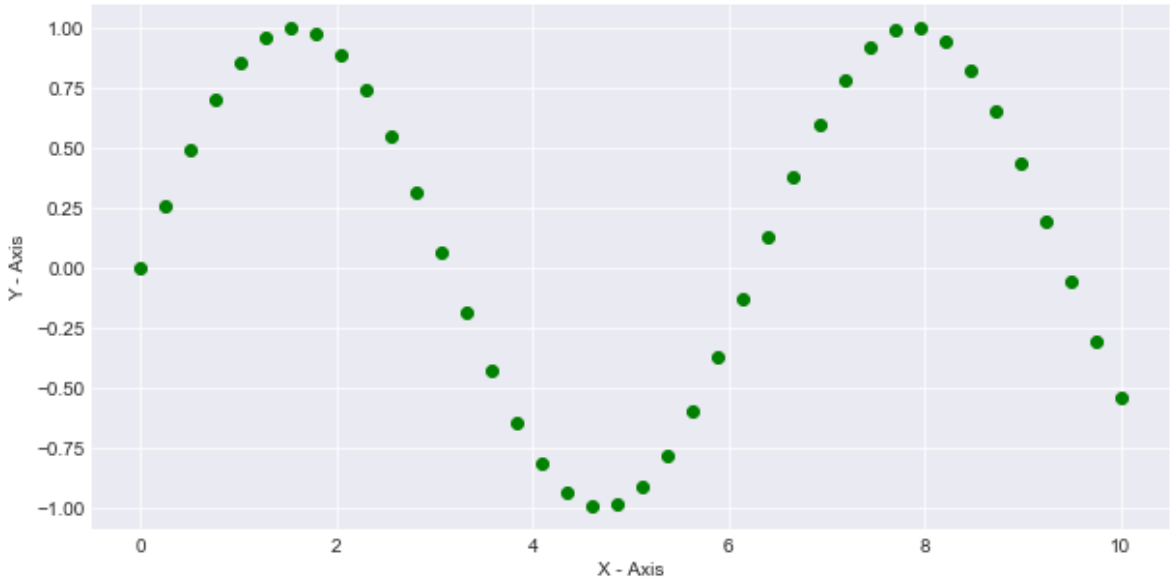
In [83]:

```
# 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()
```



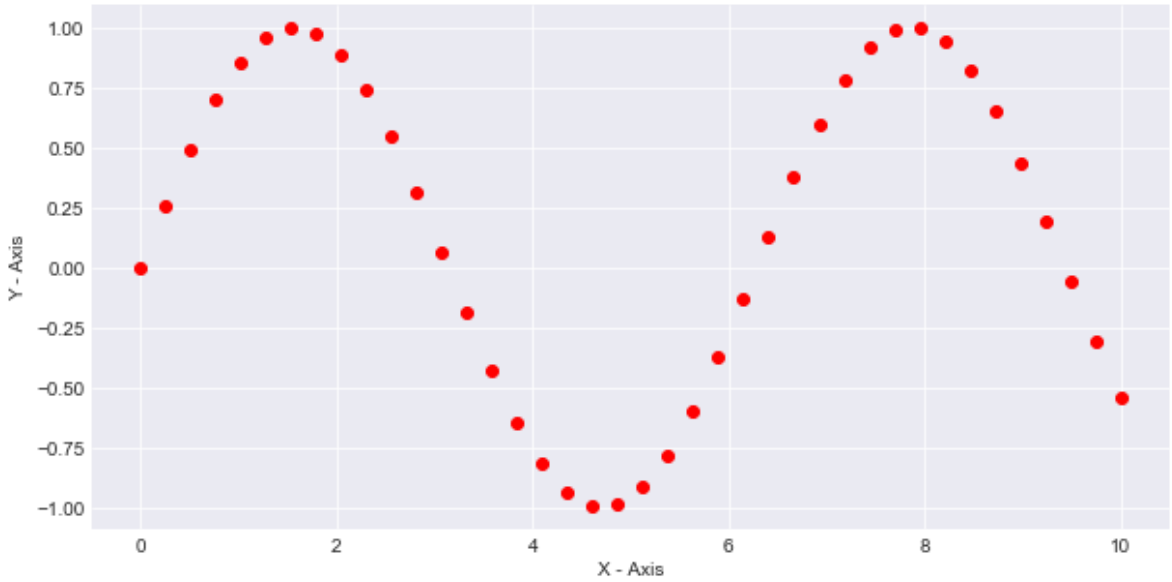
In [84]:

```
# 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()
```



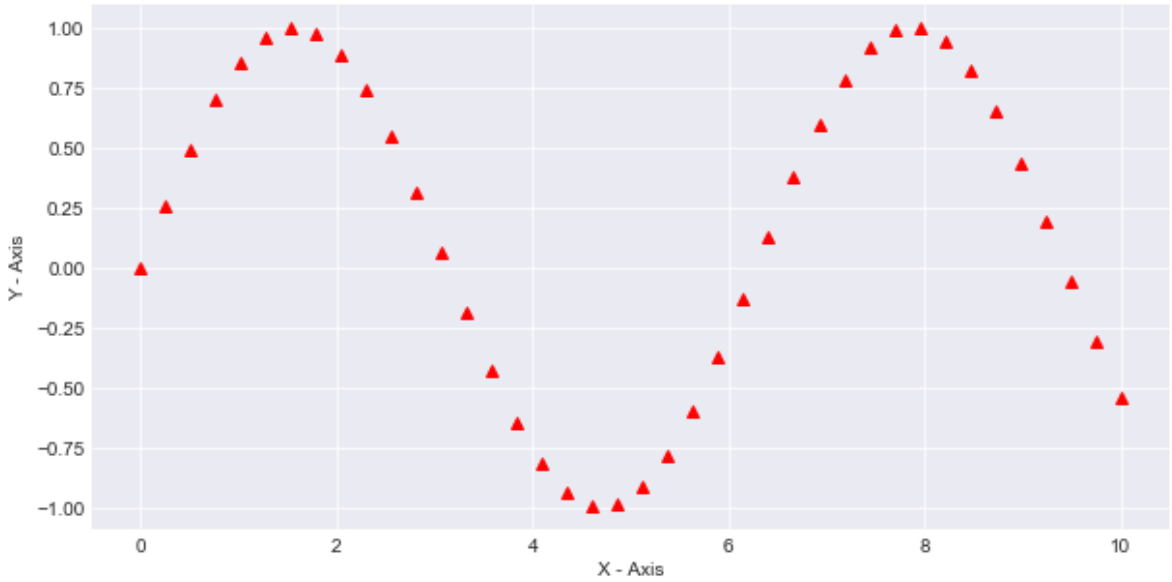
In [85]:

```
# 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()
```



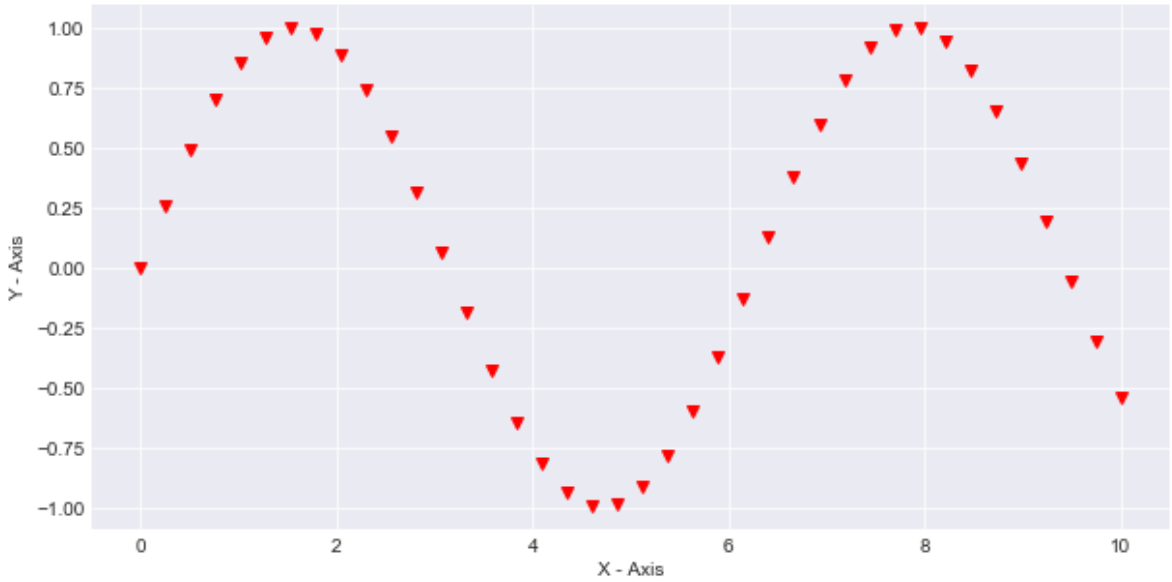
In [86]:

```
# 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()
```



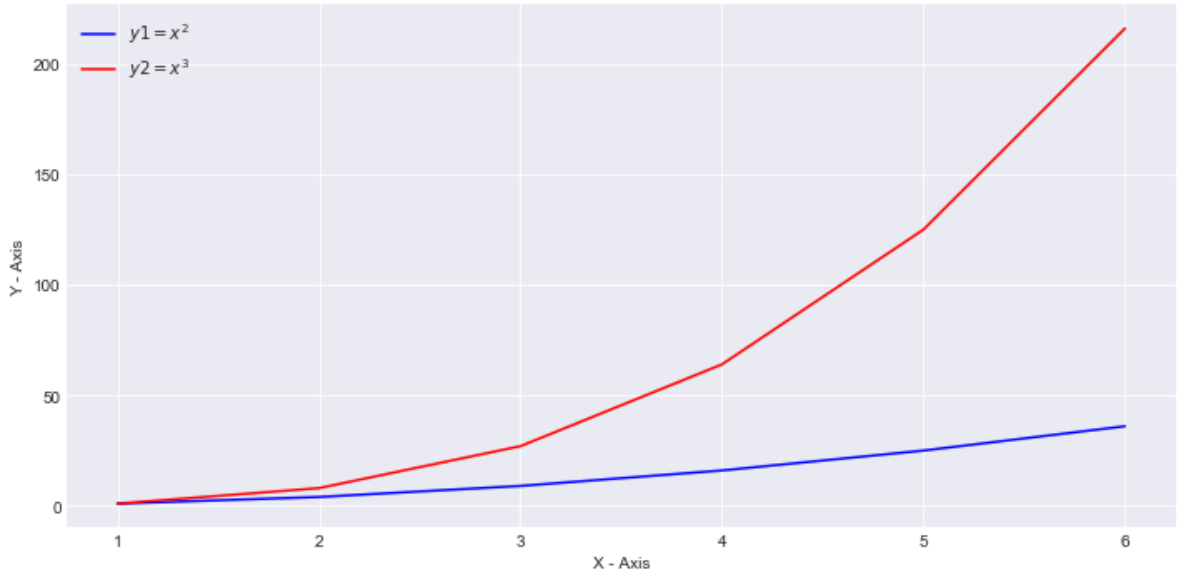
In [115]:

```
# 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()
```



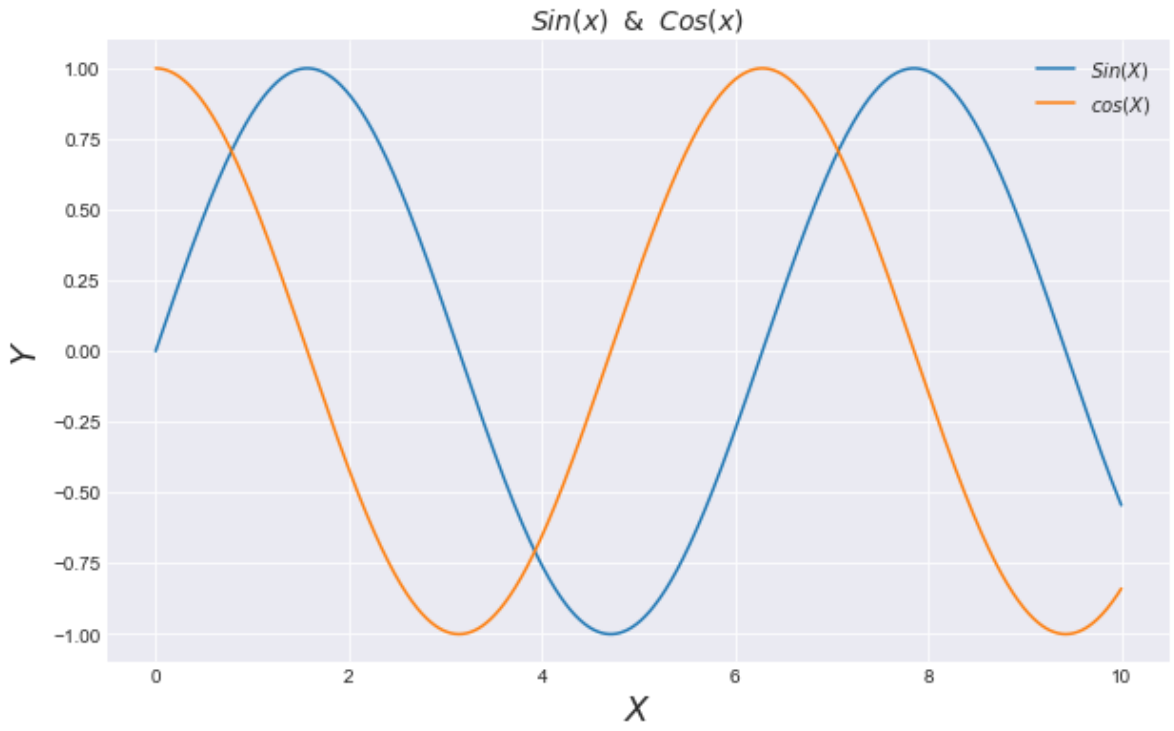
In [88]:

```
#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.ylabel("Y - Axis")
plt.legend()
plt.tight_layout()
plt.show()
```



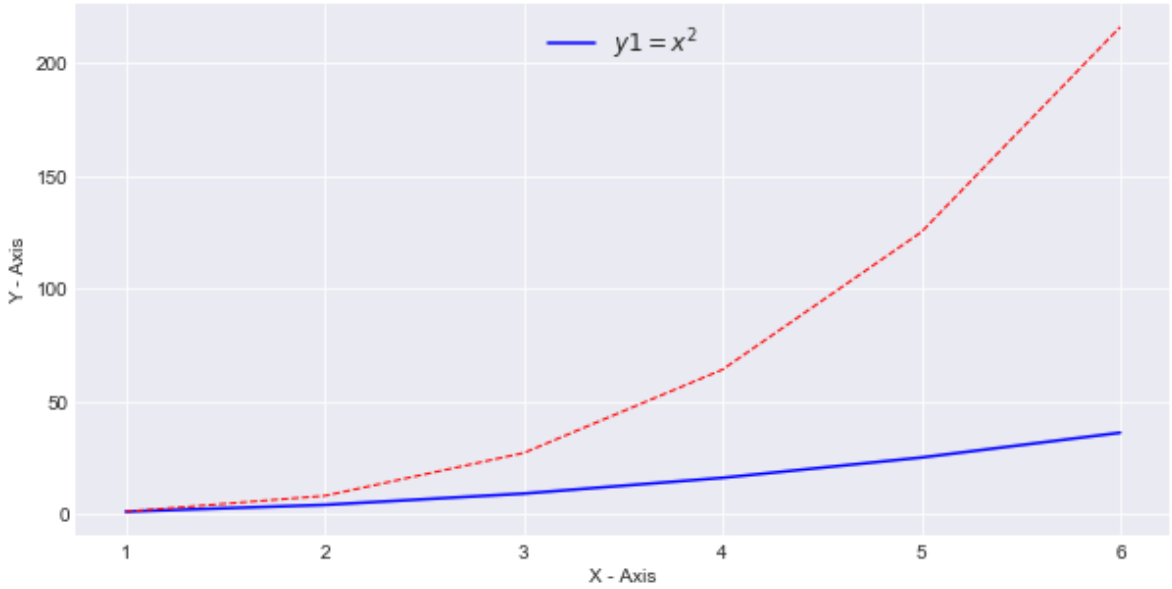
In [116]:

```
#Plotting multiple sets of data
x = np.linspace(0, 10, 2000)
plt.figure(figsize=(10,6))
plt.plot(x,np.sin(x) , label = '$Sin(X)$')
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()
```



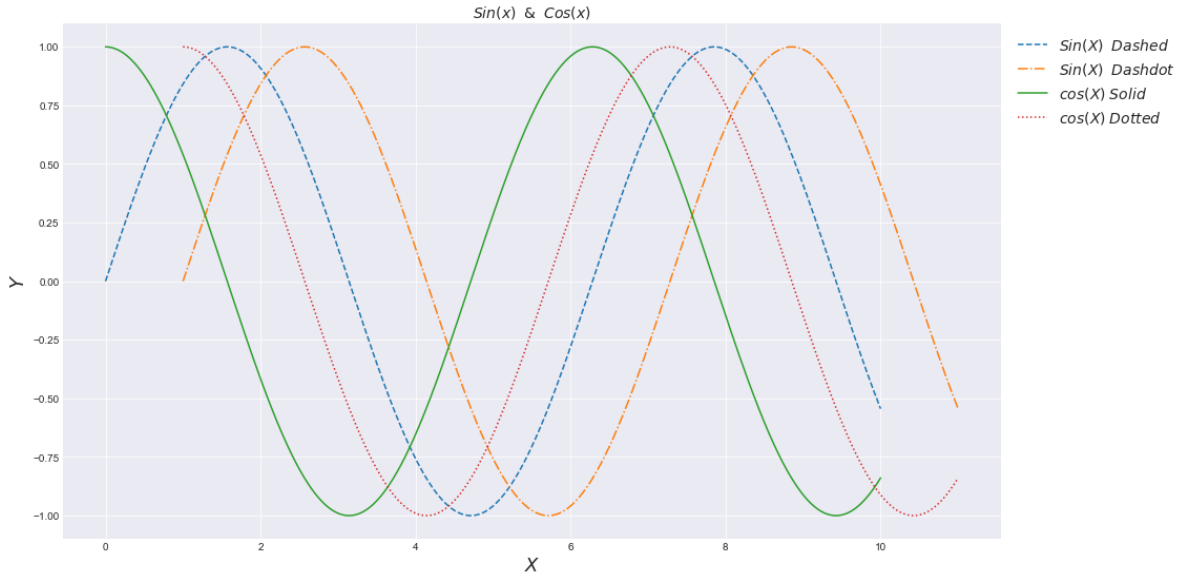
In [117]:

```
#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()
```



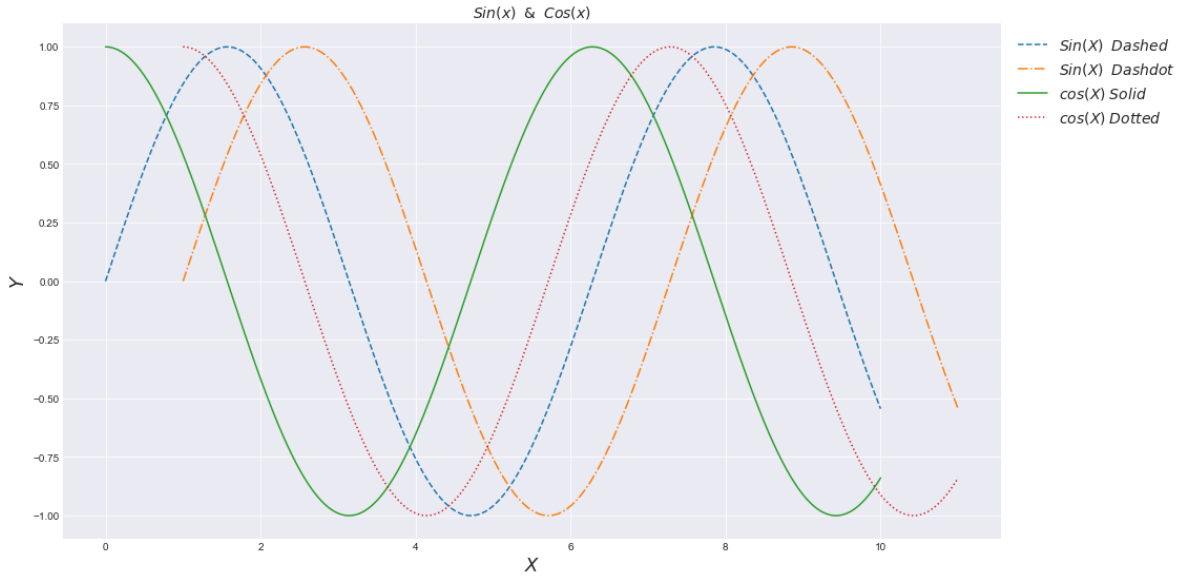
In [118]:

```
# 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 $' , linestyle='dotted')
plt.xlabel(r'$X$' , fontsize = 18)
plt.ylabel(r'$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 will b
plt.show()
```



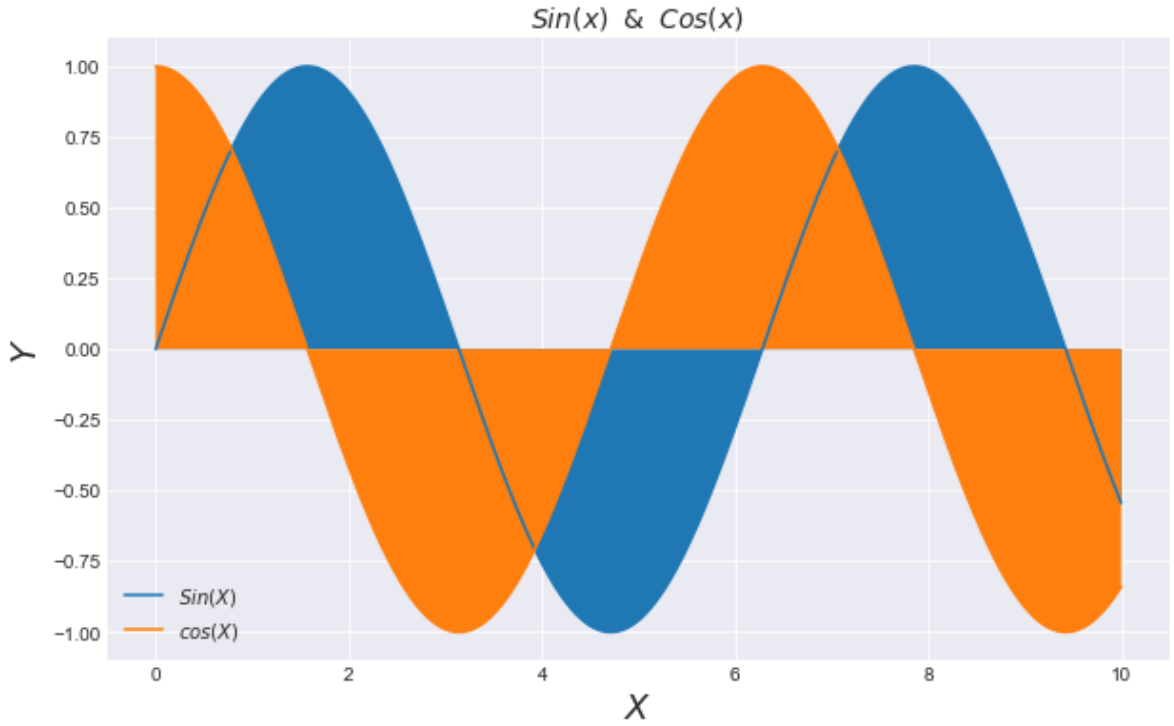
In [119]:

```
# 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 $' , linestyle=':')
plt.xlabel(r'$X$' , fontsize = 18)
plt.ylabel(r'$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 will b
plt.show()
```



