Matplotlib Library

Matplotlib:-

Matplotlib is a low level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely.

Installation Of Matplotlib:-

If you have Python and PIP already installed on a system, then installation of Matplotlib is very easy. Install it using this command in command prompt: pip install matplotlib If this command fails, then use a python distribution that already has Matplotlib installed, like Anaconda, Spyder etc.

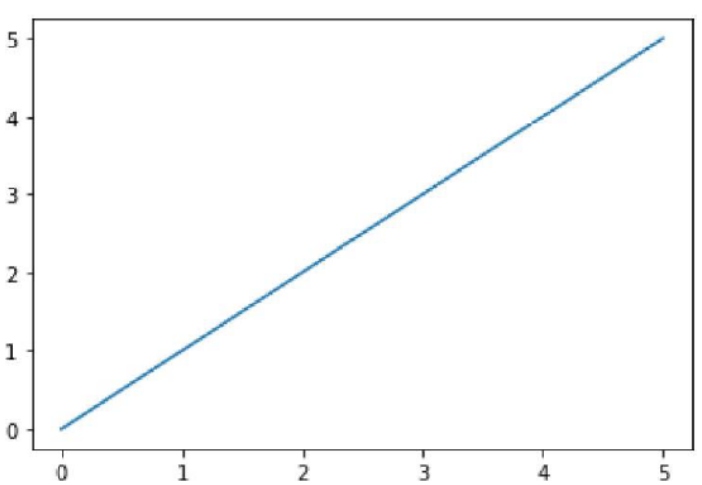
Import Matplotlib:-

Once Matplotlib is installed, import it in your applications by adding the import module statement. import matplotlib Now Matplotlib is imported and ready to use.

Matplotlib Pyplot:-

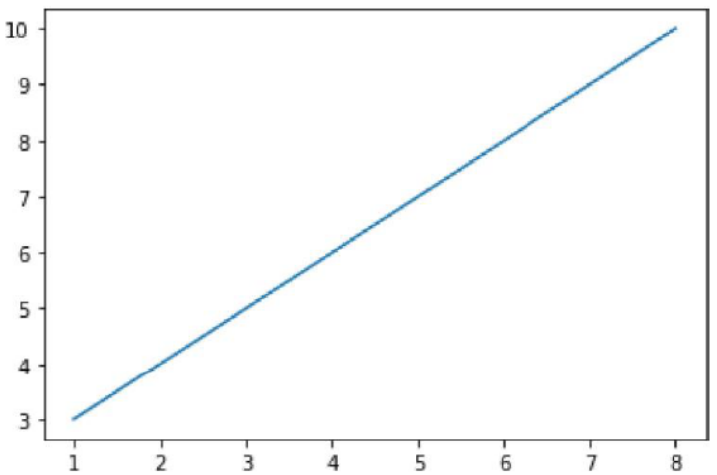
Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias: import matplotlib.pyplot as plt Now the Pyplot package can be referred to as plt.

In [1]:
```
#simple line plot
import matplotlib.pyplot as plt
x1=[0,1,2,3,4,5]
y1=[0,1,2,3,4,5]
plt.plot(x1,y1)
plt.show()
```



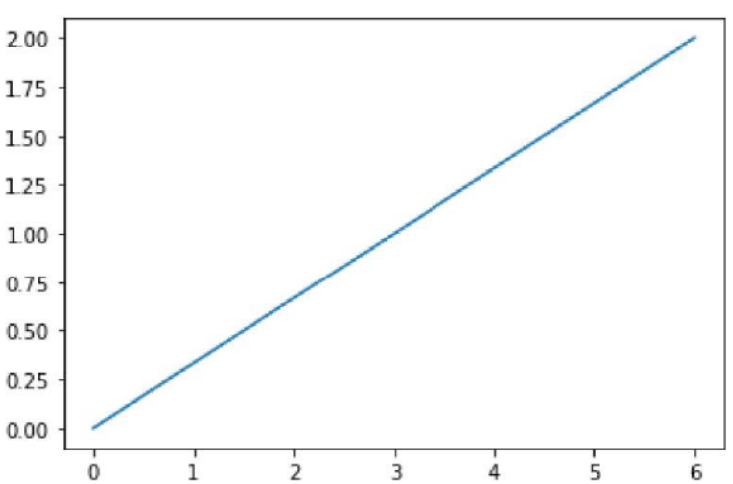
In [3]:
```
#Draw a line in a diagram from position (1, 3) to position (8, 10):

import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 8])
```

```
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show()
```



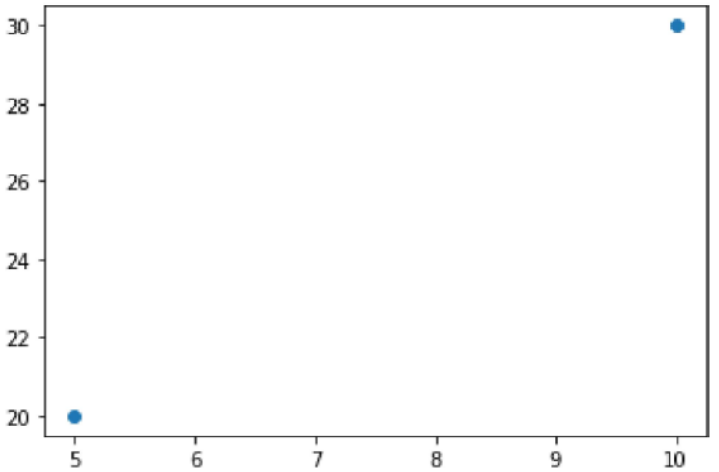
In [2]:

```
#Example:
#Draw a line in a diagram from position (0,0) to position (6,2):
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([0, 6])
ypoints = np.array([0, 2])
plt.plot(xpoints, ypoints)
plt.show()
```