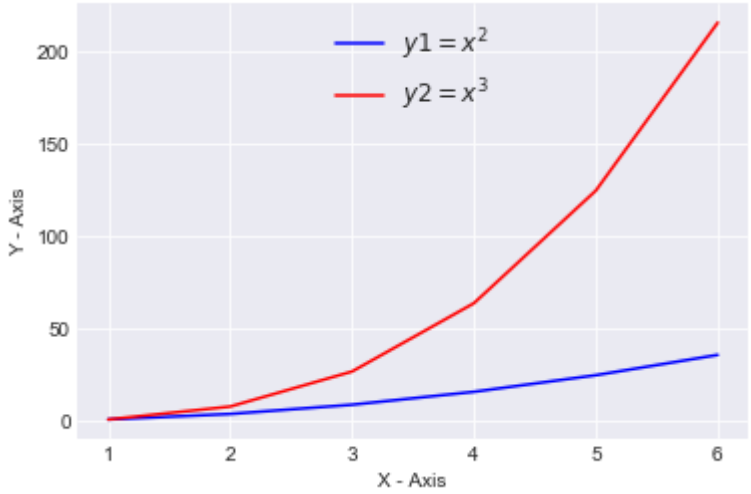
In [93]:

```
# 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()
```



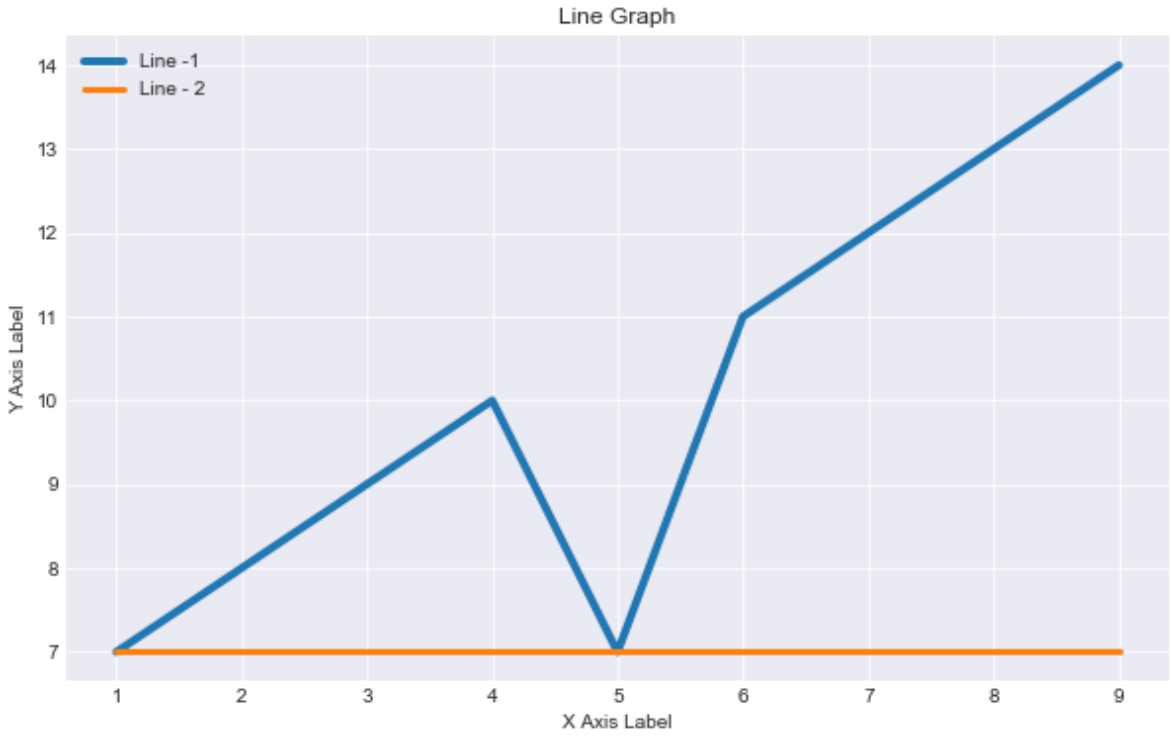
In [94]:

```
#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()
```



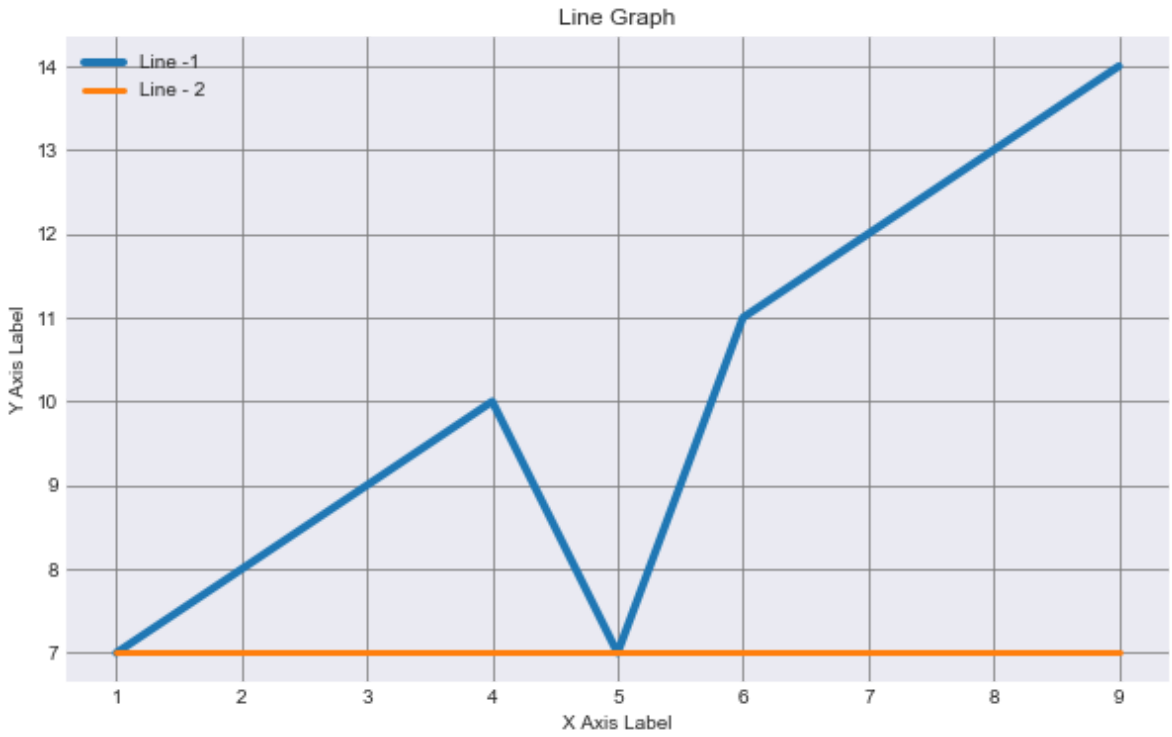
In [95]:

```
# 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()
```



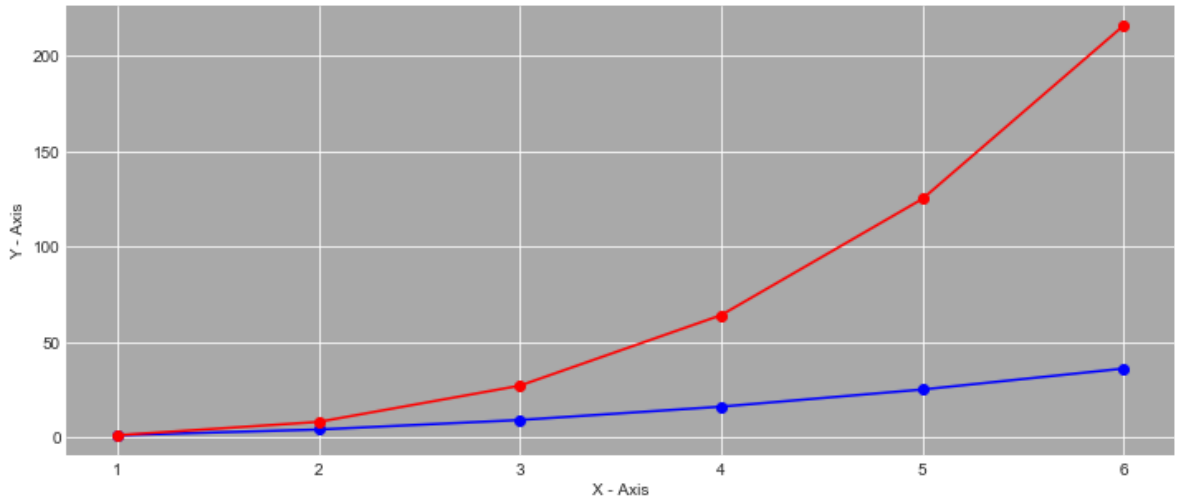
In [96]:

```
# 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 , 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()
```



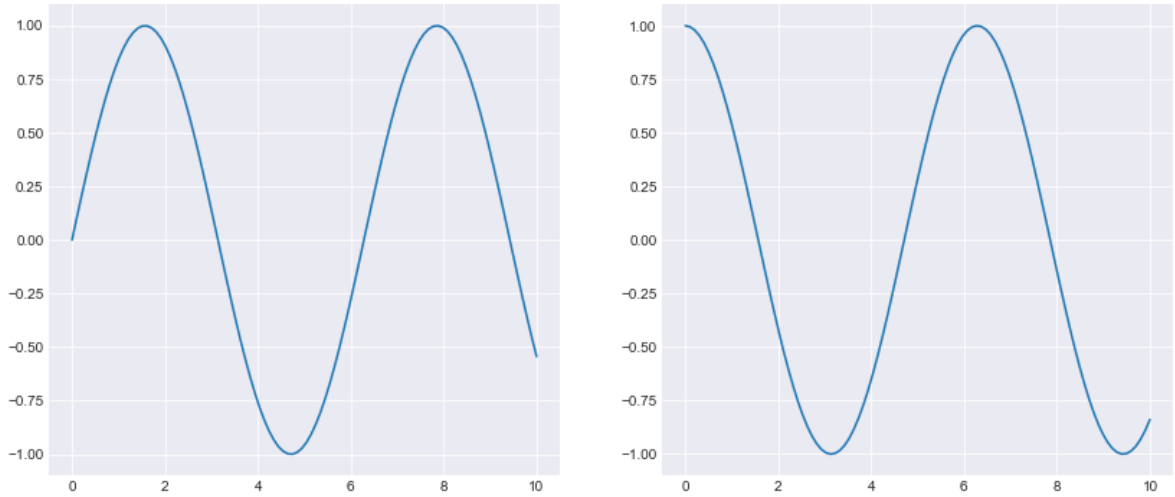
In [97]:

```
# 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()
```



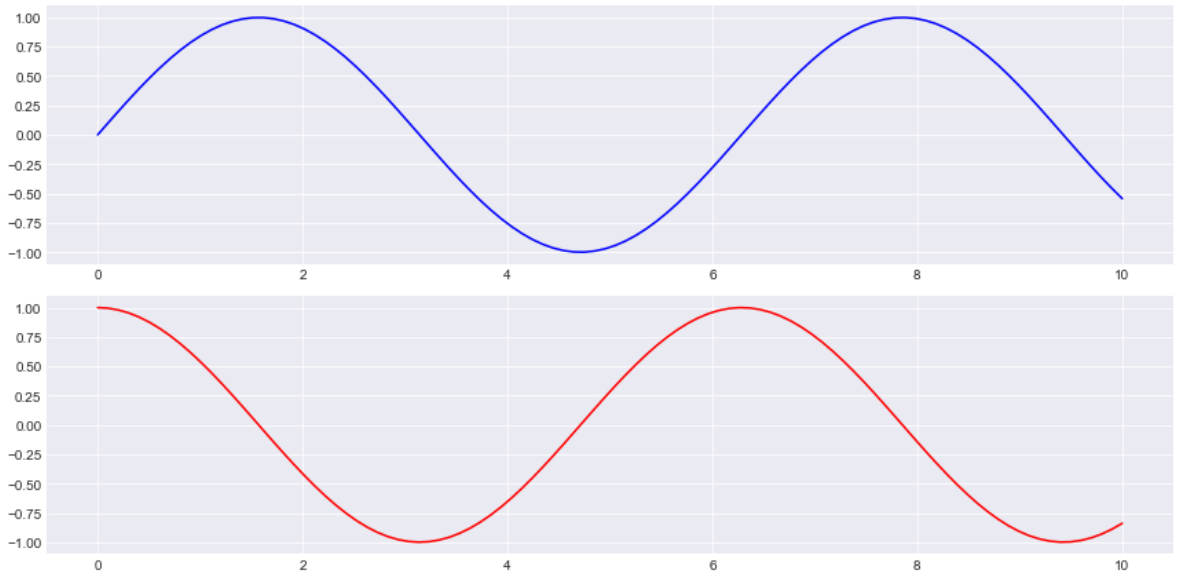
In [110]:

```
# 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()
```



In [112]:

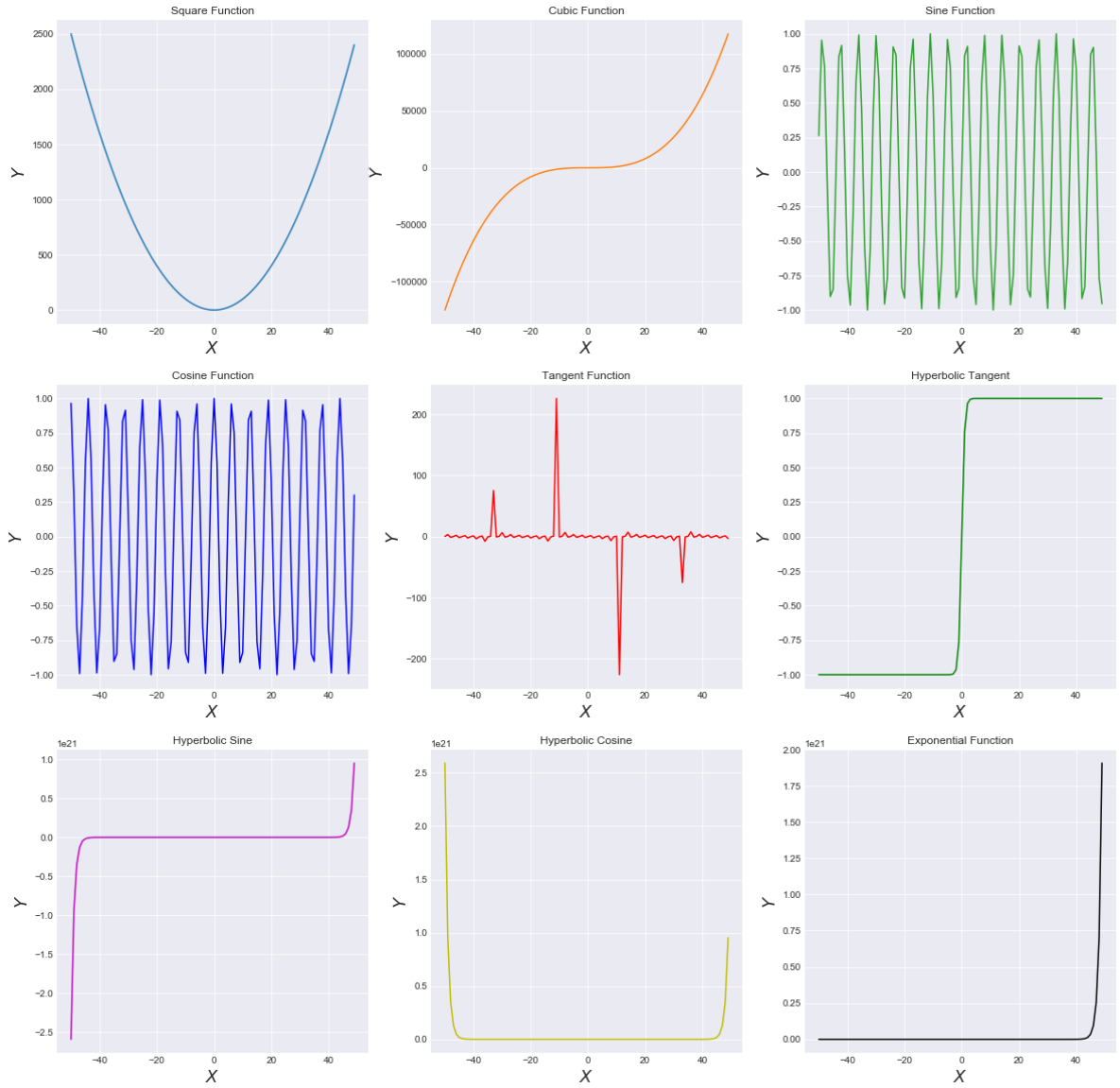
```
# Display multiple plots in one figure (2 row & 1 columns)

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.show()
```



In [113]:

```
# # 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 subplo
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_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-")
```

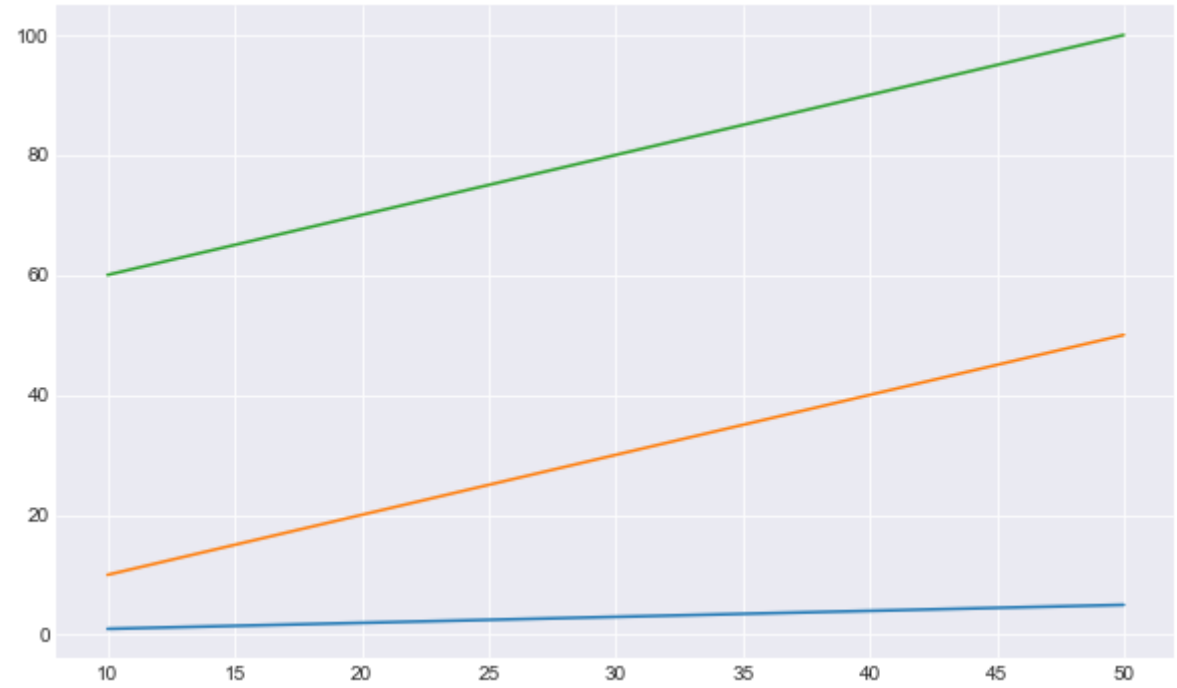
```
ax1[2,2].set_title("Exponential Function")
ax1[2,2].set_xlabel(r'$X$' , fontsize = 18)
ax1[2,2].set_ylabel(r'$Y$' , fontsize = 18)

plt.show()
```



In [114]:

```
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()
```

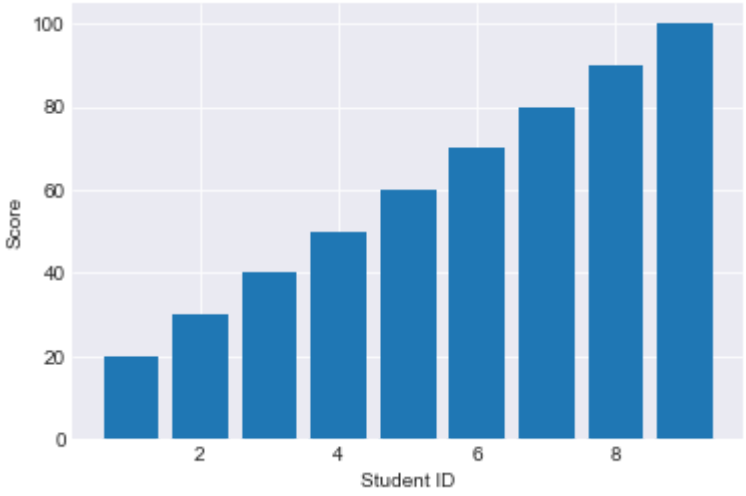
```
iteration Number :- 1
X1 Value :- [10, 20, 30, 40, 50]
Y value (i) :- [1, 2, 3, 4, 5]
iteration Number :- 2
X1 Value :- [10, 20, 30, 40, 50]
Y value (i) :- [10, 20, 30, 40, 50]
iteration Number :- 3
X1 Value :- [10, 20, 30, 40, 50]
Y value (i) :- [60, 70, 80, 90, 100]
```




# Bar Graphs

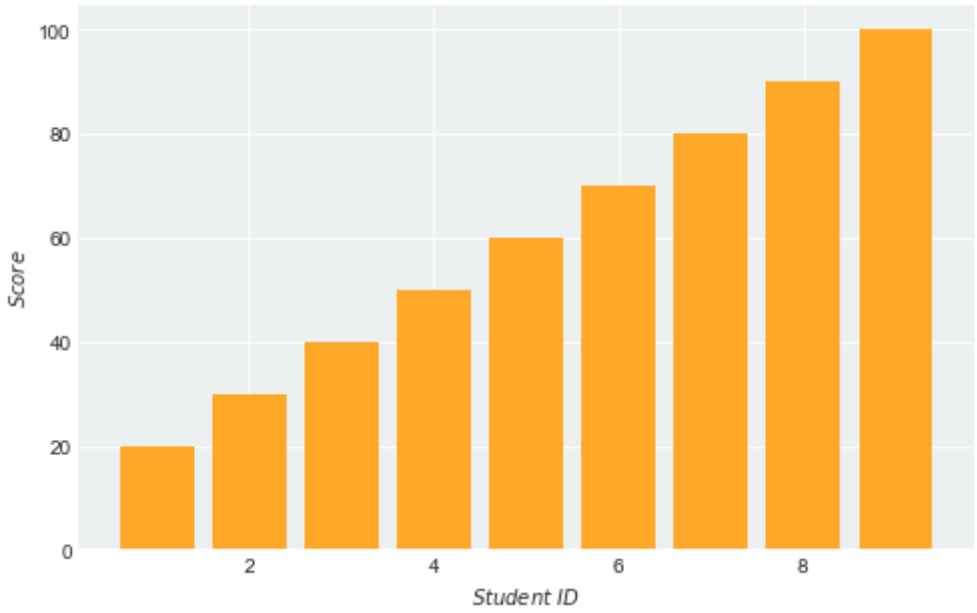
In [288]:

```
id1 = np.arange(1,10)
score = np.arange(20,110,10)
plt.bar(id1,score)
plt.xlabel('Student ID')
plt.ylabel('Score')
plt.show()
```