In [3]:

```
#Example:
#Draw two points in the diagram, one at position (10, 30) and one in position (5, 2
#Note: To plot only the markers, you can use shortcut string notation parameter 'o
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([10, 5])
ypoints = np.array([30, 20])
plt.plot(xpoints, ypoints, 'o')
plt.show()
```



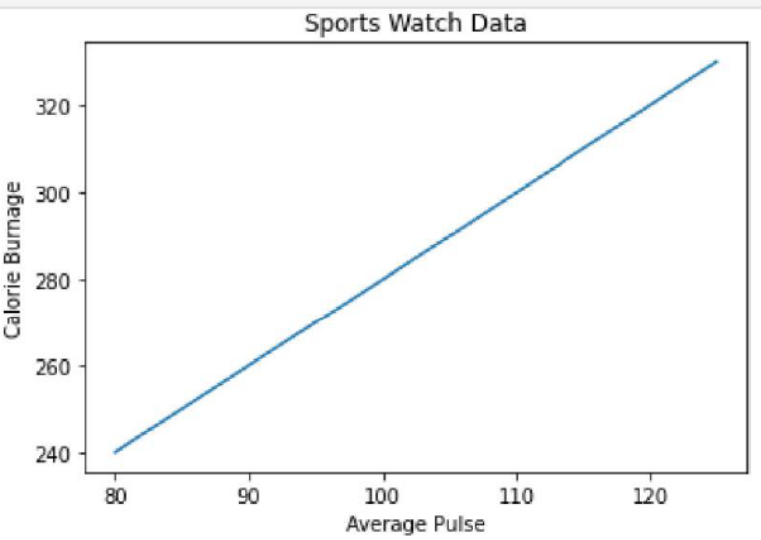
Create Labels for a Plot:-

With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.

Create a Title for a Plot:-

With Pyplot, you can use the title() function to set a title for the plot.

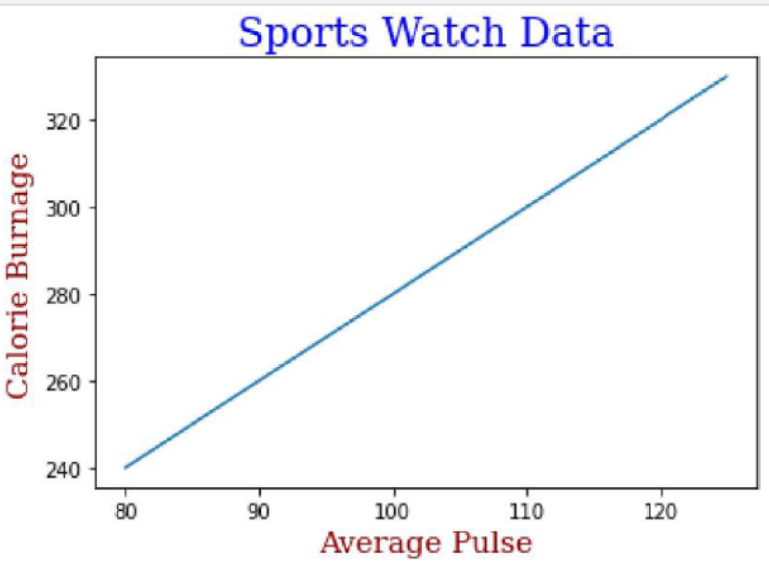
In [14]:
```
#Example
#Add labels to the x- and y-axis:

import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.title("Sports Watch Data")
plt.show()
```



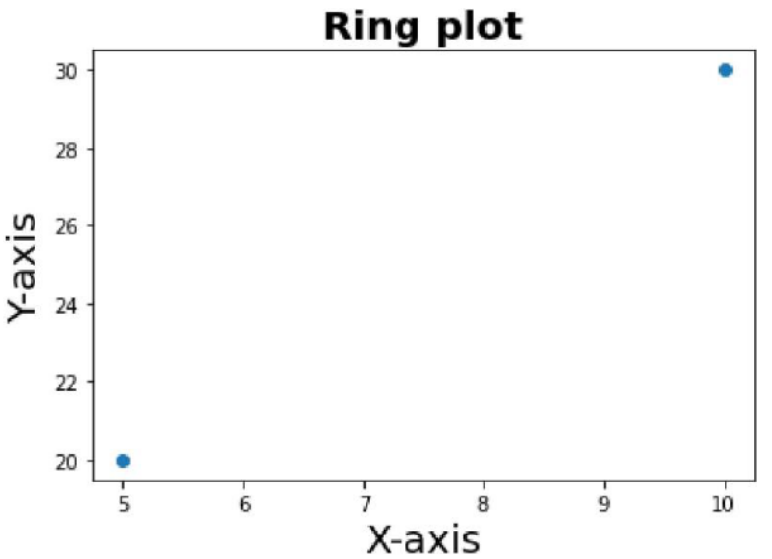
Set Font Properties for Title and Labels:-

You can use the fontdict parameter in xlabel(), ylabel(), and title() to set font properties for the title and labels.

In [15]:
```
#Example
#Set font properties for the title and labels:

import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.title("Sports Watch Data", fontdict = font1)
plt.xlabel("Average Pulse", fontdict = font2)
plt.ylabel("Calorie Burnage", fontdict = font2)

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



In [5]:
```
#simple line plot
import matplotlib.pyplot as plt
import numpy as np
x1=np.array([10,5])
y1=np.array([30,20])
plt.xlabel('X-axis',fontsize=20)
plt.ylabel('Y-axis',fontsize=20)
plt.title('Ring plot',fontsize=20,fontweight="bold")
plt.plot(x1,y1,'o')
plt.show()
```

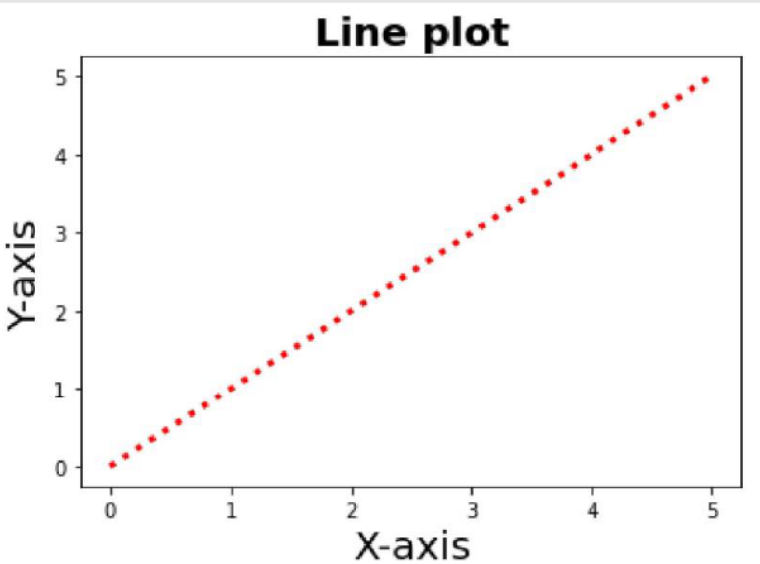


Line Styles:-

You can choose any of these styles:

'solid' (default) '-' 'dotted' ':' 'dashed' '--'
'dashdot' '-.'
'None' '' or ' '

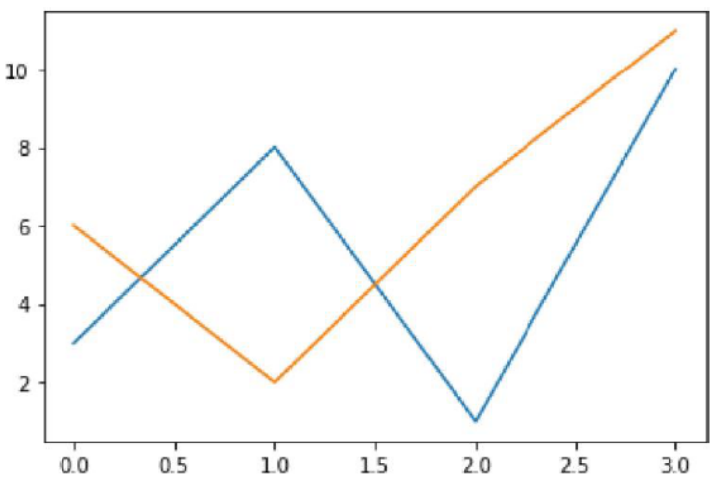
In [2]:
```
#simple line plot
import matplotlib.pyplot as plt
x1=[0,1,2,3,4,5]
y1=[0,1,2,3,4,5]
plt.xlabel('X-axis',fontsize=20)
plt.ylabel('Y-axis',fontsize=20)
plt.title('Line plot',fontsize=20,fontweight="bold")
plt.plot(x1,y1,color="red",linestyle=":",linewidth=3)
plt.show()
```



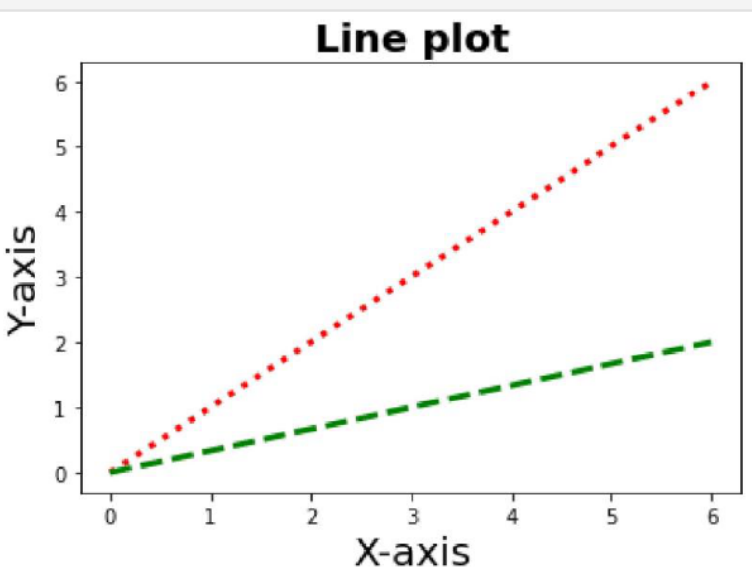
Multiple Lines:-

You can plot as many lines as you like by simply adding more plt.plot() functions:

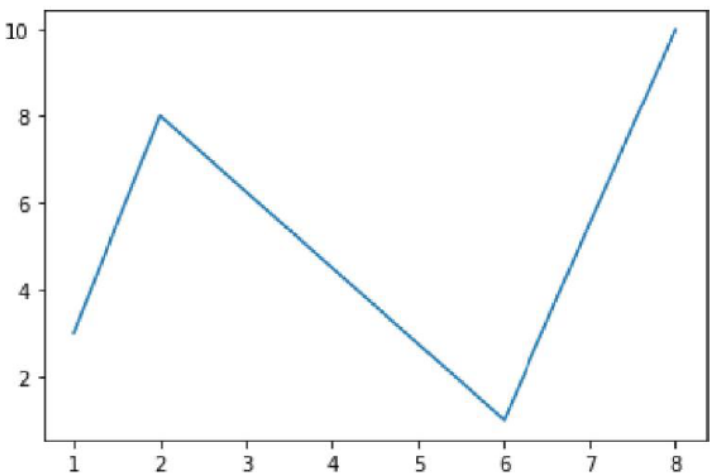
In [12]:
```
#Example
#Draw two lines by specifying a plt.plot() function for each line:

import matplotlib.pyplot as plt
import numpy as np

y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])

plt.plot(y1)
plt.plot(y2)