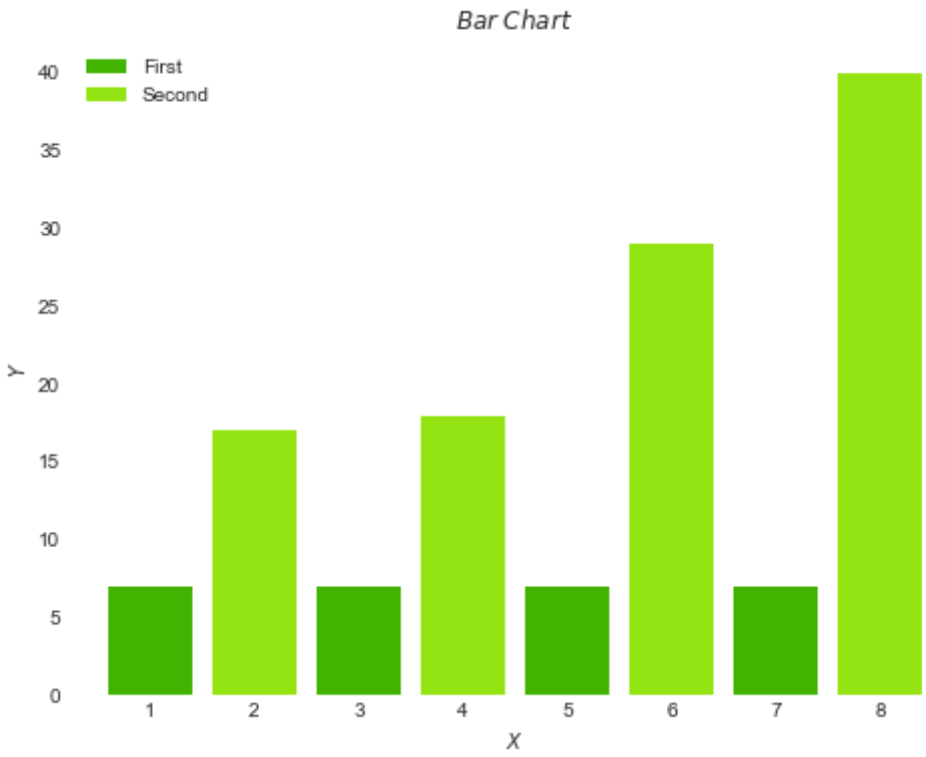
In [289]:

```
id1 = np.arange(1,10)
score = np.arange(20,110,10)
plt.figure(figsize=(8,5)) # Setting the figure size
ax = plt.axes()
ax.set_facecolor("#ECF0F1") # Setting the background color by specifying the HEX Code
plt.bar(id1,score,color = '#FFA726')
plt.xlabel(r'$Student $ $ ID$')
plt.ylabel(r'$Score$')
plt.show()
```



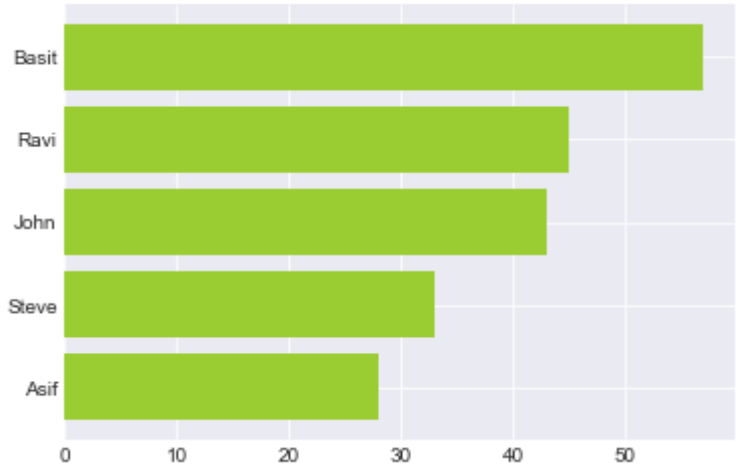
In [290]:

```
#Plotting multiple sets of data
x1= [1,3,5,7]
x2=[2,4,6,8]
y1 = [7,7,7,7]
y2= [17,18,29,40]
plt.figure(figsize=(8,6))
ax = plt.axes()
ax.set_facecolor("white")
plt.bar(x1,y1,label = "First",color = '#42B300') # First set of data
plt.bar(x2,y2,label = "Second",color = '#94E413') # Second set of data
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.title ('$Bar $ $ Chart$')
plt.legend()
plt.show()
```



In [291]:

```
# Horizontal Bar Chart
Age = [28,33,43,45,57]
Name = ["Asif", "Steve", 'John', "Ravi", "Basit"]
plt.barh(Name,Age, color ="yellowgreen")
plt.show()
```

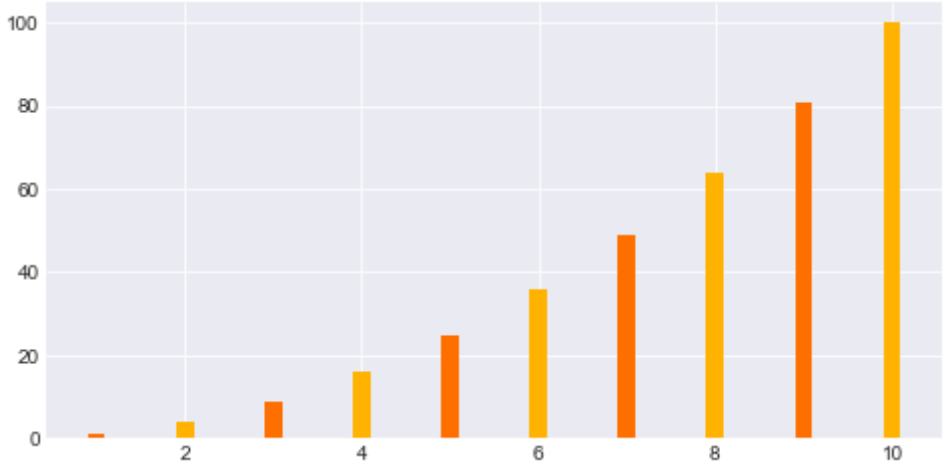



In [293]:

```
# Changing the width of Bars
num1 = np.array([1,3,5,7,9])
num2 = np.array([2,4,6,8,10])
plt.figure(figsize=(8,4))
plt.bar(num1, num1**2, width=0.2 , color = '#FF6F00')
plt.bar(num2, num2**2, width=0.2 , color = '#FFB300')
plt.plot()
```

Out[293]:

```
[]
```

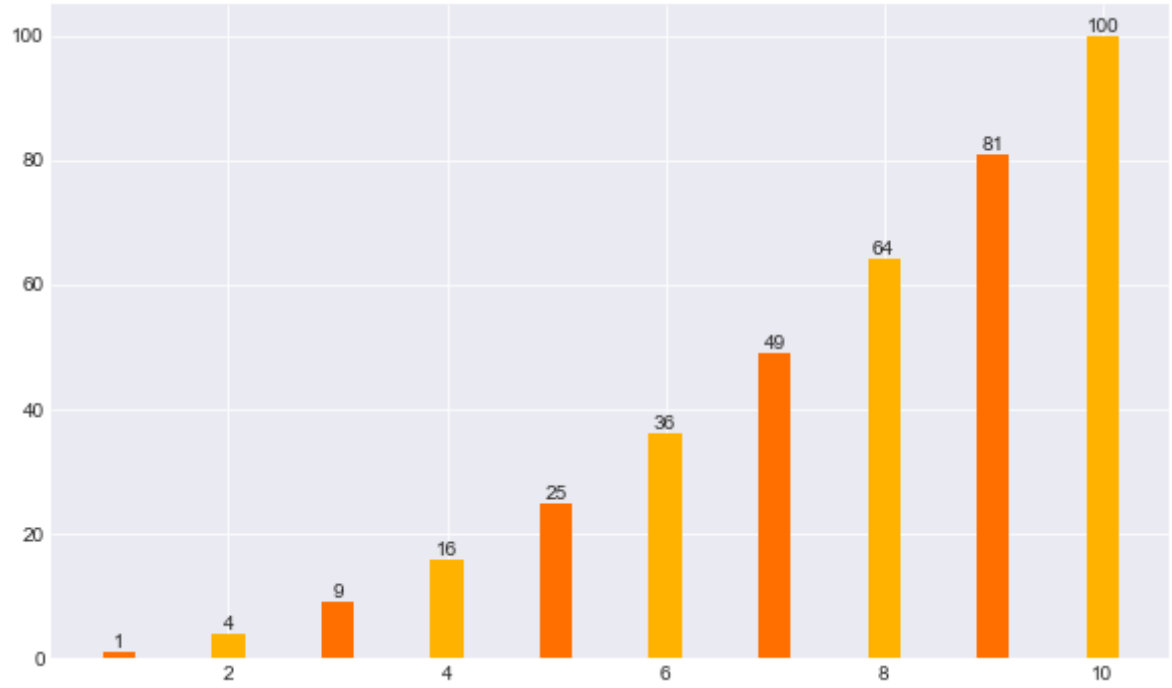


In [294]:

```
# Displaying values at the top of vertical bars
num1 = np.array([1,3,5,7,9])
num2 = np.array([2,4,6,8,10])
plt.figure(figsize=(10,6))
plt.bar(num1, num1**2, width=0.3 , color = '#FF6F00')
plt.bar(num2, num2**2, width=0.3 , color = '#FFB300')
for x,y in zip(num1,num1**2):
    plt.text(x, y+0.05, '%d' % y, ha='center' , va= 'bottom')
for x,y in zip(num2,num2**2):
    plt.text(x, y+0.05, '%d' % y, ha='center' , va= 'bottom')
plt.plot()
```

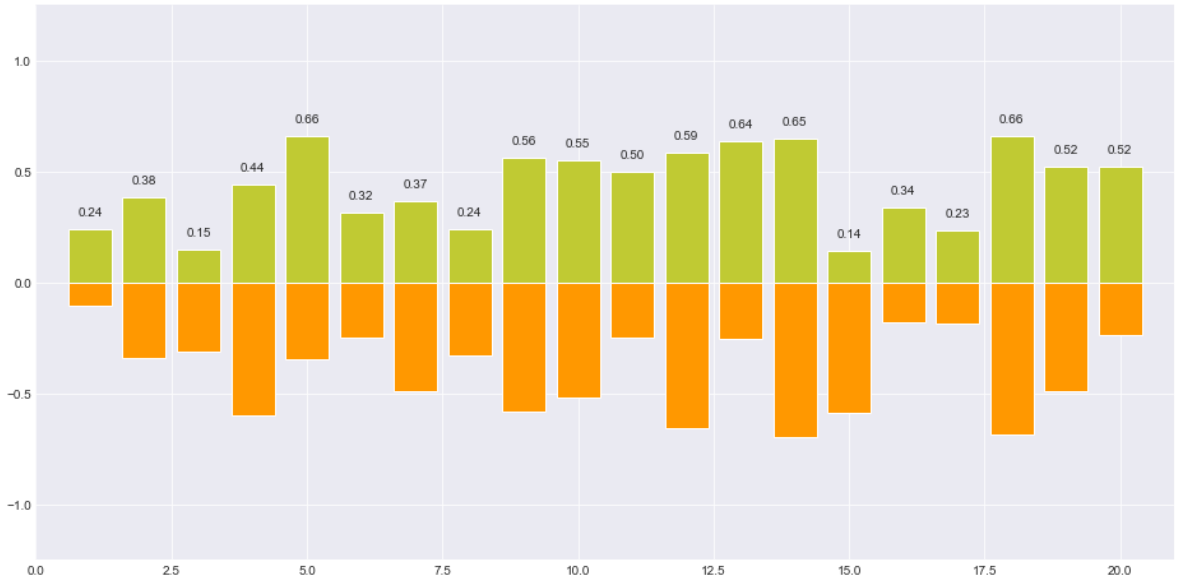
Out[294]:

```
[]
```



In [295]:

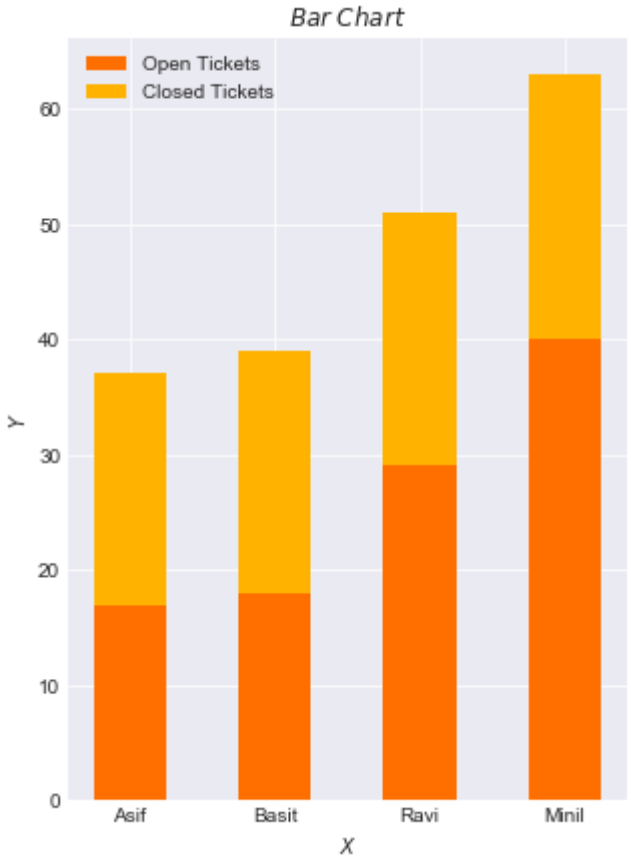
```
x = np.arange(1,21)
plt.figure(figsize=(16,8))
y1 = np.random.uniform(0.1,0.7,20)
y2 = np.random.uniform(0.1,0.7,20)


plt.bar(x, +y1, facecolor='#C0CA33', edgecolor='white') #specify edgecolor by name
plt.bar(x, -y2, facecolor='#FF9800', edgecolor='white')

for x,y in zip(x,y1):
    plt.text(x, y+0.05, '%.2f' % y, ha='center' , va= 'bottom', fontsize = 10)

plt.xlim(0,21)
plt.ylim(-1.25,+1.25)
plt.show()
```




## Stacked Vertical Bar

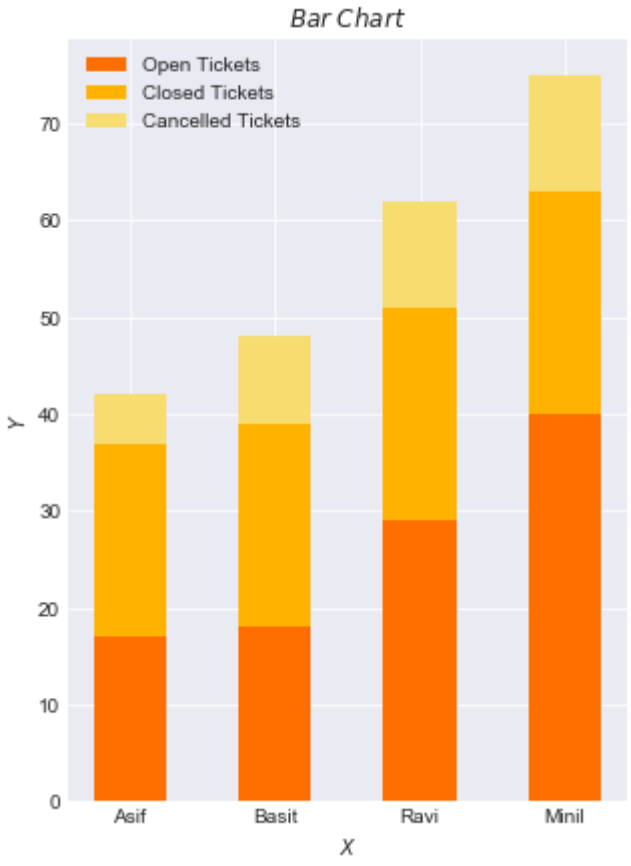
In [296]:

```
plt.style.use('seaborn-darkgrid')
x1= ['Asif','Basit','Ravi','Minil']
y1= [17,18,29,40]
y2 = [20,21,22,23]
plt.figure(figsize=(5,7))
plt.bar(x1,y1,label = "Open Tickets",width = 0.5,color = '#FF6F00')
plt.bar(x1,y2,label = "Closed Tickets",width = 0.5 ,bottom = y1 , color = '#FFB300')
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.title ('$Bar $ $ Chart$')
plt.legend()
plt.show()
```



In [297]:

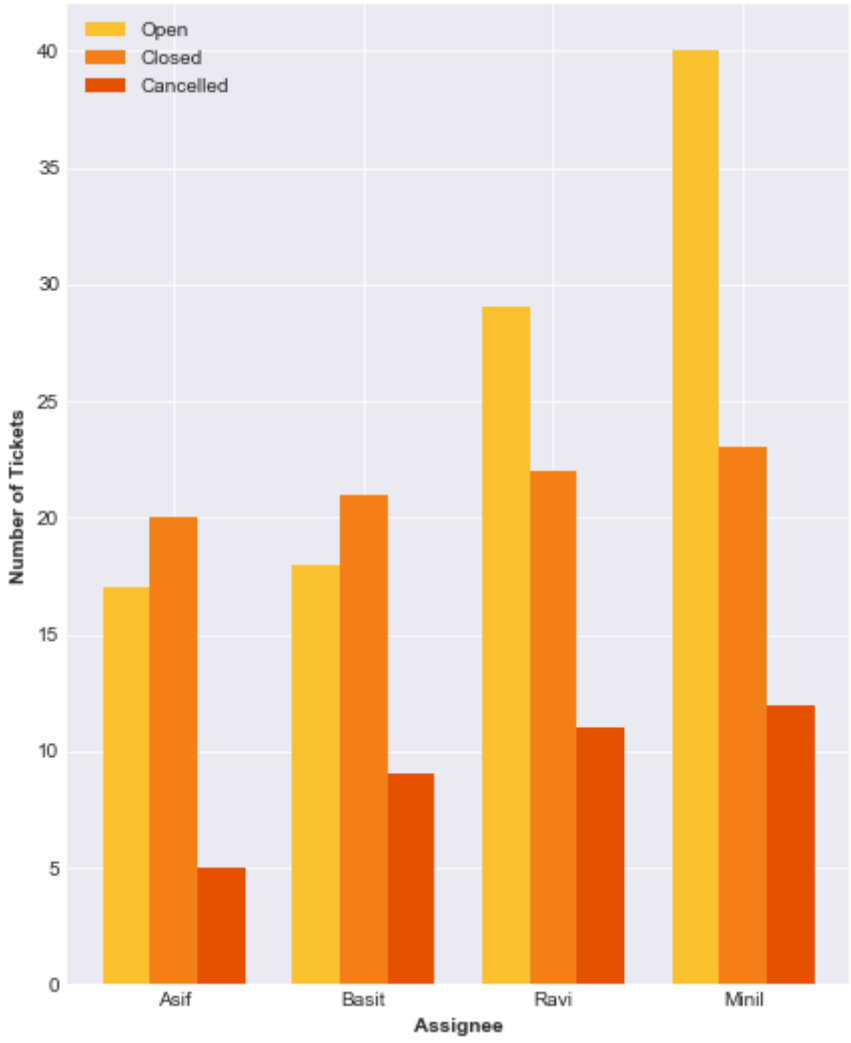
```
plt.style.use('seaborn-darkgrid')
x1= ['Asif','Basit','Ravi','Minil']
y1= np.array([17,18,29,40])
y2 =np.array([20,21,22,23])
y3 =np.array([5,9,11,12])
plt.figure(figsize=(5,7))
plt.bar(x1,y1,label = "Open Tickets",width = 0.5,color = '#FF6F00')
plt.bar(x1,y2,label = "Closed Tickets",width = 0.5 ,bottom = y1 , color = '#FFB300')
plt.bar(x1,y3,label = "Cancelled Tickets",width = 0.5 ,bottom = y1+y2 , color = '#F7DC6F')
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.title ('$Bar $ $ Chart$')
plt.legend()
plt.show()
```



## Grouped Bar Chart

In [298]:

```
# Grouped Bar Chart

plt.figure(figsize=(7,9))

# set width of bar
barWidth = 0.25

# set height of bar
y1= np.array([17,18,29,40])
y2 =np.array([20,21,22,23])
y3 =np.array([5,9,11,12])

# Set position of bar on X axis
pos1 = np.arange(len(y1))
pos2 = [x + barWidth for x in pos1]
pos3 = [x + barWidth for x in pos2]

# Make the plot
plt.bar(pos1, y1, color='#FBC02D', width=barWidth, label='Open')
plt.bar(pos2, y2, color='#F57F17', width=barWidth, label='Closed')
plt.bar(pos3, y3, color='#E65100', width=barWidth, label='Cancelled')

# Add xticks on the middle of the group bars
plt.xlabel('Assignee', fontweight='bold')
plt.ylabel('Number of Tickets', fontweight='bold')
plt.xticks([i + barWidth for i in range(len(y1))], ['Asif', 'Basit', 'Ravi', 'Minil'])

# Create legend & Show graphic
plt.legend()
plt.show()
np.arange(len(y1))
```



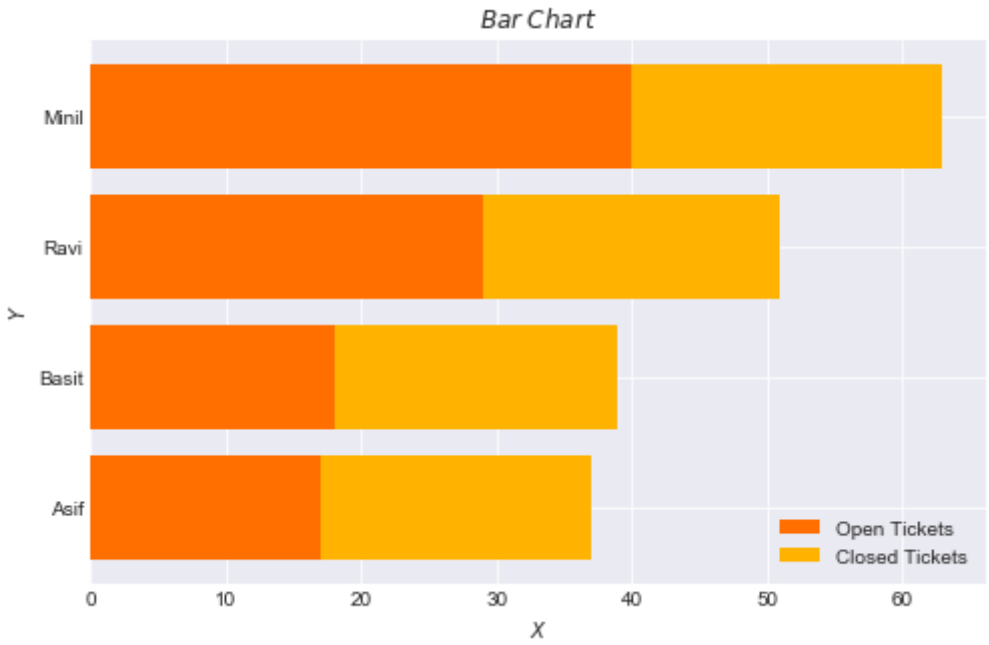

Out[298]:

```
array([0, 1, 2, 3])
```

## Stacked Vertical Bar

In [299]:

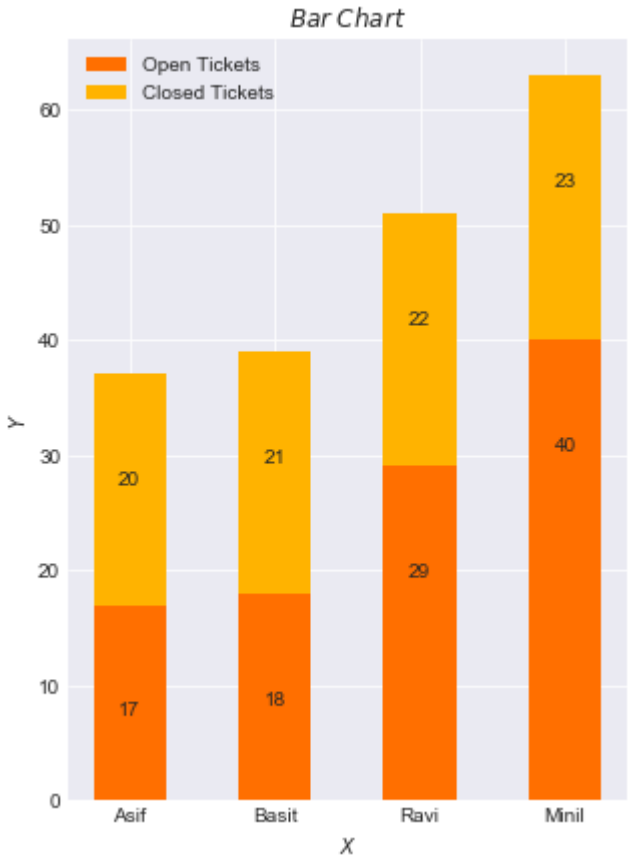
```
plt.style.use('seaborn-darkgrid')
x1= ['Asif','Basit','Ravi','Minil']
y1= [17,18,29,40]
y2 = [20,21,22,23]
plt.figure(figsize=(8,5))
plt.barh(x1,y1,label = "Open Tickets",color = '#FF6F00')
plt.barh(x1,y2,label = "Closed Tickets", left = y1 , color = '#FFB300')
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.title ('$Bar $ $ Chart$')
plt.legend()
plt.show()
```




## Displaying values in Bar Charts

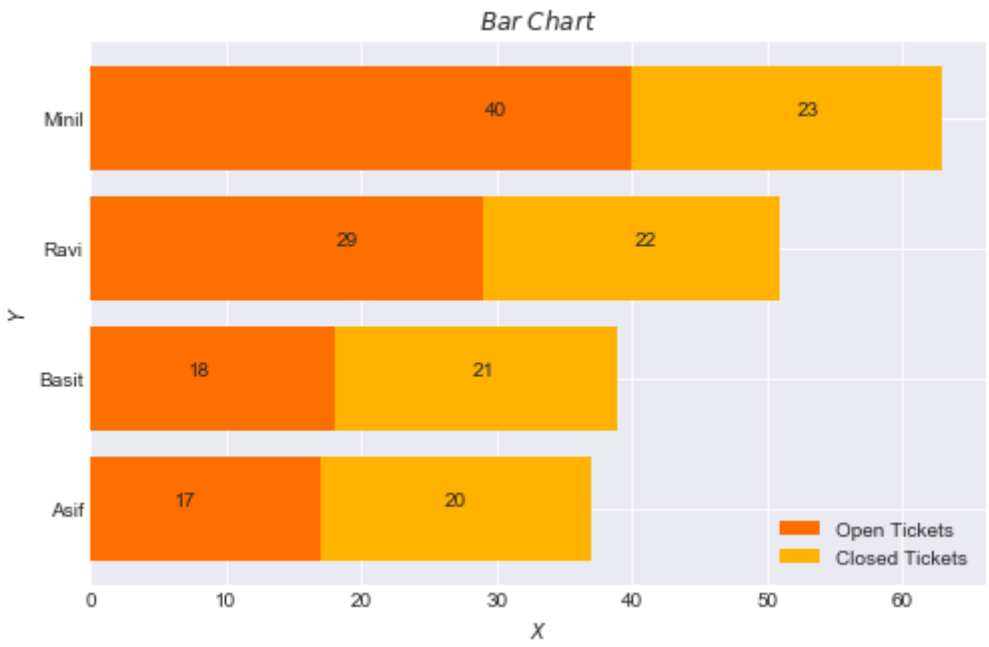
In [300]:

```
# Displaying values in the stacked vertical bars using plt.text()
plt.style.use('seaborn-darkgrid')
x1= ['Asif','Basit','Ravi','Minil']
y1= [17,18,29,40]
y2 = [20,21,22,23]
plt.figure(figsize=(5,7))
plt.bar(x1,y1,label = "Open Tickets",width = 0.5,color = '#FF6F00')
plt.bar(x1,y2,label = "Closed Tickets",width = 0.5 ,bottom = y1 , color = '#FFB300')
plt.xlabel('$X$')
plt.ylabel('$Y$')
plt.title ('$Bar $ $ Chart$')
for x,y in zip(x1,y1):
    plt.text(x, y-10, '%d' % y, ha='center' , va= 'bottom')

for x,y,z in zip(x1,y2,y1):
    plt.text(x, y+z-10, '%d' % y, ha='center' , va= 'bottom')

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



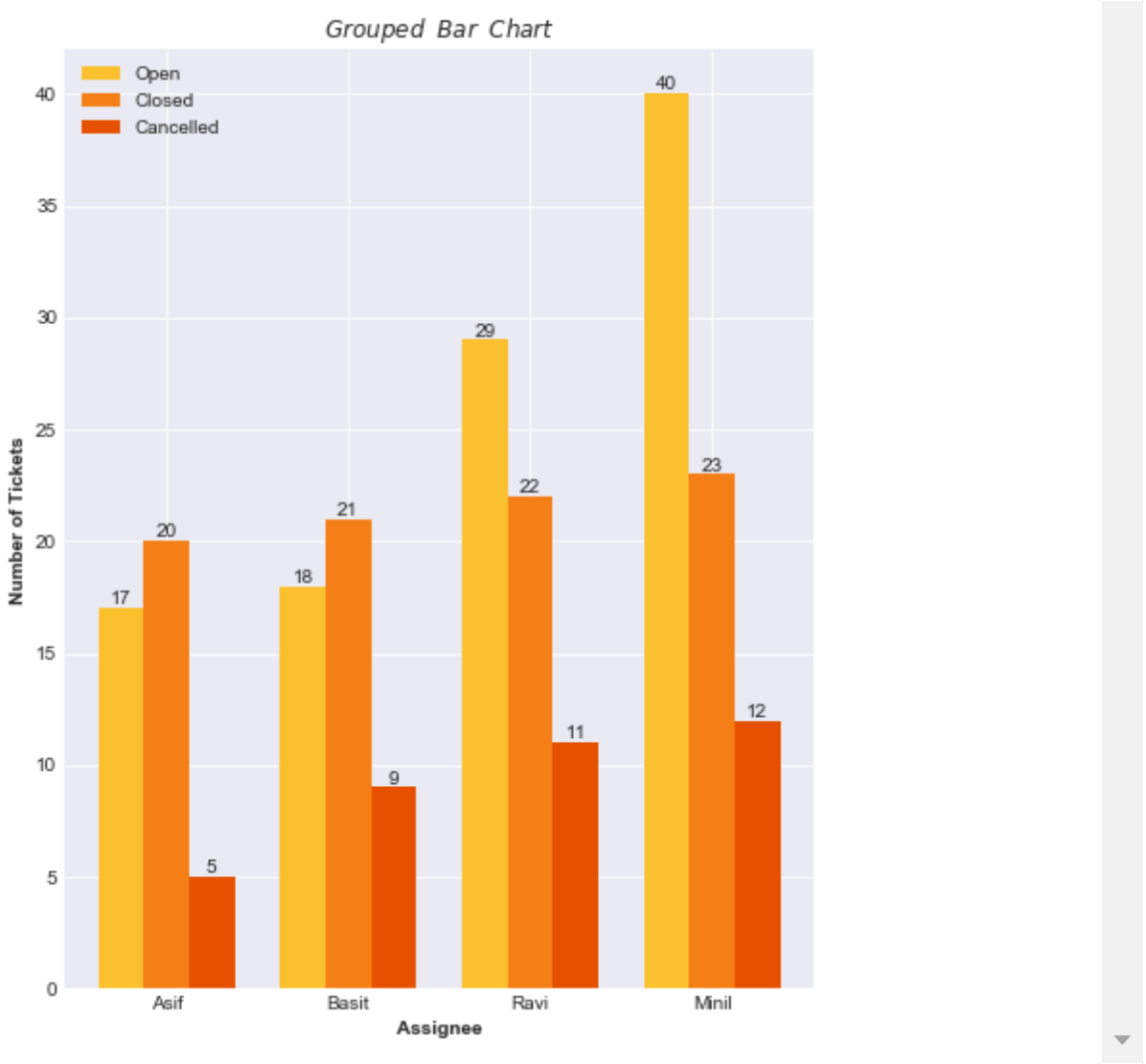
In [301]:

```
# Displaying values in the stacked horizontal bars using plt.text()
plt.style.use('seaborn-darkgrid')
x1= ['Asif','Basit','Ravi','Minil']
y1= [17,18,29,40]
y2 = [20,21,22,23]
plt.figure(figsize=(8,5))
plt.barh(x1,y1,label = "Open Tickets",color = '#FF6F00')
plt.barh(x1,y2,label = "Closed Tickets", left = y1 , color = '#FFB300')
plt.xlabel('$X$')
plt.ylabel('$Y$')

for x,y in zip(x1,y1):
    plt.text(y-10, x, '%d' % y, ha='center' , va= 'bottom')

for x,y,z in zip(x1,y2,y1):
    plt.text(y+z-10, x, '%d' % y, ha='center' , va= 'bottom')

plt.title ('$Bar $ $ Chart$')
plt.legend()
plt.show()
```



In [302]:

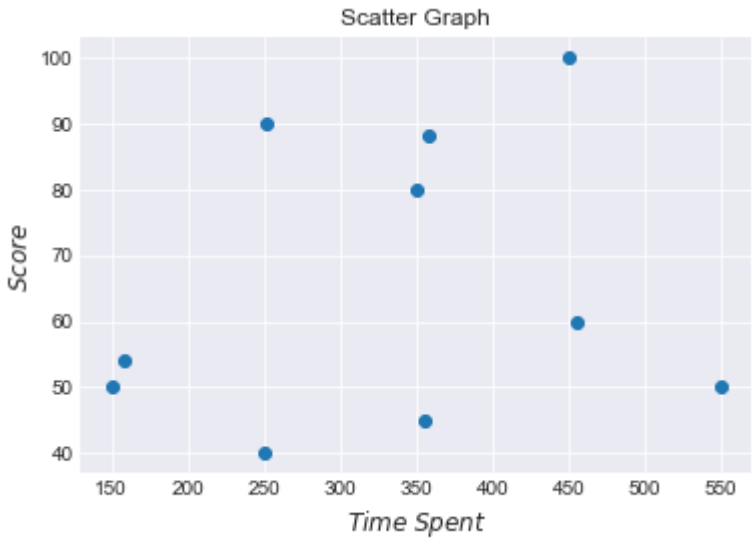
```
# Displaying values at the top of the Grouped Bar Chart using plt.text()
plt.figure(figsize=(7,9))

# set width of bar
barWidth = 0.25

# set height of bar
y1= np.array([17,18,29,40])
y2 =np.array([20,21,22,23])
y3 =np.array([5,9,11,12])

# Set position of bar on X axis
pos1 = np.arange(len(y1))
pos2 = [x + barWidth for x in pos1]
pos3 = [x + barWidth for x in pos2]

# Make the plot
plt.bar(pos1, y1, color='#FBC02D', width=barWidth, label='Open')
plt.bar(pos2, y2, color='#F57F17', width=barWidth, label='Closed')
plt.bar(pos3, y3, color='#E65100', width=barWidth, label='Cancelled')

# Add xticks on the middle of the group bars
plt.xlabel('Assignee', fontweight='bold')
plt.ylabel('Number of Tickets', fontweight='bold')
plt.xticks([i + barWidth for i in range(len(y1))], ['Asif', 'Basit', 'Ravi', 'Minil'])

for x,y in zip(pos1,y1):
    plt.text(x, y, '%d' % y, ha='center' , va= 'bottom')

for x,y in zip(pos2,y2):
    plt.text(x, y, '%d' % y, ha='center' , va= 'bottom')

for x,y in zip(pos3,y3):
    plt.text(x, y, '%d' % y, ha='center' , va= 'bottom')

plt.title ('$Grouped $  $ Bar $  $ Chart$')

# Create legend & Show graphic
plt.legend()
plt.show()
```

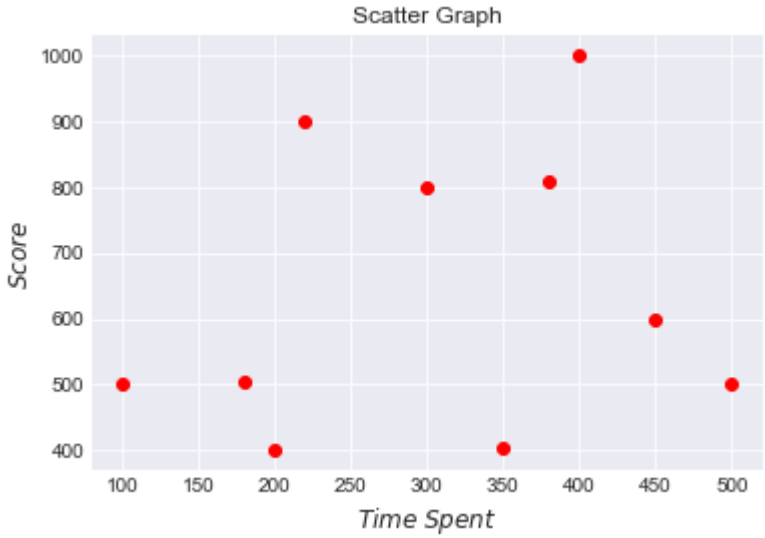


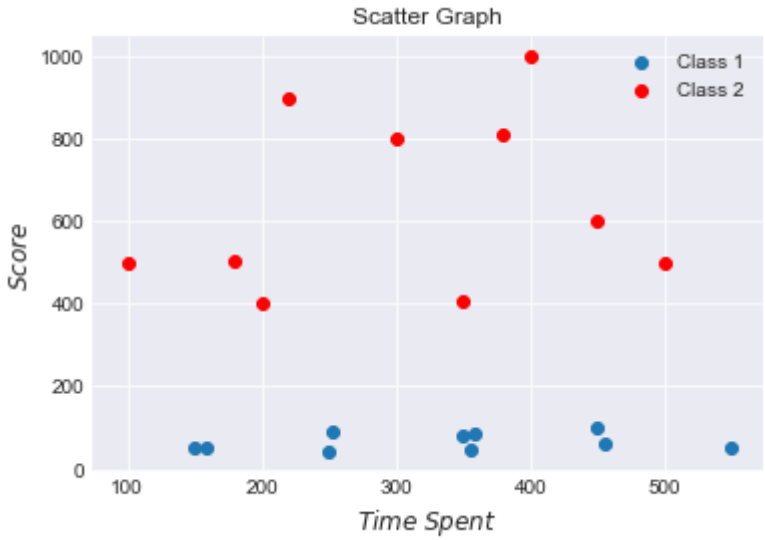

# Scatter Graphs

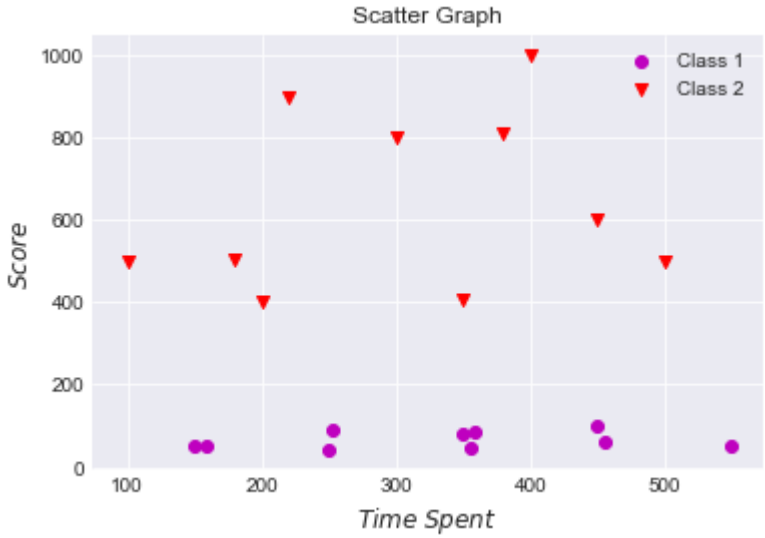
In [303]:

```
x1 = np.array([250,150,350,252,450,550,455,358,158,355])
y1 =np.array([40,50,80, 90, 100,50,60,88,54,45])


x2 = np.array([200,100,300,220,400,500,450,380,180,350])
y2 = np.array([400,500,800, 900, 1000,500,600,808,504,405])

#Graph - 1
plt.scatter(x1,y1)
plt.xlabel('$Time $ $ Spent$' , fontsize = 12)
plt.ylabel('$Score$' , fontsize = 12)
plt.title ('Scatter Graph')
plt.show()

#Graph - 2
plt.scatter(x2,y2 ,color = 'r')
plt.xlabel('$Time $ $ Spent$' , fontsize = 12)
plt.ylabel('$Score$' , fontsize = 12)
plt.title ('Scatter Graph')
plt.show()

#Graph - 3
plt.scatter(x1,y1 ,label = 'Class 1')
plt.scatter(x2,y2 ,label = 'Class 2',color ='r')
plt.xlabel('$Time $ $ Spent$' , fontsize = 12)
plt.ylabel('$Score$' , fontsize = 12)
plt.title ('Scatter Graph')
plt.legend()
plt.show()

#Graph - 4
plt.scatter(x1,y1 ,label = 'Class 1',marker='o' , color = 'm')
plt.scatter(x2,y2 ,label = 'Class 2',marker='v',color ='r')
plt.xlabel('$Time $ $ Spent$' , fontsize = 12)
plt.ylabel('$Score$' , fontsize = 12)
plt.title ('Scatter Graph')
plt.legend()
plt.show()
```



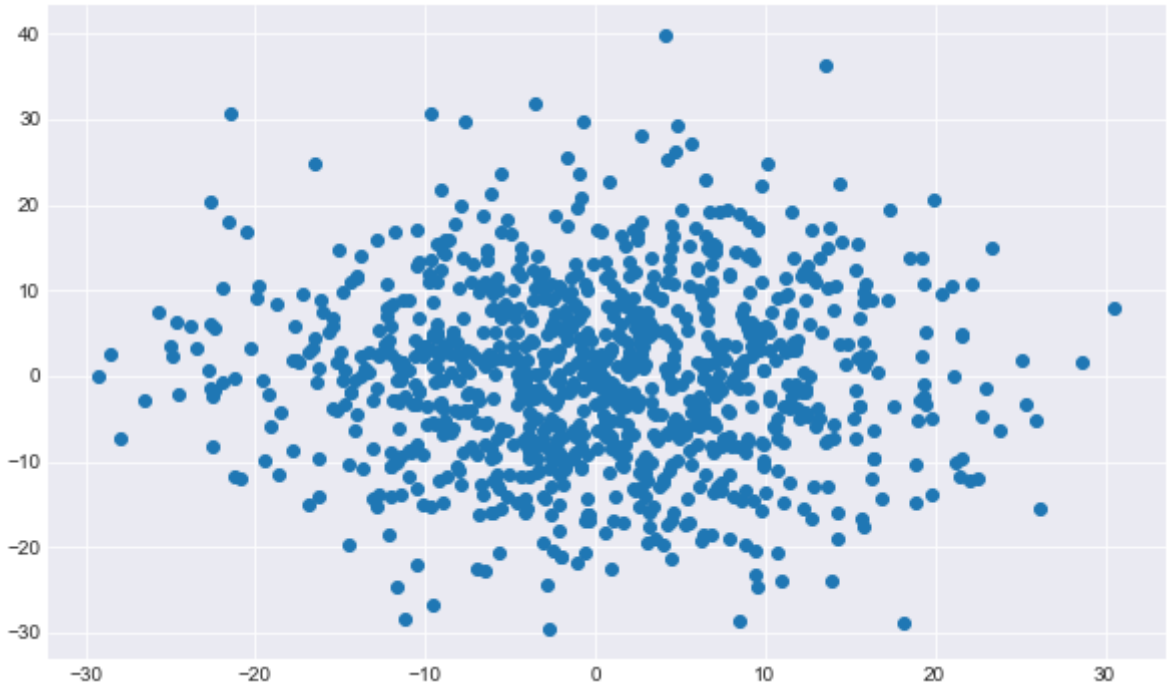

Scatter Graph
Score
Time Spent
1000
900
800
700
600
500
400
100
150
200
250
300
350
400
450
500


Scatter Graph
Class 1
Class 2
Score
Time Spent
1000
800
600
400
200
0
100
200
300
400
500


Scatter Graph
Class 1
Class 2
Score
Time Spent
1000
800
600
400
200
0
100
200
300
400
500

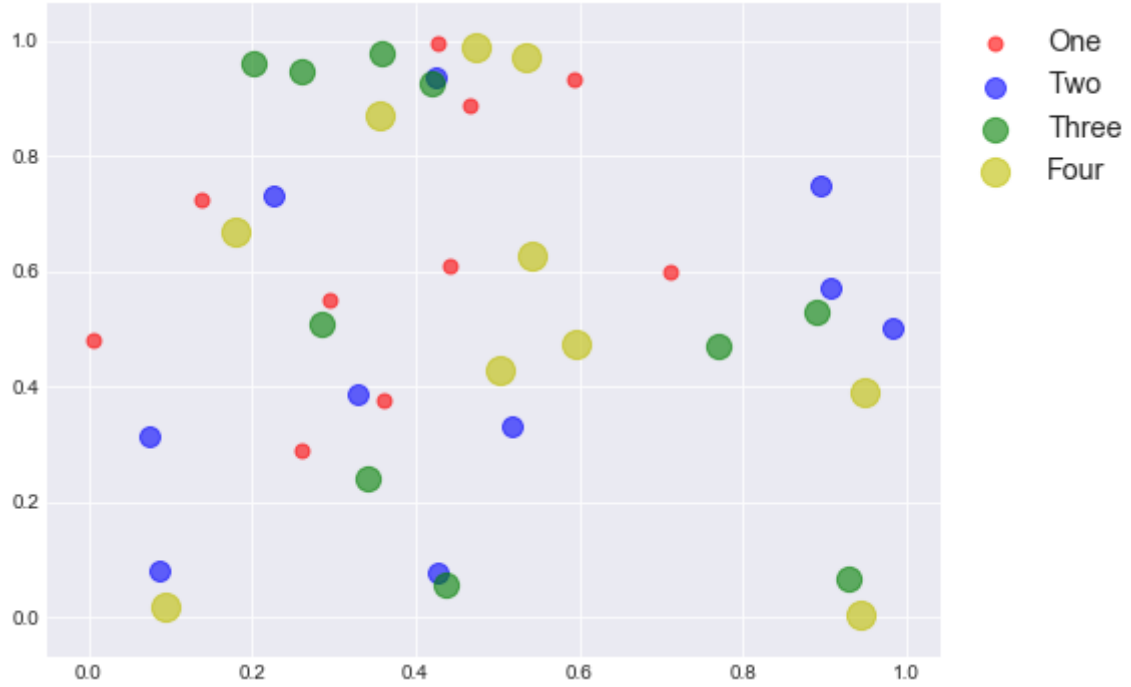
In [304]:

```
plt.figure(figsize=(10,6))
x = np.random.normal(0,10,1000)
y = np.random.normal(0,10,1000)
plt.scatter(x,y)
plt.show()
```



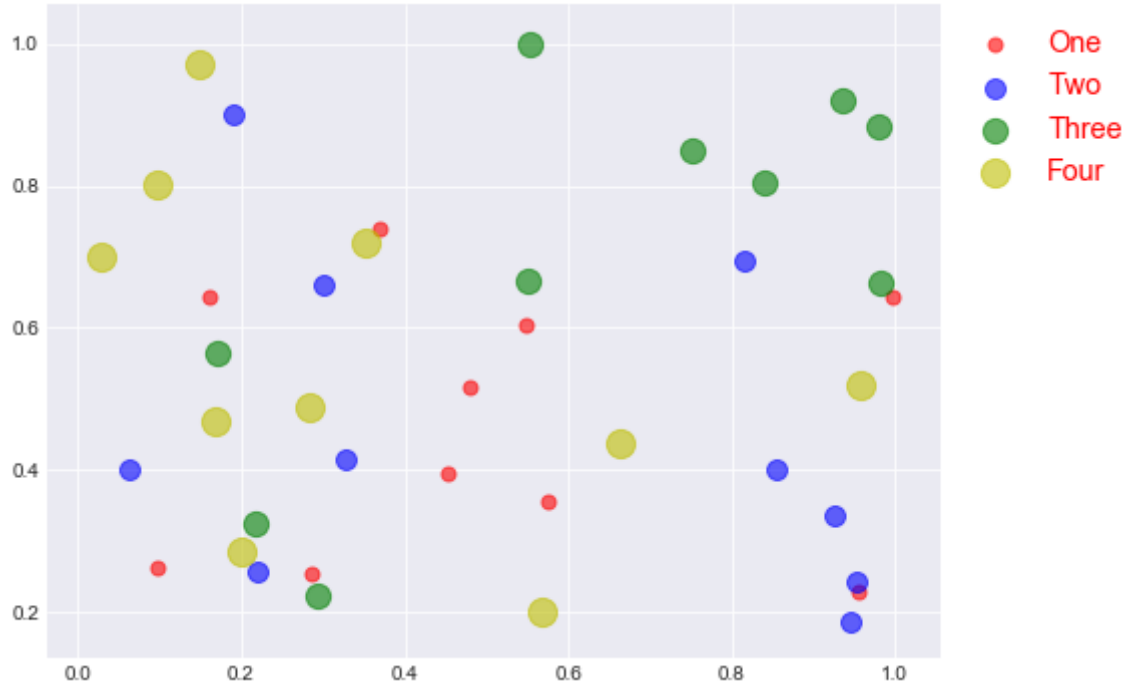
In [305]:

```
plt.figure(figsize=(8,6))
x = np.random.random(10)
y = np.random.random(10)
# "alpha" is used for softnening colors
plt.scatter(np.random.random(10),np.random.random(10),c='r', s=50 , alpha=0.6 , label = 'Or
plt.scatter(np.random.random(10),np.random.random(10),c='b', s=100 , alpha=0.6 , label = 'T
plt.scatter(np.random.random(10),np.random.random(10),c='g', s=150 , alpha=0.6 , label = 'T
plt.scatter(np.random.random(10),np.random.random(10),c='y', s=200 , alpha=0.6 , label = 'F
plt.legend(bbox_to_anchor=(1.0, 1.0) , shadow=True, fontsize='x-large')
plt.show()
```



In [306]:

```
# Changing label color
plt.figure(figsize=(8,6))
x = np.random.random(10)
y = np.random.random(10)
# "alpha" is used for softnening colors
plt.rcParams['text.color'] = 'red' # Label Color
plt.scatter(np.random.random(10),np.random.random(10),c='r', s=50 , alpha=0.6 , label = 'Or
plt.scatter(np.random.random(10),np.random.random(10),c='b', s=100 , alpha=0.6 , label = 'T
plt.scatter(np.random.random(10),np.random.random(10),c='g', s=150 , alpha=0.6 , label = 'T
plt.scatter(np.random.random(10),np.random.random(10),c='y', s=200 , alpha=0.6 , label = 'F
plt.legend(bbox_to_anchor=(1.0, 1.0) , shadow=True, fontsize='x-large')
plt.show()
```



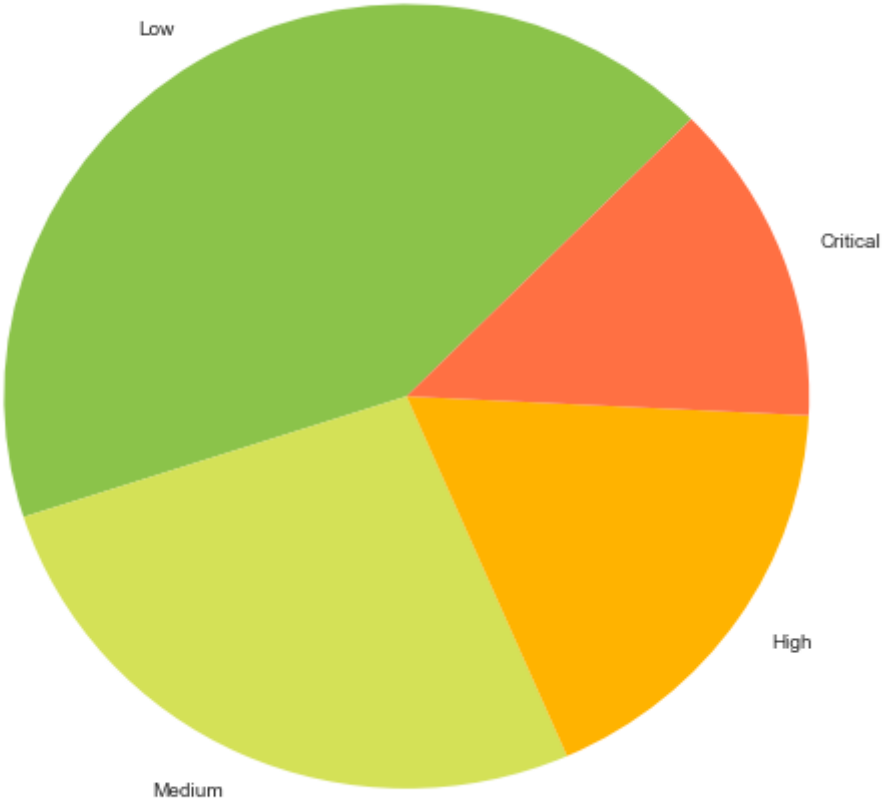

In [307]:

```
# Recover default matplotlib settings
mpl.rcParams.update(mpl.rcParamsDefault)
%matplotlib inline
plt.style.use('seaborn-darkgrid')
```

# Pie Charts

In [308]:

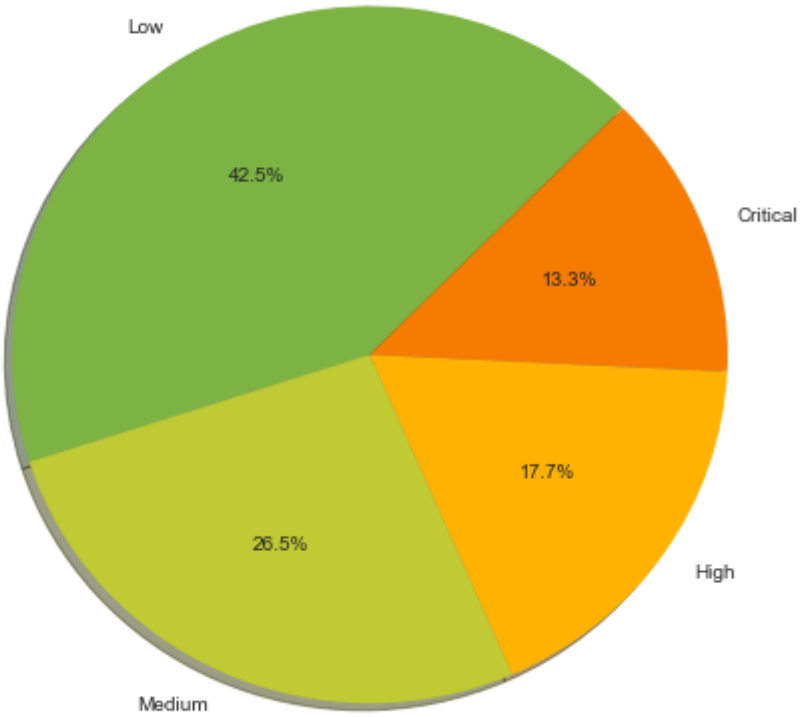
```
plt.figure(figsize=(9,9))
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']
plt.pie (area , labels= labels , colors= colors , startangle=45)
plt.show()
```




**Display percentage and actual value in Pie Chart**

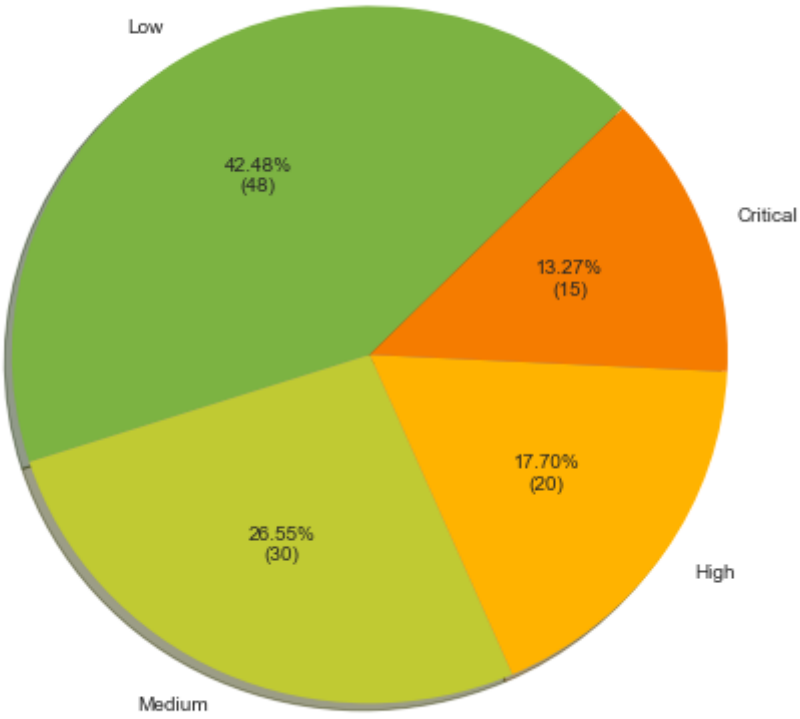
In [309]:

```
# Display percentage in Pie Chart using autopct='%1.1f%%'
plt.figure(figsize=(8,8))
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#7CB342','#C0CA33','#FFB300','#F57C00']
plt.pie (area , labels= labels , colors= colors , startangle=45 , shadow='true', autopct='%
plt.show()
```



In [310]:

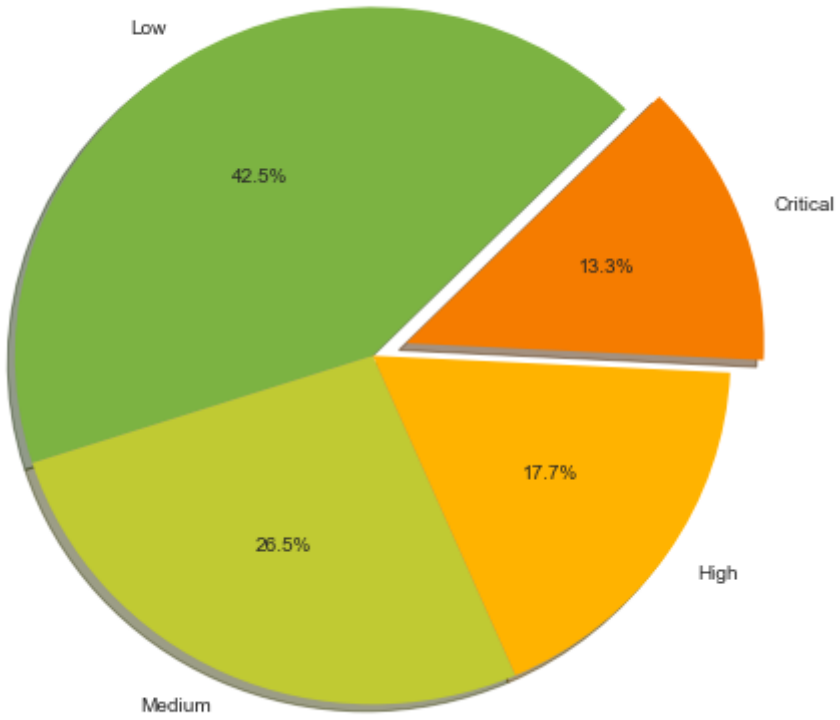
```
plt.figure(figsize=(8,8))
area = [48 , 30 , 20 , 15]
total = np.sum(area)
labels = ['Low' , 'Medium' , 'High' , 'Critical']

def val_per(x):
    return '{:.2f}%\n({:.0f})'.format(x, total*x/100)

colors = ['#7CB342','#C0CA33','#FFB300','#F57C00']
plt.pie (area , labels= labels , colors= colors , startangle=45 , shadow='true', autopct=va
plt.show()
```




**Explode Slice in Pie Chart**

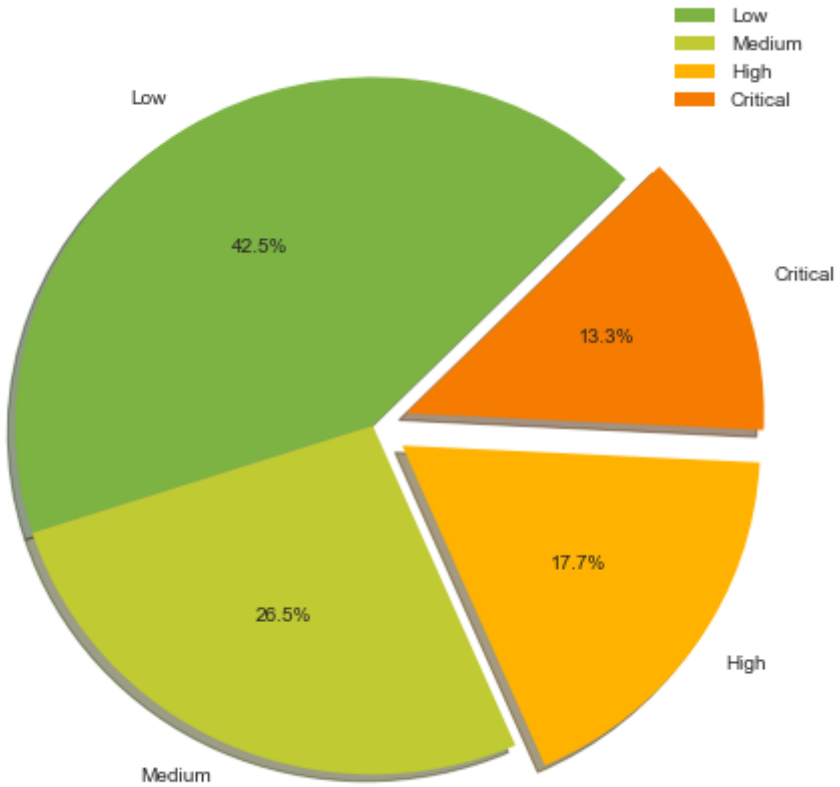
In [311]:

```
#Explode 4th Slice
plt.figure(figsize=(8,8))
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#7CB342','#C0CA33','#FFB300','#F57C00']
# explode = [0,0,0,0.1] will explode the fourth slice
plt.pie (area , labels= labels , colors= colors , startangle=45 , autopct='%1.1f%%' , shadc
plt.show()
```



In [312]:

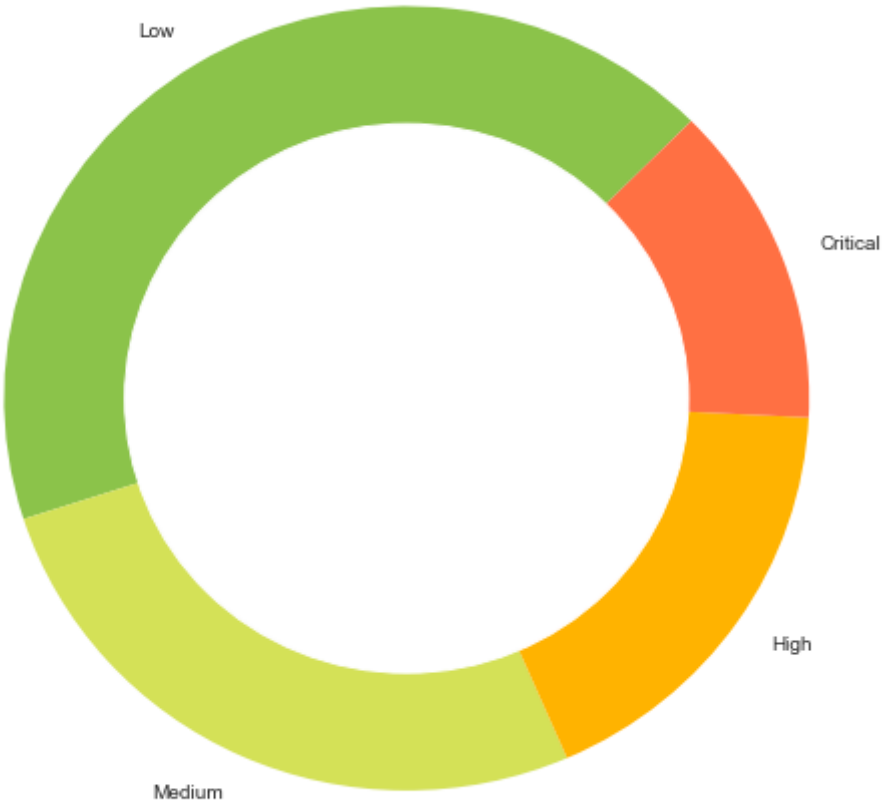
```
#Explode 3rd & 4th Slice
plt.figure(figsize=(8,8))
area = [48 , 30 , 20 , 15]
label = ['Low' , 'Medium' , 'High' , 'Critical']
color = ['#7CB342','#C0CA33','#FFB300','#F57C00']
# explode = [0,0,0.1,0.1] will explode the 3rd & 4th slice
plt.pie (area , labels= label , colors= color , startangle=45 ,autopct='%1.1f%%', shadow='t
plt.legend()
plt.show()
```



# Donut plot

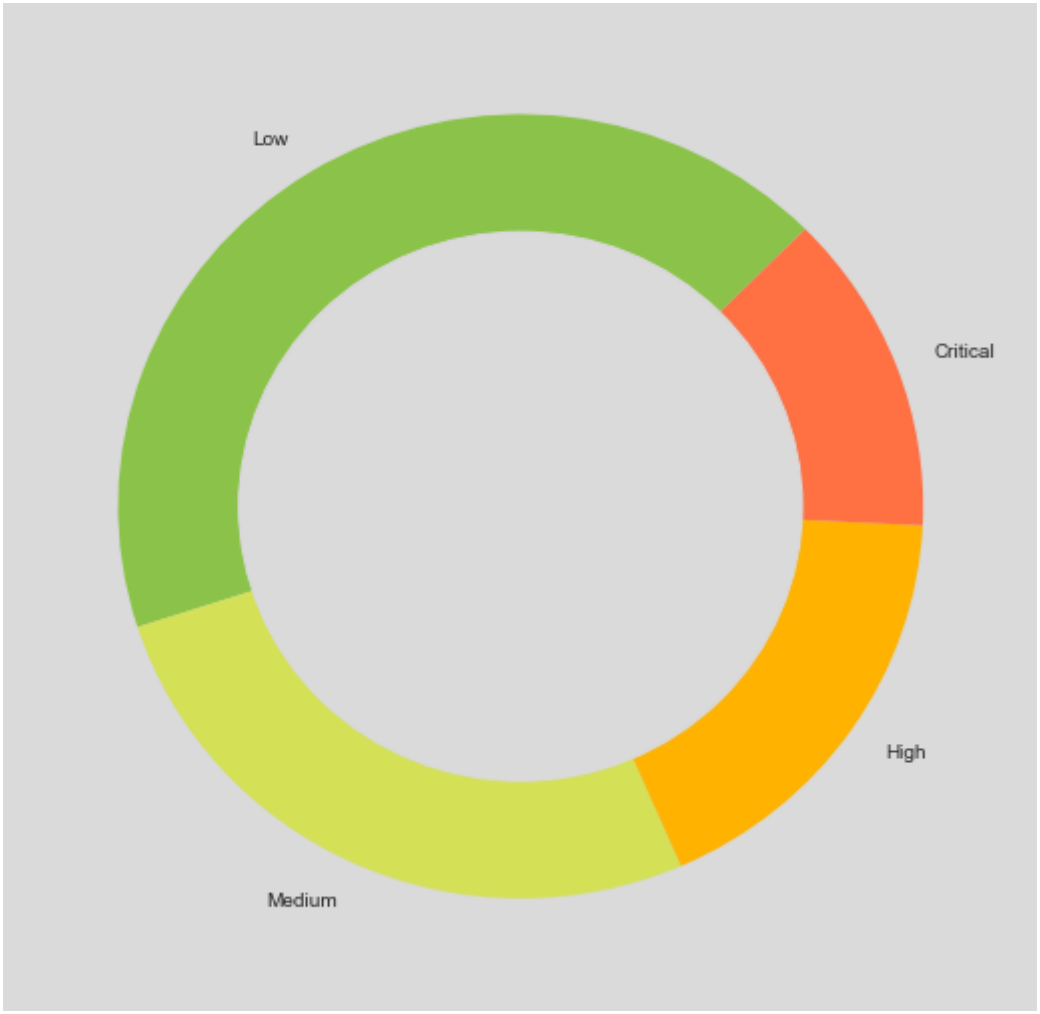
In [313]:

```
plt.figure(figsize=(9,9))
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']
plt.pie (area , labels= labels , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



In [314]:

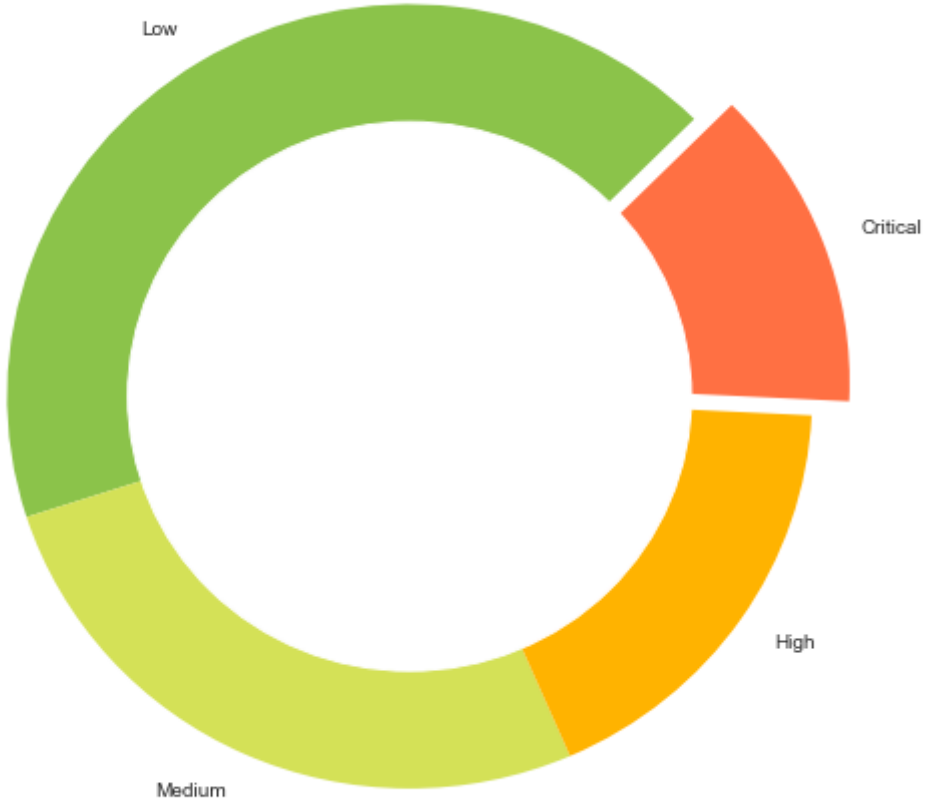
```
# Changing background color
fig = plt.figure(figsize=(9,9))
fig.patch.set_facecolor('#DADADA') # Changing background color of donut chart
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']
plt.pie (area , labels= labels , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='#DADADA') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



### Explode Slice in Donut Chart

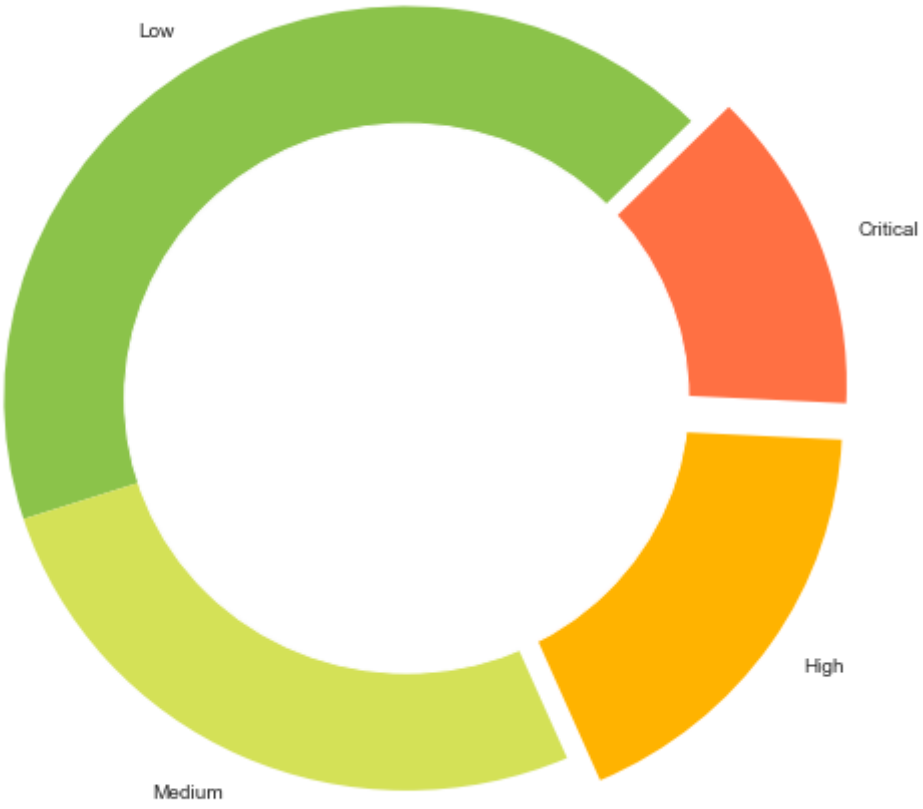
In [315]:

```
plt.figure(figsize=(9,9))
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']
plt.pie (area , labels= labels , colors= colors , startangle=45 , explode=[0,0 , 0.0 , 0.1]
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



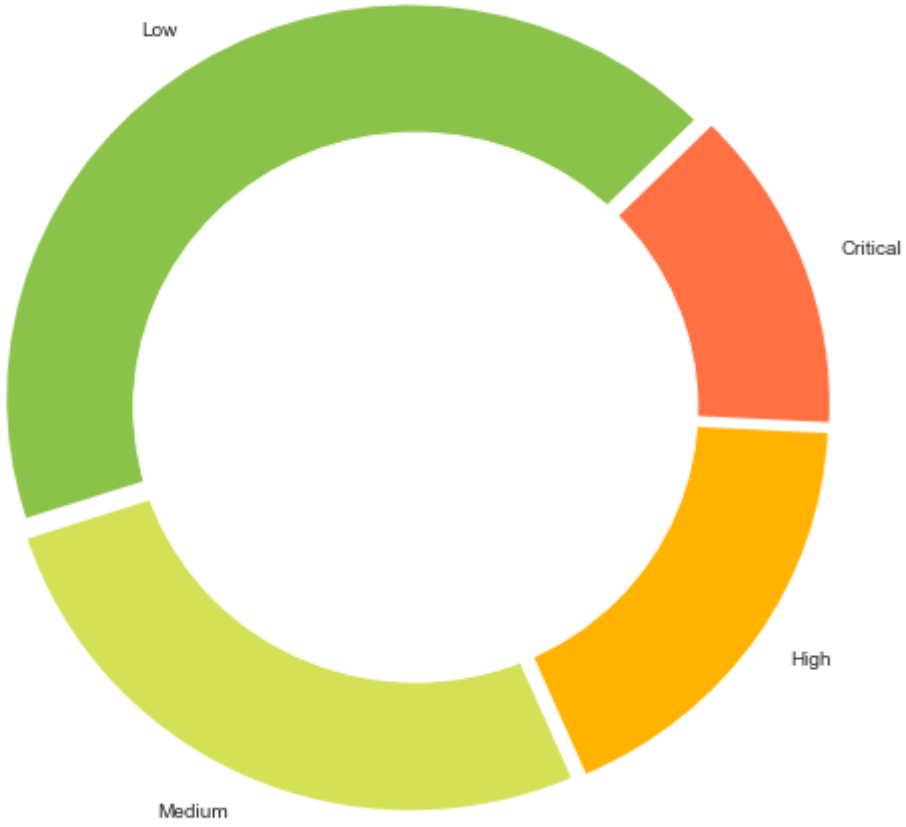
In [316]:

```
plt.figure(figsize=(9,9))
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']
plt.pie (area , labels= labels , colors= colors , startangle=45 , explode=[0,0 , 0.1 , 0.1]
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



In [317]:

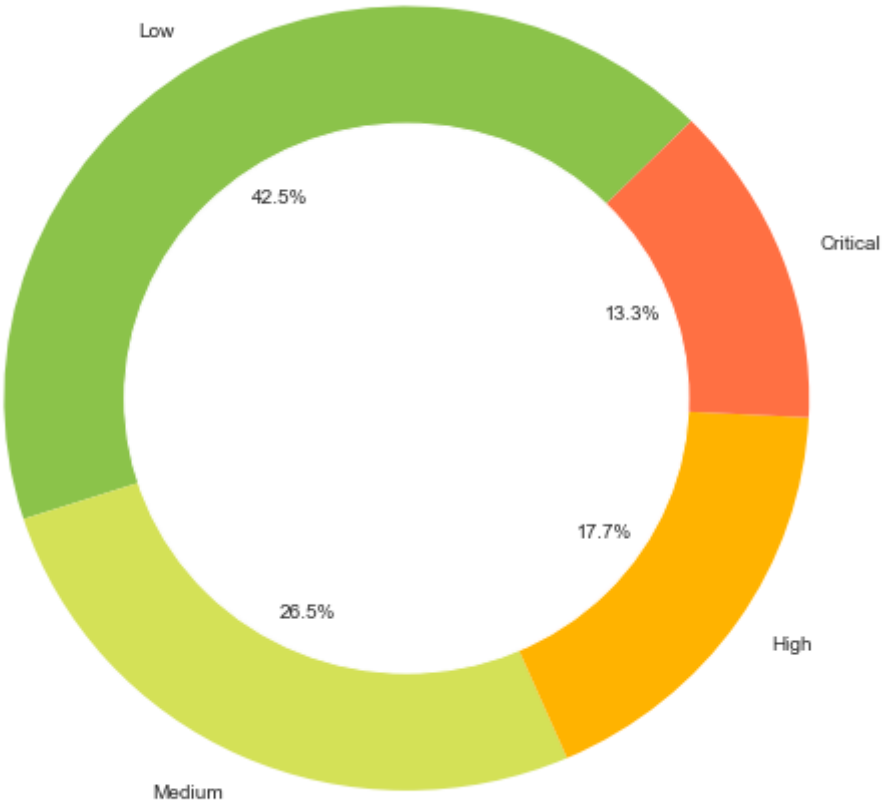
```
plt.figure(figsize=(9,9))
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']
plt.pie (area , labels= labels , colors= colors , startangle=45 , explode=[0.03,0.03 , 0.03
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```




**Displaying percentage and actual values in Donut Chart**

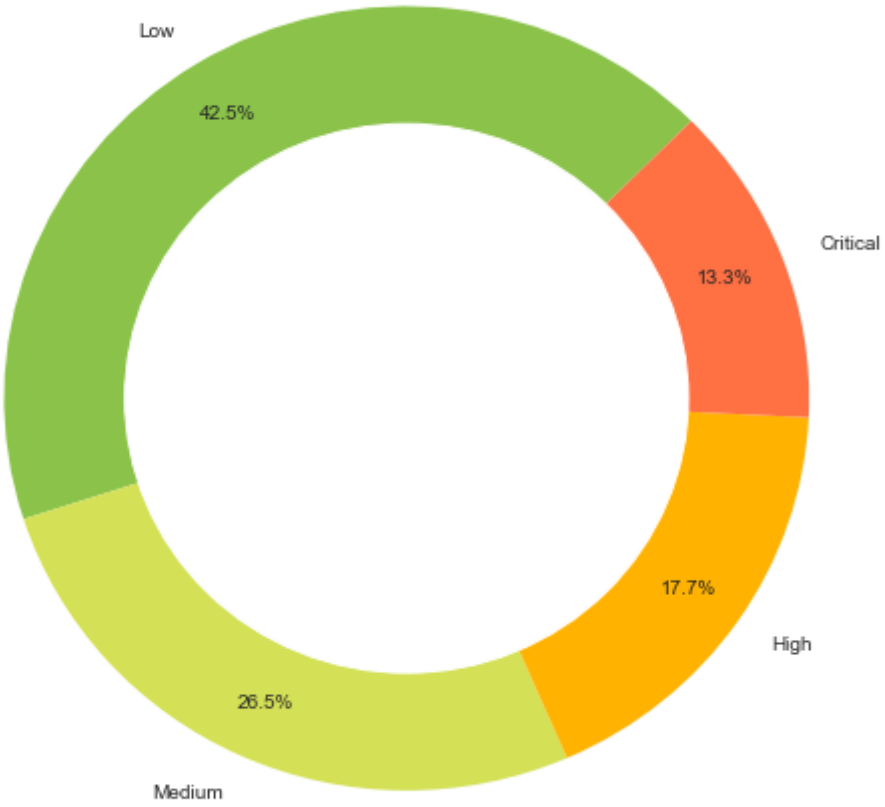
In [318]:

```
plt.figure(figsize=(9,9))
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']
plt.pie (area , labels= labels , colors= colors , startangle=45 , autopct='%1.1f%%', explod
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



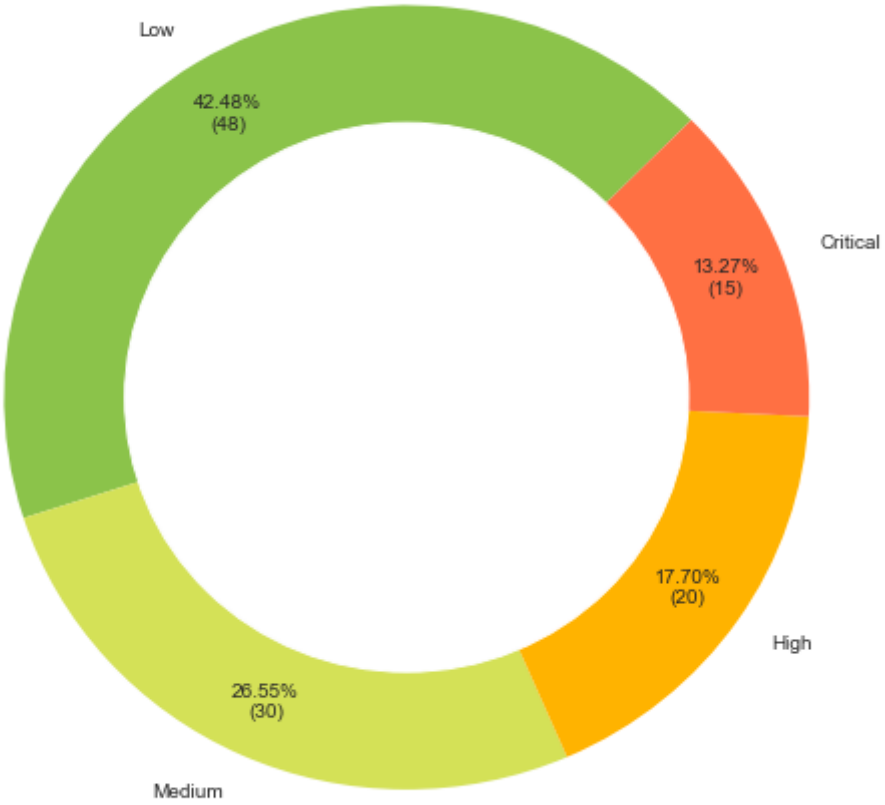
In [319]:

```
plt.figure(figsize=(9,9))
area = [48 , 30 , 20 , 15]
labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']
plt.pie (area , labels= labels , colors= colors , startangle=45 , autopct='%1.1f%%', pctdis
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



In [320]:

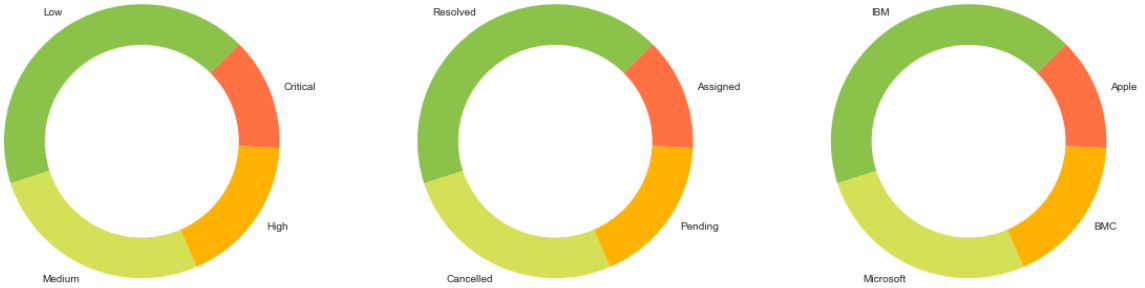
```
plt.figure(figsize=(9,9))
area = [48 , 30 , 20 , 15]
total = np.sum(area)

def val_per(x):
    return '{:.2f}%\n({:.0f})'.format(x, total*x/100)

labels = ['Low' , 'Medium' , 'High' , 'Critical']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']
plt.pie (area , labels= labels , colors= colors , startangle=45 , autopct=val_per, pctdista
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```



### Display multiple Donut plots in one figure

In [321]:

```
fig = plt.figure(figsize=(20,6))
area = [48 , 30 , 20 , 15]
priority = ['Low' , 'Medium' , 'High' , 'Critical']
status = ['Resolved' , 'Cancelled' , 'Pending' , 'Assigned']
company = ['IBM' , 'Microsoft', 'BMC' , 'Apple']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']

plt.subplot(1,3,1)
plt.pie (area , labels= priority , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)

plt.subplot(1,3,2)
plt.pie (area , labels= status , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)

plt.subplot(1,3,3)
plt.pie (area , labels= company , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)

plt.show()
```



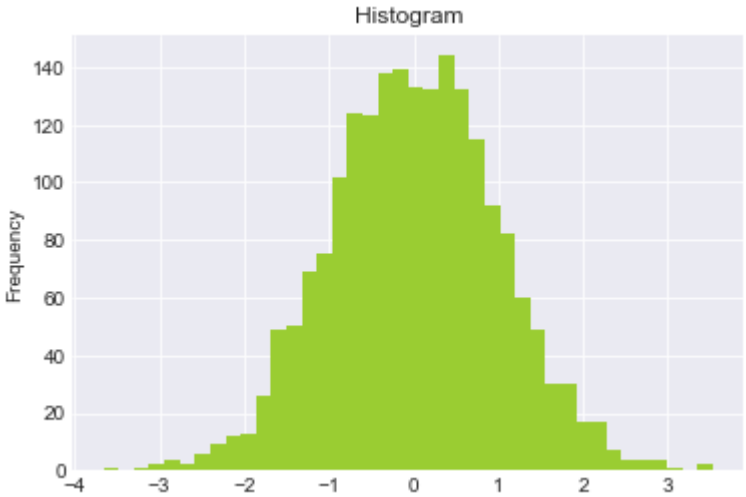
In [322]:

```
fig = plt.figure(figsize=(20,13))
area = [48 , 30 , 20 , 15]
priority = ['Low' , 'Medium' , 'High' , 'Critical']
status = ['Resolved' , 'Cancelled' , 'Pending' , 'Assigned']
company = ['IBM' , 'Microsoft', 'BMC' , 'Apple']
colors = ['#8BC34A','#D4E157','#FFB300','#FF7043']

plt.subplot(2,3,1)
plt.pie (area , labels= priority , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)

plt.subplot(2,3,2)
plt.pie (area , labels= status , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)

plt.subplot(2,3,3)
plt.pie (area , labels= company , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)


plt.subplot(2,3,4)
plt.pie (area , labels= priority , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)

plt.subplot(2,3,5)
plt.pie (area , labels= status , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)

plt.subplot(2,3,6)
plt.pie (area , labels= company , colors= colors , startangle=45)
my_circle=plt.Circle( (0,0), 0.7, color='white') # Adding circle at the centre
p=plt.gcf()
p.gca().add_artist(my_circle)

plt.show()
```


# Histogram

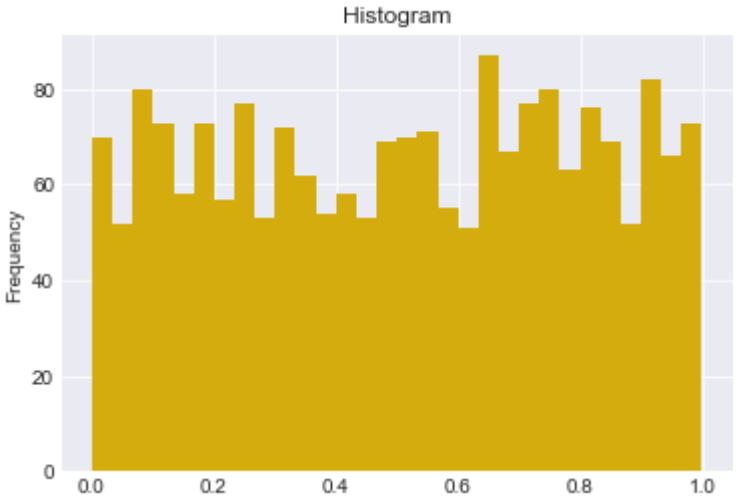
In [323]:

```
x = np.random.normal(size = 2000)
plt.hist(x, bins=40, color='yellowgreen')
plt.gca().set(title='Histogram', ylabel='Frequency')
plt.show()
```



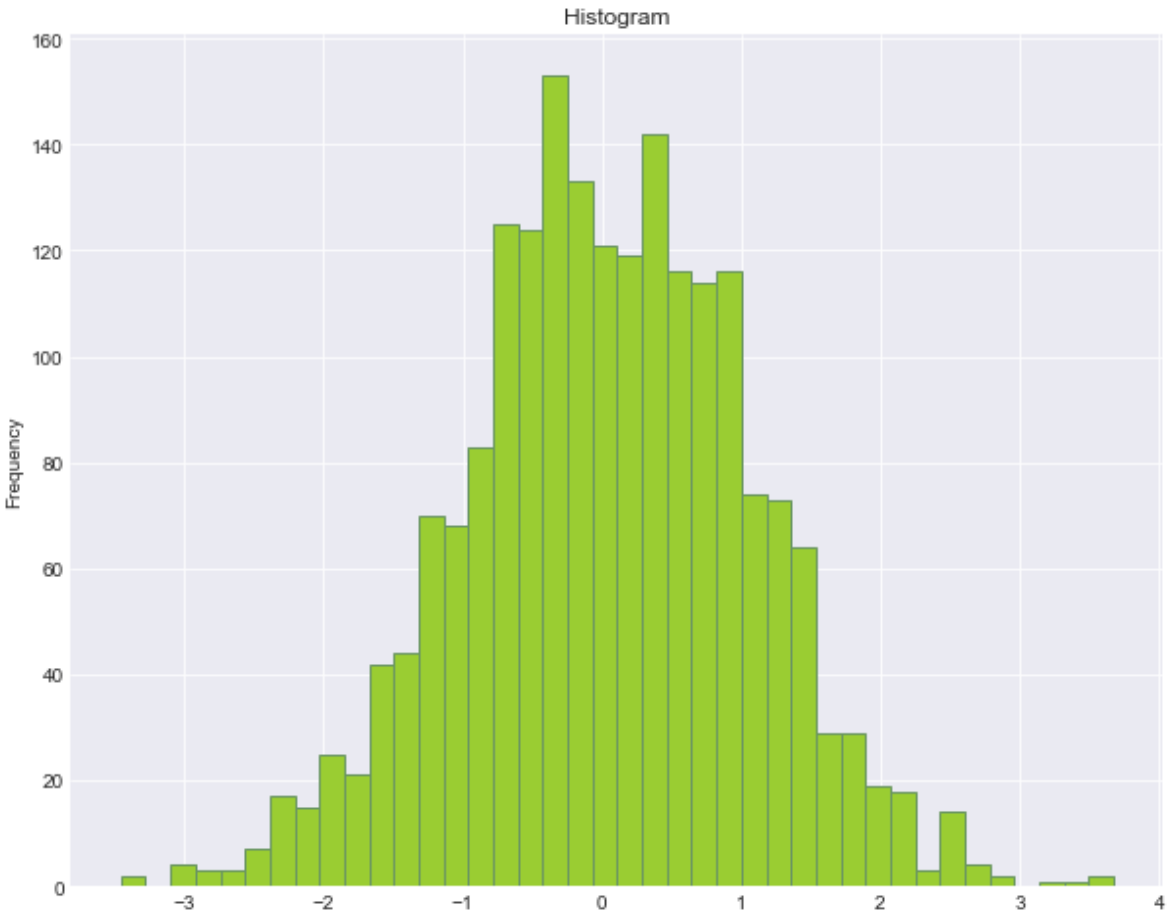
In [324]:

```
x = np.random.rand(2000)
plt.hist(x, bins=30 ,color='#D4AC0D')
plt.gca().set(title='Histogram', ylabel='Frequency')
plt.show()
```



In [325]:

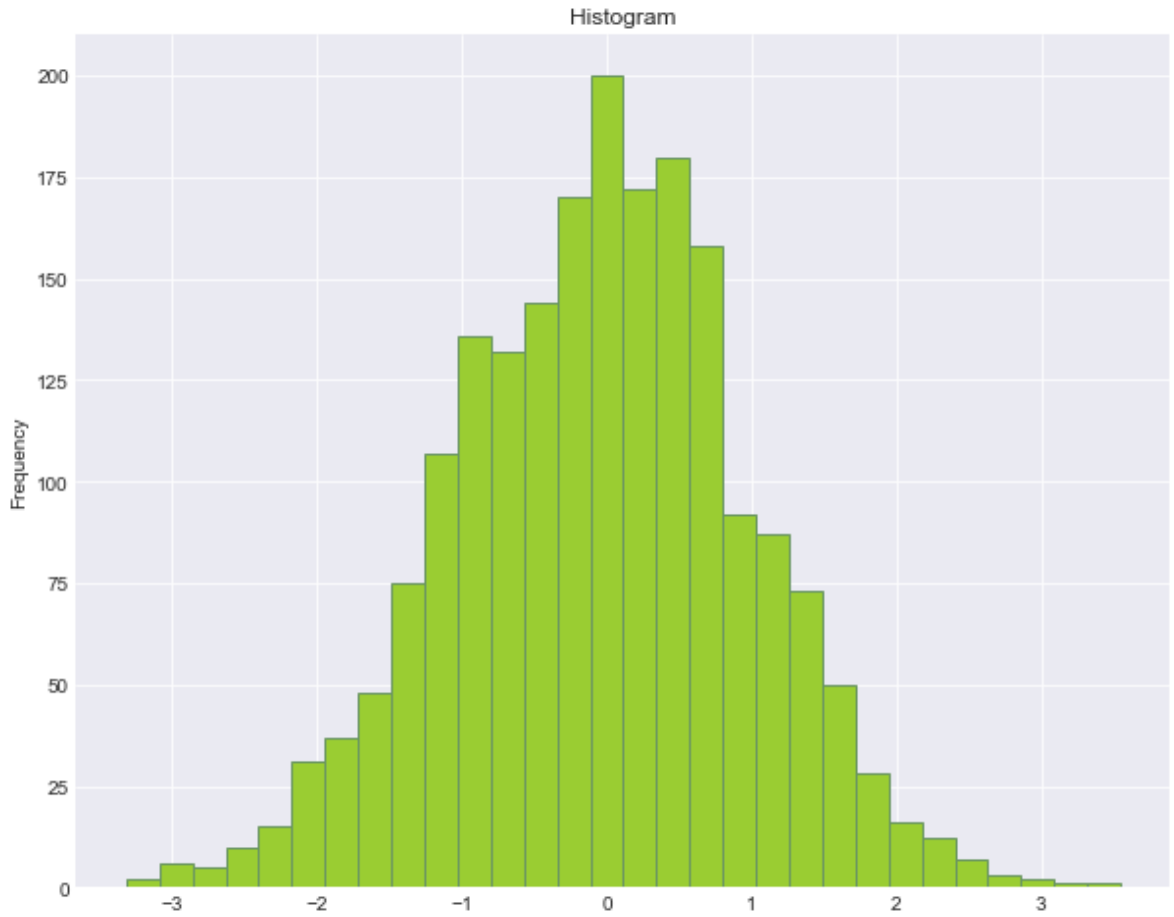
```
# Using Edge Color for readability
plt.figure(figsize=(10,8))
x = np.random.normal(size = 2000)
plt.hist(x, bins=40, color='yellowgreen' , edgecolor="#6A9662")
plt.gca().set(title='Histogram', ylabel='Frequency')
plt.show()
```

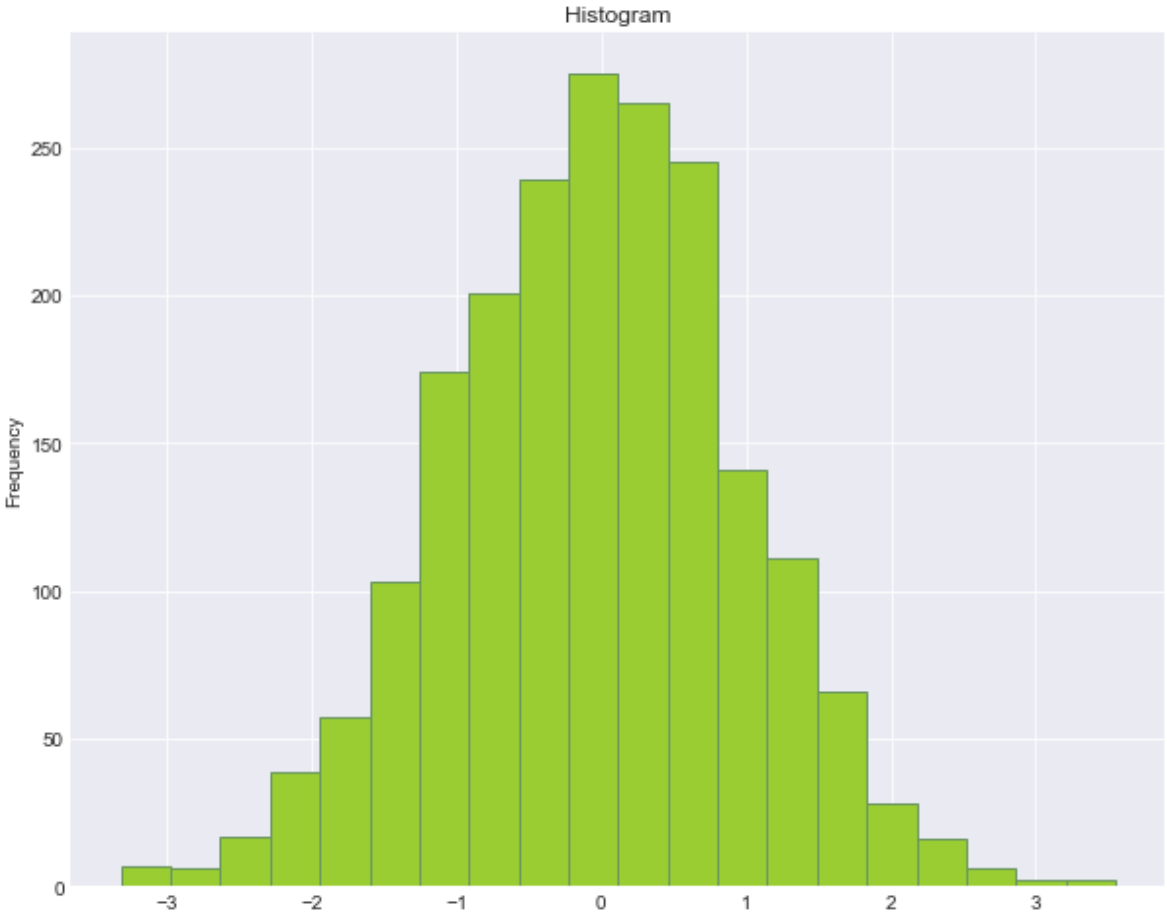


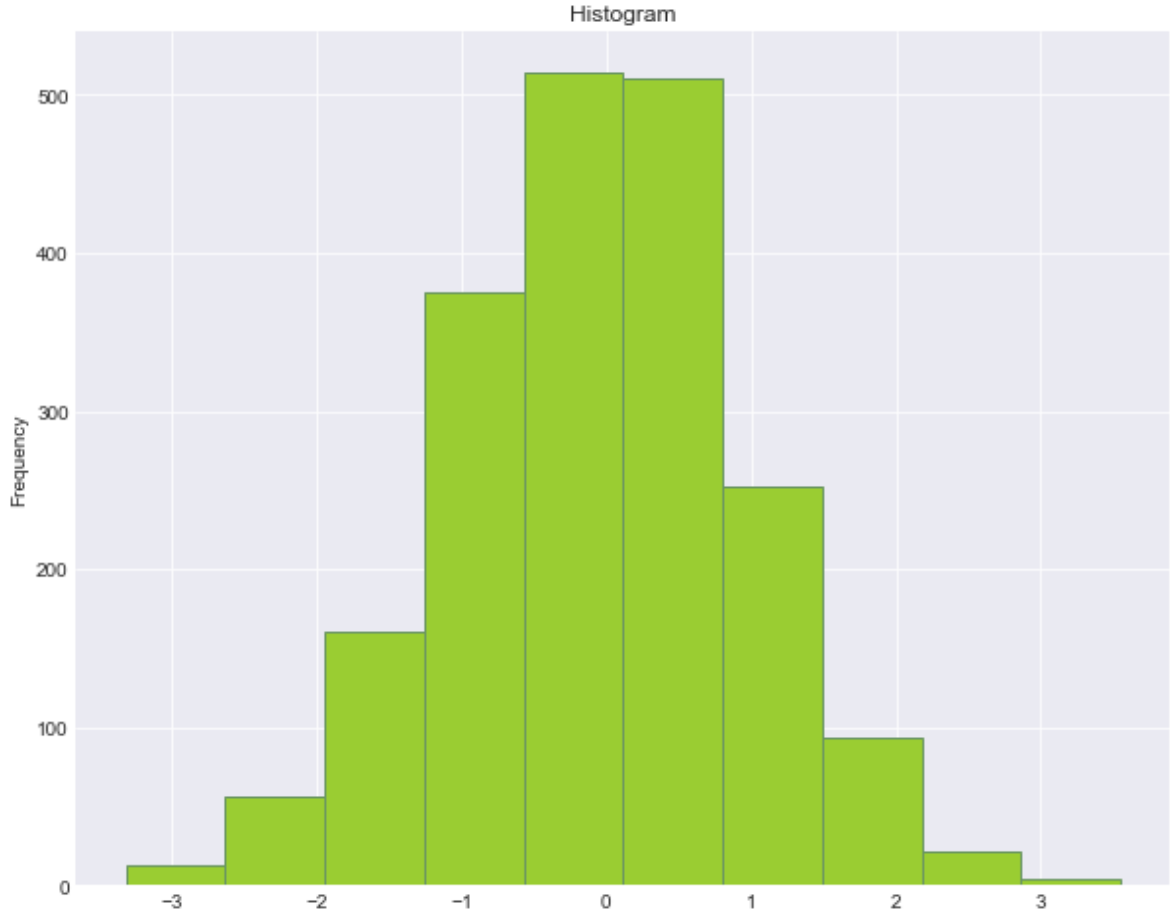

## Binning

In [326]:

```
# Binning
plt.figure(figsize=(10,8))
x = np.random.normal(size = 2000)
plt.hist(x, bins=30, color='yellowgreen' , edgecolor="#6A9662")
plt.gca().set(title='Histogram', ylabel='Frequency')
plt.show()

plt.figure(figsize=(10,8))
plt.hist(x, bins=20, color='yellowgreen' , edgecolor="#6A9662")
plt.gca().set(title='Histogram', ylabel='Frequency')
plt.show()

plt.figure(figsize=(10,8))
plt.hist(x, bins=10, color='yellowgreen' , edgecolor="#6A9662")
plt.gca().set(title='Histogram', ylabel='Frequency')
plt.show()
```



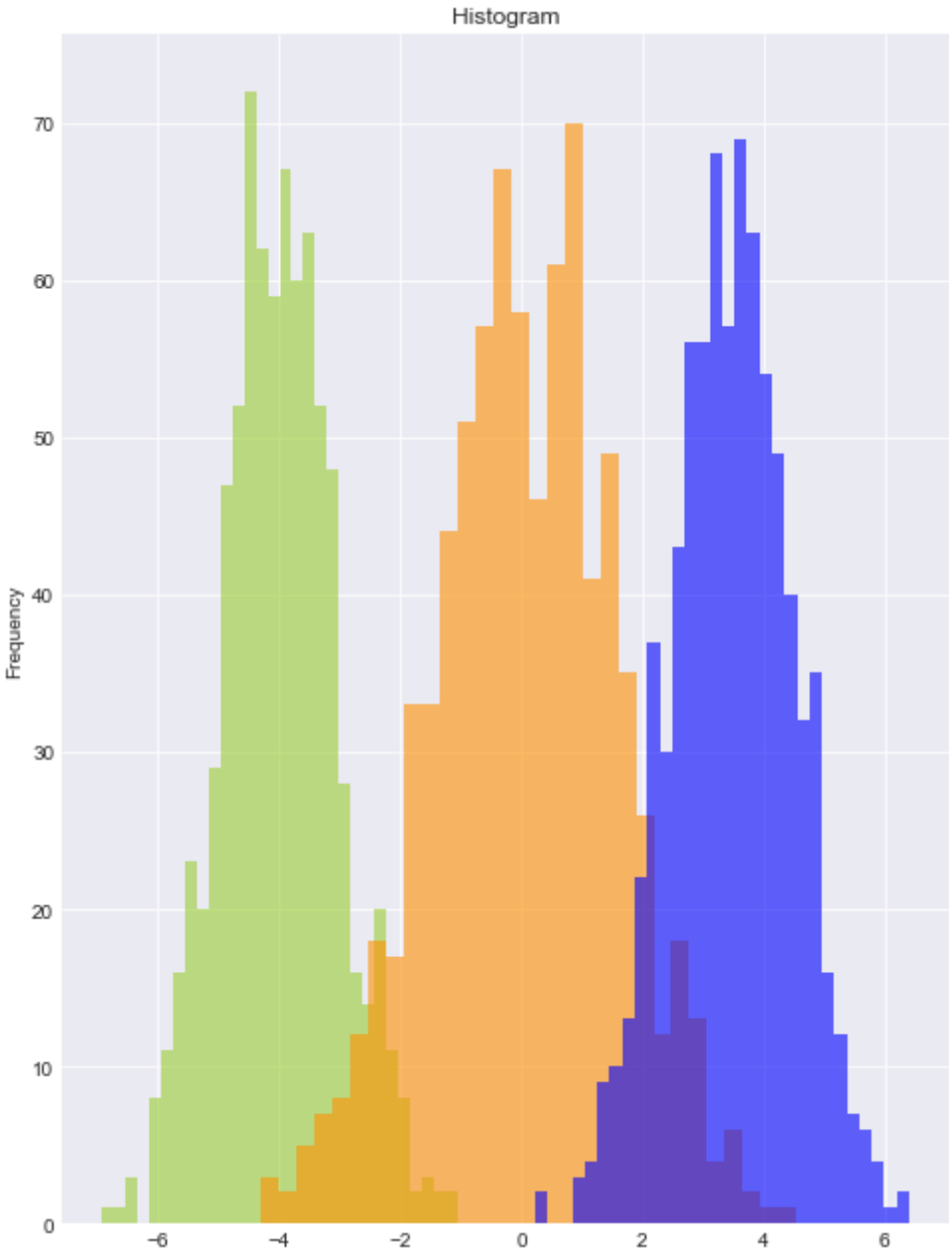

Histogram
Frequency
250
200
150
100
50
0
−3
−2
−1
0
1
2
3




## Plotting Multiple Histograms

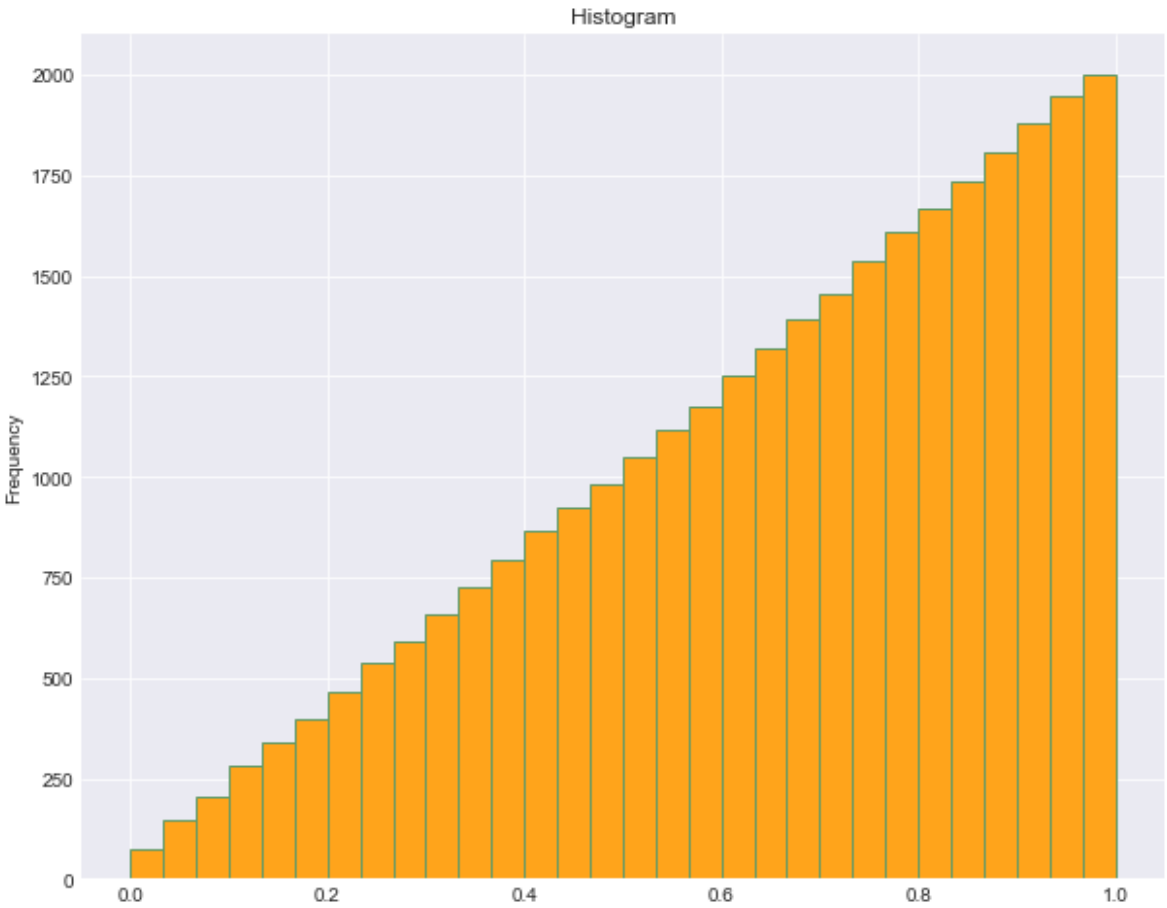
In [329]:

```
plt.figure(figsize=(8,11))
x = np.random.normal(-4,1,size = 800)
y = np.random.normal(0,1.5,size = 800)
z = np.random.normal(3.5,1,size = 800)
plt.hist(x, bins=30, color='yellowgreen' , alpha=0.6)
plt.hist(y, bins=30, color='#FF8F00' , alpha=0.6)
plt.hist(z, bins=30, color='blue' , alpha=0.6)
plt.gca().set(title='Histogram', ylabel='Frequency')
plt.show()
```



In [330]:

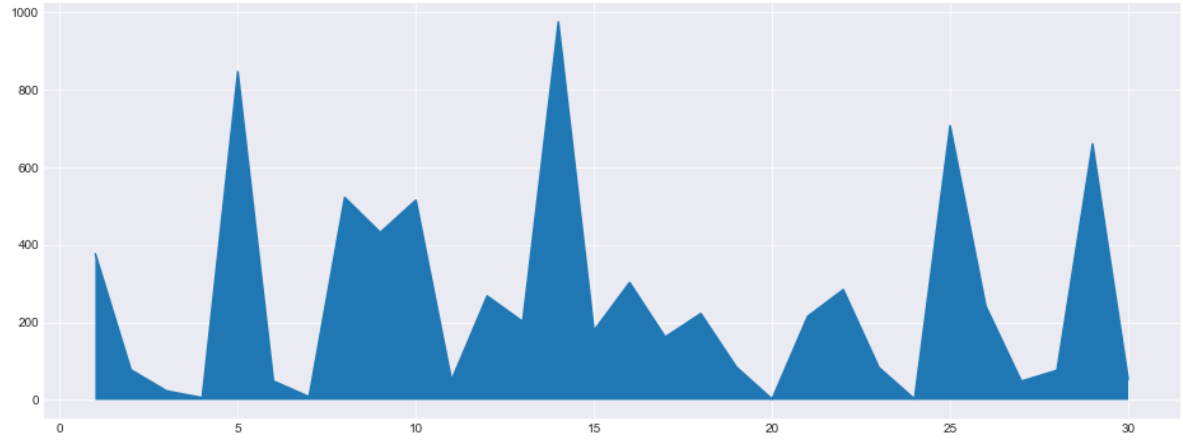
```
# Using Histogram to plot a cumulative distribution function
plt.figure(figsize=(10,8))
x = np.random.rand(2000)
plt.hist(x, bins=30 ,color='#ffa41b' , edgecolor="#639a67",cumulative=True)
plt.gca().set(title='Histogram', ylabel='Frequency')
plt.show()
```




# Area Plot

In [331]:

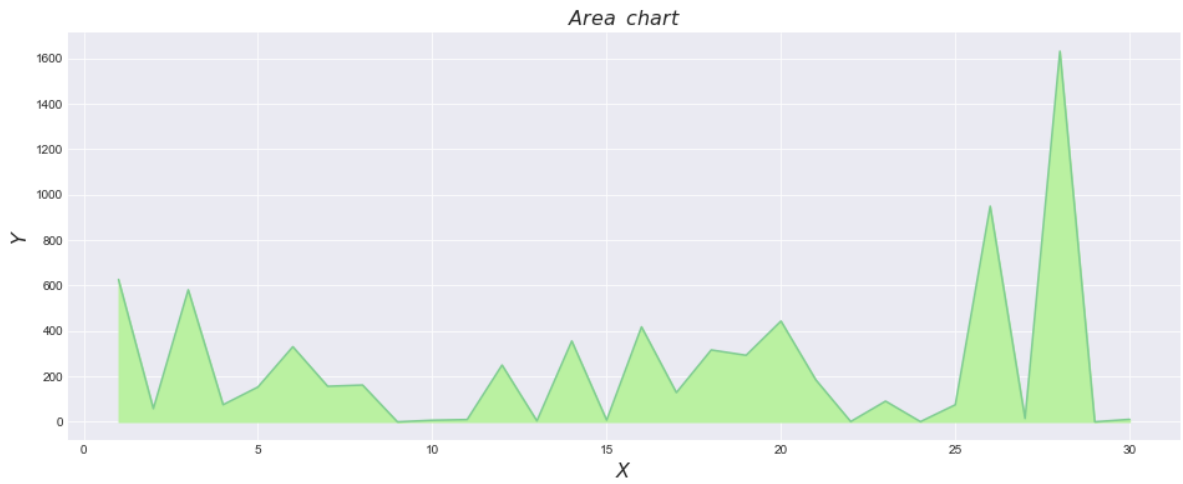
```
x = np.arange(1,31)
y = np.random.normal(10,11,size=30)
y = np.square(y)
plt.figure(figsize=(16,6))
plt.plot(x,y)
plt.fill_between(x, y)
plt.show()
```




### Changing Fill Color

In [334]:

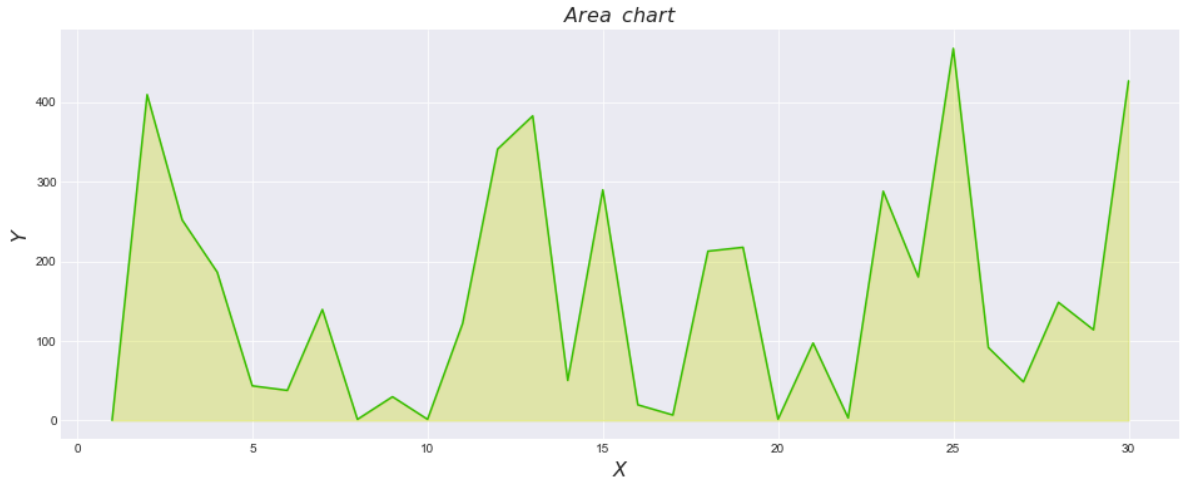
```
x = np.arange(1,31)
y = np.random.normal(10,11,size=30)
y = np.square(y)
plt.figure(figsize=(16,6))