plt.show()
```



In [3]:
```
#Adding two lines in same graph
import matplotlib.pyplot as plt
import numpy as np
x1=np.array([0,6])
y1=np.array([0,6])
y2=np.array([0,2])
plt.xlabel('X-axis',fontsize=20)
plt.ylabel('Y-axis',fontsize=20)
plt.title('Line plot',fontsize=20,fontweight="bold")
plt.plot(x1,y1,color="red",linestyle=":",linewidth=3)
plt.plot(x1,y2,color="green",linestyle="--",linewidth=3)
plt.show()
```



In [4]:
```
#Multiple points:
#Draw a line in a diagram from position (1, 3) to (2, 8)
#then to (6, 1) and finally to position (8, 10):

import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

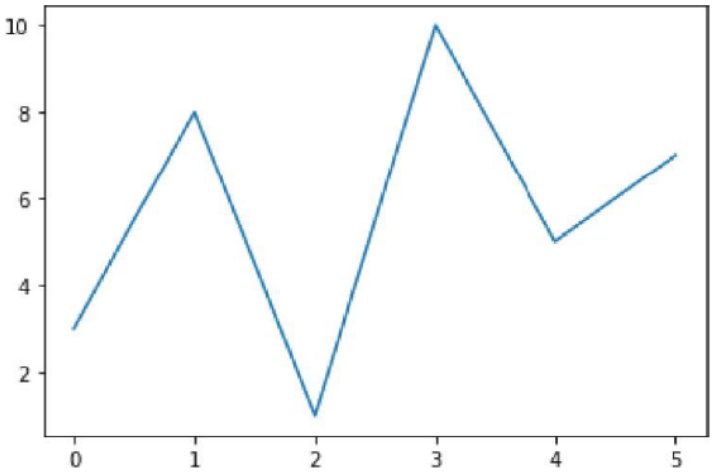


Default X-Points:

If we do not specify the points in the x-axis, they will get the default values 0, 1, 2, 3, (etc. depending on the length of the y-points.

So, if we take the same example as above, and leave out the x-points, the diagram will look like this:

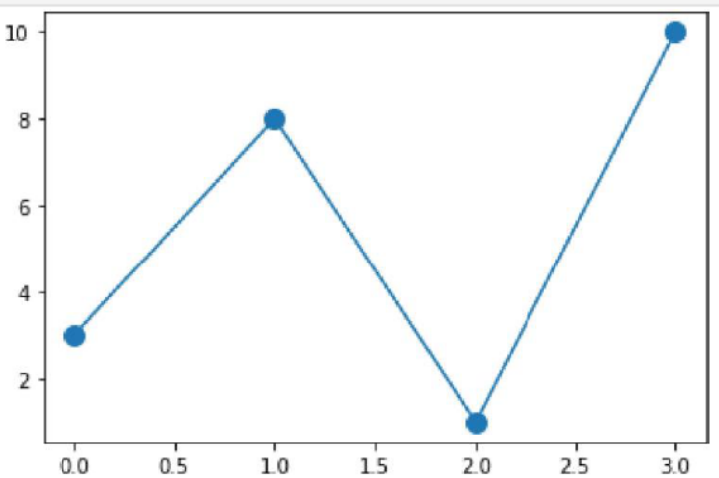
In [5]:
```
#Example
#Plotting without x-points:

import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints)
plt.show()
```



Markers:

You can use the keyword argument marker to emphasize each point with a specified marker: Marker Size You can use the keyword argument markersize or the shorter version, ms to set the size of the markers:

In [10]:
```
#Example
#Mark each point with a circle:

import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms=10)
plt.show()
```



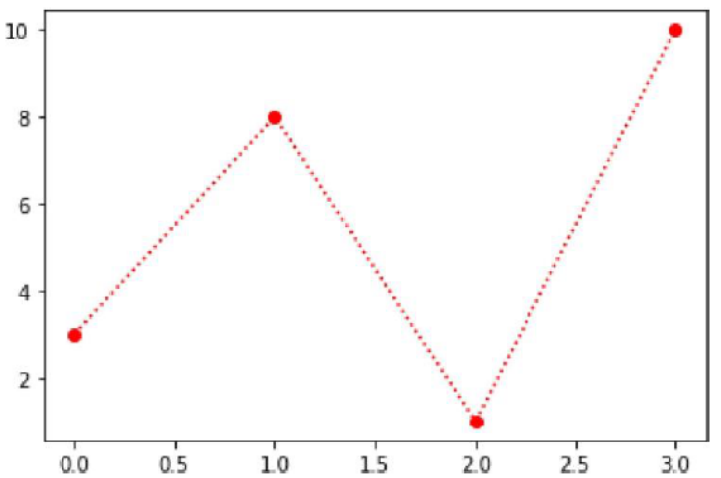
Format Strings fmt:-

You can use also use the shortcut string notation parameter to specify the marker.

This parameter is also called fmt, and is written with this syntax:

marker|line|color

In [7]:
```
#Example
#Mark each point with a circle:

import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, 'o:r')
plt.show()
```



Color Reference:-

'r' Red 'g' Green

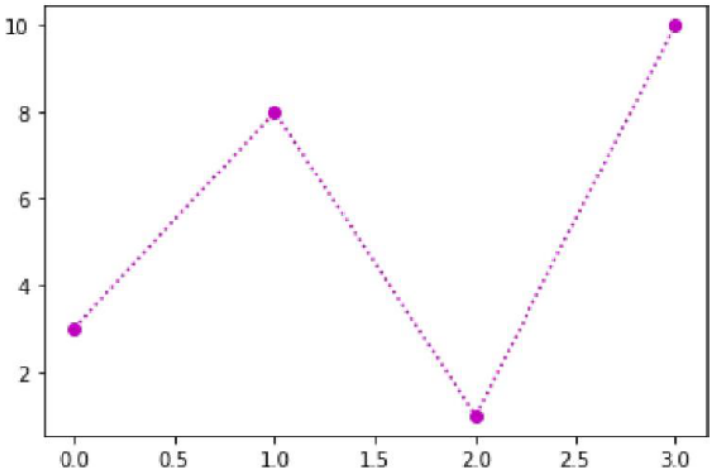
'b' Blue

'c' Cyan

'm' Magenta 'y' Yellow
'k' Black

'w' White

In [8]:
```
#Example
#Mark each point with a circle:

import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, 'o:m')
plt.show()
```



The subplot() Function:-

The subplot() function takes three arguments that describes the layout of the figure.

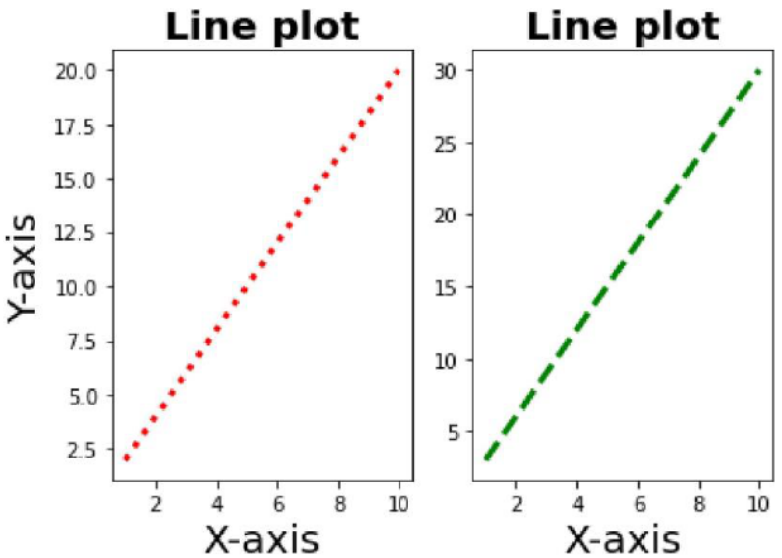
The layout is organized in rows and columns, which are represented by the first and second argument.

The third argument represents the index of the current plot.

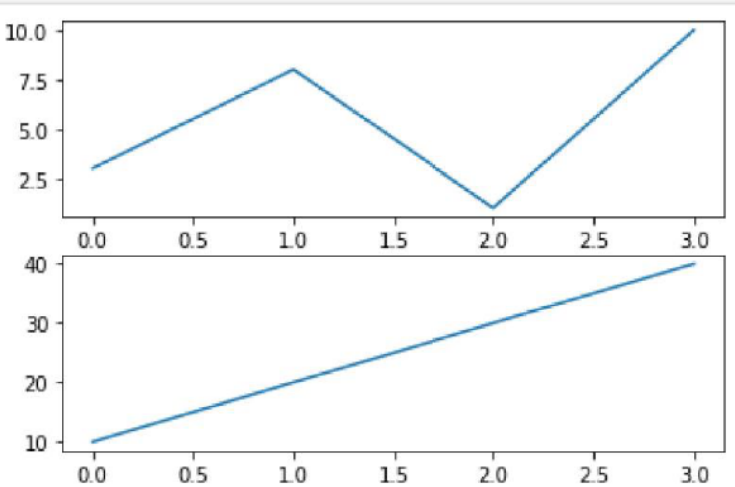
plt.subplot(1, 2, 1) the figure has 1 row, 2 columns, and this plot is the first plot.

plt.subplot(1, 2, 2) the figure has 1 row, 2 columns, and this plot is the second plot.

In [4]:
```
#Adding two lines in different graph
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(1,11)
y1=2*x
y2=3*x

plt.subplot(1,2,1)
plt.title('Line plot',fontsize=20,fontweight="bold")
plt.xlabel('X-axis',fontsize=20)
plt.ylabel('Y-axis',fontsize=20)
plt.plot(x,y1,color="red",linestyle=":",linewidth=3)

plt.subplot(1,2,2)
plt.title('Line plot',fontsize=20,fontweight="bold")
plt.xlabel('X-axis',fontsize=20)
plt.plot(x,y2,color="green",linestyle="--",linewidth=3)

plt.show()
```

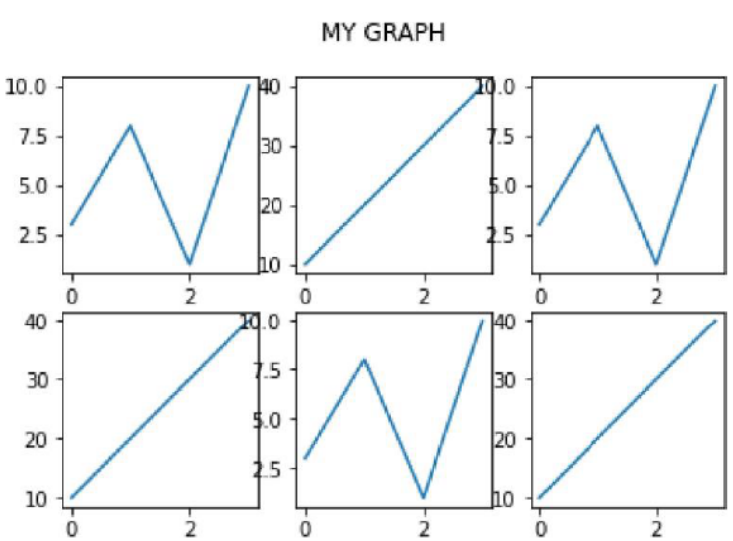


In [17]:
```
#if we want a figure with 2 rows an 1 column
#(meaning that the two plots will be displayed on top of each other instead of side

#Draw 2 plots on top of each other:

import matplotlib.pyplot as plt
import numpy as np

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 1, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 1, 2)
plt.plot(x,y)

plt.show()
```



In [19]:
```
#Draw 6 plots:

import matplotlib.pyplot as plt
import numpy as np
```

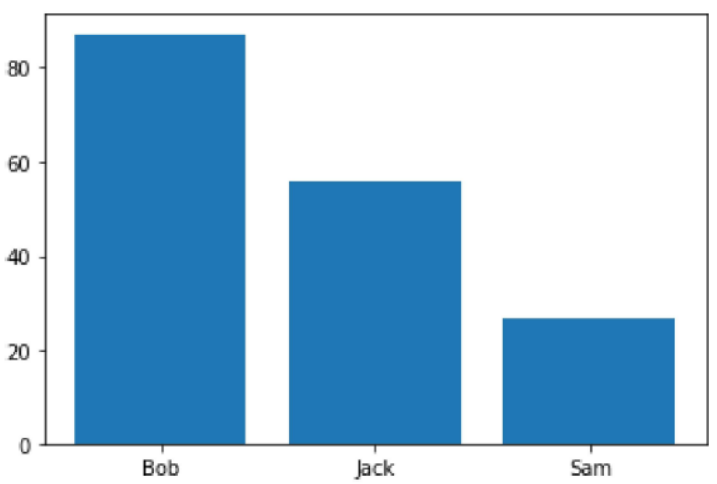
```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 1)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 2)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 3)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 4)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 5)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 6)
plt.plot(x,y)
plt.suptitle("MY GRAPH")
plt.show()
```



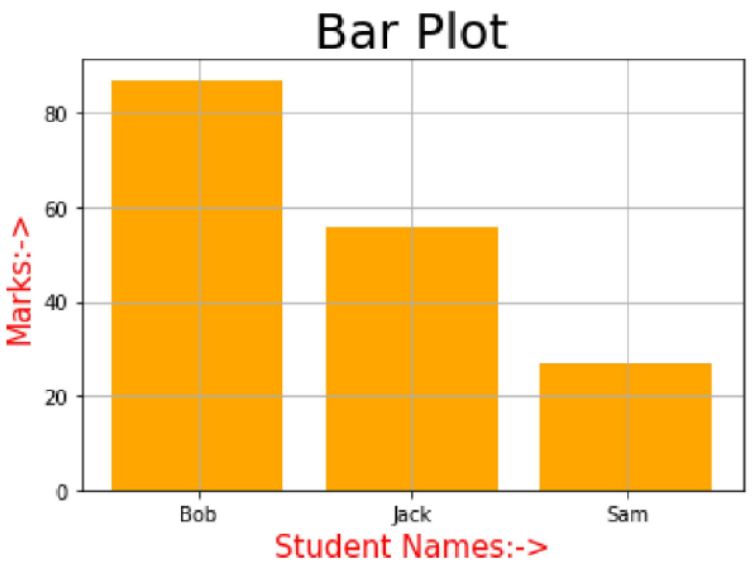
Bar Plot:-

The bar plot or bar chart is usually a graph/chart that is mainly used to represent the category of data with rectangular bars with lengths and heights that are proportional to the values which they represent. You can plot these bars either vertically or horizontally. We use the bar plot to mainly show the comparison between discrete categories of data. In the bar plot, One axis of the plot is used to represent the specific categories being compared, while the other axis usually represents the measured values corresponding to those categories.

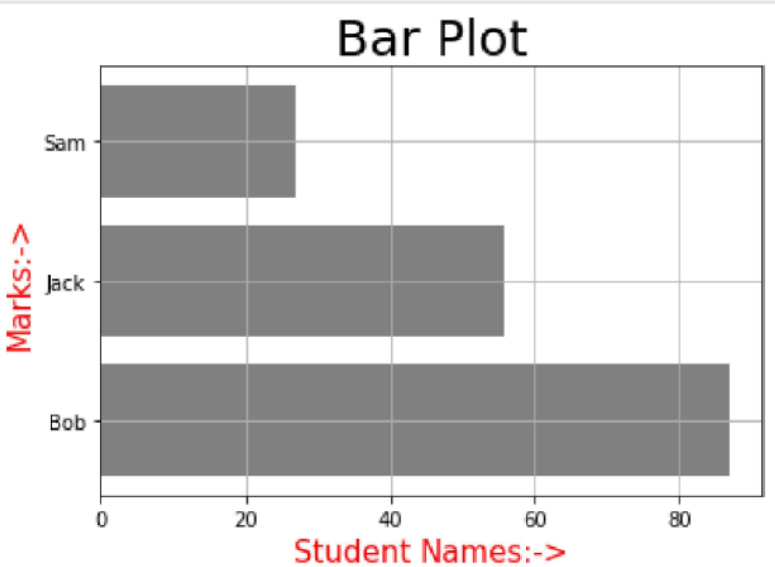
```
#Bar plot
import matplotlib.pyplot as plt
student={"Bob":87,"Jack":56,"Sam":27}
names=list(student.keys())
values=list(student.values())

plt.bar(names,values)
plt.show()
```



In [8]:
```
#Bar plot
import matplotlib.pyplot as plt
student={"Bob":87,"Jack":56,"Sam":27}
names=list(student.keys())
values=list(student.values())

plt.bar(names,values,color="orange")
plt.title("Bar Plot",fontsize=25)
plt.xlabel("Student Names:->",fontsize=15,color="red")
plt.ylabel("Marks:->",fontsize=15,color="red")
plt.grid(True)

plt.show()
```



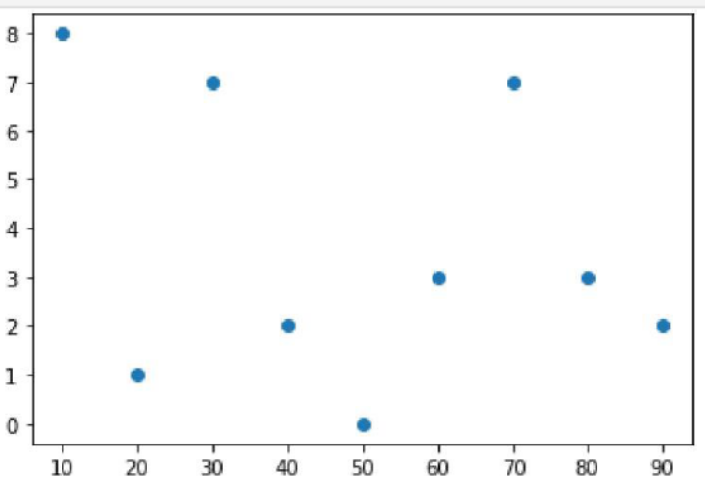
In [9]:
```
#Horizontal Bar plot
import matplotlib.pyplot as plt
student={"Bob":87,"Jack":56,"Sam":27}
names=list(student.keys())
values=list(student.values())

plt.barh(names,values,color="grey")
plt.title("Bar Plot",fontsize=25)
plt.xlabel("Student Names:->",fontsize=15,color="red")
plt.ylabel("Marks:->",fontsize=15,color="red")
plt.grid(True)
plt.show()
```



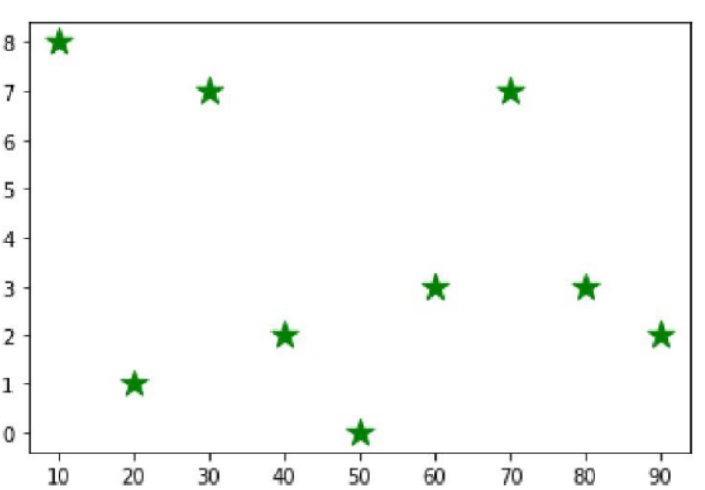
Scatter Plot:-

Scatter plots are used to plot data points on horizontal and vertical axis in the attempt to show how much one variable is affected by another. The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

In [10]:
```
#Scatter Plot
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]

plt.scatter(x,a)
plt.show()
```

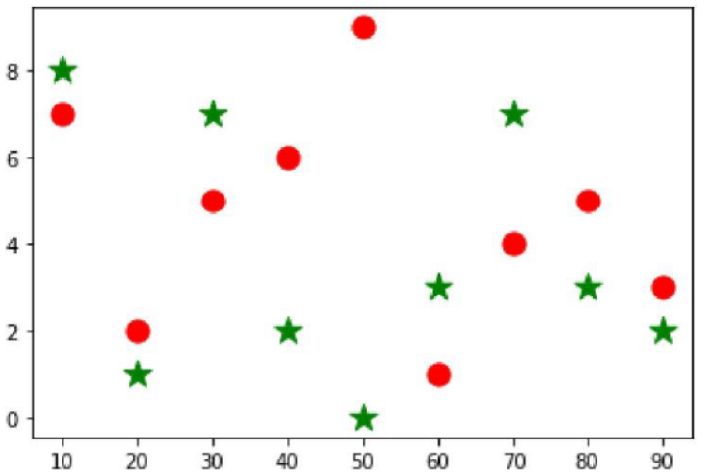


In [11]:
```
#Scatter Plot
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]

plt.scatter(x,a,marker="*",color="green",s=200)
plt.show()
```

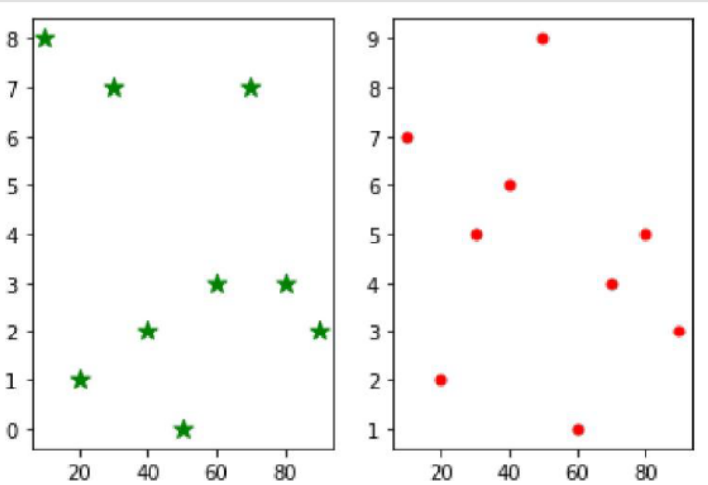
In [12]:



```
#Scatter Plot
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]
b=[7,2,5,6,9,1,4,5,3]

plt.scatter(x,a,marker="*",color="green",s=200)
```

```
plt.scatter(x,b,marker=".",color="red",s=500)

plt.show()
```



In [13]:
```
#Adding subplot
#Scatter Plot
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]
b=[7,2,5,6,9,1,4,5,3]

plt.subplot(1,2,1)
plt.scatter(x,a,marker="*",color="green",s=100)

plt.subplot(1,2,2)
plt.scatter(x,b,marker=".",color="red",s=100)
plt.show()
```



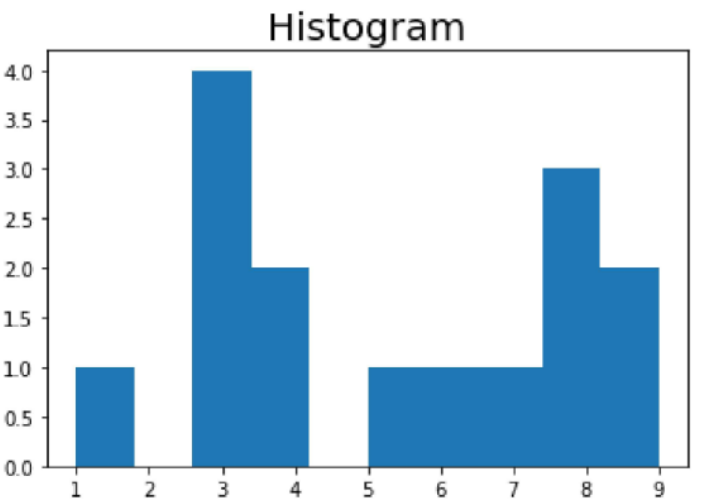
Histogram:-

A histogram is an accurate representation of the distribution of numerical data. So Histogram is a type of bar graph and it was invented by Karl Pearson The Histogram is mainly used to represent the data that is provided in some groups. Histograms usually consist of bins of data,where each bin consists of minimum and maximum values. To estimate the probability distribution of the continuous variable, histogram is used.

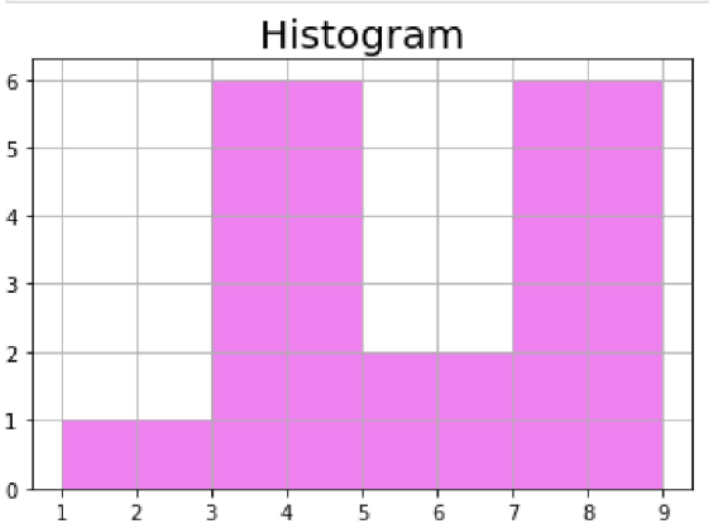
In [14]:

```
#Histogram
data=[1,3,3,3,3,9,9,5,4,4,8,8,8,6,7]
```

```
plt.hist(data)
plt.title("Histogram",fontsize=20)
plt.show()
```



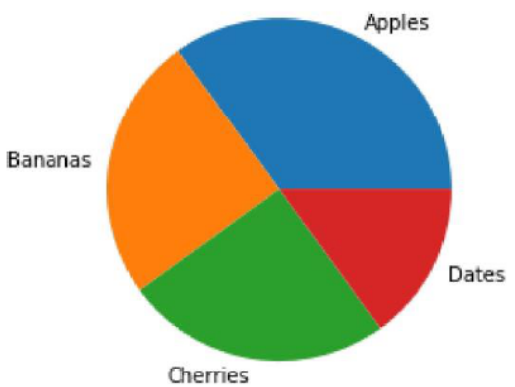

```
#Histogram
data=[1,3,3,3,3,9,9,5,4,4,8,8,8,6,7]

plt.hist(data,bins=4,color="violet")
plt.grid(True)
plt.title('Histogram',fontsize=20)
plt.show()
```



Pie Chart:-

A Pie Chart can only display one series of data. Pie charts show the size of items (called wedge) in one data series, proportional to the sum of the items. The data points in a pie chart are shown as a percentage of the whole pie. You can use the method plt.pie() to create a Pie Chart.

In [20]:




```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.show()
```

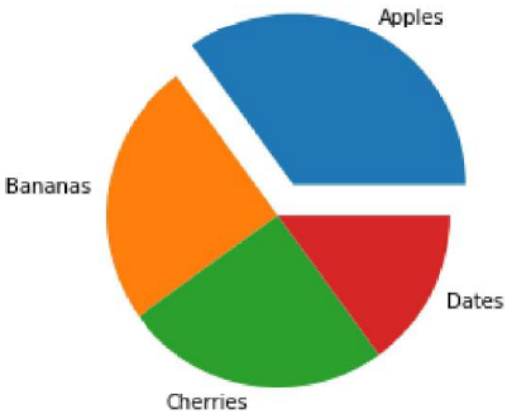
Explode:-

Maybe you want one of the wedges to stand out? The explode parameter allows you to do that.

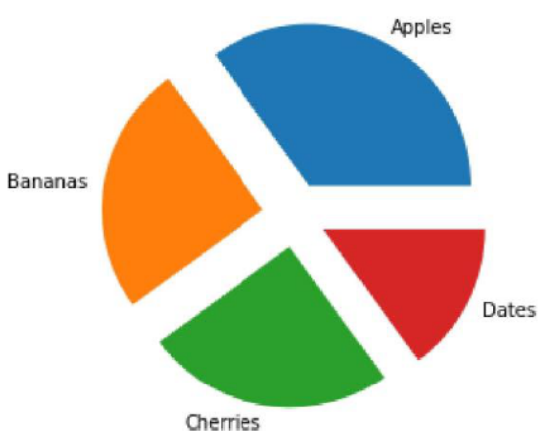
```
#Example
#Pull the "Apples" wedge 0.2 from the center of the pie:

import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]

plt.pie(y, labels = mylabels, explode = myexplode)
plt.show()
```

The explode parameter, if specified, and not None, must be an array with one value for each wedge.

Each value represents how far from the center each wedge is displayed:




```
#Pull the "All" wedge 0.2 from the center of the pie:

import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0.2, 0.2, 0.2]

plt.pie(y, labels = mylabels, explode = myexplode)
plt.show()
```




```
#Pie chart
fruit=['Apple','Orange','Mango','Guava']
```

```
no=[67,34,100,29]

plt.pie(no,labels=fruit,autopct='%0.1f%%',colors=['yellow','grey','blue','brown'])
plt.show()
```