plt.fill_between( x, y, color="#baf1a1")  # #Changing Fill color
plt.plot(x, y, color='#7fcd91') # Color on edges

plt.title("$ Area $  $ chart $" , fontsize = 16)
plt.xlabel("$X$" , fontsize = 16)
plt.ylabel("$Y$" , fontsize = 16)


plt.show()
```




**Changing Fill Color and its transperancy**

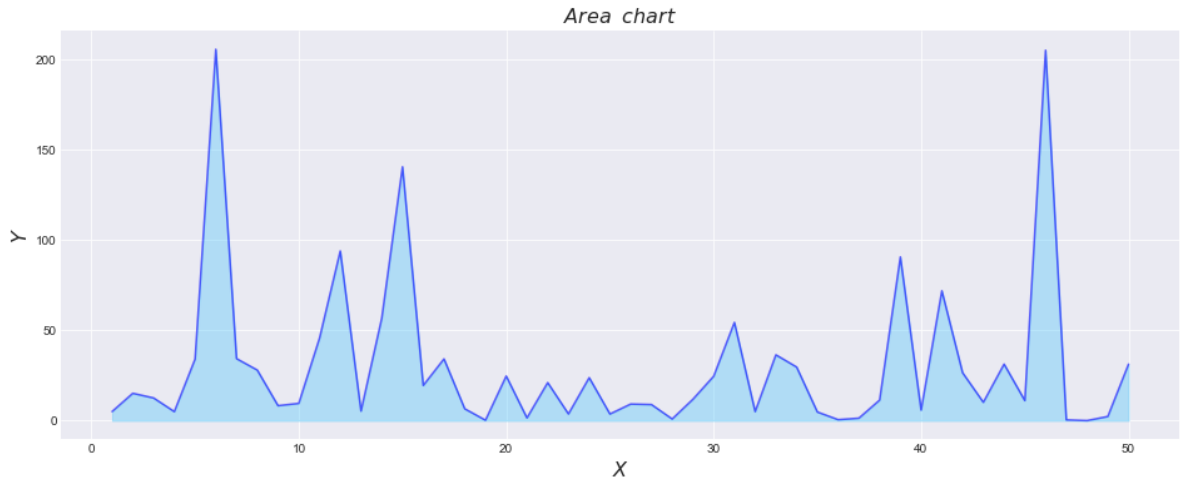
In [336]:

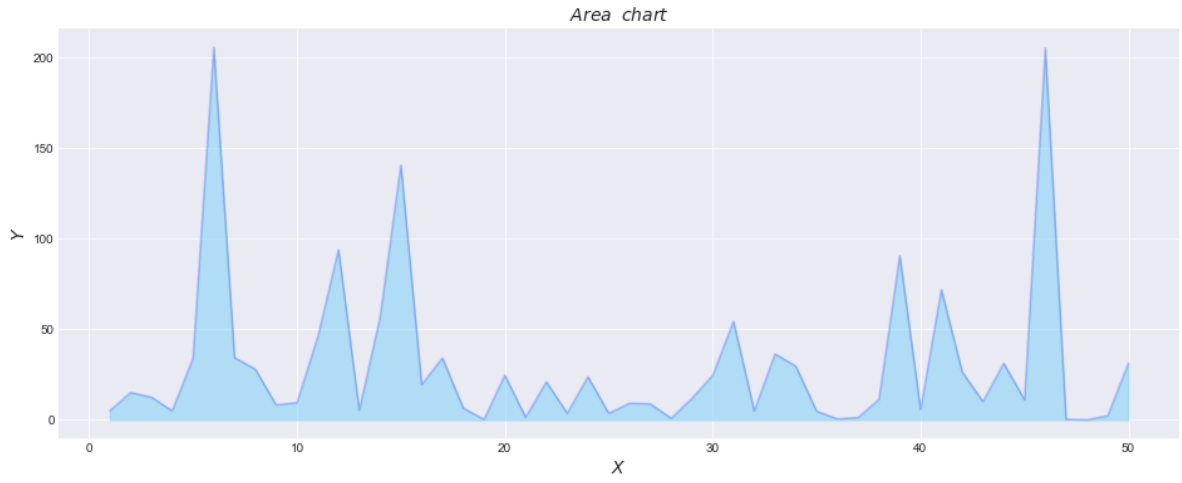
```
x = np.arange(1,31)
y = np.random.normal(10,11,size=30)
y = np.square(y)
plt.figure(figsize=(16,6))

plt.fill_between( x, y, color="#C8D700" , alpha = 0.3) # Changing transperancy using Alpha
plt.plot(x, y, color='#36BD00')

plt.title("$ Area $  $ chart $" , fontsize = 16)
plt.xlabel("$X$" , fontsize = 16)
plt.ylabel("$Y$" , fontsize = 16)

plt.show()
```



In [338]:

```
x = np.arange(1,51)
y = np.random.normal(1,5,size=50)
y = np.square(y)

plt.figure(figsize=(16,6))
plt.fill_between( x, y, color="#5ac8fa", alpha=0.4)
plt.plot(x, y, color="blue", alpha=0.6) # Bold line on edges
plt.title("$ Area $  $ chart $" , fontsize = 16)
plt.xlabel("$X$" , fontsize = 16)
plt.ylabel("$Y$" , fontsize = 16)
plt.show()

plt.figure(figsize=(16,6))
plt.fill_between( x, y, color="#5ac8fa", alpha=0.4)
plt.plot(x, y, color="blue", alpha=0.2) # Less stronger line on edges
plt.title("$ Area $  $ chart $" , fontsize = 14)
plt.xlabel("$X$" , fontsize = 14)
plt.ylabel("$Y$" , fontsize = 14)
plt.show()
```

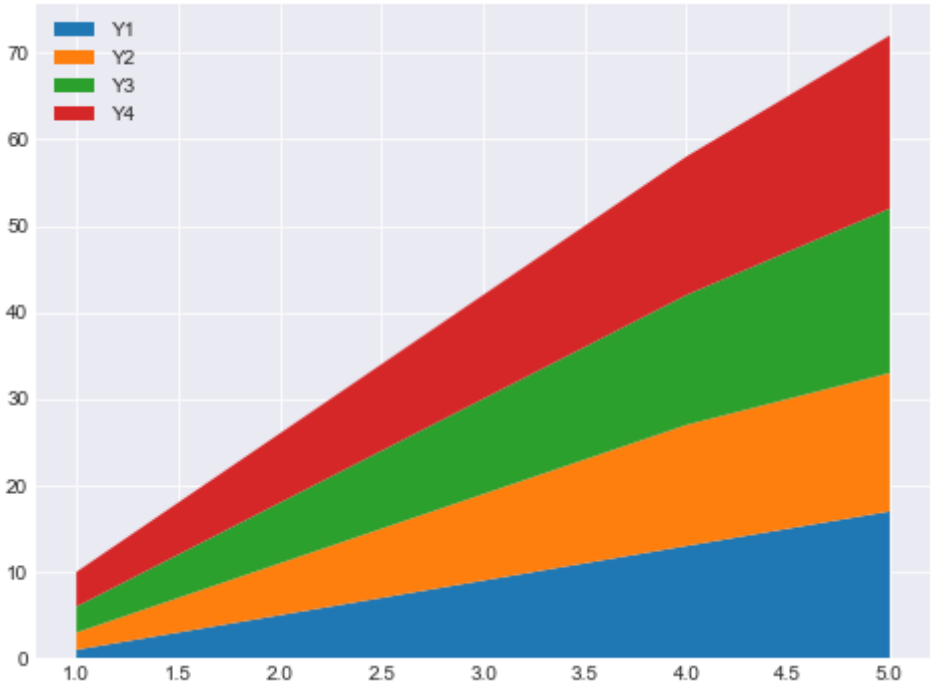



### Stacked Area plot

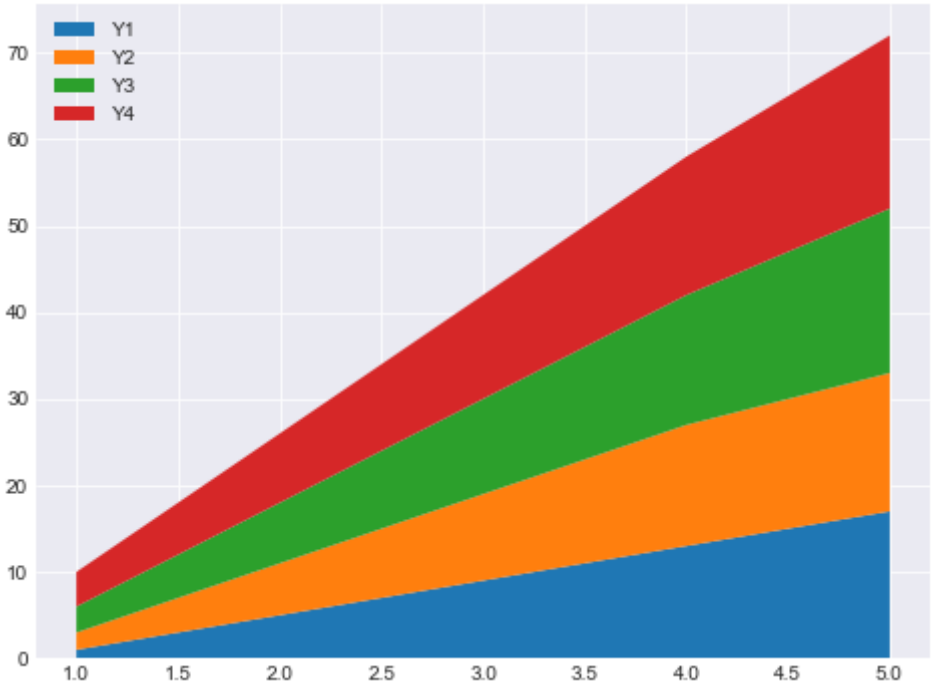
In [339]:

```
x=np.arange(1,6)
y1 =  np.array([1,5,9,13,17])
y2 = np.array([2,6,10,14,16])
y3 = np.array([3,7,11,15,19])
y4 = np.array([4,8,12,16,20])

plt.figure(figsize=(8,6))
plt.stackplot(x,y1,y2,y3,y4, labels=['Y1','Y2','Y3','Y4'])
plt.legend(loc='upper left')
plt.show()
```



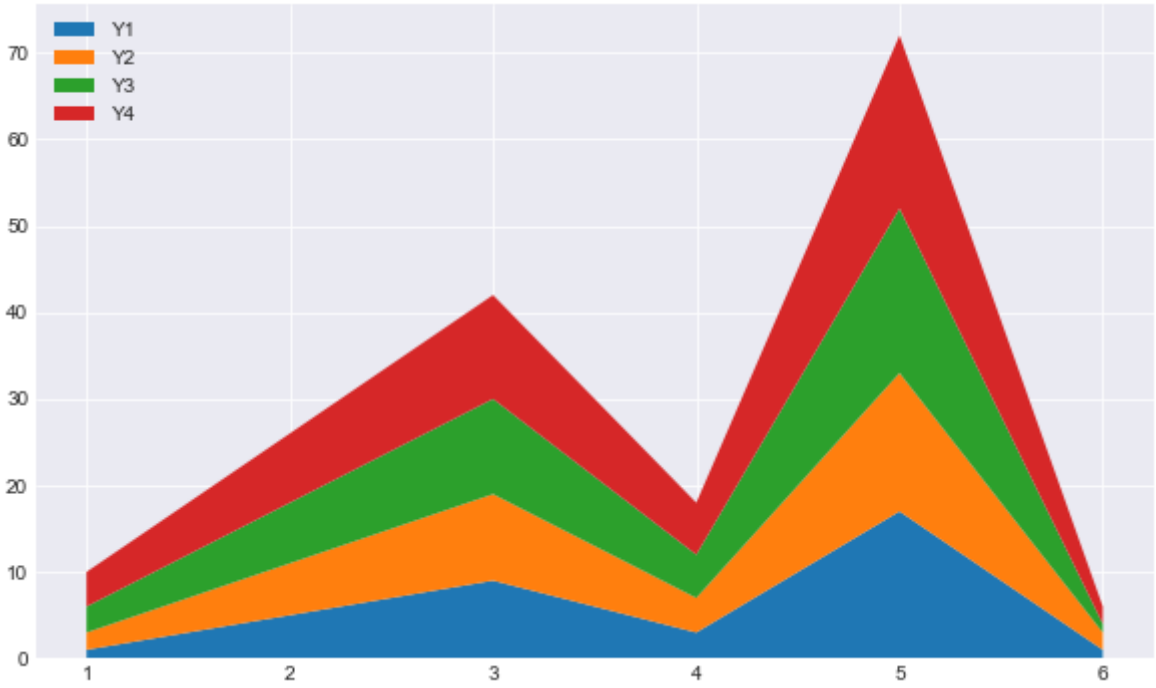
In [340]:

```
x=np.arange(1,6)
y=[ [1,5,9,13,17], [2,6,10,14,16], [3,7,11,15,19] , [4,8,12,16,20] ]
plt.figure(figsize=(8,6))
plt.stackplot(x,y , labels=['Y1','Y2','Y3','Y4'])
plt.legend(loc='upper left')
plt.show()
```



In [341]:

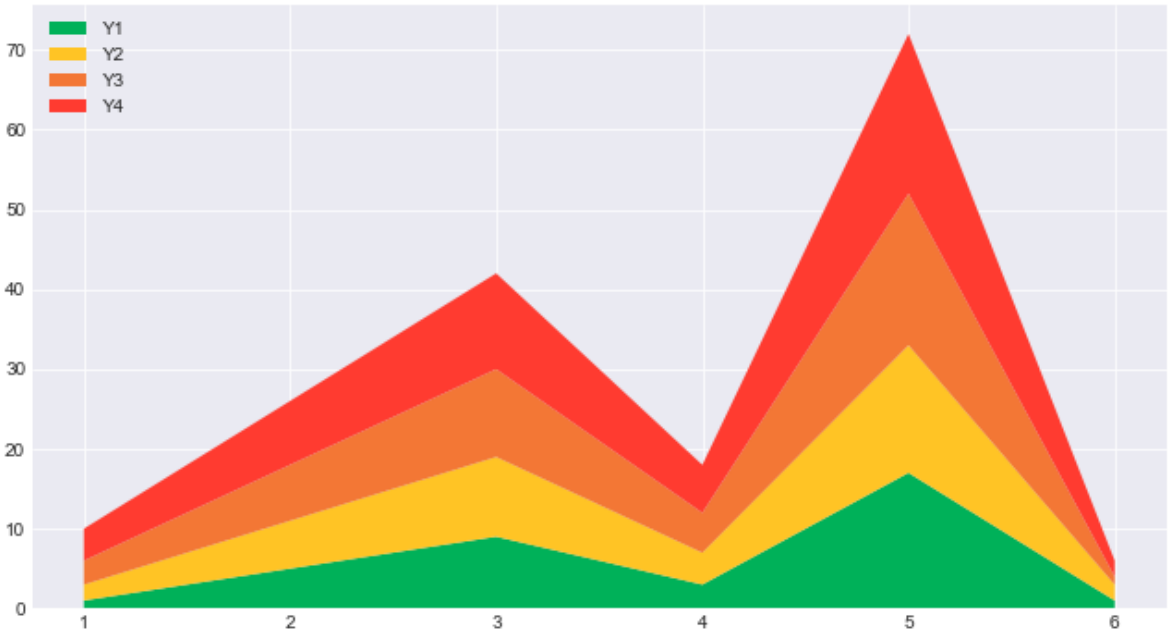
```
x=np.arange(1,7)
y=[ [1,5,9,3,17,1], [2,6,10,4,16,2], [3,7,11,5,19,1] , [4,8,12,6,20,2] ]
plt.figure(figsize=(10,6))
plt.stackplot(x,y , labels=['Y1','Y2','Y3','Y4'])
plt.legend(loc='upper left')
plt.show()
```

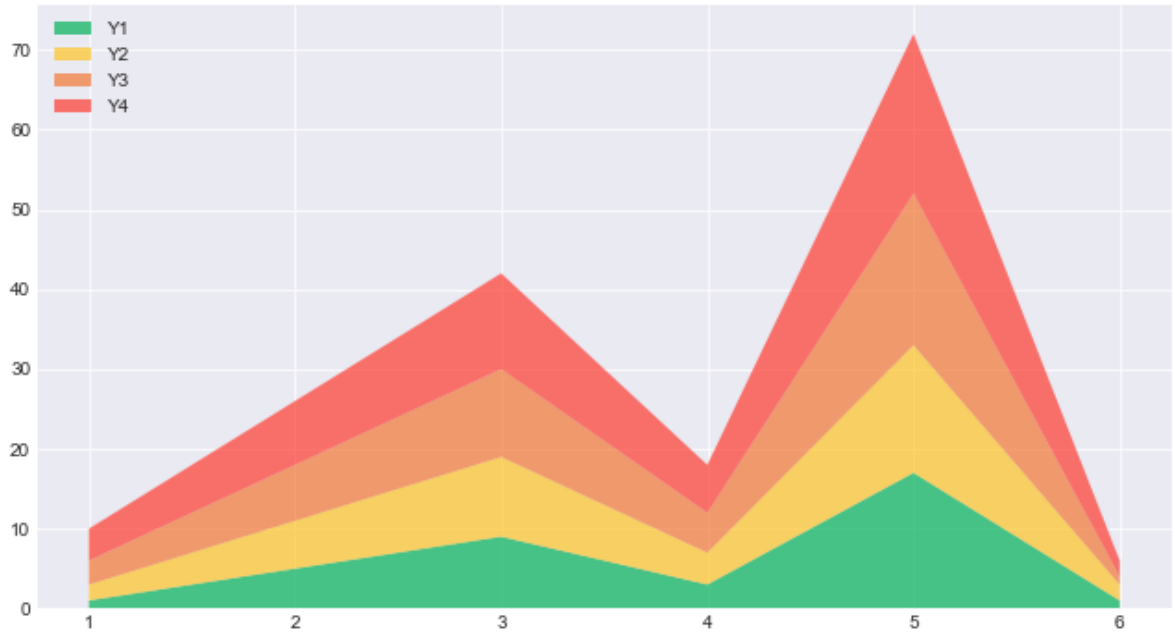


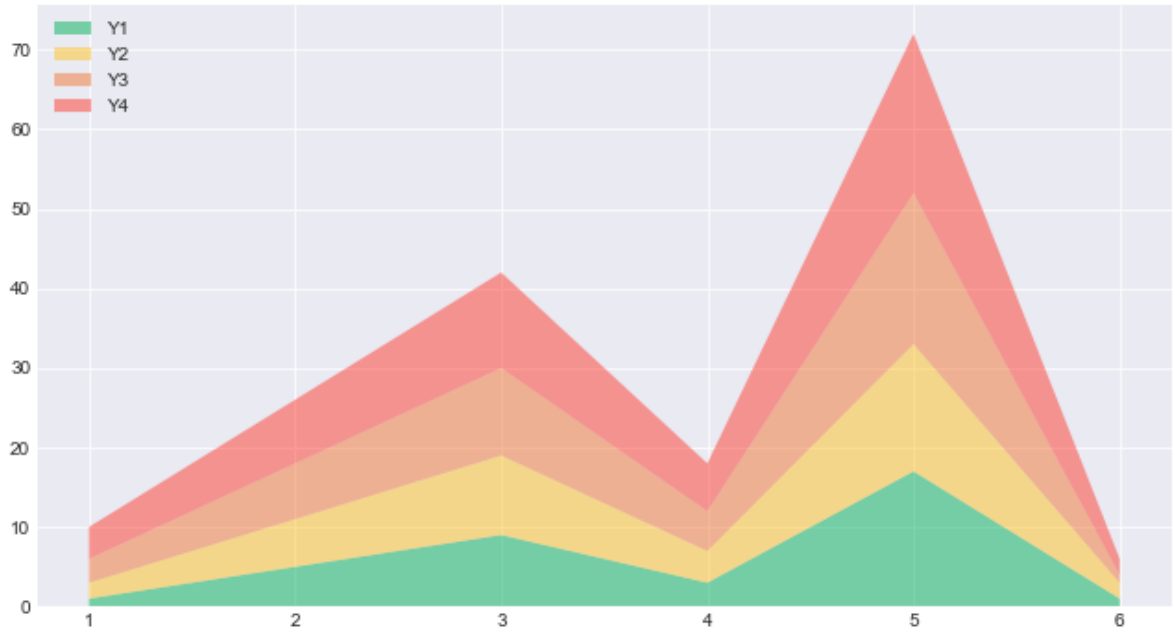

**Changing Fill Color and its transperancy in Stacked Plot**

In [342]:

```
x=np.arange(1,7)
y=[ [1,5,9,3,17,1], [2,6,10,4,16,2], [3,7,11,5,19,1] , [4,8,12,6,20,2] ]

plt.figure(figsize=(11,6))
plt.stackplot(x,y , labels=['Y1','Y2','Y3','Y4'] , colors= ["#00b159" , "#ffc425", "#f37735
plt.legend(loc='upper left')
plt.show()

plt.figure(figsize=(11,6))
plt.stackplot(x,y, labels=['Y1','Y2','Y3','Y4'], colors= ["#00b159" , "#ffc425", "#f37735",
plt.legend(loc='upper left')
plt.show()

plt.figure(figsize=(11,6))
plt.stackplot(x,y, labels=['Y1','Y2','Y3','Y4'], colors= ["#00b159" , "#ffc425", "#f37735",
plt.legend(loc='upper left')
plt.show()
```




Y1
Y2
Y3
Y4
0
10
20
30
40
50
60
70
1
2
3
4
5
6


Y1
Y2
Y3
Y4
0
10
20
30
40
50
60
70
1
2
3
4
5
6

# END

In [ ]: