In [ ]:
```
                          "Python Matplotlib "
```

In [ ]:
```
"MatplotLib"
```

Matplotlib is a low level of graph plotting library in Python that serves as a visualisation utility.

Matplotlib was created by john D.hunter.

Matplotlib is a open source and we can use freely.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

In [ ]:
```
"Installation of MatplotLib"
```

In [ ]:
```
we have already Python and pip installed in our system,then installation of Matplotlib is very easy.

Install it using this command.
```

In [2]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\gadam\anaconda3\lib\site-packages (3.8.4)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\gadam\anaconda3\lib\site-packages (from matplotlib)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\gadam\anaconda3\lib\site-packages (from matplotlib) (0.1
1.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\gadam\anaconda3\lib\site-packages (from matplotlib)
(4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\gadam\anaconda3\lib\site-packages (from matplotlib)
(1.4.4)
Requirement already satisfied: numpy>=1.21 in c:\users\gadam\anaconda3\lib\site-packages (from matplotlib) (1.26
.4)
Requirement already satisfied: packaging>=20.0 in c:\users\gadam\anaconda3\lib\site-packages (from matplotlib) (
23.2)
Requirement already satisfied: pillow>=8 in c:\users\gadam\anaconda3\lib\site-packages (from matplotlib) (10.3.0
)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\gadam\anaconda3\lib\site-packages (from matplotlib)
(3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\gadam\anaconda3\lib\site-packages (from matplotl
ib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\gadam\anaconda3\lib\site-packages (from python-dateutil>=2.7
->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
In this anaconda,already every library is inbuilt .
```

In [ ]:
```
"Import MatplotLib"
```

Once Matplot is installed ,import it in your applications by adding the 'import module' statement.

In [5]:
```
import matplotlib
```

In [ ]:
```
"checking Matplotlib Version"
```

In [ ]:
```
The version string is stored under __version__ attribute.
```

In [7]:
```
import matplotlib
print(matplotlib.__version__)
```

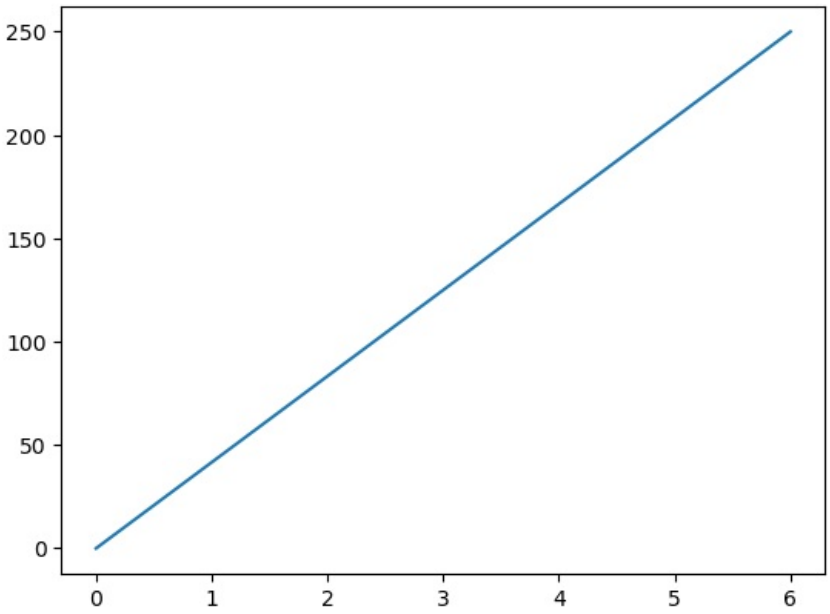
```
3.8.4
```

In [ ]:
```
"Pyplot"
```

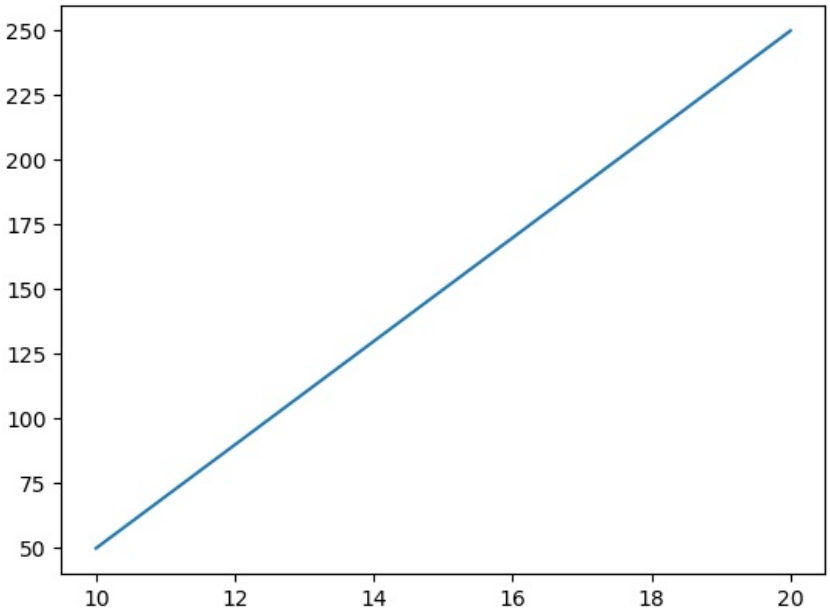
Most of the matplotlib utilities lies under the 'pyplot' submodule ,and are useally imported under the plt alis.

In [10]:
```
import matplotlib.pyplot as plt
```

In [14]:
```
# Example:
# Draw a line in a daigram from position (0,0) to (6,250)
import matplotlib.pyplot as plt
import numpy as np
xpoints=np.array([0,6])
ypoints=np.array([0,250])
plt.plot(xpoints,ypoints)
plt.show()
```



In [16]:
```
import matplotlib.pyplot as plt
import numpy as np
xpoints=np.array([10,20])
ypoints=np.array([50,250])
plt.plot(xpoints,ypoints)
plt.show()
```



In [ ]:
```
"Plotting x and y points"
```

The 'plot()' function is used to draw points(markers) in a daigram.

By default ,plot() function draws a line from point to point.

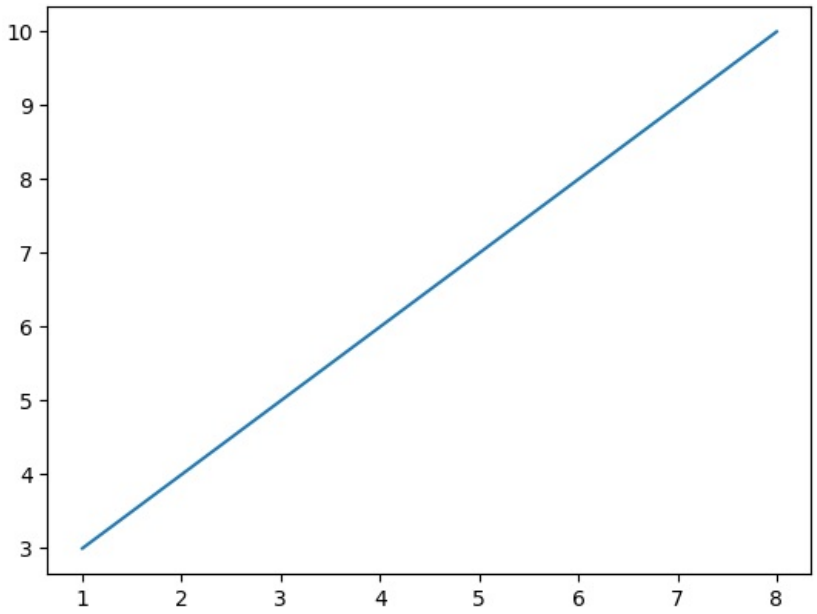
The function takes parameters for specifying points in the daigram

Parameter 1 is an array containing the points on the x-axis

Parameters 2 is an array contaning the points on the y-axis.

If we ned to plot a line from (1,3) to (8,10) we have to pass two arrays [1,8] amd [3,10] to the plot funtion.

In [21]:
```
# Draw a line in a diagram from position (1, 3) to position (8, 10):
xpoints=np.array([1,8])
ypoints=np.array([3,10])
plt.plot(xpoints,ypoints)
plt.show()
```

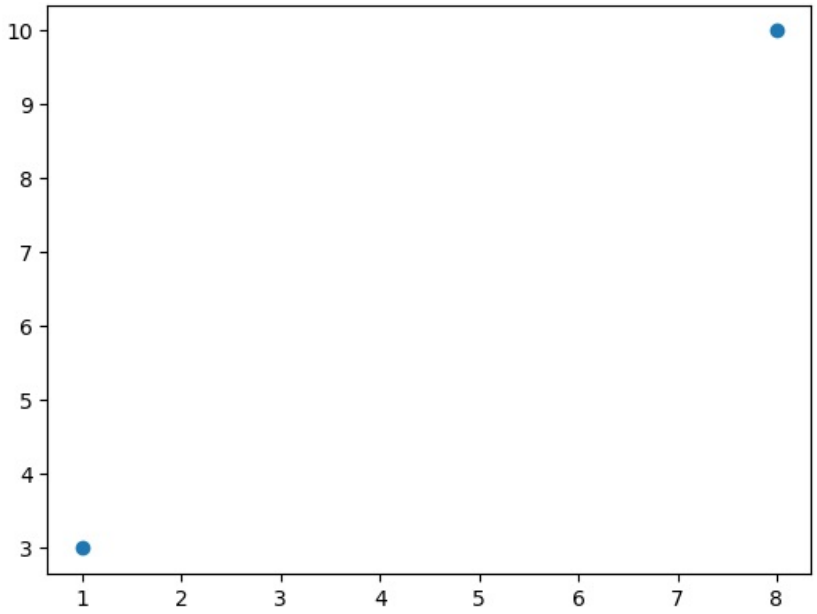


In [ ]:
```
The x-axis is the horizontal axis.
The y-axis is the vertical axis.
```

In [ ]:
```
"Plotting without line"
```

to plot only the markers ,you can use shortcut string notation parameter 'o' which means 'rings'

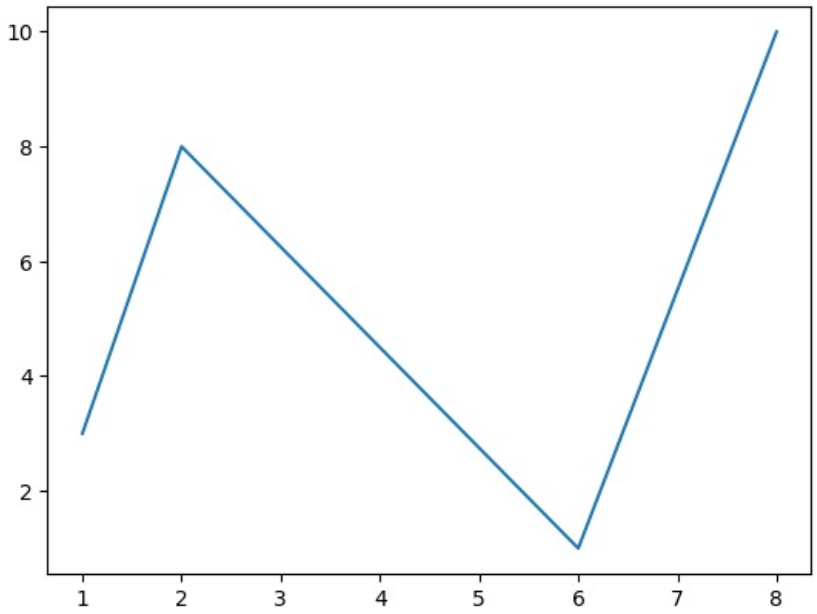
In [28]:
```
#draw the two points in the daigram ,one at position (1,3) and one in position (8,10):
xpoints=np.array([1,8])
ypoints=np.array([3,10])
plt.plot(xpoints,ypoints,'o')
plt.show()
```



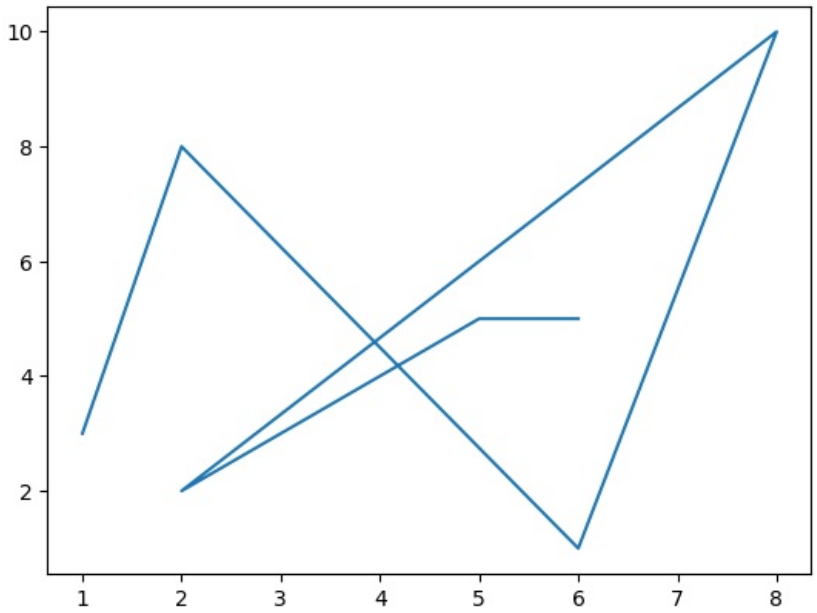
In [ ]:
```
"Multiple Points"
```

we can plot as many points as you like ,just make sure you have the same number of points in the both axis.

In [31]:
```
# Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):
xpoints=np.array([1,2,6,8])
ypoints=np.array([3,8,1,10])
plt.plot(xpoints,ypoints)
plt.show()
```



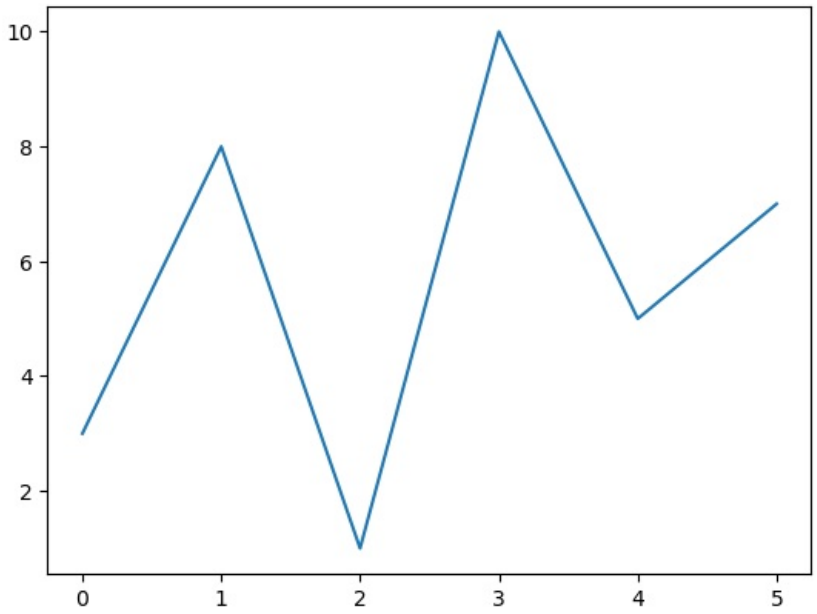
In [37]:
```
xpoints=np.array([1,2,6,8,2,5,6])
ypoints=np.array([3,8,1,10,2,5,5])
plt.plot(xpoints,ypoints)
plt.show()
```



In [ ]:
```
"Default X-points"
```

If we didn.t mention the points on te=he x-axis ,they will get by default values 0,1,2,3,4 etc,depending on the length of the y-points

In [42]:
```
#xpoints=np.array([1,2,6,8,2,5,6])
ypoints=np.array([3,8,1,10,5,7])
plt.plot(ypoints)
plt.show()
```

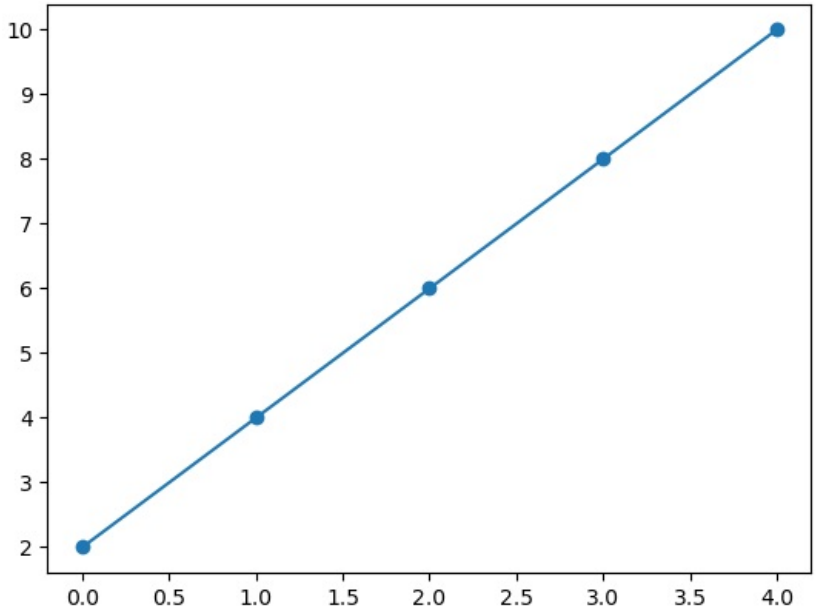


In [ ]:
```
# The x-points in the example above are [0, 1, 2, 3, 4, 5].
```

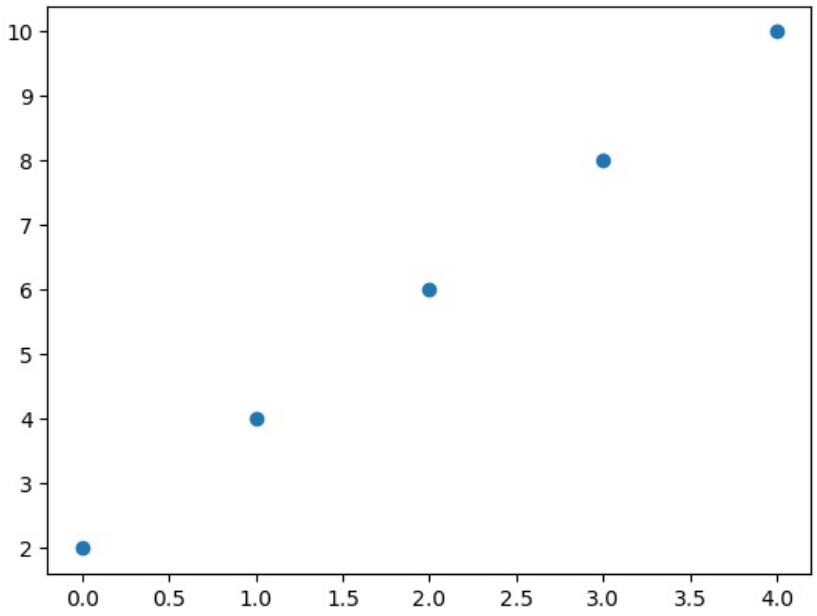
In [ ]:
```
"Matplot Markers"
```

you can use the keyword arguemnt 'marker' to emphasize the each point with a specified marker.

In [49]:
```
ypoints =np.array([2,4,6,8,10])
plt.plot(ypoints,marker='o')
plt.show()
```

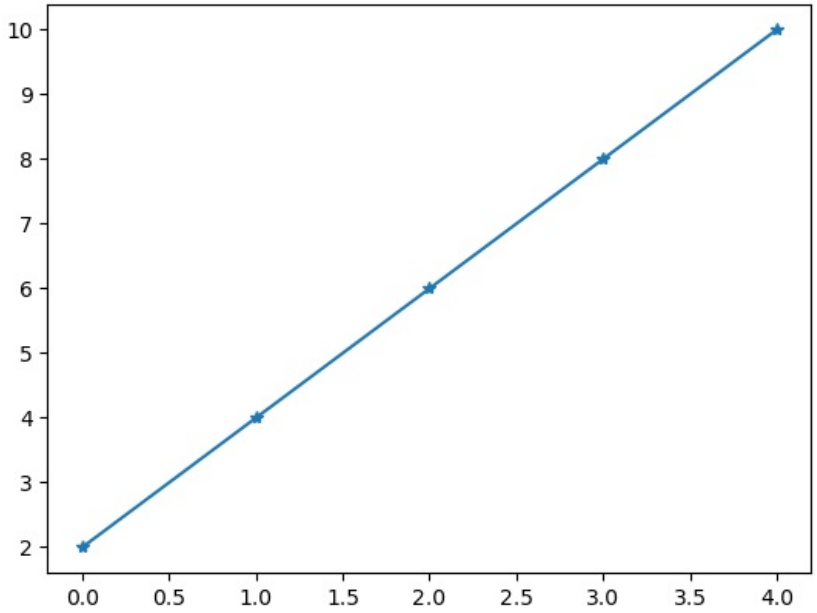


In [51]:
```
ypoints =np.array([2,4,6,8,10])
plt.plot(ypoints,'o')
plt.show()
```



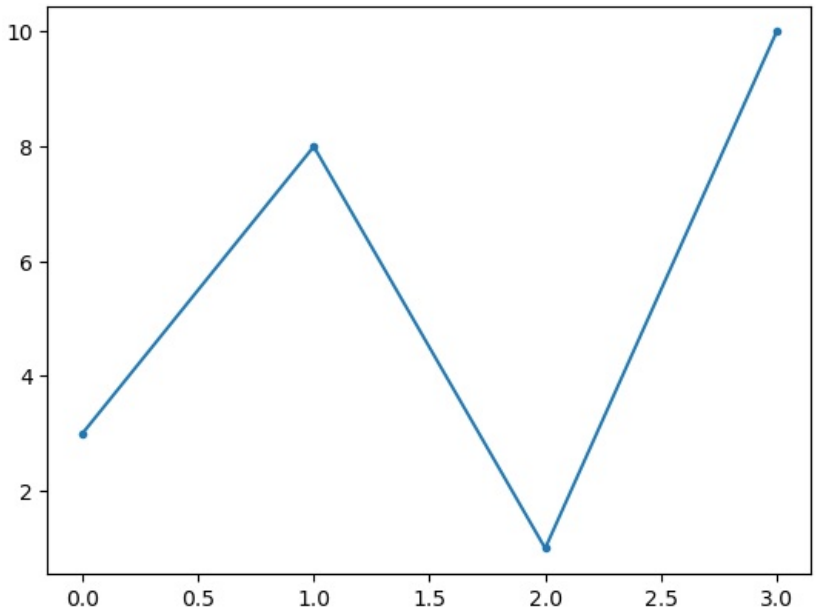
In [ ]:
```
"Mark each point with a star:"
```

In [59]:
```
plt.plot(ypoints,marker='*')
plt.show()
```

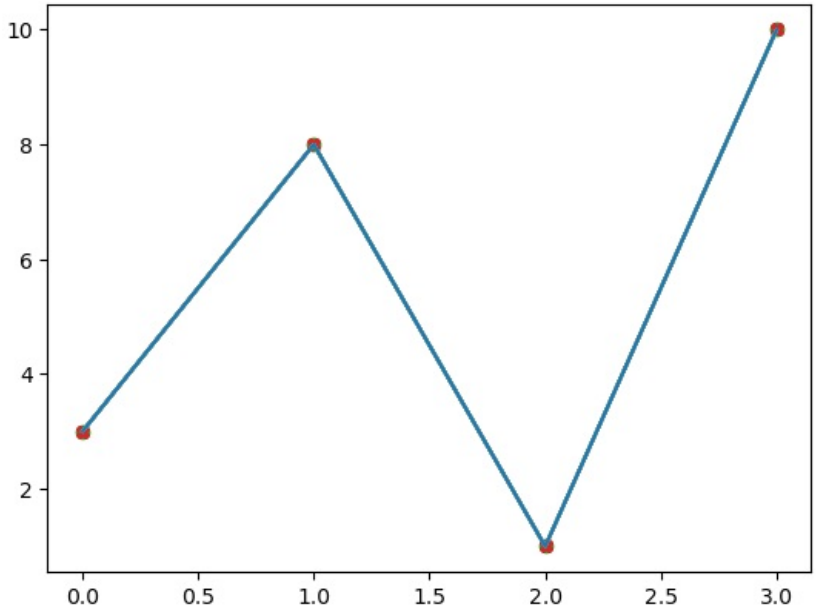


In [ ]:
```
"Marker reference"
```

In [103…
```
1.point
plt.plot(ypoints,marker='.')
plt.show()
```

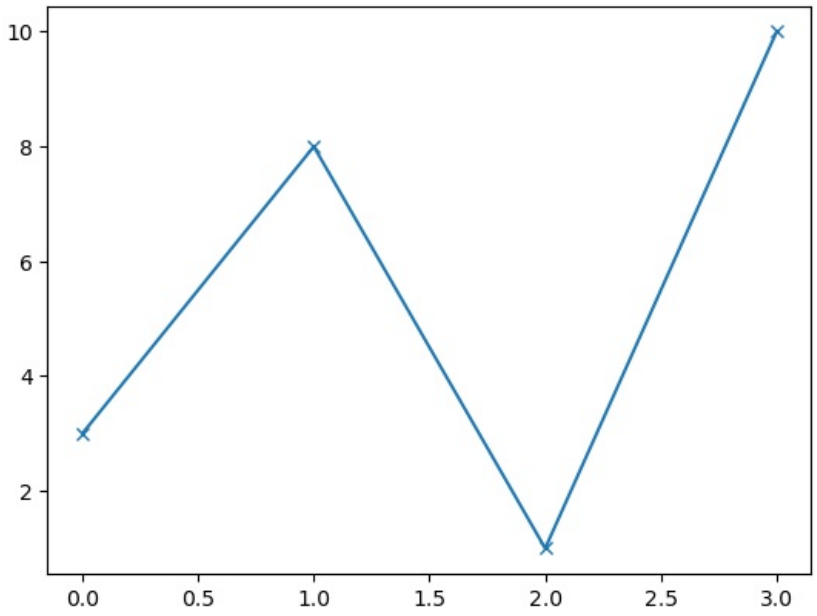


In [98]:
```
# pip install matplotlib
# 2.Pixel
plt.plot(ypoints,marker=',')
plt.show()
```



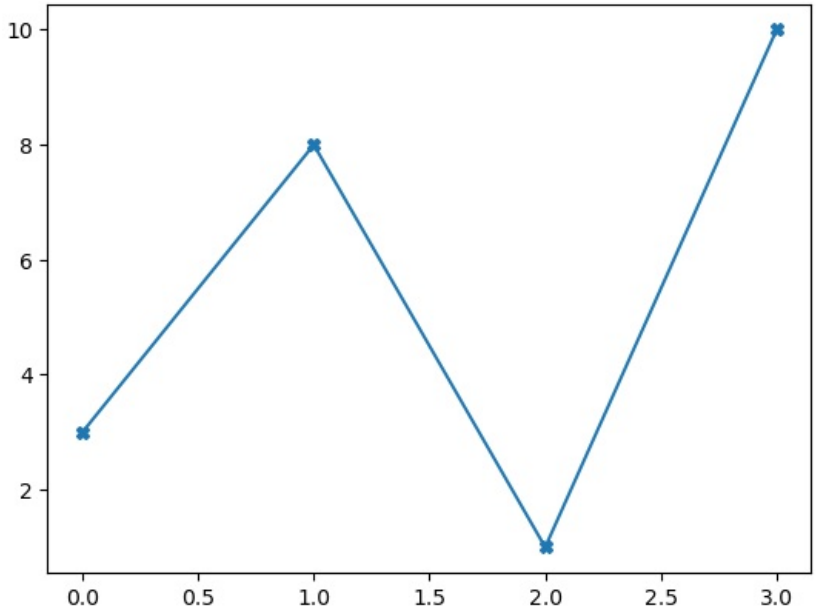
In [96]:
```
%matplotlib inline
```

In [105…
```
3.X
plt.plot(ypoints,marker='x')
plt.show()
```



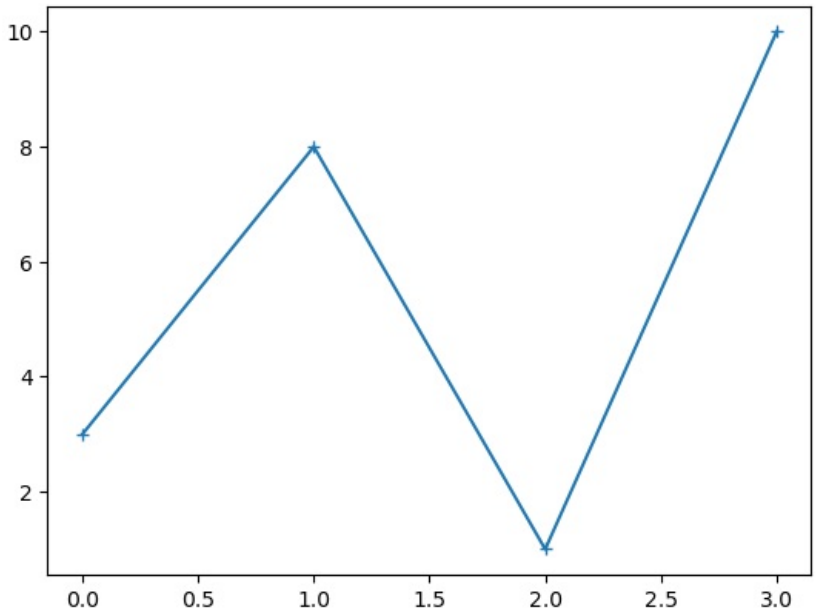
In [107…

```
4.X (filled)
plt.plot(ypoints,marker='X')
plt.show()
```

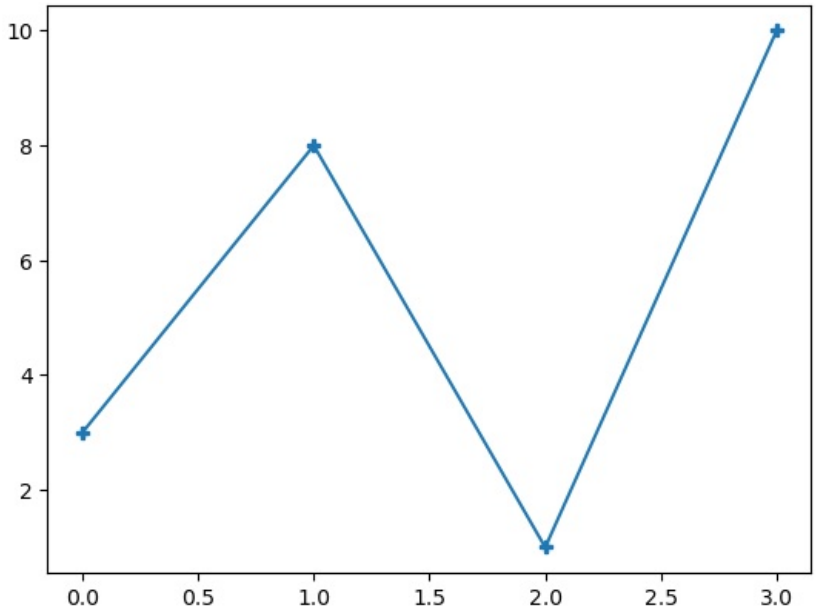


In [111…

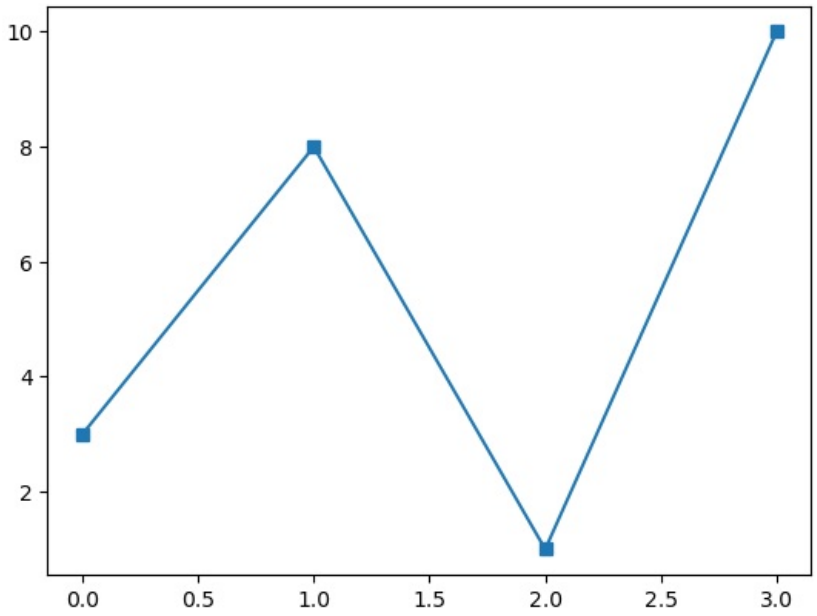
```
 6.plus
plt.plot(ypoints,marker='+')
plt.show()
```



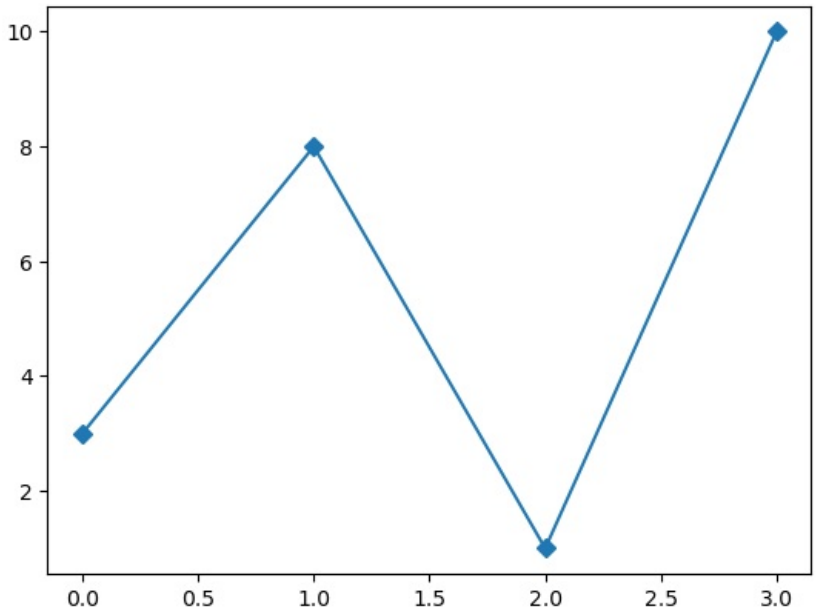
In [113…
```
7.P
plt.plot(ypoints,marker='P')
plt.show()
```



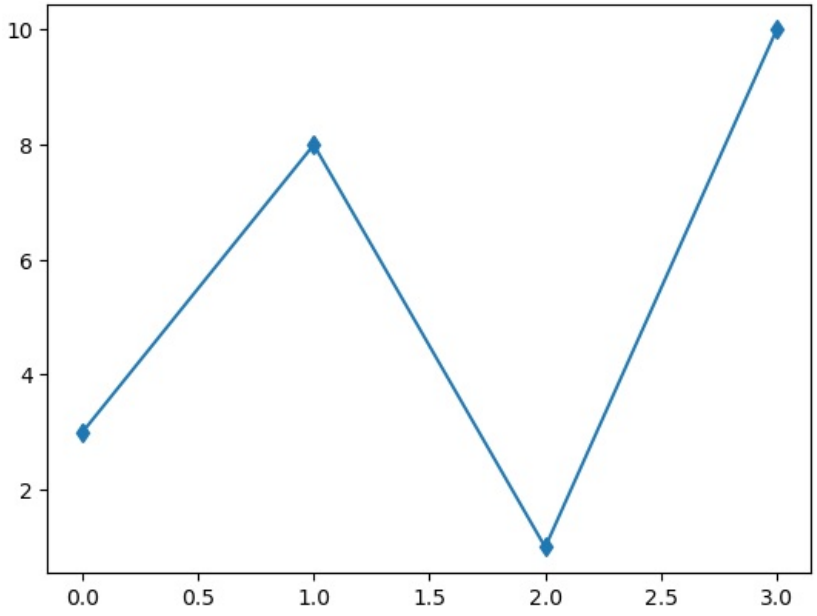
In [115…
```
plt.plot(ypoints,marker='s')
plt.show()
Sqaure
```



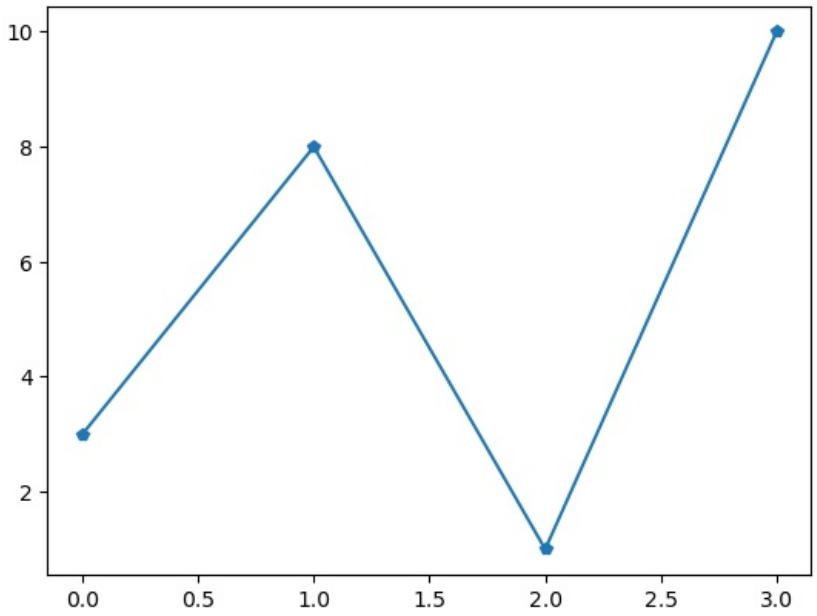
In [117…
```
plt.plot(ypoints,marker='D')
plt.show()
Daimond
```



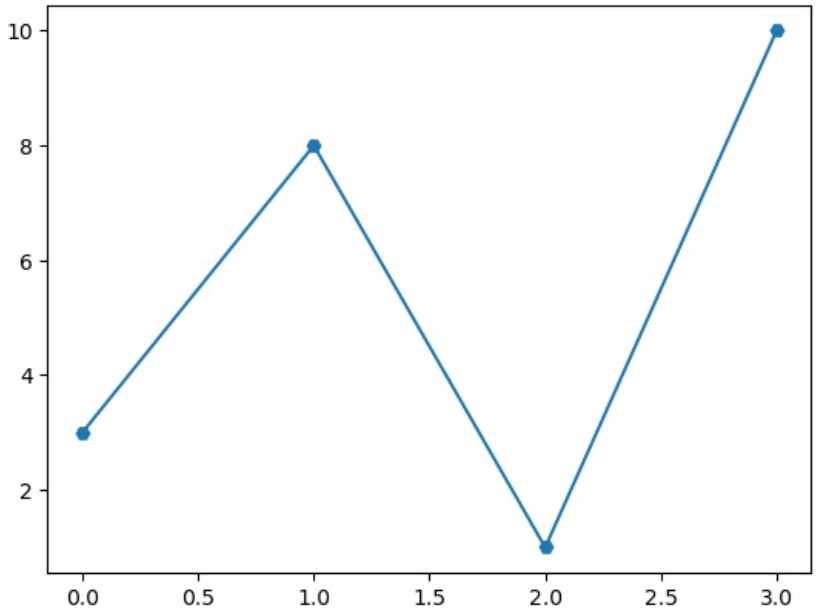
In [119…
```
plt.plot(ypoints,marker='d')
plt.show()
Daimondthin
```



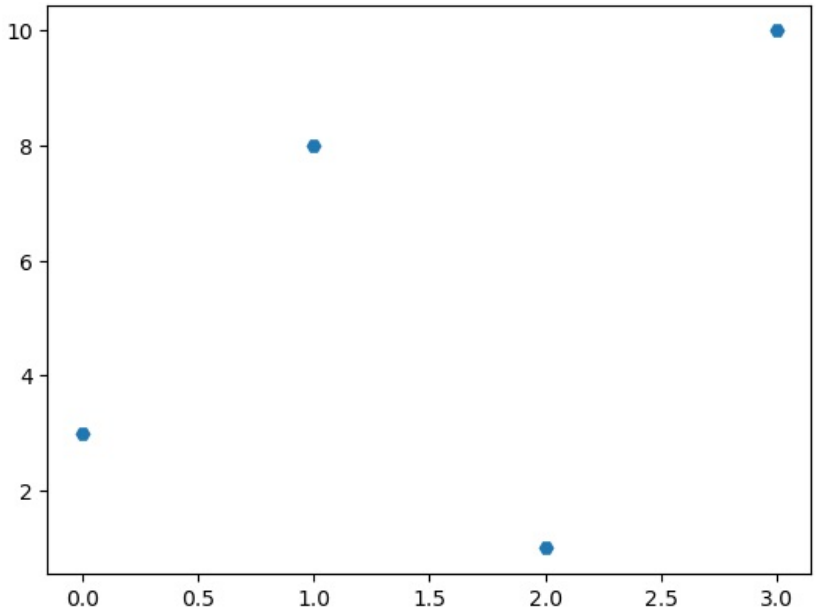
In [121…
```
plt.plot(ypoints,marker='p')
plt.show()
pentagon
```



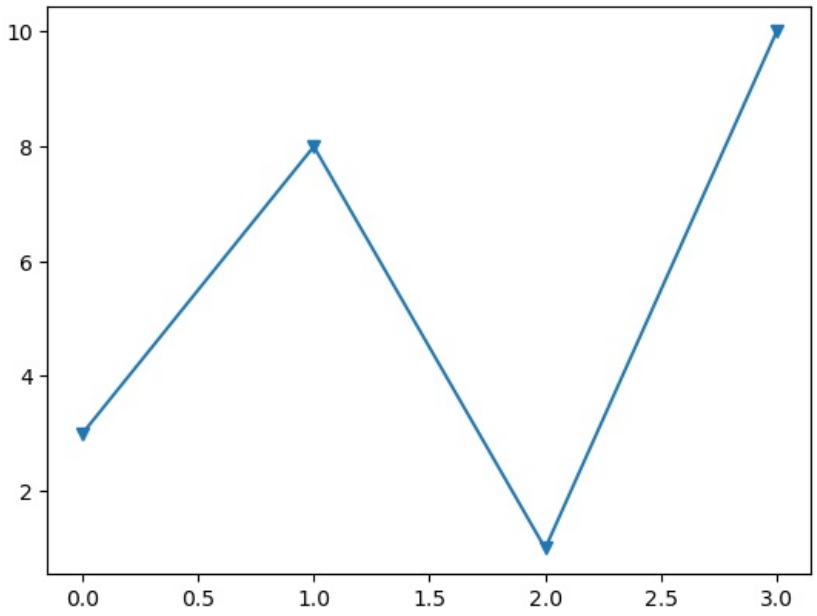
In [125…
```
plt.plot(ypoints,marker='H')
plt.show()
Hexagon
```



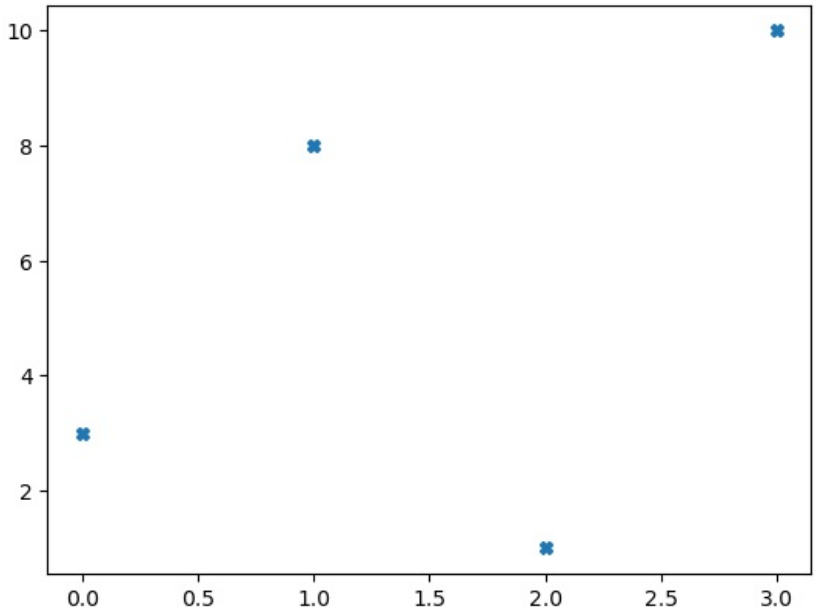
In [127…
```
plt.plot(ypoints,'H')
plt.show()
```



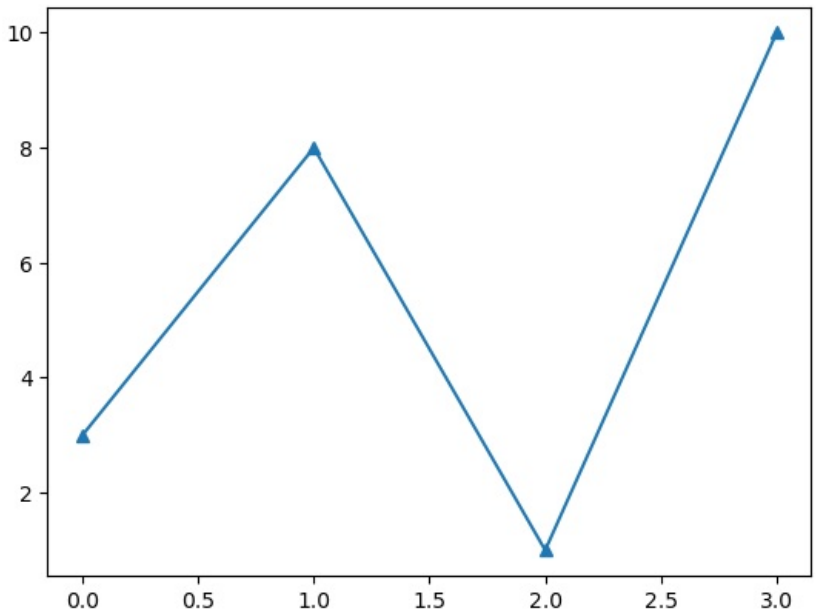
In [129…
```
plt.plot(ypoints,marker='v')
plt.show()
triangle Down
```



In [133…
```
plt.plot(ypoints,'X')
plt.show()
```

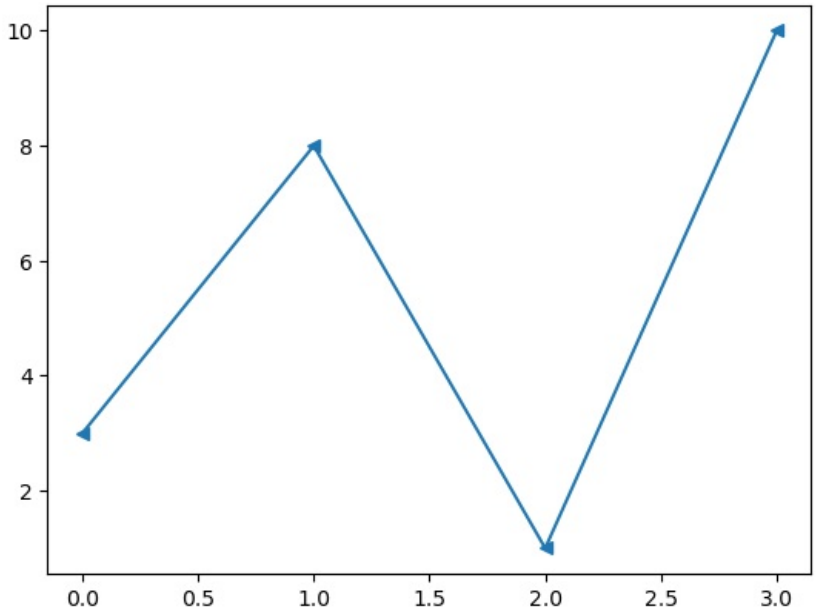


In [135…
```
plt.plot(ypoints,marker='^')
plt.show()
        Triangle Up
```

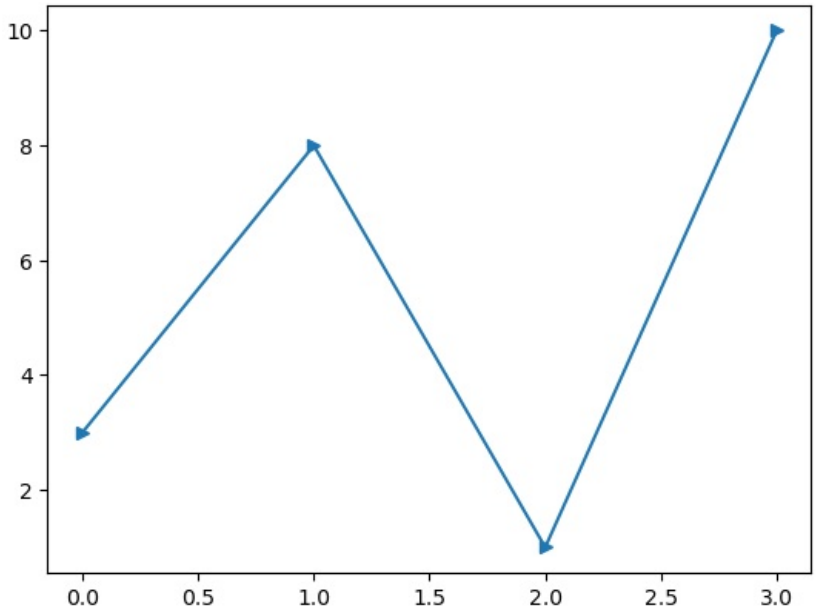


In [137…
```
plt.plot(ypoints,marker='<')
```

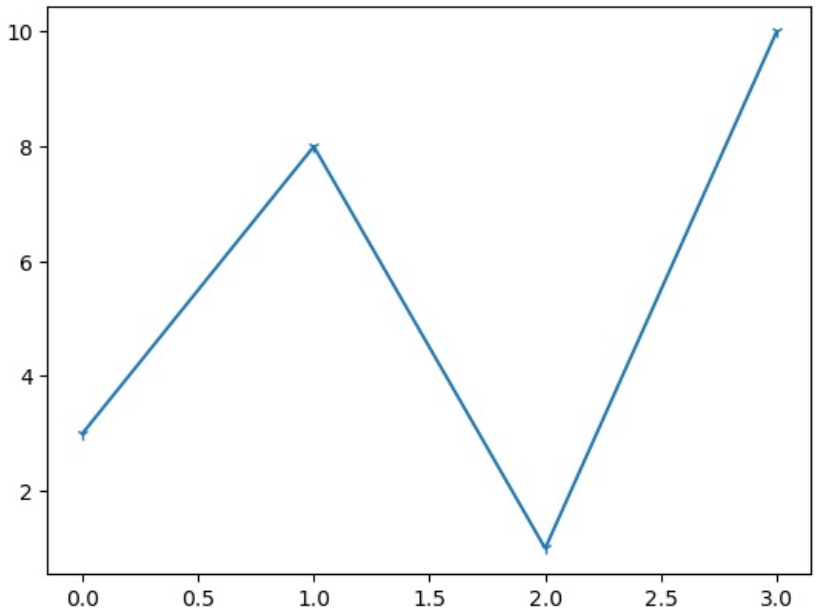
```
plt.show()
Triangle Left
```



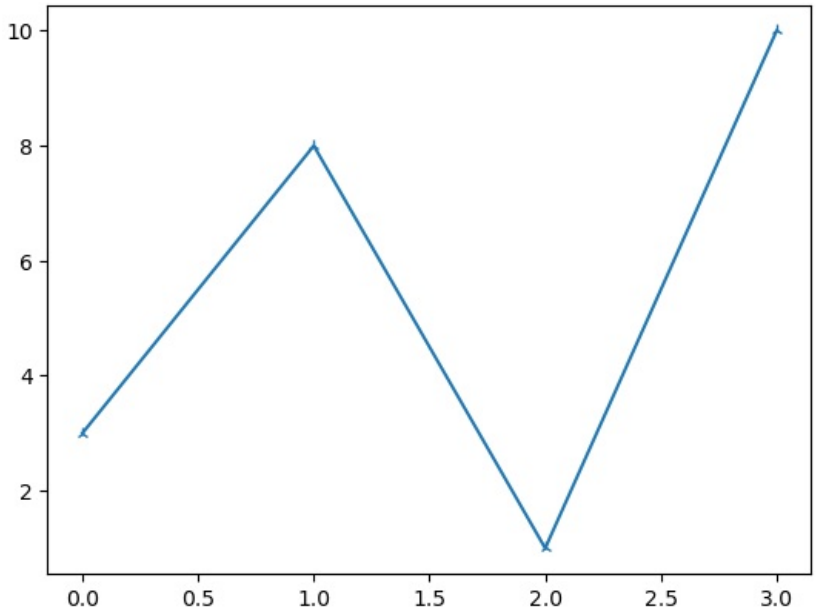
In [141…
```
plt.plot(ypoints,marker='>')
plt.show()
         Triangle Right
```



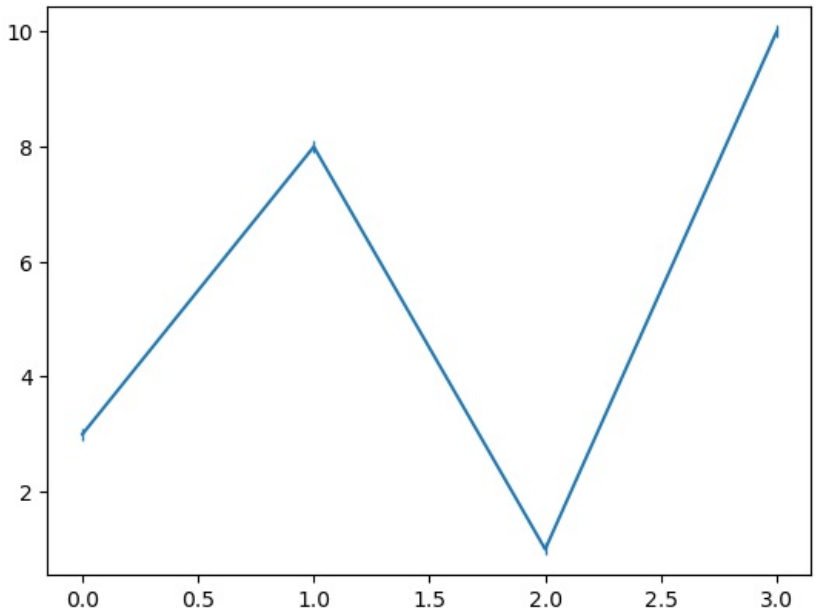
In [143…
```
plt.plot(ypoints,marker='1')
plt.show()
```



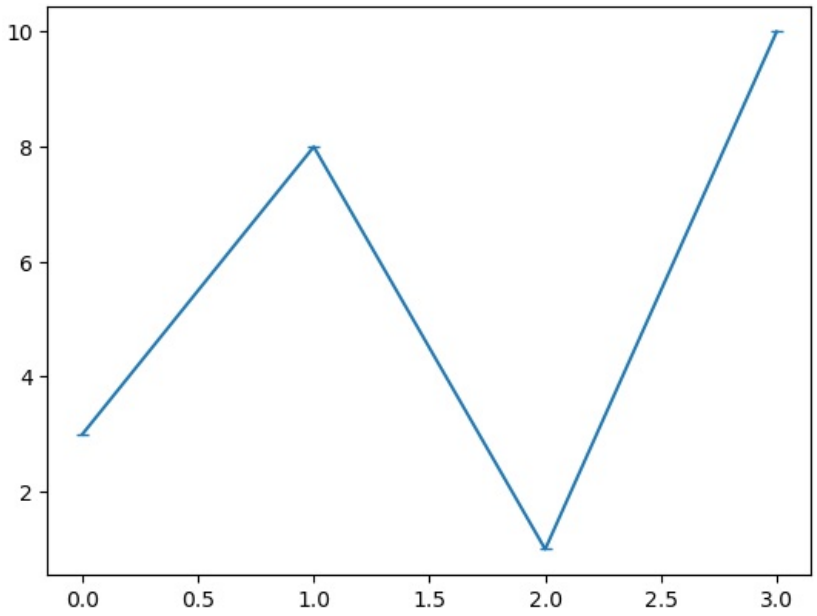
In [145…
```
plt.plot(ypoints,marker='2')
plt.show()
```



In [147…
```
plt.plot(ypoints,marker='|')
plt.show()
```



In [149…
```
plt.plot(ypoints,marker='_')
plt.show()
```



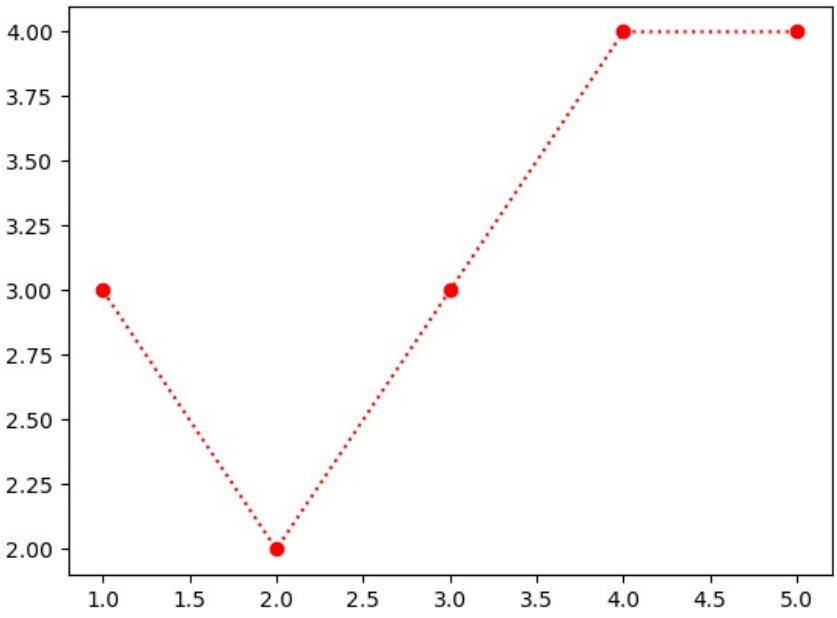
In [ ]:
```
"Format Strings 'fmt' "
```

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

This parameter is also called 'fmt' and is written with this syntax.

In [ ]:
```
marker|line|color
```

In [152…
```
xpoints=np.array([1,2,3,4,5])
ypoints=np.array([3,2,3,4,4])
plt.plot(xpoints,ypoints,'o:r')# circle and red
plt.show()
```

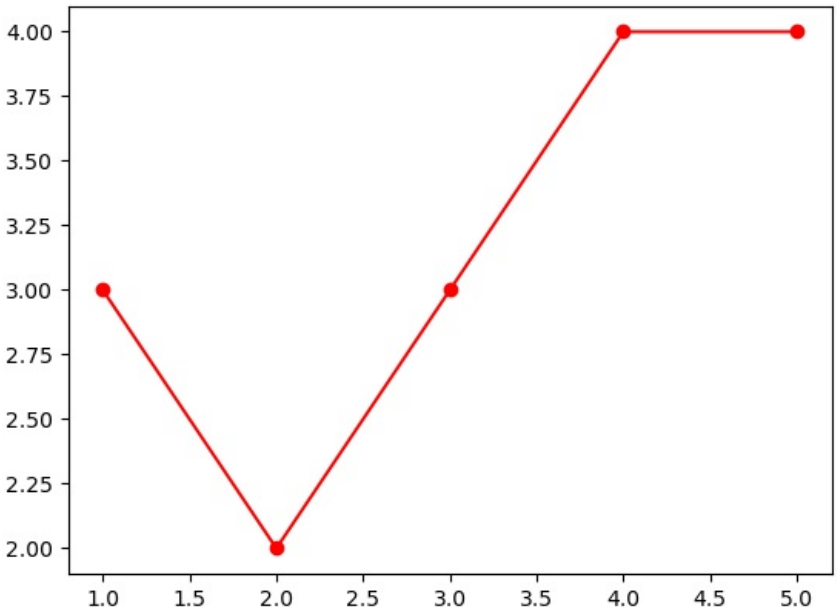


Marker can be any value from the Marker Reference Above.

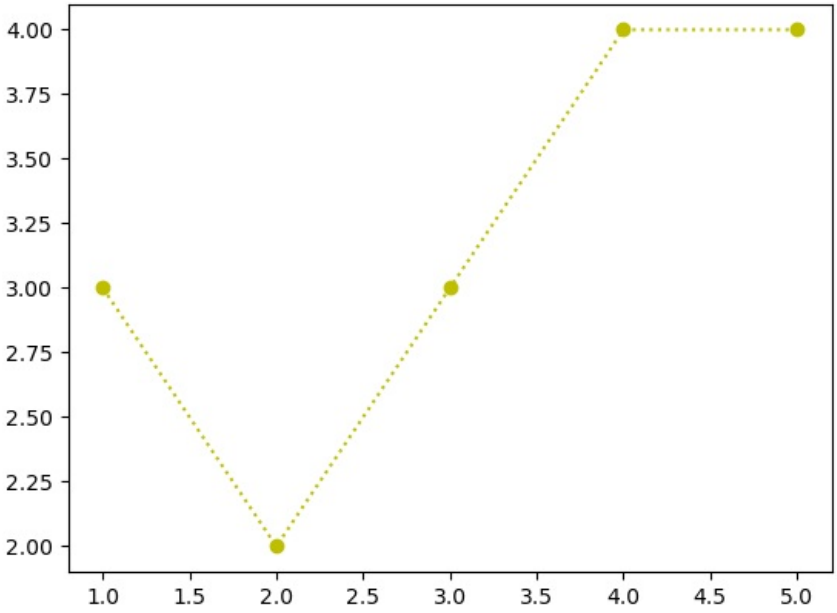
the line value can be one of the following

In [ ]:
```
"Line Reference"
```

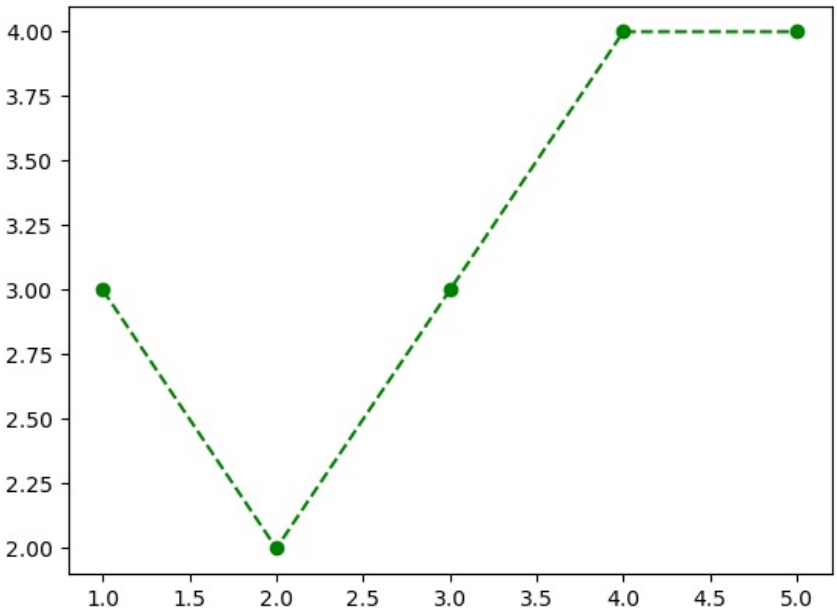
In [173…
```
xpoints=np.array([1,2,3,4,5])
ypoints=np.array([3,2,3,4,4])
plt.plot(xpoints,ypoints,'o-r')# circle and red
plt.show()
```



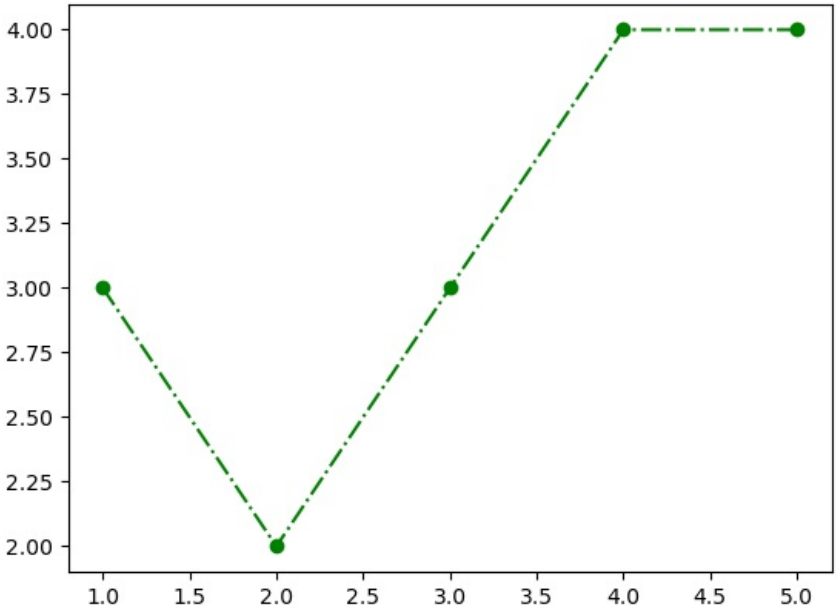
In [177…
```
xpoints=np.array([1,2,3,4,5])
ypoints=np.array([3,2,3,4,4])
plt.plot(xpoints,ypoints,'o:y')# circle and red
plt.show()
```



In [191…
```
xpoints=np.array([1,2,3,4,5])
ypoints=np.array([3,2,3,4,4])
plt.plot(xpoints,ypoints,'o--g')# circle and red
plt.show()
```

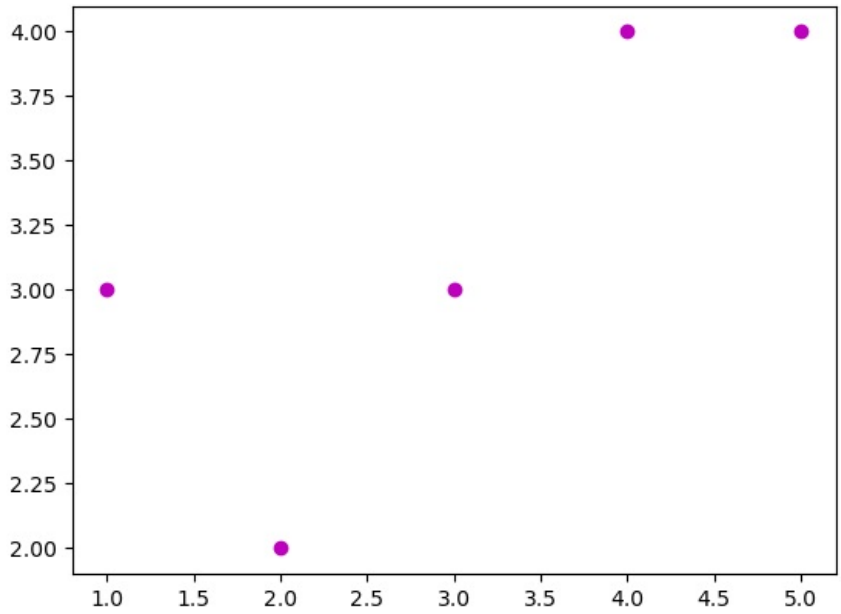


In [200…
```
xpoints=np.array([1,2,3,4,5])
ypoints=np.array([3,2,3,4,4])
plt.plot(xpoints,ypoints,'o-.g')# circle and red
plt.show()
```



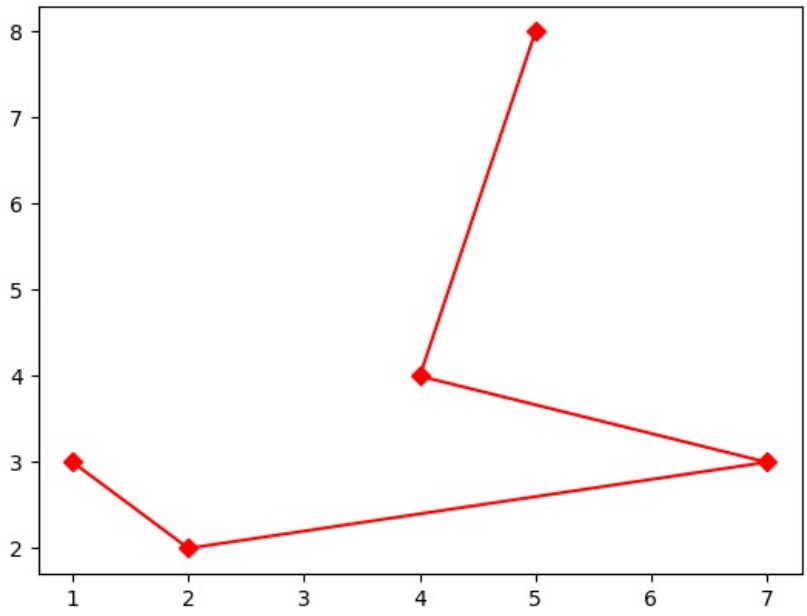
Note: If you leave out the line value in the fmt parameter, no line will be plotted.

In [204…
```
xpoints=np.array([1,2,3,4,5])
ypoints=np.array([3,2,3,4,4])
plt.plot(xpoints,ypoints,'om')# circle and red
plt.show()
```

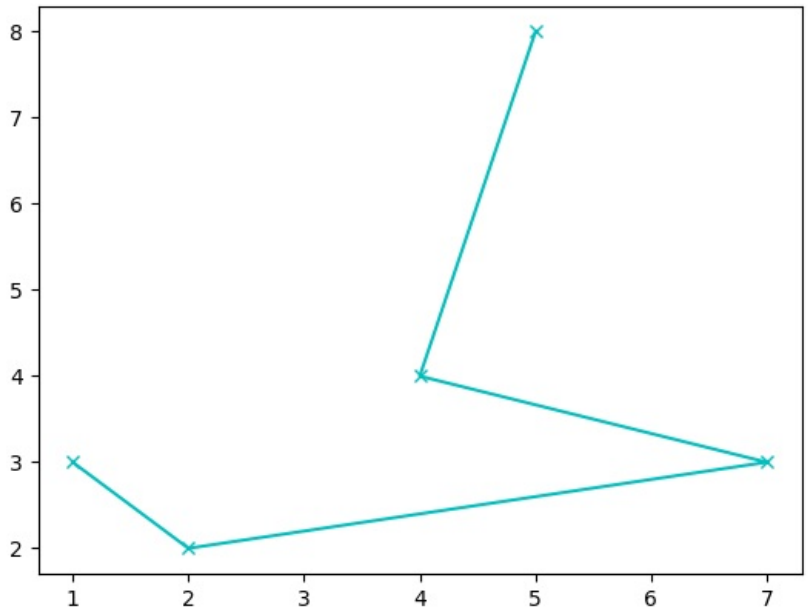


In [ ]:
```
"Color Reference"
```

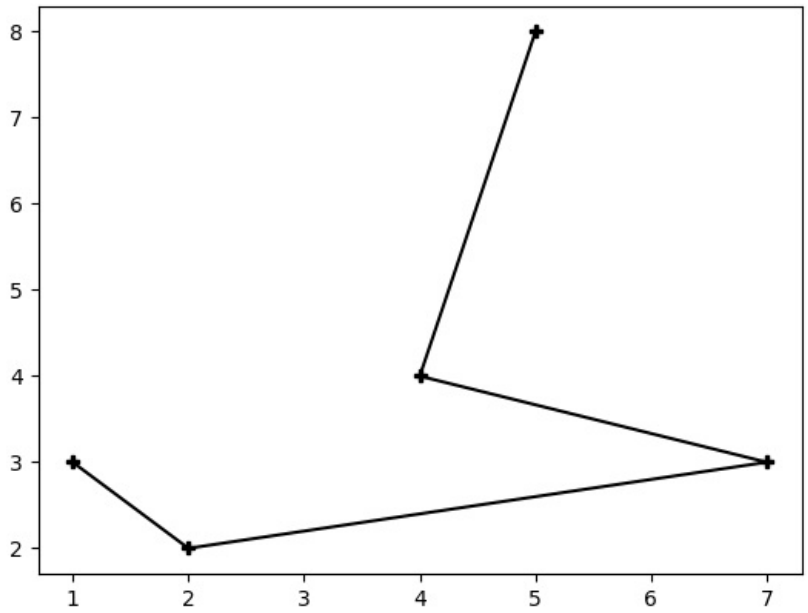
In [210…
```
xpoints=np.array([1,2,7,4,5])
ypoints=np.array([3,2,3,4,8])
plt.plot(xpoints,ypoints,'D-r')# circle and red
plt.show()
```



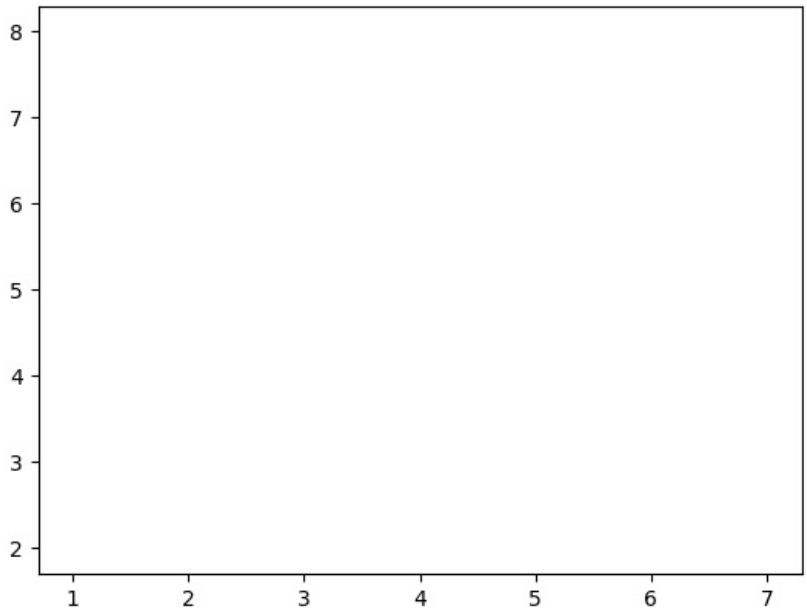
In [212…
```
xpoints=np.array([1,2,7,4,5])
ypoints=np.array([3,2,3,4,8])
plt.plot(xpoints,ypoints,'x-c')# circle and red
plt.show()
```



In [218…
```
xpoints=np.array([1,2,7,4,5])
ypoints=np.array([3,2,3,4,8])
plt.plot(xpoints,ypoints,'P-k')# circle and red
plt.show()
```



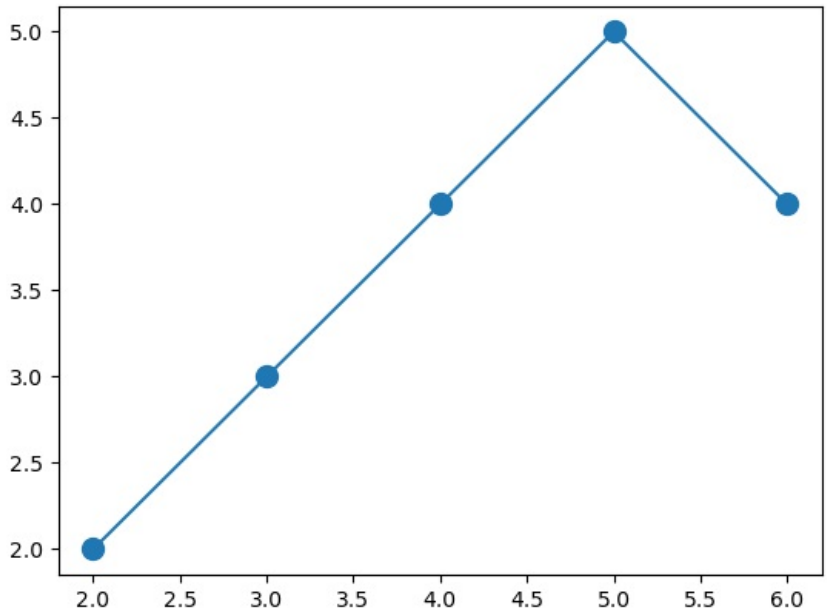
In [220…
```
xpoints=np.array([1,2,7,4,5])
ypoints=np.array([3,2,3,4,8])
plt.plot(xpoints,ypoints,'D-w')# circle and red
plt.show()
```



In [ ]:
```
"Marker size"
```

we can use the keyword arguement 'markersize' or the shorter version ,'ms' to set size of the markers:

In [233…
```
x=np.array([2,3,4,5,6])
y=np.array([2,3,4,5,4])
plt.plot(x,y,marker='o',ms=10,)
plt.show()
```

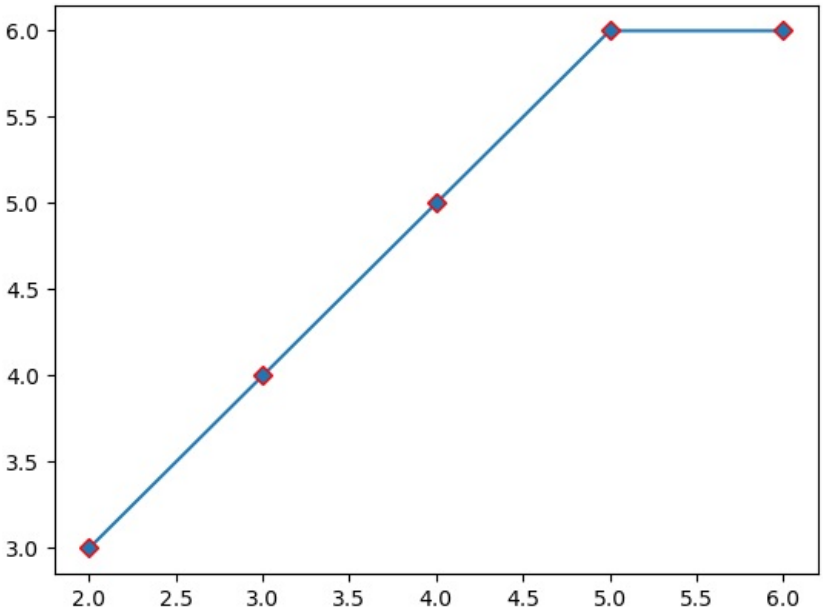


In [ ]:
```
"Marker Color"
```

we can use the keyword argument 'markeredgecolor' or the shorter 'mec' to set the color of the edge of the markers

In [249…
```
x=np.array([2,3,4,5,6])
y=np.array([3,4,5,6,6])
plt.plot(x,y,marker='D',mec='r')
```

Out[249…
```
[<matplotlib.lines.Line2D at 0x24c2827bbc0>]
```

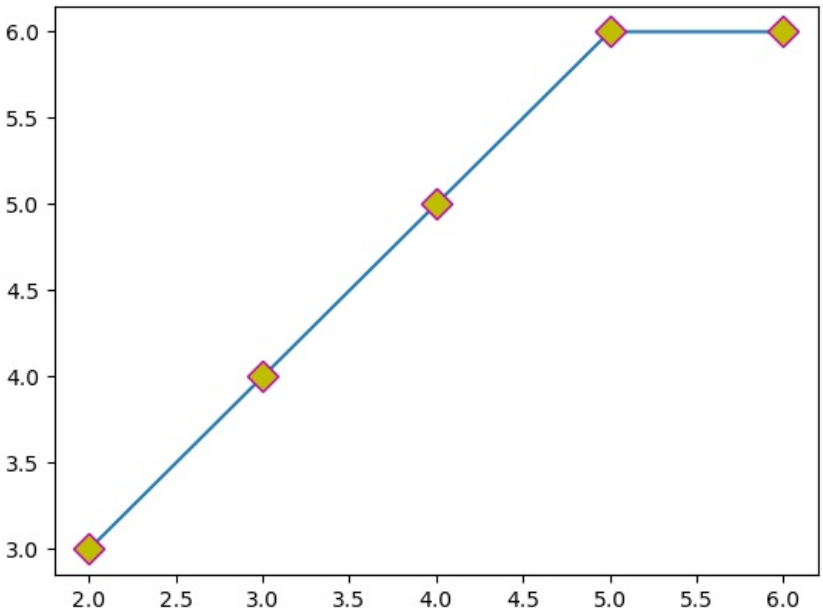


In [ ]:
```
"markerfacecolor"
```

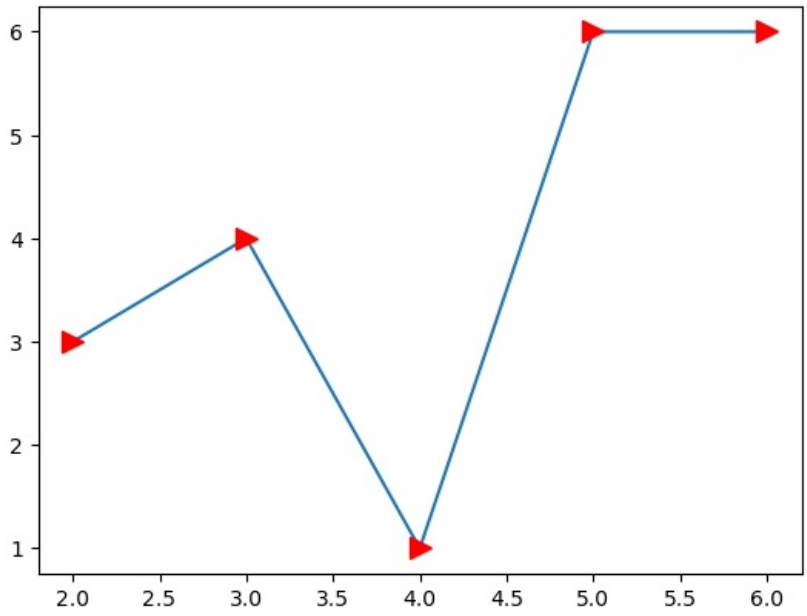
"markerfacecolor" or the shorter 'mfc' to set the color inside the edge of the markers:

In [268…
```
x=np.array([2,3,4,5,6])
y=np.array([3,4,5,6,6])
plt.plot(x,y,marker='D',mec='m',mfc='y',ms=10)
```

Out[268… `[<matplotlib.lines.Line2D at 0x24c2c1d0440>]`

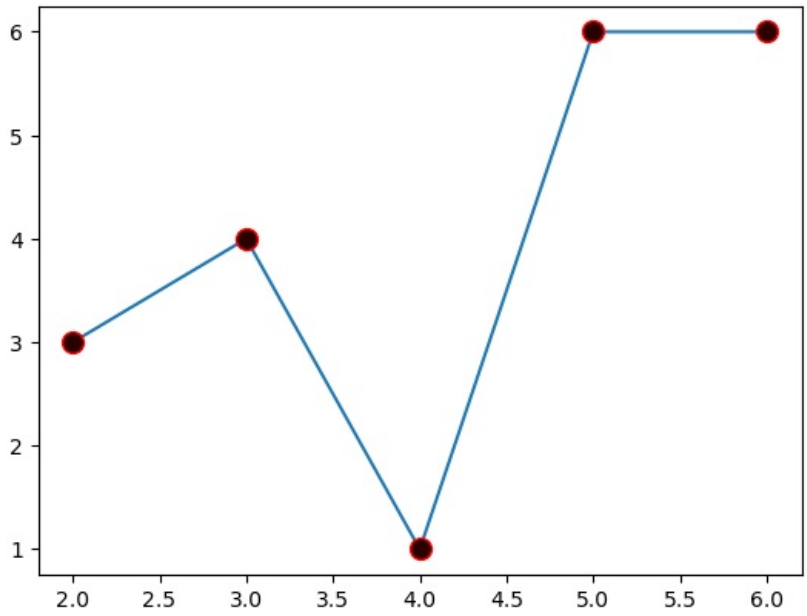


In [286…
```
x=np.array([2,3,4,5,6])
y=np.array([3,4,1,6,6])
plt.plot(x,y,marker='>',mec='r',mfc='r',ms=10)
plt.show()
```



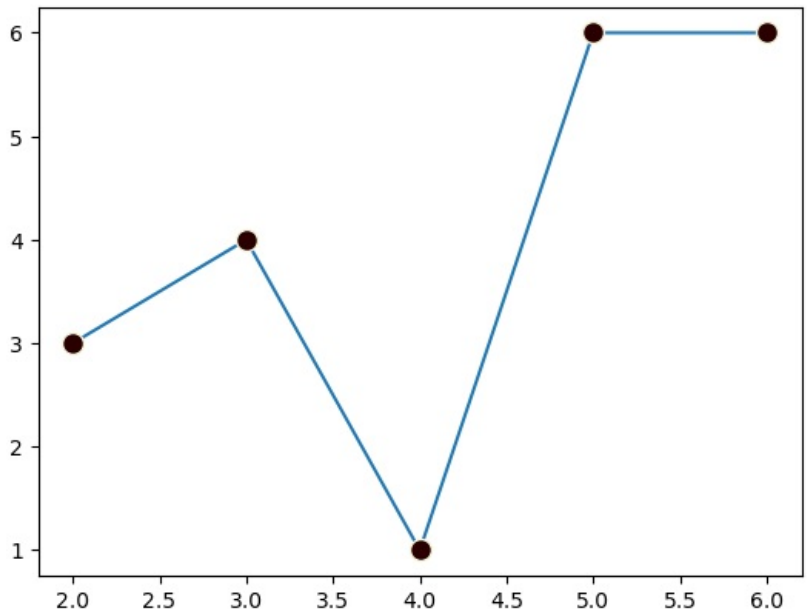
In [298…
```
plt.plot(x,y,marker='o',ms=10,mec='#ff0000',mfc='#2a0000')
```

Out[298… [<matplotlib.lines.Line2D at 0x24c2c9ff440>]

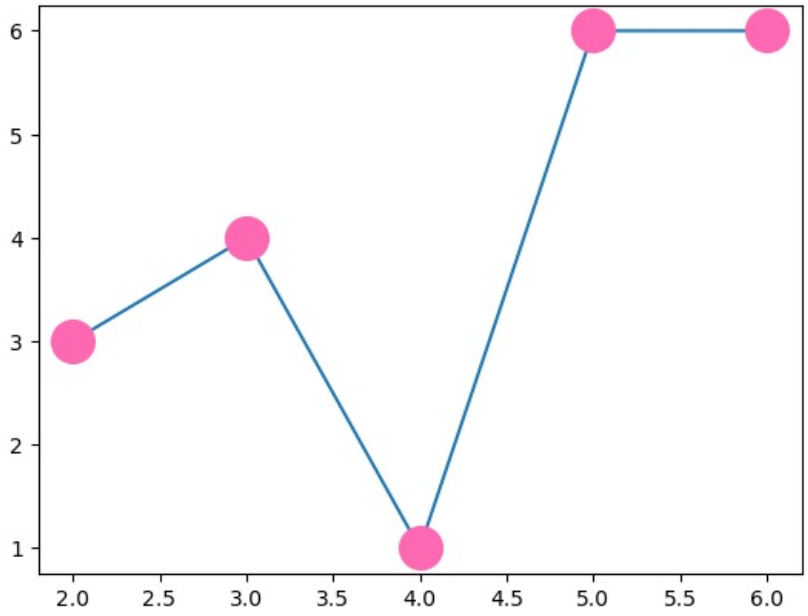


In [300…
```
plt.plot(x,y,marker='o',ms=10,mec='#F5F5DC',mfc='#2a0000')
```

Out[300… [<matplotlib.lines.Line2D at 0x24c2ca43410>]



In [302…]:
```
plt.plot(x,y,marker='o',ms=20,mec='hotpink',mfc='hotpink')
plt.show()
```

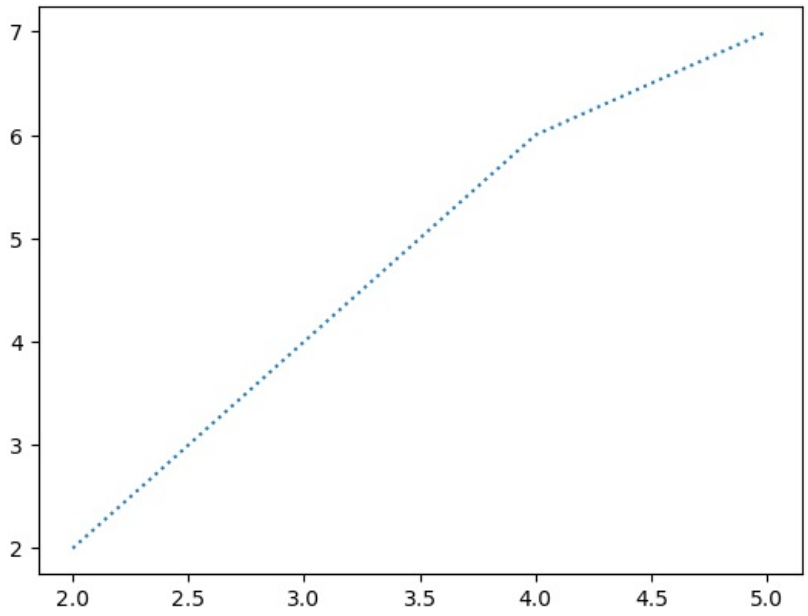


In [ ]:
```
"Matplotlib line"
```

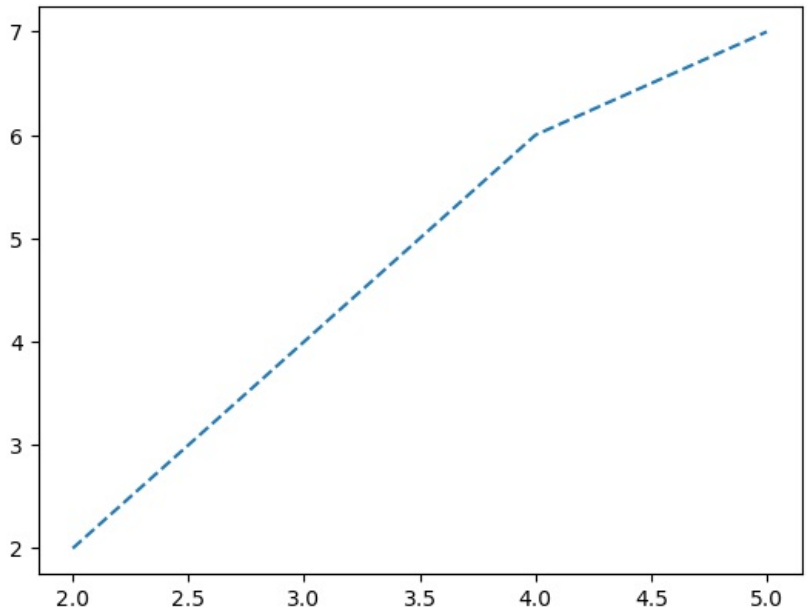
In [ ]:
```
"Linestyle"
```

we can use the keyword for changing the linestyle of the plotted line using 'linestyle or ls'

In [4]:
```
import matplotlib.pyplot as plt
import numpy as np
x=np.array([2,3,4,5])
y=np.array([2,4,6,7])
plt.plot(x,y,linestyle='dotted')
plt.show()
```



In [6]:
```
plt.plot(x,y,linestyle='dashed')
plt.show()
```



In [ ]:
```
"shorted syntax"
```

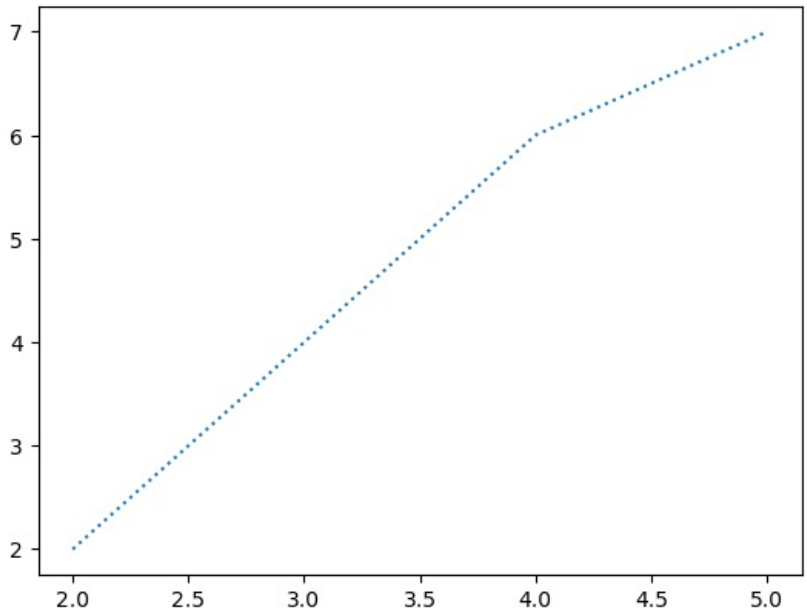
the line style can be writtern in a shorter syntax :

linestyle can be written as ls

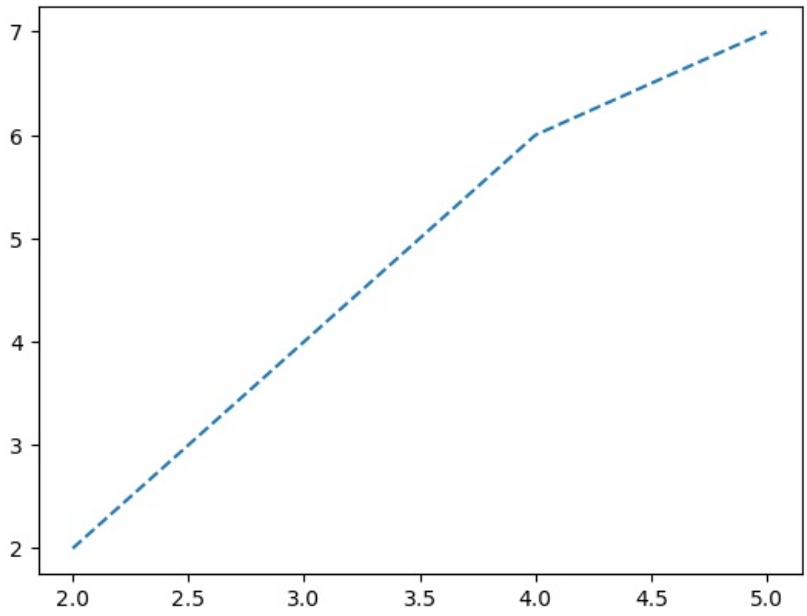
dotted can be written as :,

dashed can be written as --.

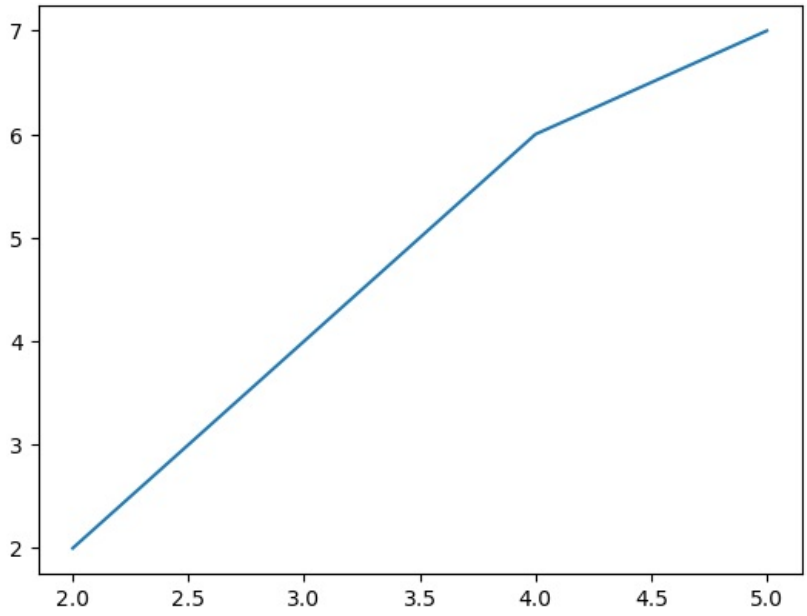
In [13]:
```
plt.plot(x,y,ls=':')
plt.show()
```



In [15]:
```
plt.plot(x,y,ls='--')
plt.show()
```



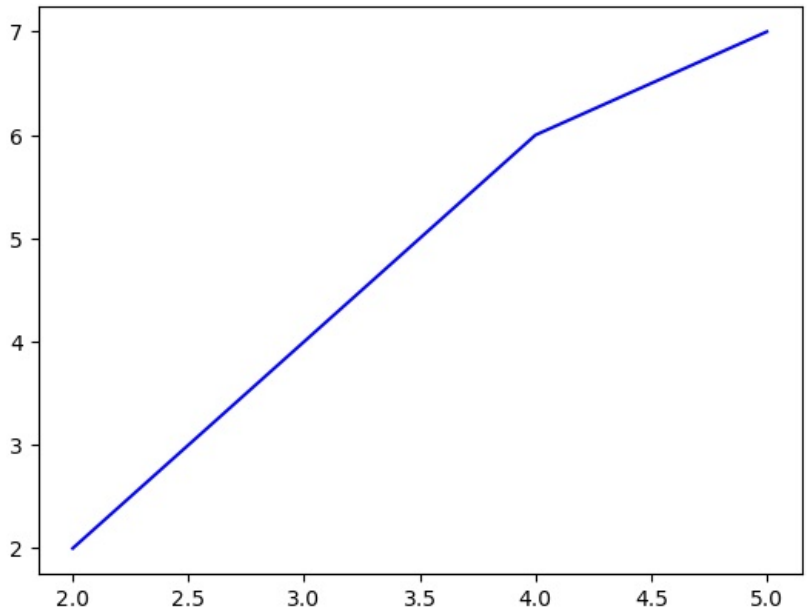
In [17]:
```
plt.plot(x,y,ls='-')
plt.show()
```



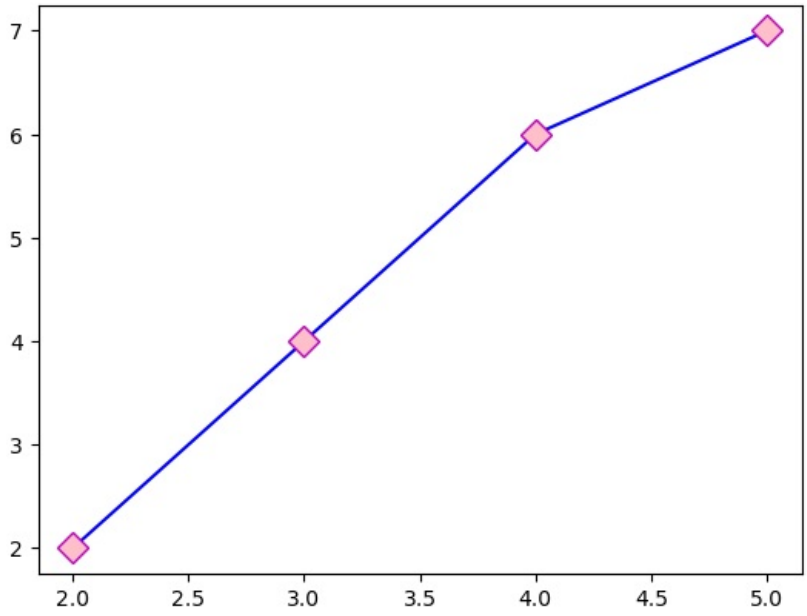
In [ ]:
```
"linecolor"
```

we can use the keyword argument 'color' or the shorter 'c' to set the color of the line.

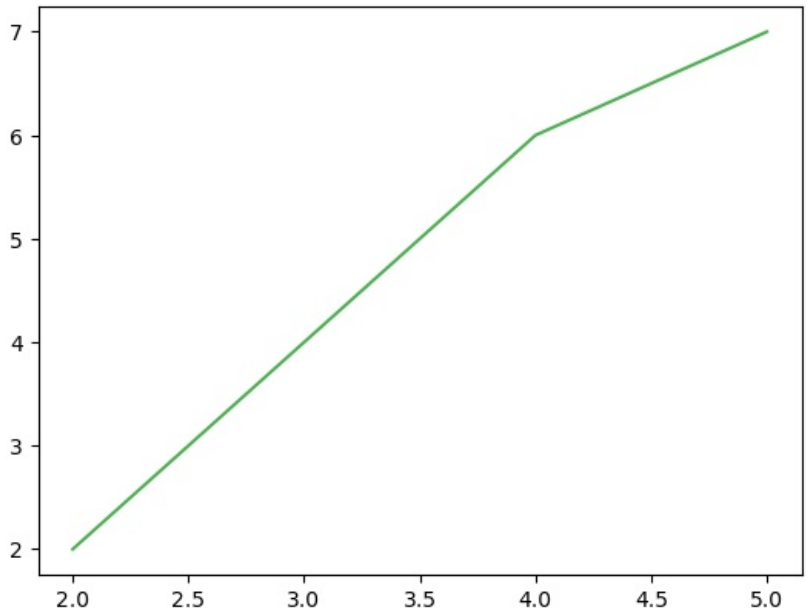
In [27]:
```
plt.plot(x,y,ls='-',color='b')
plt.show()
```



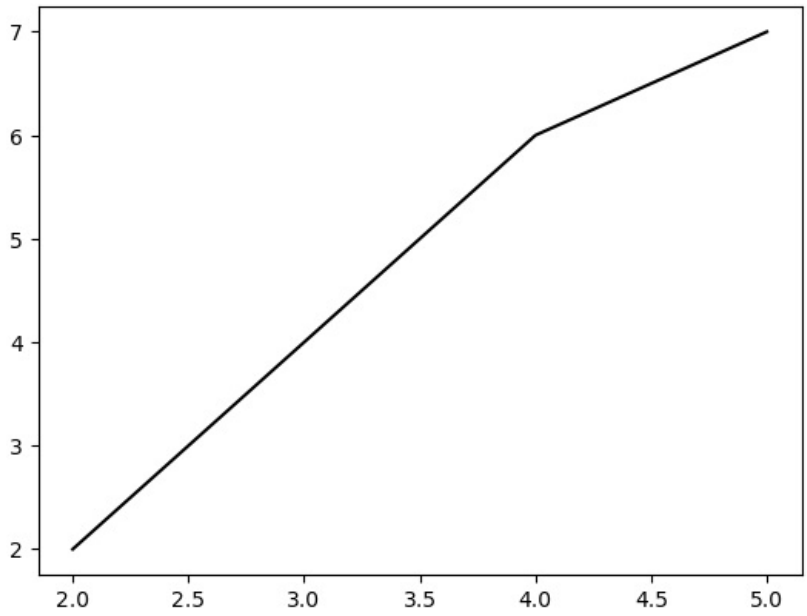
In [39]:
```
plt.plot(x,y,ls='-',color='b',marker='D',ms=10,mec='m',mfc='pink')
plt.show()
```



In [43]:
```
plt.plot(x,y,color='#4CAF50')
plt.show()
```



In [49]:
```
plt.plot(x,y,c='black')
plt.show()
```



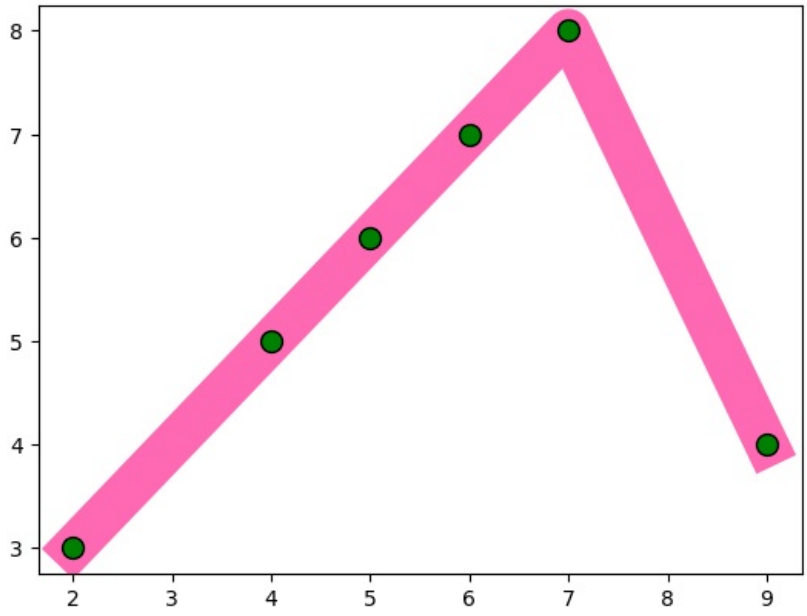
In [ ]:
```
"linewidth"
```

we can use the keyword argument 'linewidth' or the shorter 'lw' to change the width of the line.

the value is floating number in the points.

In [58]:
```
import matplotlib.pyplot as plt
import numpy as np

x=np.array([2,4,5,6,7,9])
y=np.array([3,5,6,7,8,4])
plt.plot(x,y,ls='-',linewidth='20.5',color="hotpink",marker='o',ms=10,mec='black',mfc='green')
```

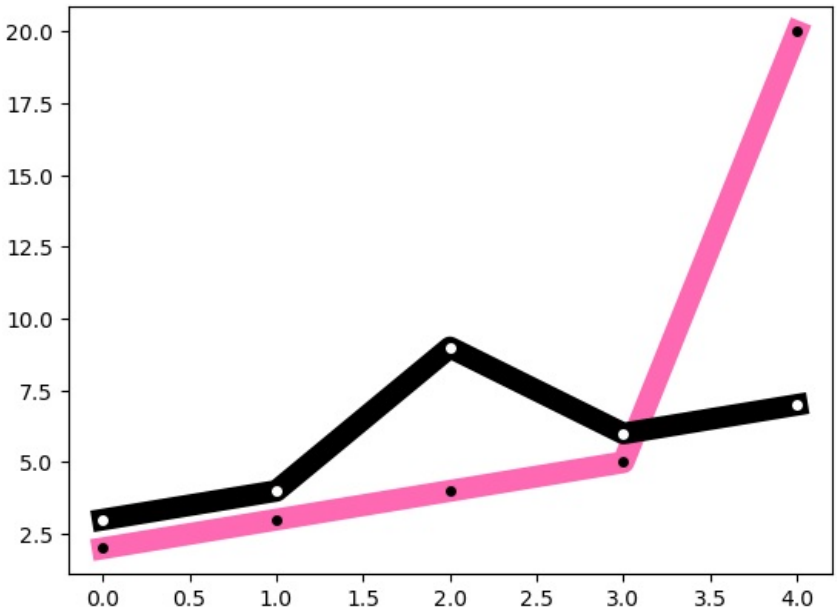
Out[58]:
```
[<matplotlib.lines.Line2D at 0x1326d6cd5e0>]
```



In [ ]:
```
"multiple lines"
```

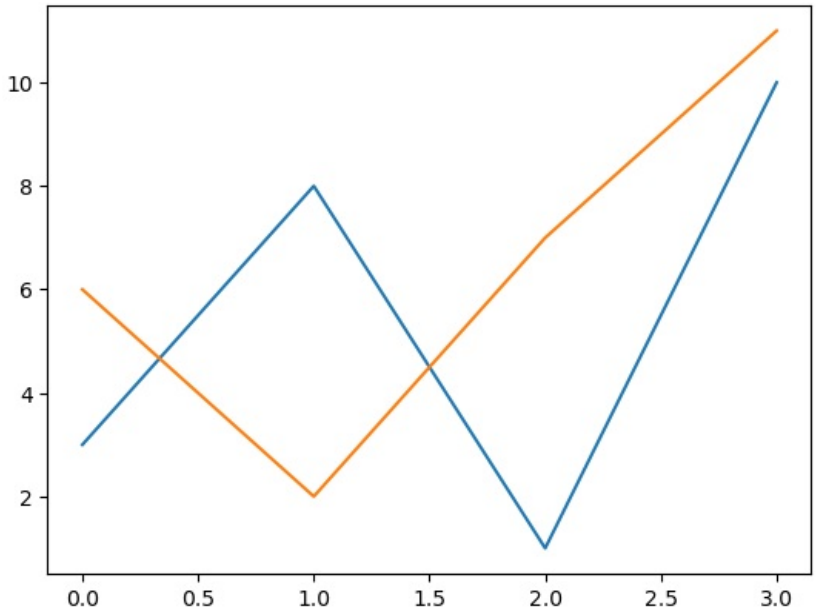
we can plot as many lines as you like by simply adding more plt.plot() functions.

In [93]:
```
# Draw two lines by specifying a plt.plot() function for each line:

#by default x values should be taken by the function
#x1=[0,1,2,3, etc..
y1=np.array([2,3,4,5,20])
y2=np.array([3,4,9,6,7])
plt.plot(y1,ls='-',linewidth='10.3',color='hotpink',marker='o',mfc='black')
plt.plot(y2,ls='-',linewidth='10.2',color='black',marker='o',mfc='w')
plt.show()
```



we can also many lines by adding the points for the x-axis and y-axis for each line in the same plt.plot() function.

In [96]:
```
import matplotlib.pyplot as plt
import numpy as np

x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])
x2 = np.array([0, 1, 2, 3])
y2 = np.array([6, 2, 7, 11])

plt.plot(x1, y1, x2, y2)
plt.show()
```

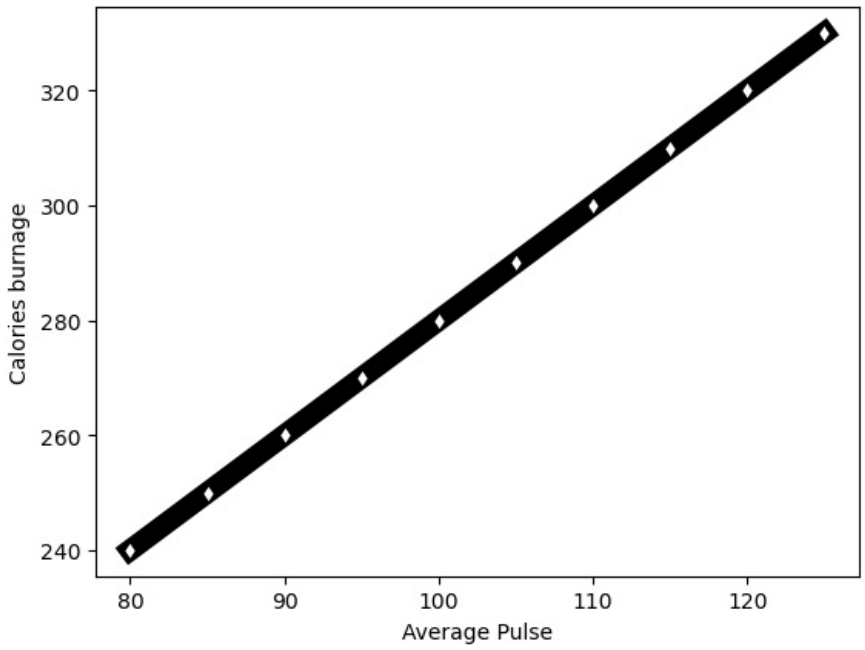


In [ ]:
```
"MatplotLib Labels and title"
```

In [ ]:
```
"create labels for the plot"
```

with pyplot we can use the 'xlabel()' and 'ylabel()' functions to set a label for the x-and u=y-axis.

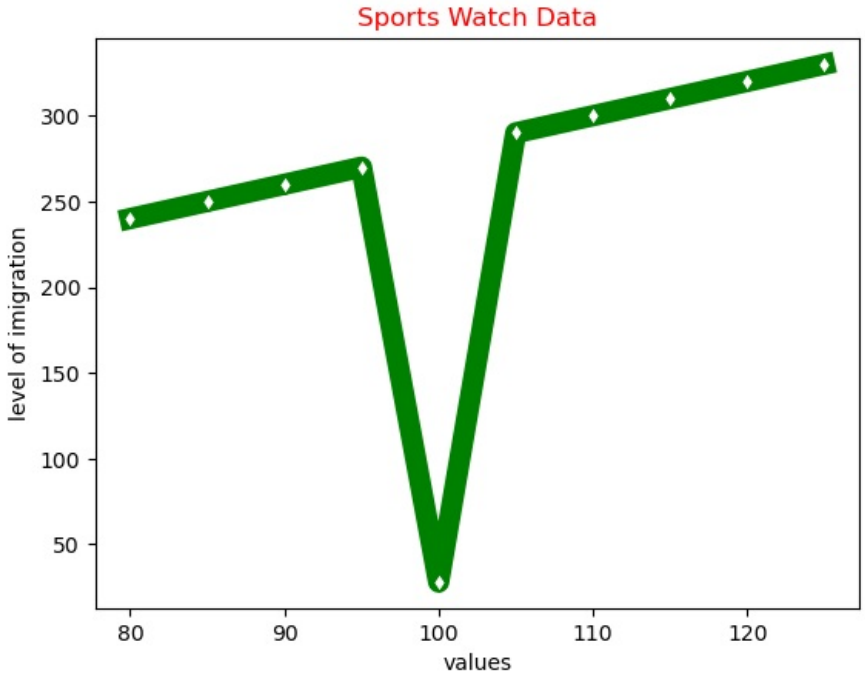
In [124…
```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y=np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.plot(x,y,linestyle='-',lw=10,color='black',marker='d',mfc='w')

plt.xlabel("Average Pulse")
plt.ylabel("Calories burnage")
plt.show()
```



In [ ]:
```
"create a title for the plot"
```

with pyplot we can use the 'title()' function to set a title for the plot

In [134…
```
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, 28, 290, 300, 310, 320, 330])
plt.plot(x,y,linestyle='-',lw=10,color='green',marker='d',mfc='w')
plt.xlabel('values')
plt.ylabel('level of imigration',color='black')
plt.title('Sports Watch Data',color='red')
plt.show()
```

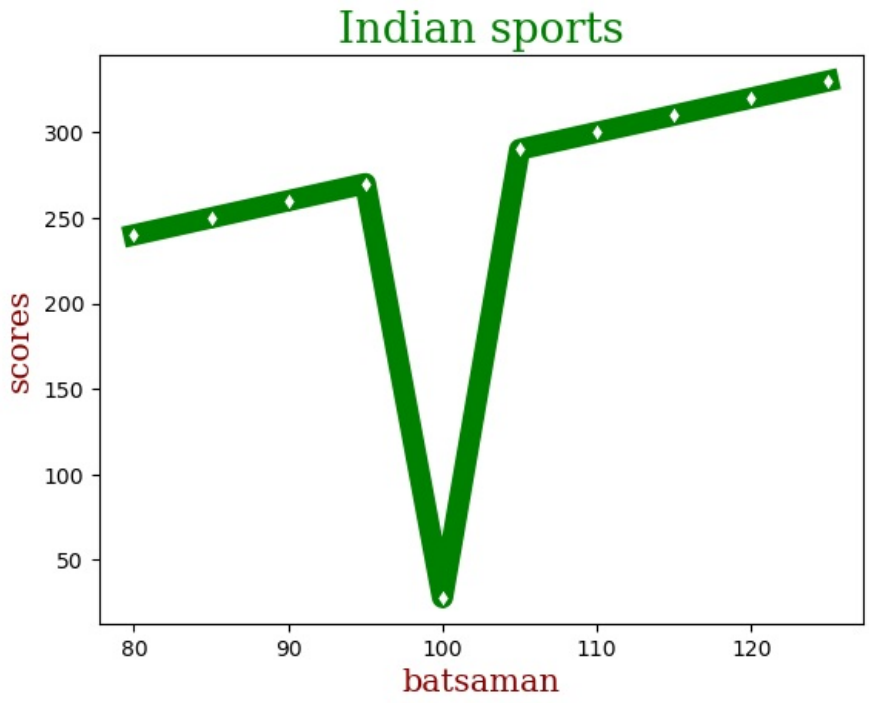


In [ ]:
```
"Set font properties for the labels and titles"
```

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

In [151…
```
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, 28, 290, 300, 310, 320, 330])
plt.plot(x,y,linestyle='-',lw=10,color='green',marker='d',mfc='w')
font1={'family':'serif','color':'green','size':20}
font2={'family':'serif','color':'darkred','size':15}
plt.title('Indian sports',fontdict=font1)
plt.xlabel('batsaman',fontdict=font2)
plt.ylabel('scores',fontdict=font2)
```

```
plt.show()
```

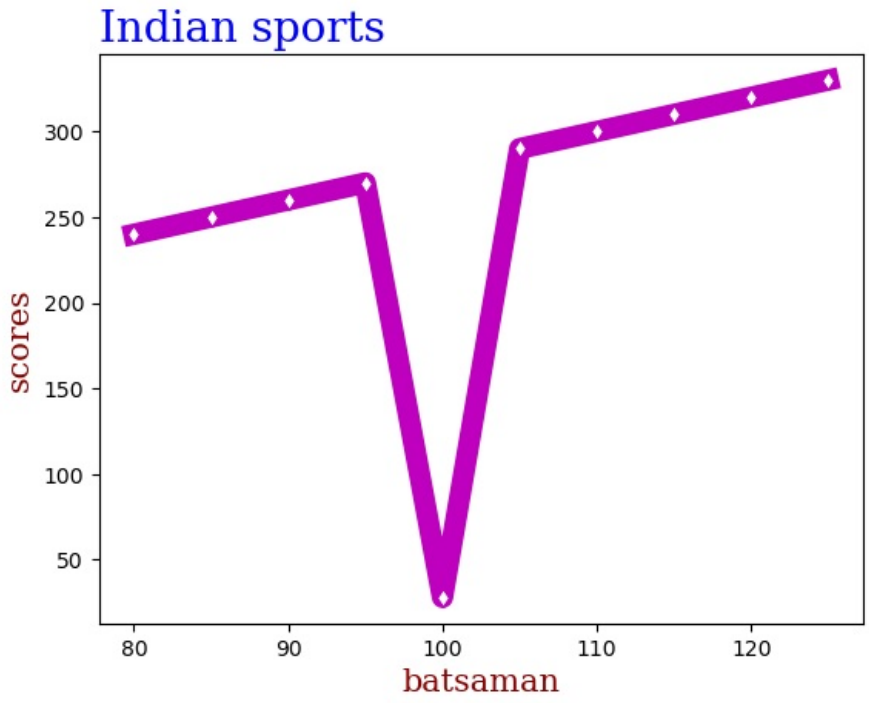


In [ ]:
```
"Position of the title"
```

we can use the 'loc' parameter for the 'title()' position change perpose

legal values are:'left','right','center',default value is 'center'

In [169…
```
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, 28, 290, 300, 310, 320, 330])
plt.plot(x,y,linestyle='-',lw=10,color='m',marker='d',mfc='w')
font1={'family':'serif','color':'b','size':20}
font2={'family':'serif','color':'darkred','size':15}
plt.title('Indian sports',fontdict=font1,loc='left')
plt.xlabel('batsaman',fontdict=font2)
plt.ylabel('scores',fontdict=font2)
plt.show()
```



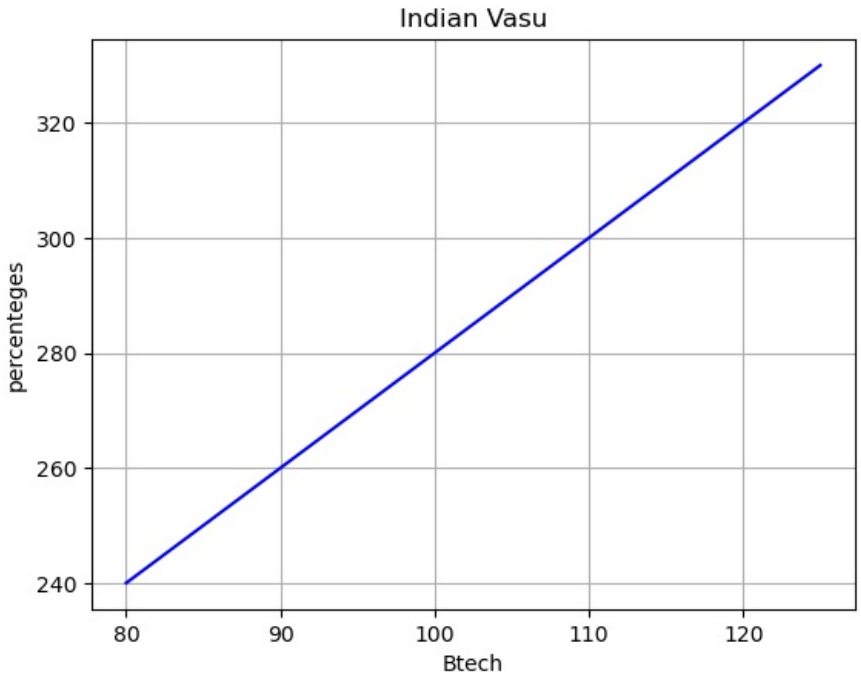
In [ ]:

In [ ]:
```
"MatplotLib Adding Grid Lines"
```

In [ ]:
```
"Add Grid Lines to a plot"
```

with pyplot,we can use the 'grid()' function to add grid lines to the plot

In [47]:
```
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,color='b',)
plt.title("Indian Vasu")
plt.xlabel("Btech")
plt.ylabel("percenteges")
plt.grid()
plt.show()
```

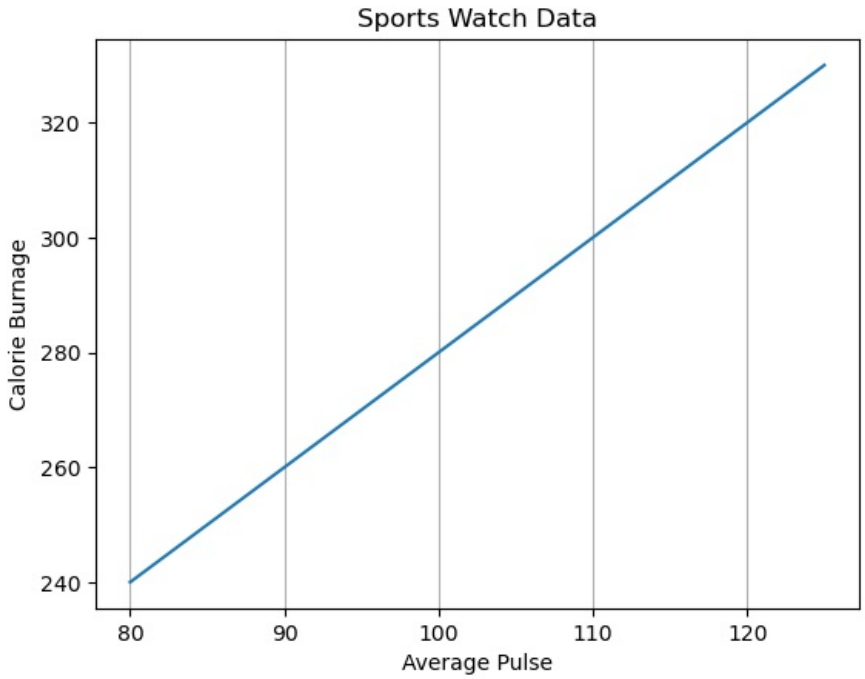


In [ ]:
```
"Specify which Grid lines to Display"
```

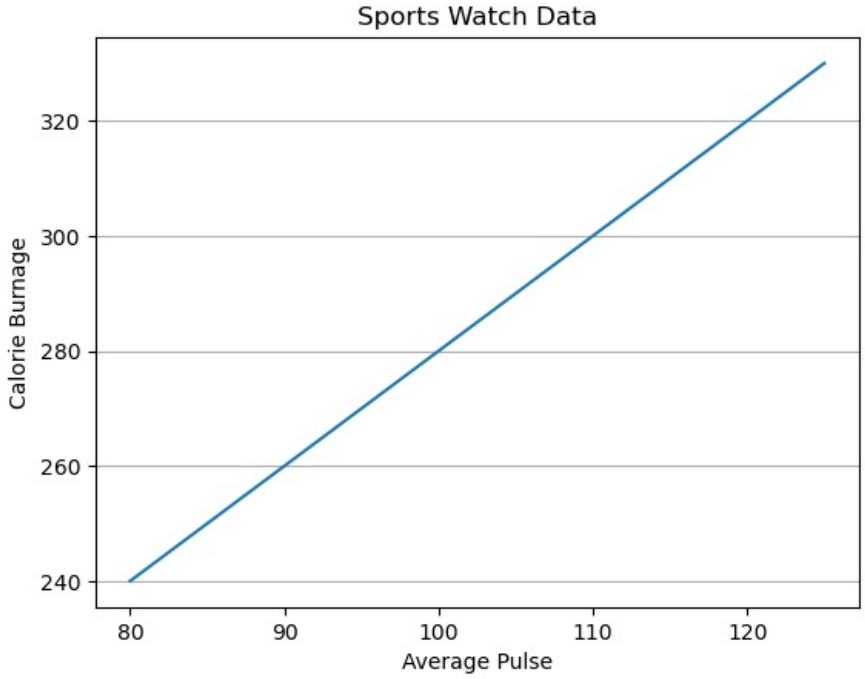
we can use the 'axis' parameter in the 'grid()' function to specify whcih grid lines to specify

Legal values are both:x,y and 'both'.Default values is 'both'

In [52]:
```
#Display only grid lines for the x-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.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.plot(x,y)
plt.grid(axis='x')
plt.show()
```



In [56]:
```
#Display only grid lines for the 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.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.plot(x,y)
plt.grid(axis='y')
plt.show()
```

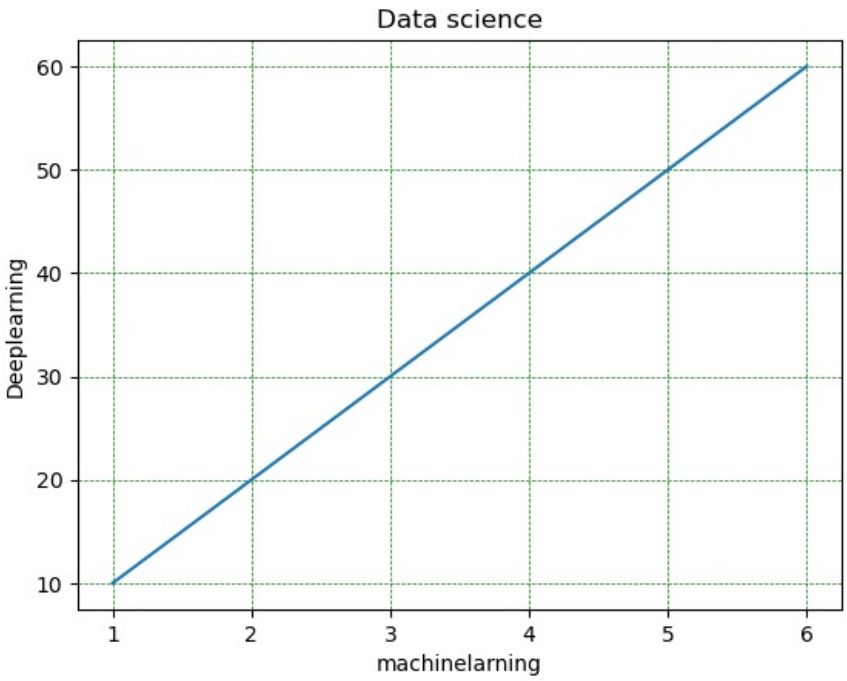


In [ ]:
```
"set line properties for the grid lines"
```

we can also set the line properties of the grid ,like this:grid(color='color',linestyle='linestyle',linewidth=number)

In [71]:
```
# Set the line properties of the grid:
import numpy as np
import matplotlib.pyplot as plt

x=np.array([1,2,3,4,5,6])
y=np.array([10,20,30,40,50,60])
plt.plot(x,y)
plt.title("Data science")
plt.xlabel('machinelarning')
plt.ylabel("Deeplearning")
```

```
plt.grid(color='green',linestyle='--',linewidth=0.5)
plt.show()
```

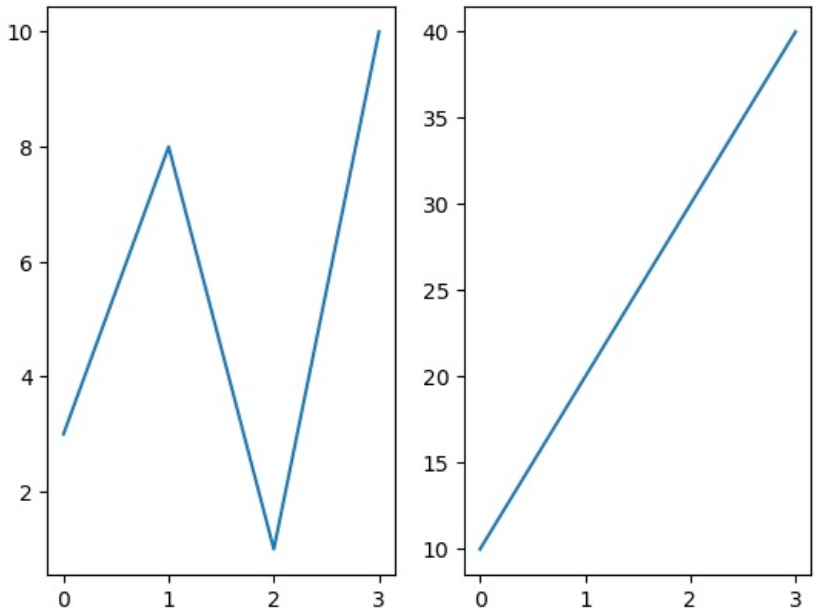


In [ ]:
```
"Matplotlib subplot"
```

In [ ]:
```
"Display Multiple Plots"
```

with the 'subplot()' function we can draw the multiple plots in one figure

In [110…
```
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(1, 2, 1)#row,column,plot
plt.plot(x,y)

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

plt.subplot(1, 2, 2)#row,colomn,plot
plt.plot(x,y)

plt.show()
```



In [ ]:
```
"The subplot function explanation"
```

The 'subplot' function takes three arguments that describes the layout

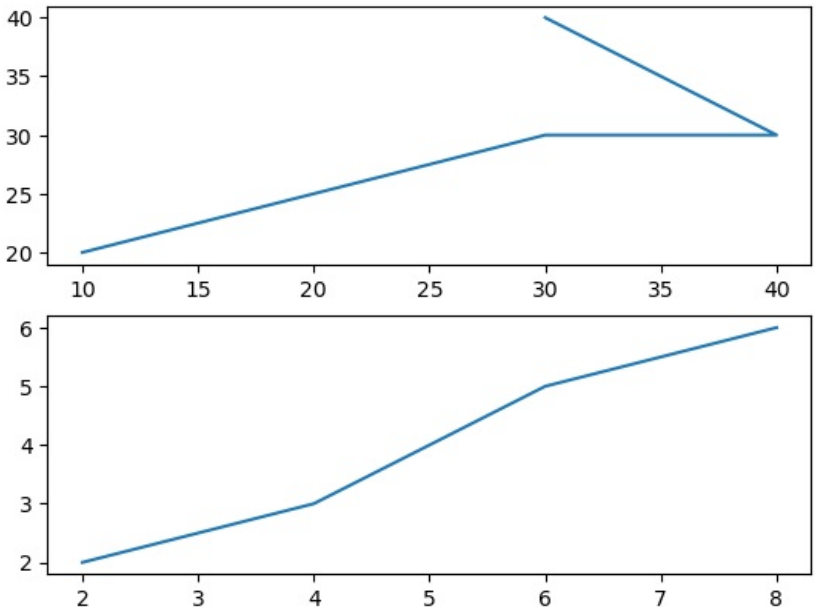
The layout is organised in rows and columns ,which are represented by the first and second arguement.

The third arguement represents the index of the current plot.

In [ ]:
```
plt.subplot(1,2,1)
#the figure has 1 row and 2 columns and this plot is the first plot.
```

In [ ]:
```
plt.subplot(1,2,2)
#the figure has 1 row,2 columns and this plot is the second plot
```

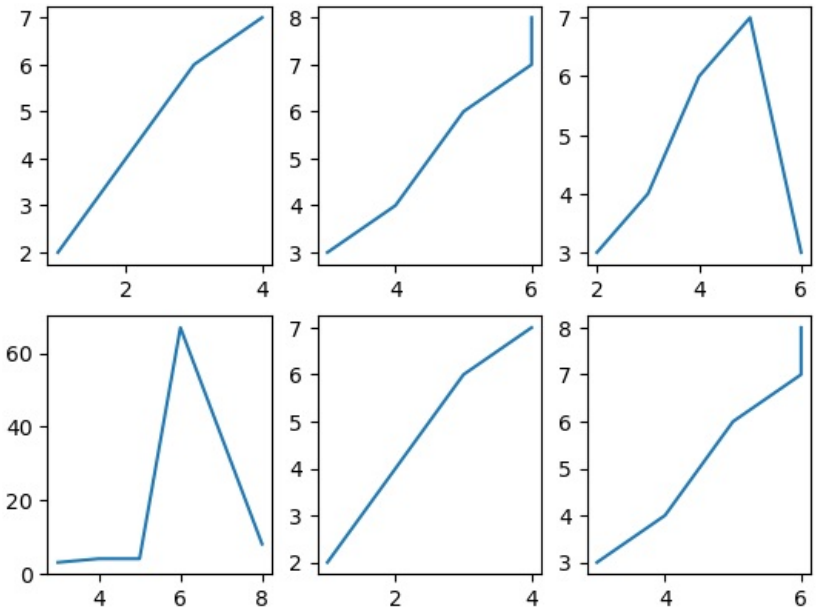
In [126…
```
x=np.array([10,30,40,30])
y=np.array([20,30,30,40])
plt.subplot(2,1,1)
plt.plot(x,y)


x=np.array([2,4,6,8])
y=np.array([2,3,5,6])
plt.subplot(2,1,2)
plt.plot(x,y)
plt.show()
```



we can draw the many plots you like on one figure just descibe the number of rows and columns and the index of the plot

In [139…
```
x=np.array([1,2,3,4])
y=np.array([2,4,6,7])
plt.subplot(2,3,1)
plt.plot(x,y)

x=np.array([3,4,5,6,6])
y=np.array([3,4,6,7,8])
plt.subplot(2,3,2)
plt.plot(x,y)

x=np.array([2,3,4,5,6])
y=np.array([3,4,6,7,3])
plt.subplot(2,3,3)
plt.plot(x,y)

x=np.array([3,4,5,6,8])
y=np.array([3,4,4,67,8])
plt.subplot(2,3,4)
plt.plot(x,y)

x=np.array([1,2,3,4])
y=np.array([2,4,6,7])
plt.subplot(2,3,5)
plt.plot(x,y)

x=np.array([3,4,5,6,6])
y=np.array([3,4,6,7,8])
plt.subplot(2,3,6)
plt.plot(x,y)
```

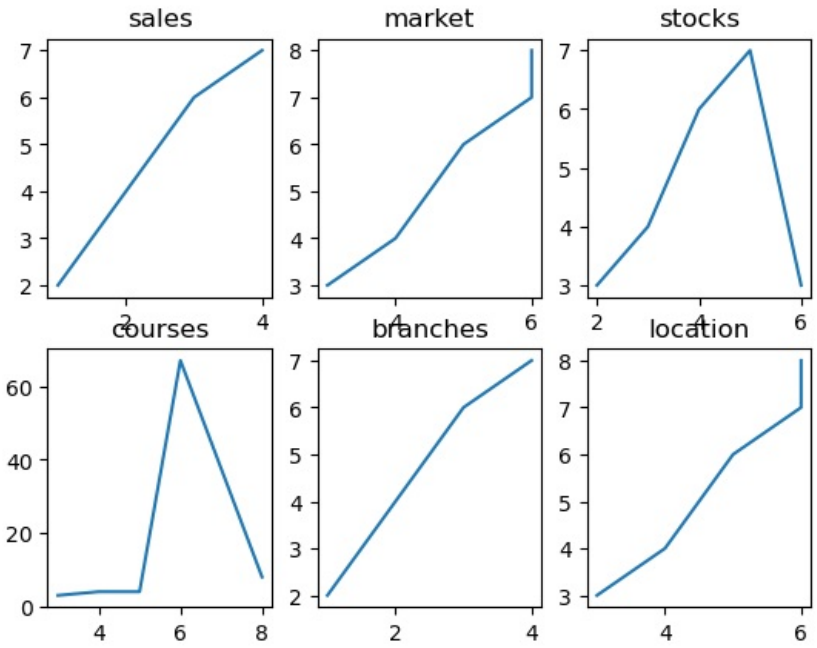
```
plt.show()
```



In [ ]:
```
"Title"
```

we can add the title each plot with 'title()' function.

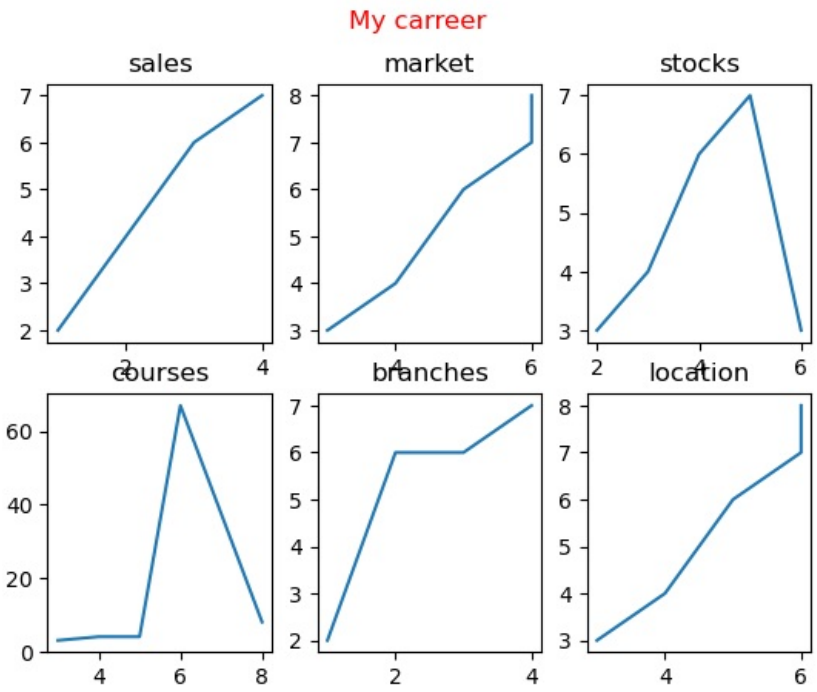
In [144…
```
x=np.array([1,2,3,4])
y=np.array([2,4,6,7])
plt.subplot(2,3,1)
plt.plot(x,y)
plt.title("sales")
x=np.array([3,4,5,6,6])
y=np.array([3,4,6,7,8])
plt.subplot(2,3,2)
plt.plot(x,y)
plt.title("market")

x=np.array([2,3,4,5,6])
y=np.array([3,4,6,7,3])
plt.subplot(2,3,3)
plt.plot(x,y)
plt.title("stocks")

x=np.array([3,4,5,6,8])
y=np.array([3,4,4,67,8])
plt.subplot(2,3,4)
plt.plot(x,y)
plt.title("courses")

x=np.array([1,2,3,4])
y=np.array([2,4,6,7])
plt.subplot(2,3,5)
plt.plot(x,y)
plt.title("branches")

x=np.array([3,4,5,6,6])
y=np.array([3,4,6,7,8])
plt.subplot(2,3,6)
plt.plot(x,y)
plt.title("location")

plt.show()
```



In [ ]:
```
"Super Title"
```

we can add the super title of the entire the plot with 'suptitle()' function

In [152…
```
x=np.array([1,2,3,4])
y=np.array([2,4,6,7])
plt.subplot(2,3,1)
plt.plot(x,y)
plt.title("sales")
x=np.array([3,4,5,6,6])
y=np.array([3,4,6,7,8])
plt.subplot(2,3,2)
plt.plot(x,y)
plt.title("market")

x=np.array([2,3,4,5,6])
y=np.array([3,4,6,7,3])
plt.subplot(2,3,3)
plt.plot(x,y)
plt.title("stocks")

x=np.array([3,4,5,6,8])
y=np.array([3,4,4,67,8])
plt.subplot(2,3,4)
plt.plot(x,y)
plt.title("courses")

x=np.array([1,2,3,4])
y=np.array([2,6,6,7])
plt.subplot(2,3,5)
plt.plot(x,y)
plt.title("branches")

x=np.array([3,4,5,6,6])
y=np.array([3,4,6,7,8])
plt.subplot(2,3,6)
plt.plot(x,y)
plt.title("location")

plt.suptitle("My carreer",color='red')

plt.show()
```



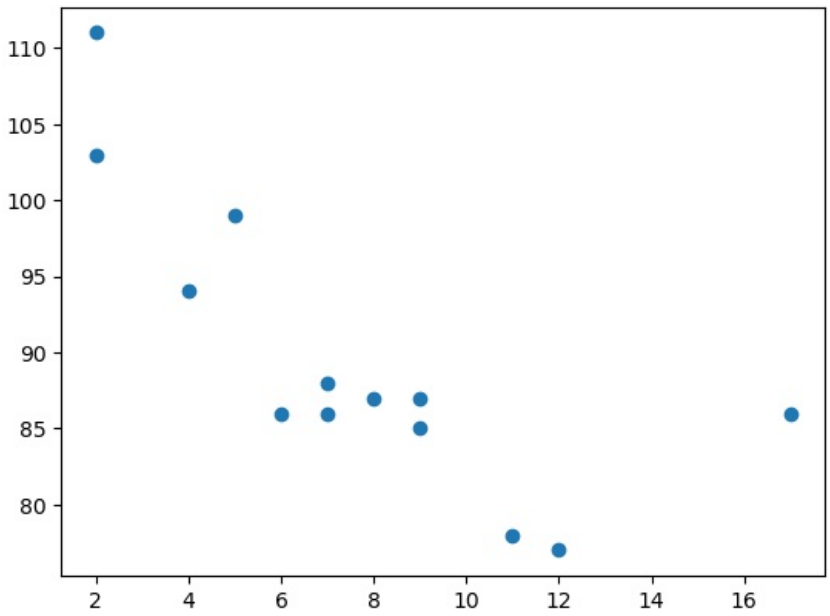
In [ ]:
```
"Matplotlib Scatter"
```

In [ ]:
```
"creating scatter plots"
```

with pyplot,we can use the 'scatter()' function to draw the scatter plot.

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 another for values on the y-axis

In [171…
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

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



The observations in the example above is the result of 13 cars passing by.

the X-axis shows how old the car is.

the y-axis shows how the speed of the car.

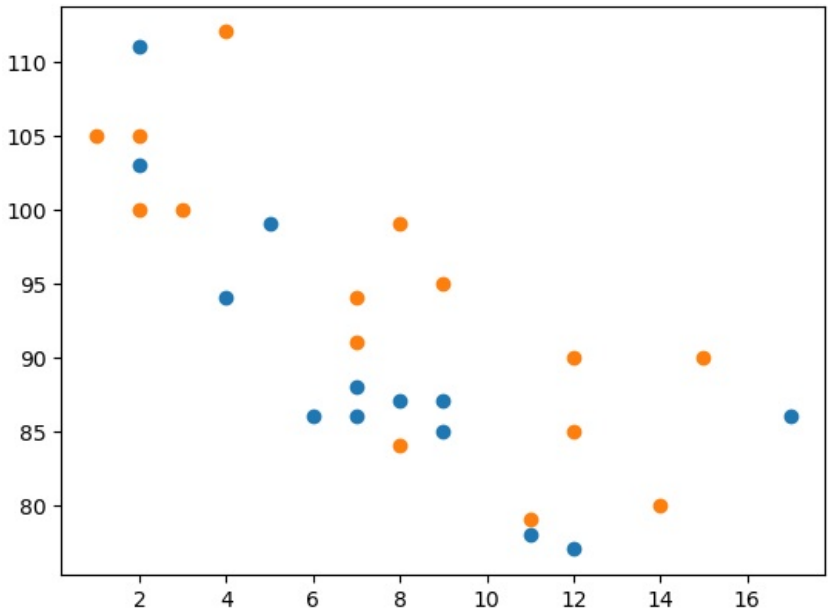
are there any relationships between the obseravations.

It seems the newer the car,the faster it drives ,but that could be a coincidence ,after all we only registered 13 cars.

In [ ]:
```
"compare plots"
```

In the above example above ,there seems to be a relationship between speed and age ,but what if we plot the observations from another day as well? will the scatter plot tells us something else?

In [179…
```
#day one,the age and speed of the  13 car
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)
x=np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y=np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x,y)
plt.show()
```



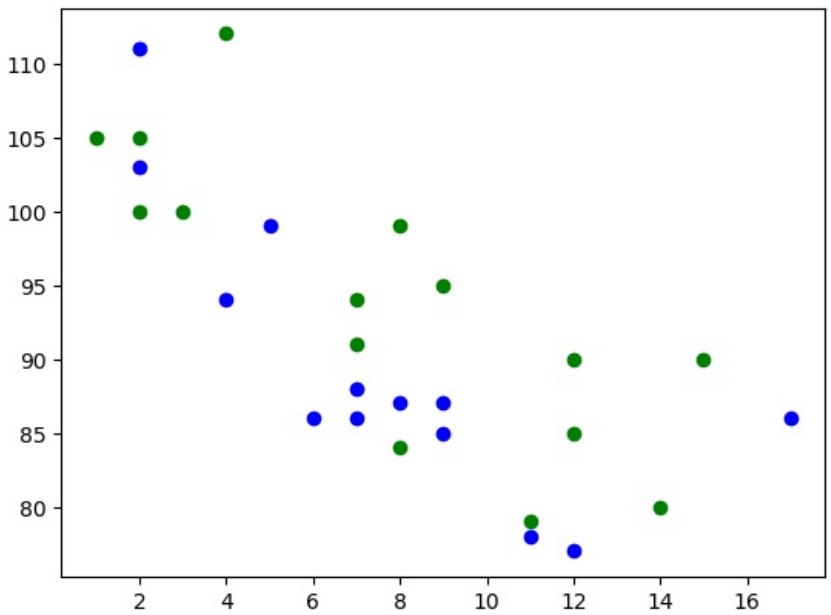
Note: The two plots are plotted with two different colors, by default blue and orange,

By comparing the two plots, I think it is safe to say that they both gives us the same conclusion: the newer the car, the faster it drives.

In [ ]:
```
"Colors"
```

we can set your own the for each scatter plot with the 'color' or the 'c' arguement.

In [184…
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y,color='blue')
x=np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y=np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x,y,color='green')
plt.show()
```

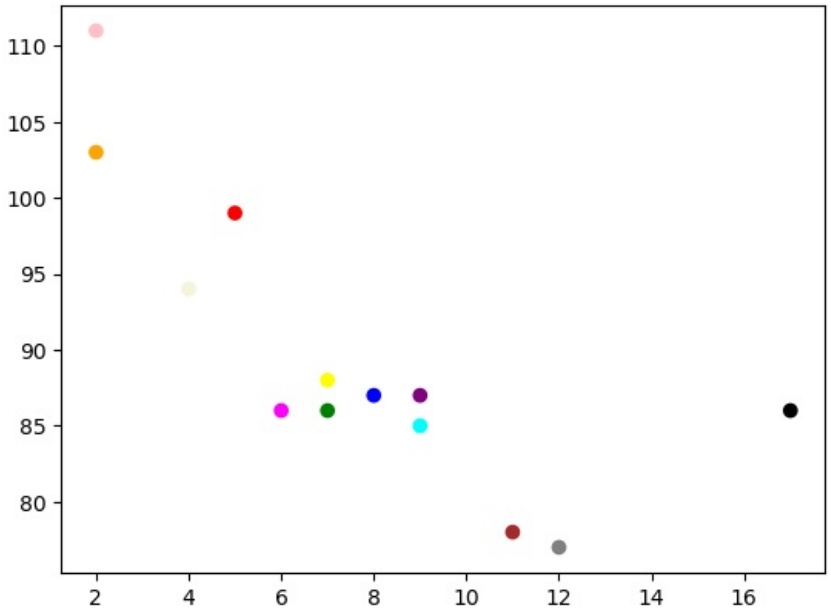


In [ ]:
```
"color Each Dot"
```

we can even set a specific color for each dot by using an array of colors as value for the 'c' argument

Note: we can use the 'color' argument for this we have to use only 'c' argument.

In [192…
```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors=np.array(['red','green','blue','yellow','pink','black','orange','purple','beige','brown','gray','cyan','
plt.scatter(x,y,c=colors)
plt.show()
```



In [ ]:
```
"colormap"
```

the matplotlib module has a number of available colormaps.

```
   A colormap is like a list of colors ,where each color has a value that ranges from 0 to 100.
```

In [195…
```
# here is one ecxample :
from PIL import Image
Image.open('img_colorbar.png')
```

Out[195…

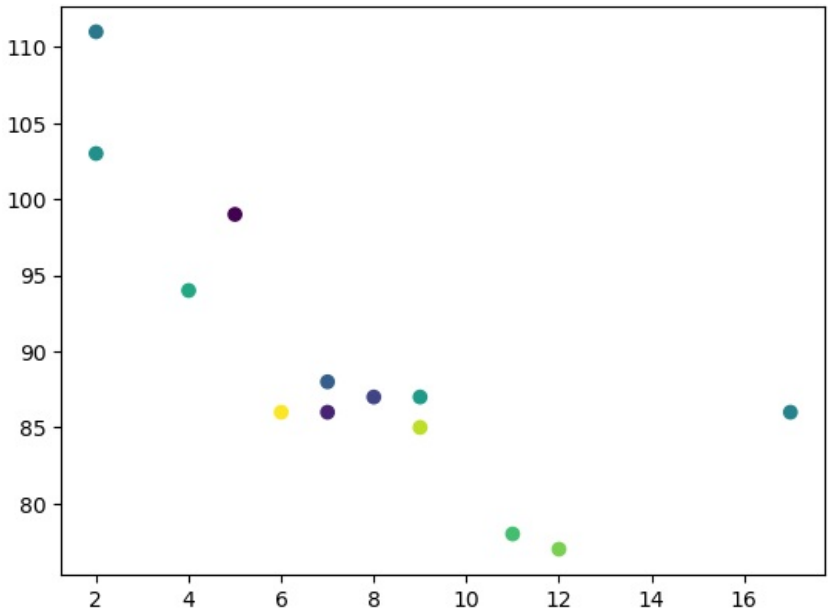
This colormap is called 'virdis' and as we can see it ranges from 0,which is a purple color,up to 100,which is a yellow color.

In [ ]:
```
"How to use the colormap"
```

we can specify the colormap with a keyword argument 'cmap' with the value of the colormap,in this case 'viridis',which is one of the built in colomap in the Matplotlib.

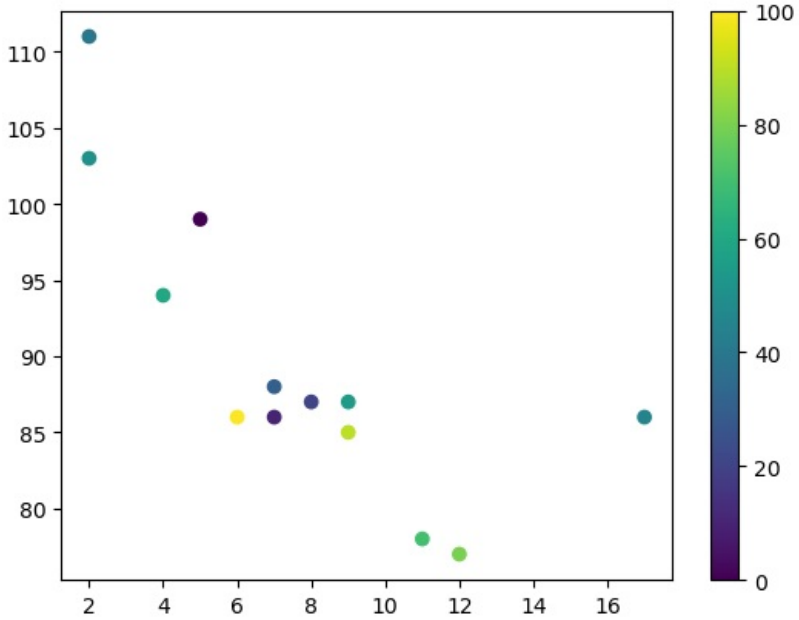
In addition we have to create an array with value(from 0 to 100),one value for each point in the scatter plot.

In [205…
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x,y ,cmap='viridis',c=colors)
plt.show()
```



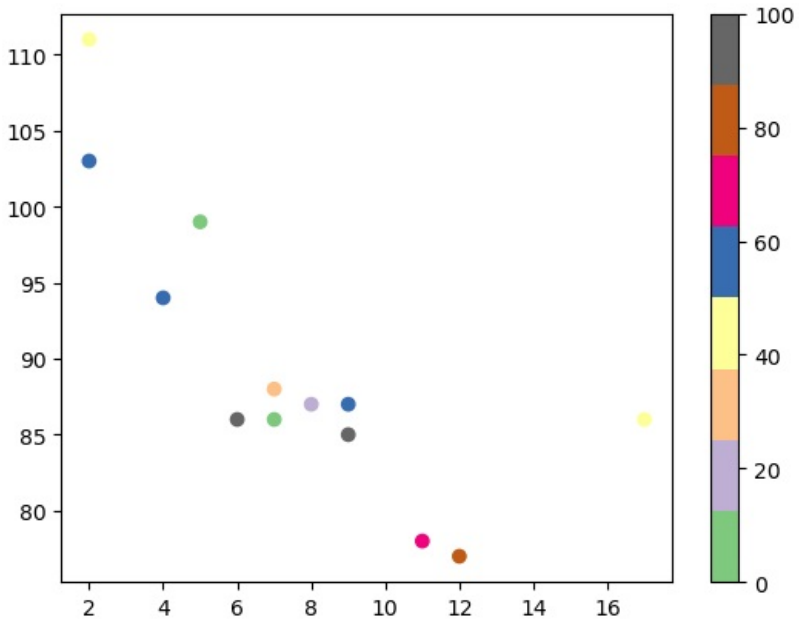
Now,we can include colormap in the drawing by inculuding the 'plt.colorbar()' statement

In [210…
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x,y ,cmap='viridis',c=colors)
plt.colorbar()
plt.show()
```



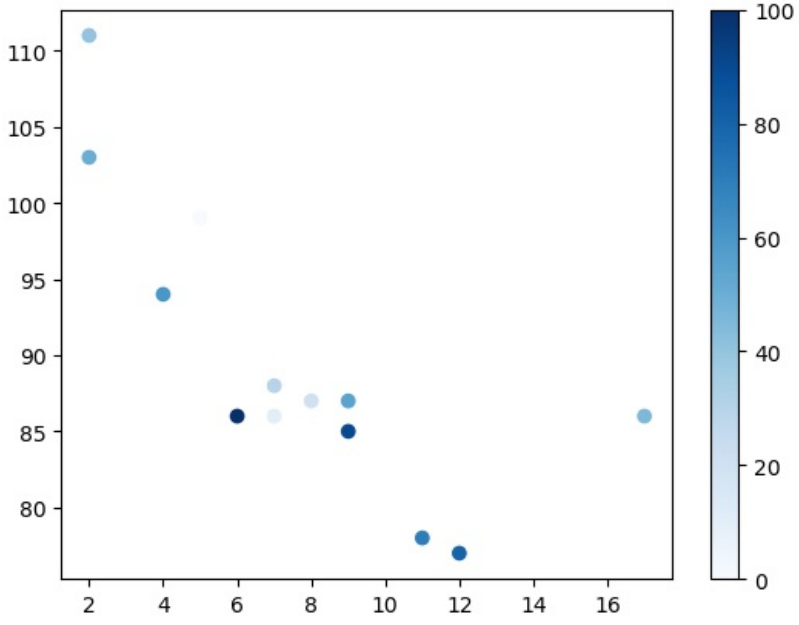
In [212…
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x,y ,cmap='Accent',c=colors)
plt.colorbar()
```

```
plt.show()
```



In [218…

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

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x,y ,cmap='Blues',c=colors)
plt.colorbar()
plt.show()
```



In [ ]:

```
Available ColorMaps
You can choose any of the built-in colormaps:

Name                    Reverse
Accent                  Accent_r
Blues                   Blues_r
BrBG                    BrBG_r
BuGn                    BuGn_r
BuPu                    BuPu_r
CMRmap                  CMRmap_r
Dark2                   Dark2_r
GnBu                    GnBu_r
Greens                  Greens_r
Greys                   Greys_r
OrRd                    OrRd_r
Oranges                 Oranges_r
PRGn                    PRGn_r
Paired                  Paired_r
Pastel1                 Pastel1_r
Pastel2                 Pastel2_r
```

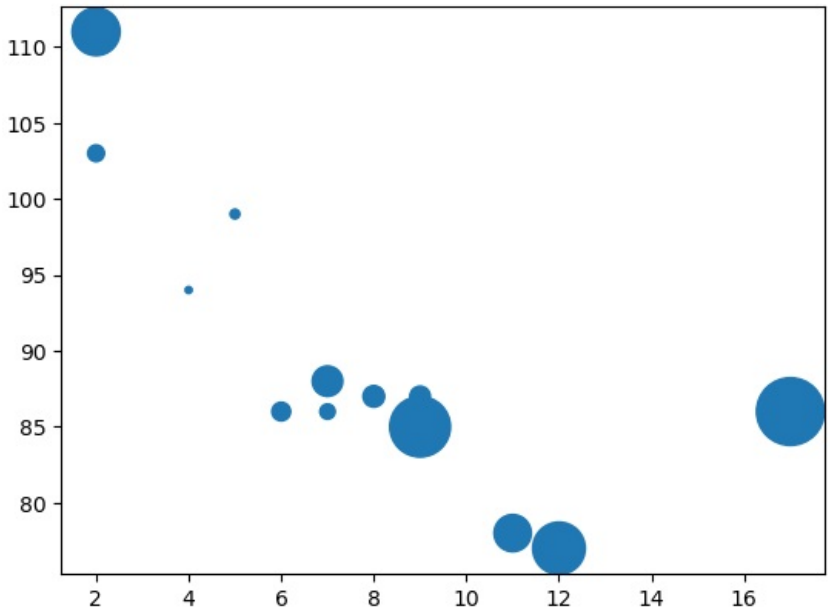
```
PiYG                    PiYG_r
PuBu                    PuBu_r
PuBuGn                  PuBuGn_r
PuOr                    PuOr_r
PuRd                    PuRd_r
Purples                 Purples_r
RdBu                    RdBu_r
RdGy                    RdGy_r
RdPu                    RdPu_r
RdYlBu                  RdYlBu_r
RdYlGn                  RdYlGn_r
Reds                    Reds_r
Set1                    Set1_r
Set2                    Set2_r
Set3                    Set3_r
Spectral                        Spectral_r
Wistia                  Wistia_r
YlGn                    YlGn_r
YlGnBu                  YlGnBu_r
YlOrBr                  YlOrBr_r
YlOrRd                  YlOrRd_r
afmhot                  afmhot_r
autumn                  autumn_r
binary                  binary_r
bone                    bone_r
brg                     brg_r
bwr                     bwr_r
cividis                 cividis_r
cool                    cool_r
coolwarm                        coolwarm_r
copper                  copper_r
cubehelix                       cubehelix_r
flag                    flag_r
gist_earth                      gist_earth_r
gist_gray                       gist_gray_r
gist_heat                       gist_heat_r
gist_ncar                       gist_ncar_r
gist_rainbow                    gist_rainbow_r
gist_stern                      gist_stern_r
gist_yarg                       gist_yarg_r
gnuplot                 gnuplot_r
gnuplot2                        gnuplot2_r
gray                    gray_r
hot                     hot_r
hsv                     hsv_r
inferno                 inferno_r
jet                     jet_r
magma                   magma_r
nipy_spectral                   nipy_spectral_r
ocean                   ocean_r
pink                    pink_r
plasma                  plasma_r
prism                   prism_r
rainbow                 rainbow_r
seismic                 seismic_r
spring                  spring_r
summer                  summer_r
tab10                   tab10_r
tab20                   tab20_r
tab20b                  tab20b_r
tab20c                  tab20c_r
terrain                 terrain_r
twilight                        twilight_r
twilight_shifted                        twilight_shifted_r
viridis                 viridis_r
winter                  winter_r
```

In [ ]:
```
"size"
```

we can change the size of the dots with the 's' arguement.

Just like colors,make sure the arrays for sizes has the same length as the arrays for the x-axis and y-axis.

In [221…
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x,y,s=sizes)
plt.show()
```

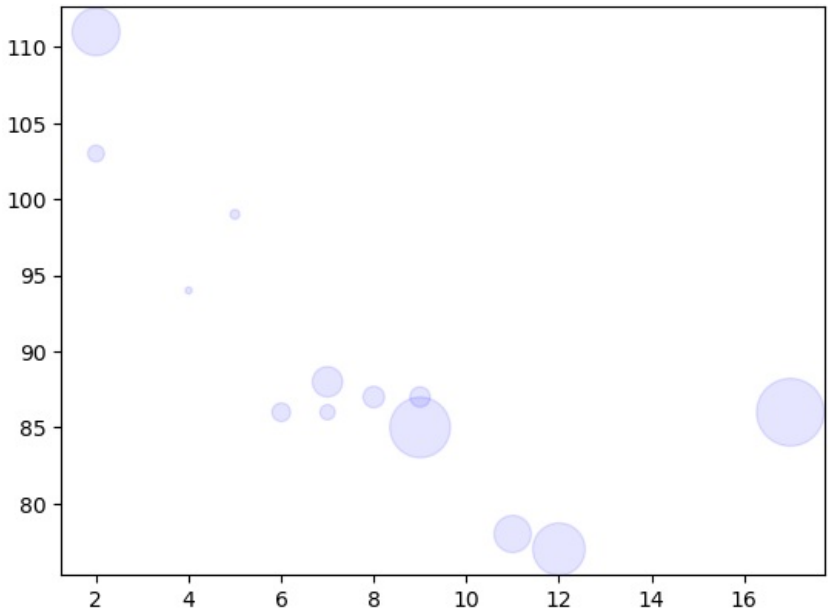


In [ ]:
```
"Alpha"
```

we can adjust the transparency of the dots with the 'alpha ' afgument.

just like colors,make sure the array for sizes has the same length as arrays for x-axis and y-axis.

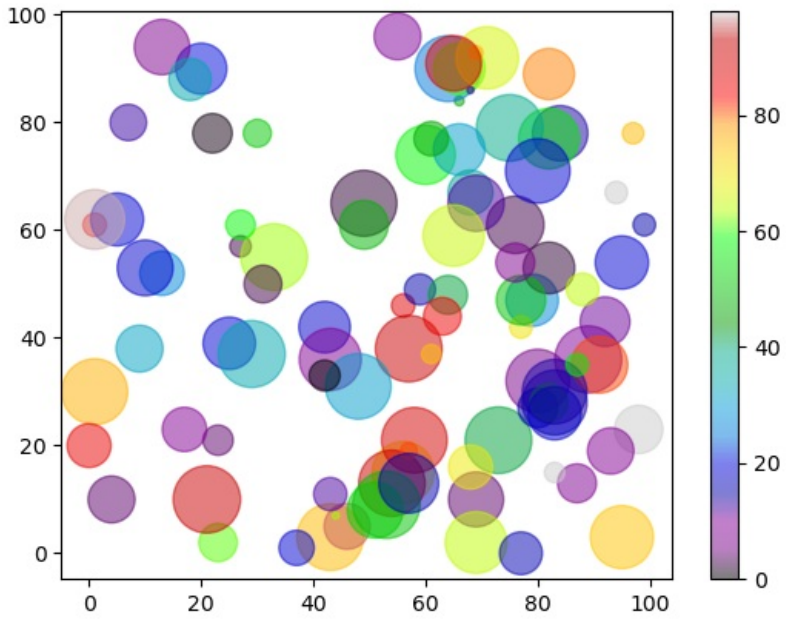
In [232…
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x,y,s=sizes,alpha=0.1,color='blue') #alpha how much the dark the color
plt.show()
```



In [ ]:
```
"combine color size and alpha"
```

we can combine a colormap with different sizes of the dots.this is best visualised if the dots are transaparent.

In [251…
```
x=np.random.randint(100,size=(100))
y=np.random.randint(100,size=(100))
colors=np.random.randint(100,size=(100))
sizes=10*np.random.randint(100,size=(100))
plt.scatter(x,y,c=colors,s=sizes,alpha=0.5,cmap='nipy_spectral')
plt.colorbar()
plt.show()
```



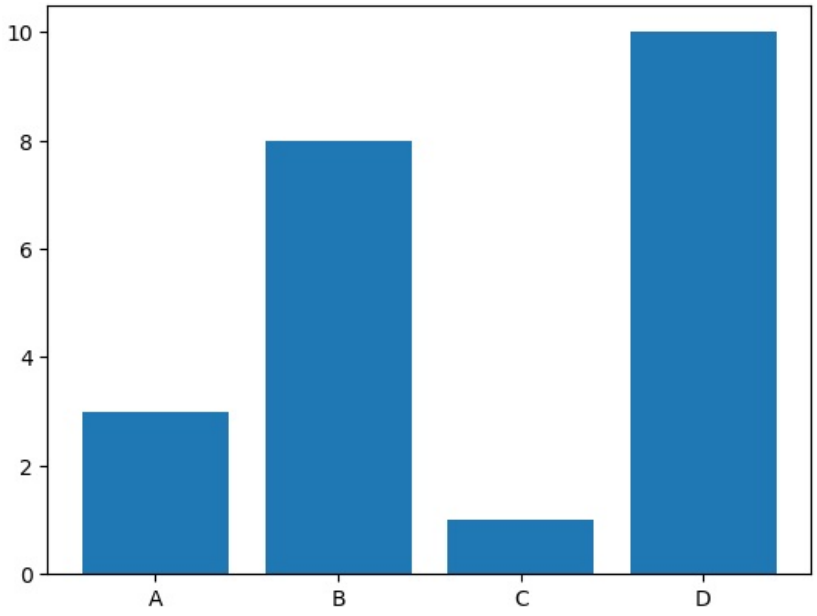
In [ ]:

In [ ]:
```
"MatplotLib Bars"
```

In [ ]:
```
"Creating Bars"
```

With pyplot,we can use the 'bar()' function to draw bar graphs.

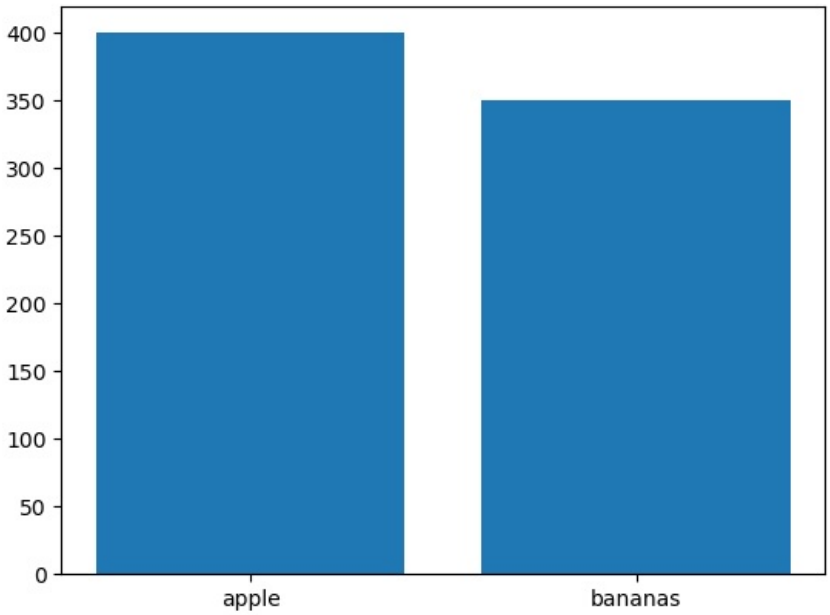
In [256…
```
import numpy as np
import matplotlib.pyplot as plt
x=np.array(['A','B','C','D'])
y=np.array([3,8,1,10])

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



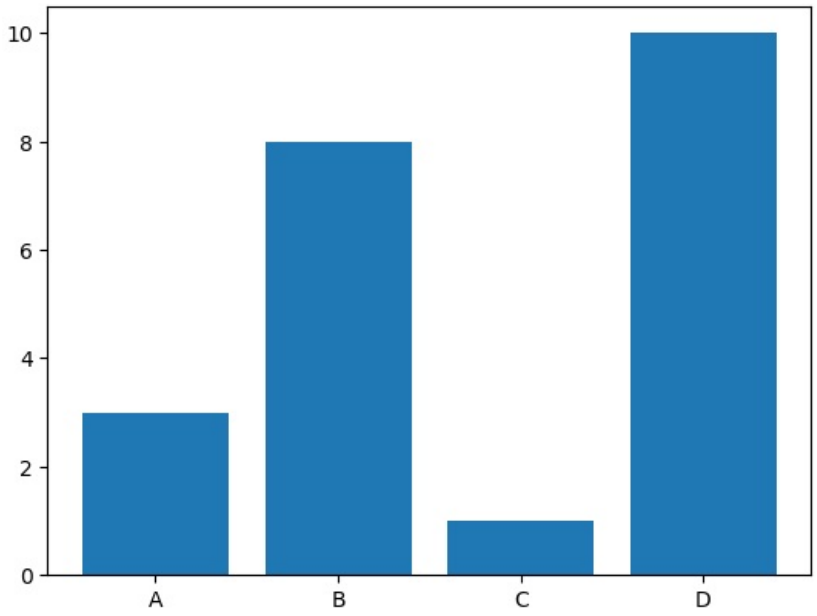
The 'bars()' function takes arguemenets that describes the layout of the bars.

The categories and their values represented by first and second arguement as arrays

In [261…
```
x=["apple","bananas"]
y=[400,350]
plt.bar(x,y)
plt.show()
```



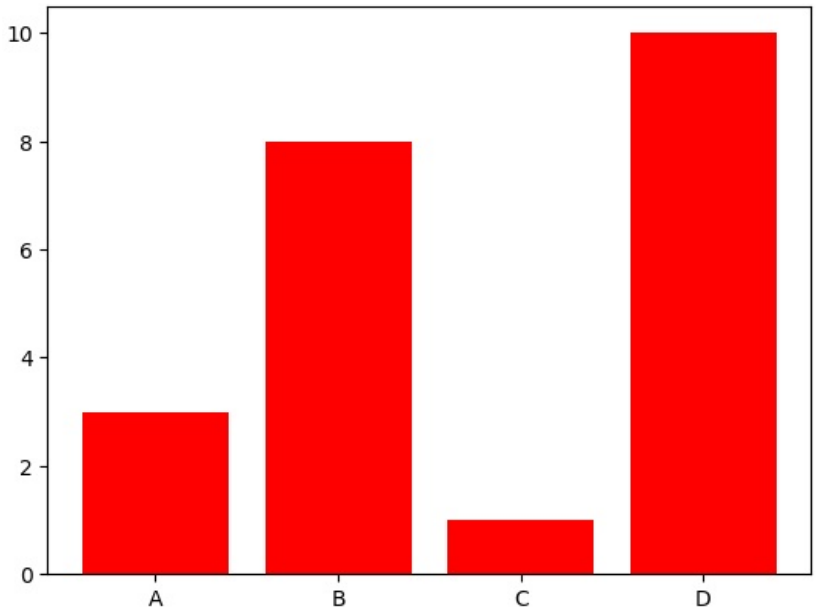
In [263…
```
x=np.array(["A","B","C","D"])
y=np.array([3,8,1,10])

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



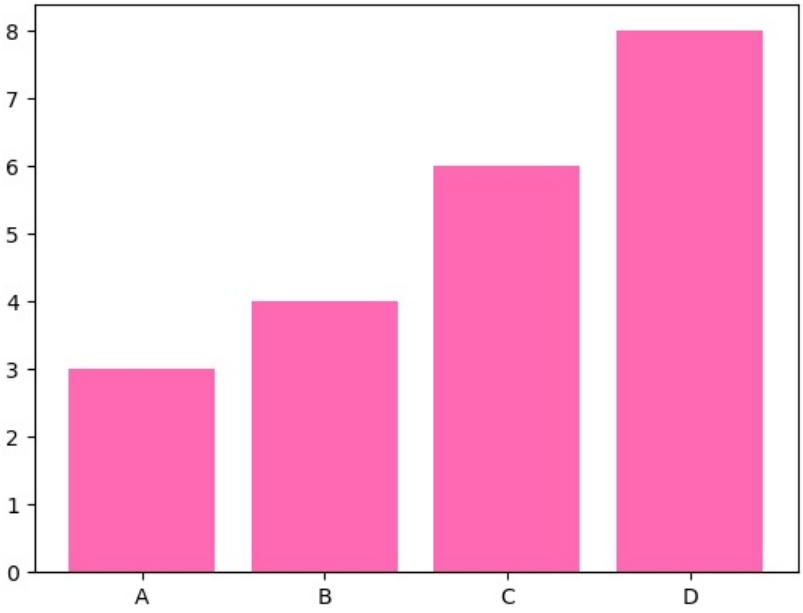
In [ ]:
```
"Bar Color "
```

The 'bar()' and 'barh()' take the keyword argument 'color' to set the color of the bars.

In [266…
```
x=np.array(["A","B","C","D"])
y=np.array([3,8,1,10])
plt.bar(x,y,color='red')
plt.show()
```

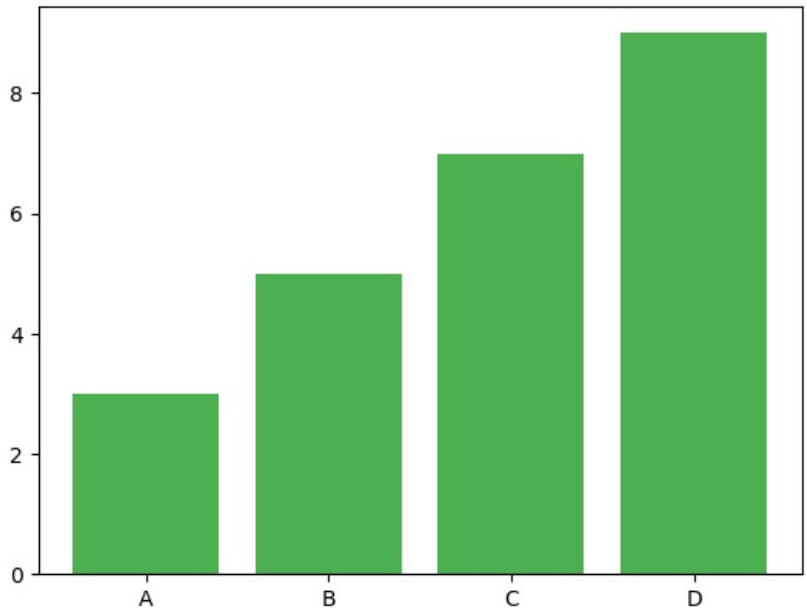


In [270…
```
x=np.array(["A","B","C","D"])
y=np.array([3,4,6,8])
plt.bar(x,y,color='hotpink')
plt.show()
```



In [ ]:
```
"Color Hex"
```

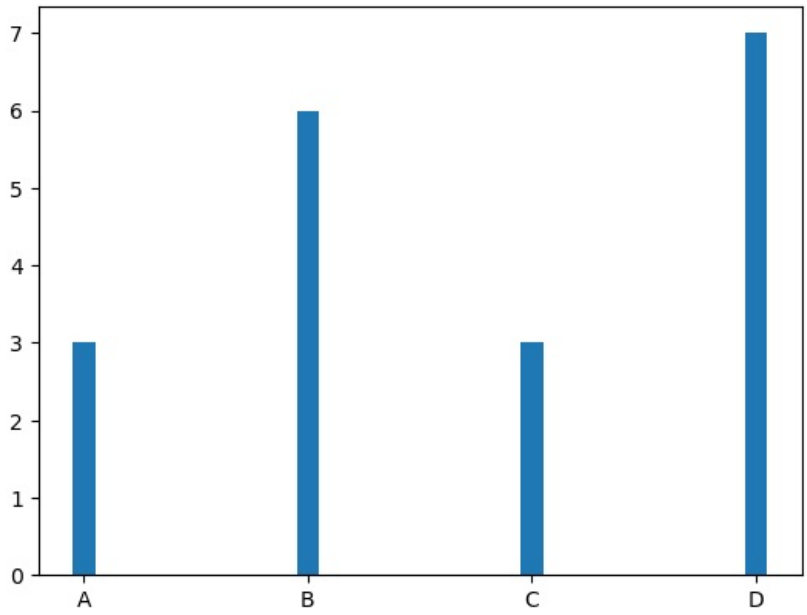
In [274…
```
x=np.array(["A","B","C","D"])
y=np.array([3,5,7,9])
plt.bar(x,y,color='#4CAF50')
plt.show()
```



In [ ]:
```
"Bar Width"
```

The 'bar()' takes the keyword argument 'width' to set the width of the bars.

In [279…
```
x=np.array(["A","B","C","D"])
y=np.array([3,6,3,7])
plt.bar(x,y,width=0.1)
plt.show()
```



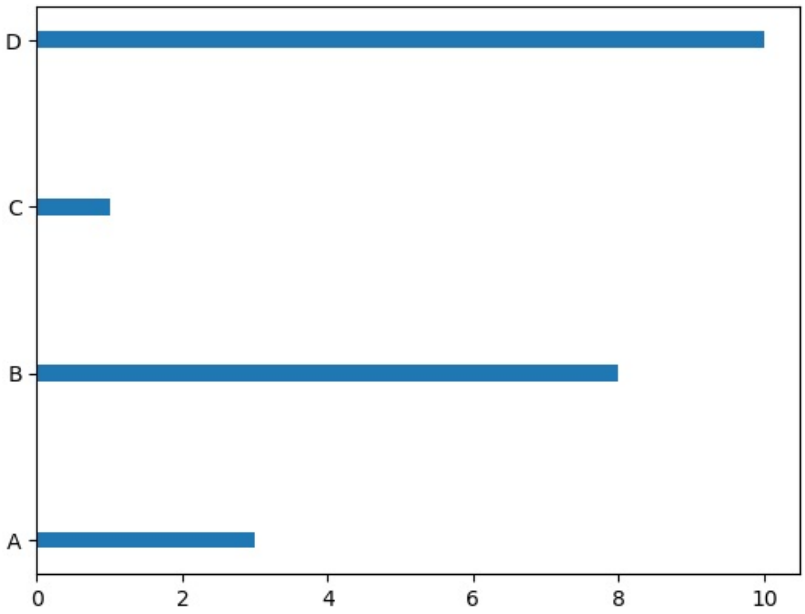
In [ ]:
```
#The default width value is 0.8
```

Note:For horizontal bars,use 'height' instead of 'width'

In [ ]:
```
"Bar height"
```

the 'barh()' takes the keyword argument 'height' to set the height of the bars:

In [289…
```
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x,y,height=0.1)
plt.show()
```



In [ ]:
```
"MatplotLib Histograms"
```

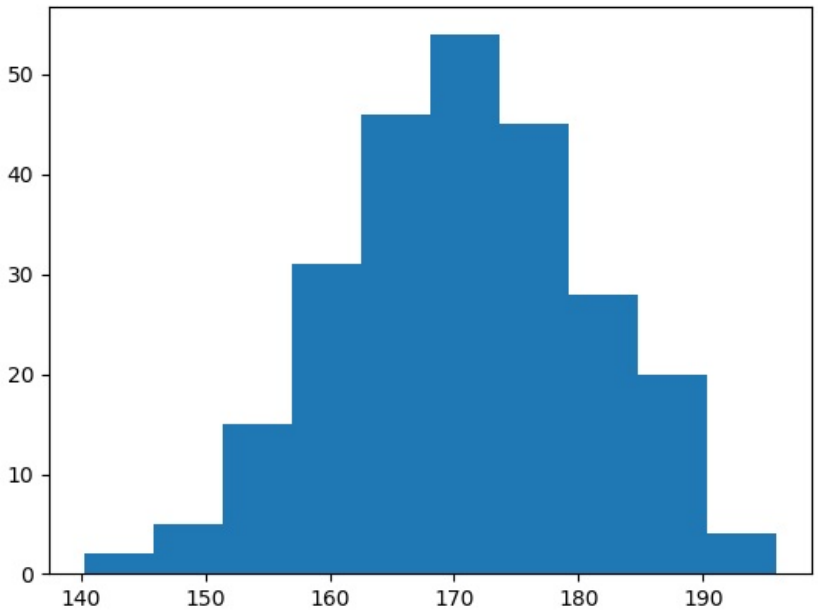
A histogram is a graph showing frequency ditsributions.

It is a graph showing the number of observations within each given interval.

example.Say you ask for the height of 250 people ,you might end up with a histogram like this:

In [292…
```
Image.open('img_matplotlib_histogram1.png')
```

Out[292…



In [ ]:
```
2 people from 140 to 145cm
5 people from 145 to 150cm
15 people from 151 to 156cm
31 people from 157 to 162cm
46 people from 163 to 168cm
53 people from 168 to 173cm
45 people from 173 to 178cm
28 people from 179 to 184cm
21 people from 185 to 190cm
4 people from 190 to 195cm
```

In [ ]:
```
"create histogram"
```

In MatpltLib,we can use the hist() function to create histograms.

The hist() fuction will use an array of numbers to create a histogram the array is sent into the function as an argument.

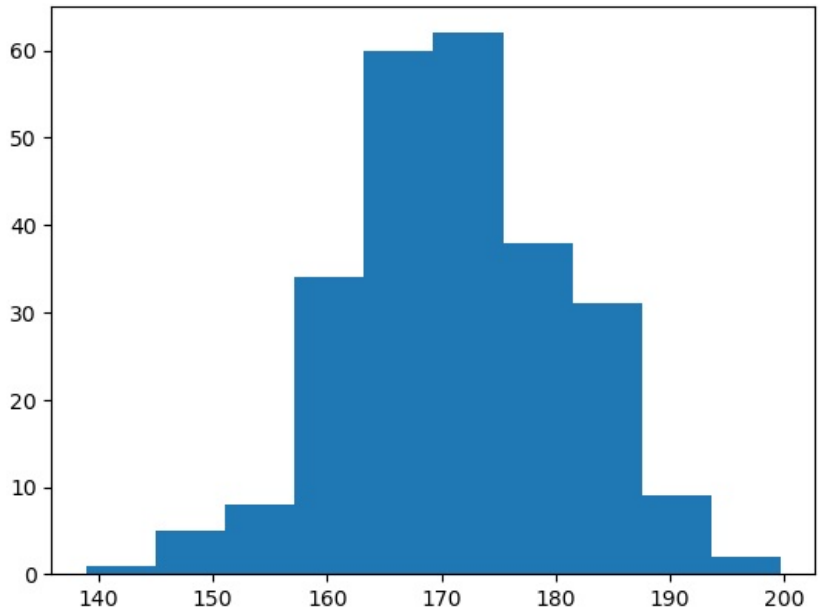
For simplicity we use NumPy to randomly generate an array with 250 values.where the values will concentrate around 170,and the standard deviation is 10.

In [301…
```
# A Normal Data Distribution by NumPy:

import numpy as np

x = np.random.normal(170, 10, 250)

print(x)
```

```
[180.93646285 159.18470587 172.34170212 174.21313298 162.29396405
 190.64523797 155.02755531 166.1885937  174.3116683  150.36056899
 170.11354196 182.12586138 171.25249738 167.21578982 166.22863556
 167.29528322 167.95904225 161.30728352 188.7508061  175.51920109
 163.67118859 179.08809084 178.73396561 168.30566482 158.98479711
 156.93433055 162.67570126 160.63813648 180.93089686 158.68375726
 177.51361987 175.11008506 173.21221441 166.20307142 189.46320878
 160.74258092 158.26665221 182.30853042 174.11164648 161.46494148
 190.70068726 179.09819041 165.93973449 171.13106283 166.10574562
 167.68918025 182.87022725 173.4894282  177.04576794 175.66257396
 166.37027639 170.75814691 163.01813327 157.46424951 173.33050384
 155.43734902 175.67622974 180.63049959 180.8797925  168.27355647
 160.12953845 154.18804277 168.48739514 167.94594022 152.1903446
 168.87742243 156.80075809 171.26741    175.26434197 165.85815788
 176.17871462 168.46568659 184.44318275 151.14749081 160.36072343
 182.74469112 183.16895199 179.45973424 175.69216806 181.79996577
 171.37959742 151.41728221 153.33815582 192.08585536 162.77366817
 179.42490743 179.09845255 161.18196227 171.72001179 166.73862945
 165.58649956 167.37372389 176.9348609  158.14105731 165.51190609
 167.81456883 168.35009306 162.83019666 170.53141796 162.96624042
 180.66491548 185.00038444 171.01026059 168.25096902 146.43740349
 169.61582145 159.32640009 167.75438884 170.08164082 180.41629758
 178.07782776 175.33044696 159.09796777 161.39388284 168.31771344
 164.50676545 171.82440901 175.48387471 161.1102781  158.68175735
 176.9794532  188.14133453 170.76180038 165.95129438 170.53084171
 171.83539355 165.77525847 165.10954577 163.96434287 169.65780754
 156.39501535 182.55282716 157.1523057  152.07271636 181.18409308
 167.32947501 178.54164457 177.31491377 145.40800485 179.03516663
 179.76912269 158.2898013  172.12259642 173.04402313 179.11042829
 148.46861436 156.07943268 175.07519094 163.84392785 165.129304
 175.89197212 162.52849137 170.56100639 173.2577     182.96735831
 160.64903539 167.33752896 156.48905435 161.29214823 171.72748786
 178.7901158  183.93408817 156.33733881 179.32922606 173.62140026
 176.43658949 178.89010227 174.94822813 172.77674299 175.39973346
 147.50387559 175.61333974 177.31656506 159.96026612 175.32914567
 182.82041559 167.25258584 165.15137835 145.32609145 167.47053986
 180.82557738 184.91086436 178.60817172 181.62442628 160.89366399
 165.50265826 160.02769533 177.92787717 168.11356589 156.8610681
 168.20882009 171.19080959 155.8181746  179.35293839 178.66954466
 180.34138414 164.96261544 177.85461893 176.12524038 162.90236733
 170.95057857 172.97140851 166.75436907 160.43055701 166.07149801
 180.9792448  170.88131643 160.21486004 176.3763182  166.37196874
 173.52253793 160.44459167 165.47882827 174.30077748 157.57211781
 184.15552345 163.06969633 184.86546217 154.85514535 160.52964781
 163.39522461 182.6664321  183.41430476 168.1119417  166.41079956
 175.90173528 179.26770357 181.91647759 172.55232052 179.72062184
 176.39385445 160.02323724 180.85767072 182.02304234 163.43547436
 173.16060774 158.12466161 187.12388887 160.19521552 173.78327275
 172.92485407 162.59023647 169.95709077 197.34060436 177.61696563
 171.45448448 161.48157201 168.88101084 175.00801651 169.09780634]
```

the 'hist()' function will read the array and produce a histogram.

In [306…
```
import matplotlib.pyplot as plt
import numpy as np
x=np.random.normal(170,10,250)
plt.hist(x)
plt.show()
```



In [ ]:
```
"Matplotlib Pie Charts"
```

In [ ]:
```
"Creating Pie charts"
```

In [ ]:
```
with pyplot, we can use the 'pie()' function to draw pie charts
```

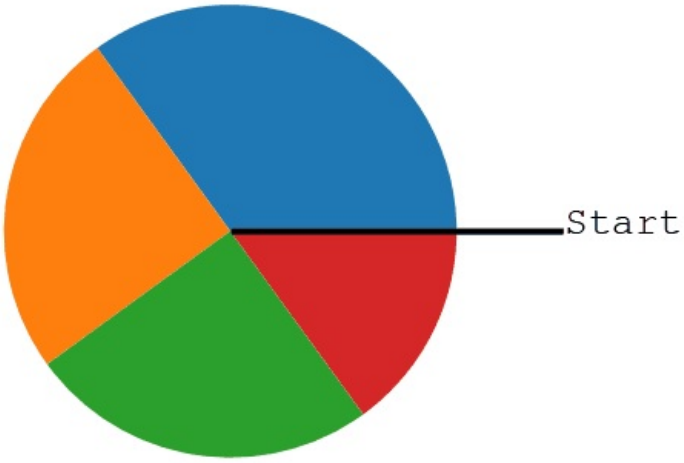
In [310…
```
y=np.array([12,21,24,43])
plt.pie(y)
plt.show()
```

As you can see the pie chart draws one piece (called a wedge) for each value in the array (in this case [35, 25, 25, 15]).

By default the plotting of the first wedge starts from the x-axis and moves counterclockwise:

In [313…
```
Image.open('img_matplotlib_pie_start.png')
```

Out[313…



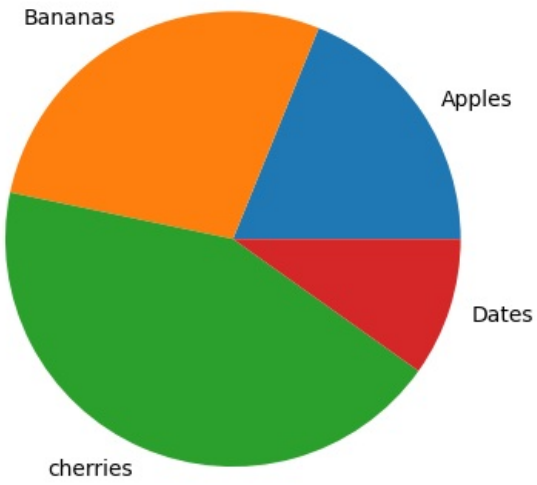
In [ ]:
```
Note: the size of each wedge is determined by comparing the value with all other values,by using this formula;

The value divided by the sum of all vaues: x/sum(x).
```

In [ ]:
```
"Labels"
```

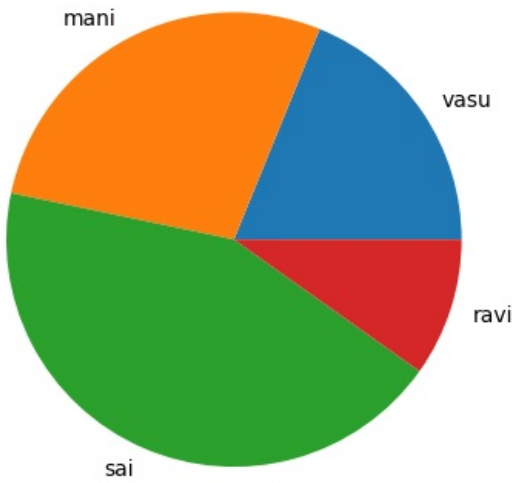
Add Lebels to the pie chart with the'labels' parameter

The 'labels' parameter must be an array with one label for each wedge:

In [316…
```
y=np.array([23,34,53,12])
mylabels=["Apples","Bananas","cherries","Dates"]
plt.pie(y,labels=mylabels)
plt.show()
```



In [320…
```
plt.pie(y,labels=["vasu","mani","sai","ravi"])
plt.show()
```



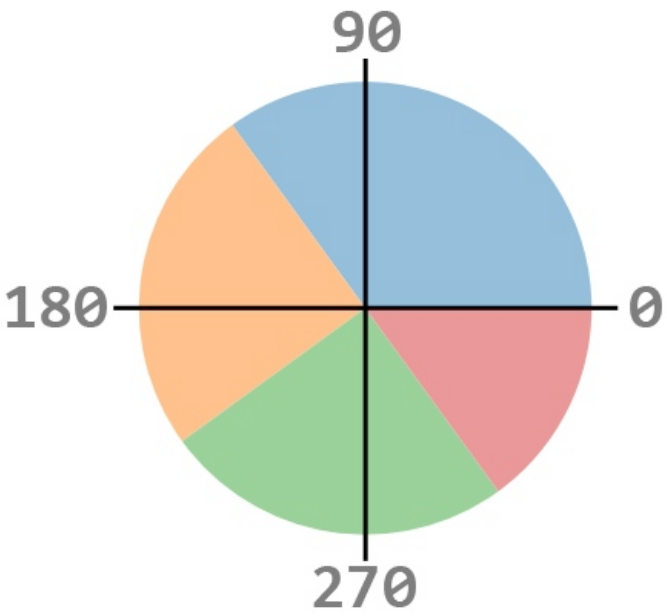
In [ ]:
```
"Start Angle"
```

As mentioned the default start angle is at the x-axis,but you can change the start angle by specifying a 'startangle' parameter

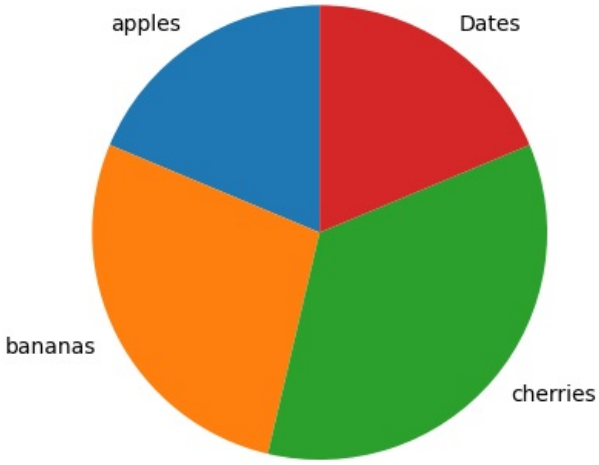
the 'startangle' parameter is defined with an angle in degrees ,default anglw is 0.

In [325…
```
Image.open('img_matplotlib_pie_angles.png')
```

Out[325…



In [327…
```
# Start the first wedge at 90 degrees

y=np.array([23,34,43,23])
mylabels=["apples","bananas","cherries","Dates"]
plt.pie(y,labels=mylabels,startangle=90)
plt.show()
```



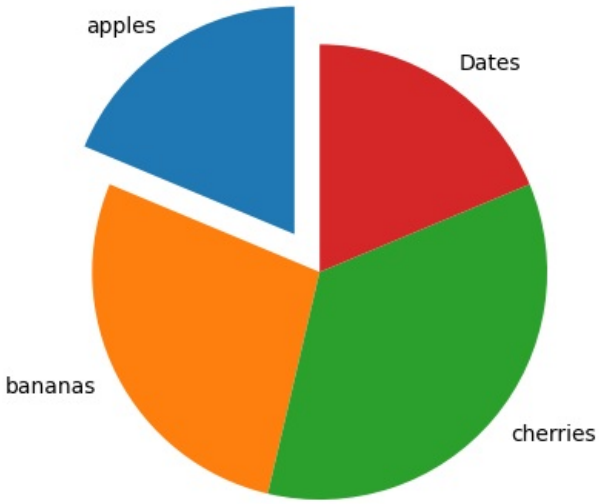
In [ ]:
```
"Explode"
```

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

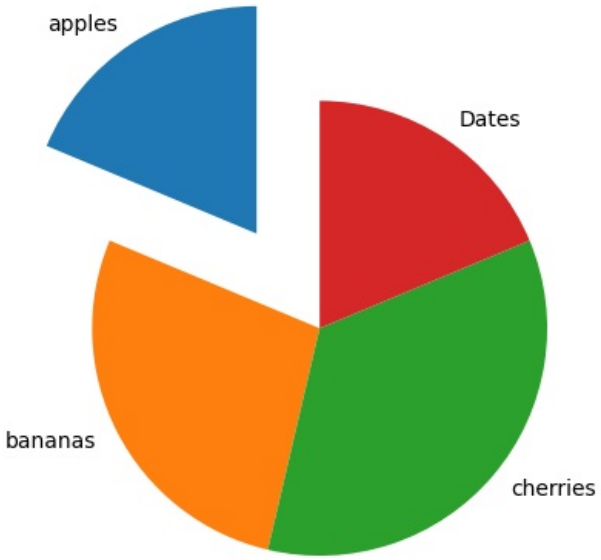
The 'explode' parameter ,if specified and not 'NOne' ,must be an array with one value for each wedge.

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

In [344…
```
#Pull the "Apples" wedge 0.2 from the center of pie:


y=np.array([23,34,43,23])
mylabels=["apples","bananas","cherries","Dates"]
myexplode=[0.2,0,0,0]
plt.pie(y,labels=mylabels,startangle=90,explode=myexplode)
plt.show()
```



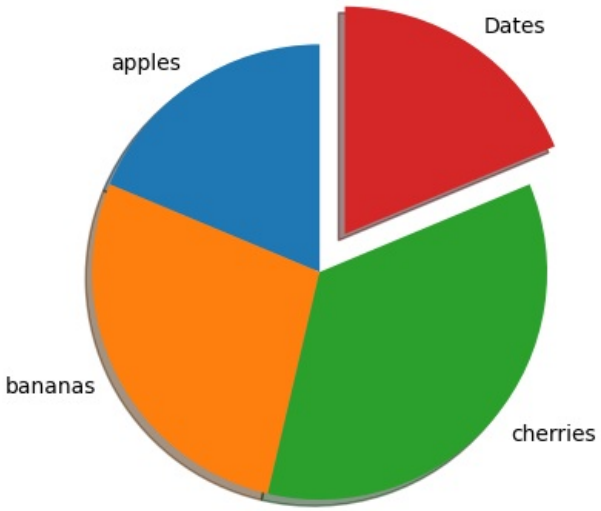
In [348…
```
y=np.array([23,34,43,23])
mylabels=["apples","bananas","cherries","Dates"]
plt.pie(y,labels=mylabels,startangle=90,explode=[0.5,0,0,0])
plt.show()
```



In [ ]:
```
"Shadow"
```

Add a shadow to the pie chart by setting the 'shadows' parameter to 'True':

In [355…
```
y=np.array([23,34,43,23])
mylabels=["apples","bananas","cherries","Dates"]
plt.pie(y,labels=mylabels,startangle=90,explode=[0,0,0,0.2],shadow=True)
plt.show()
```

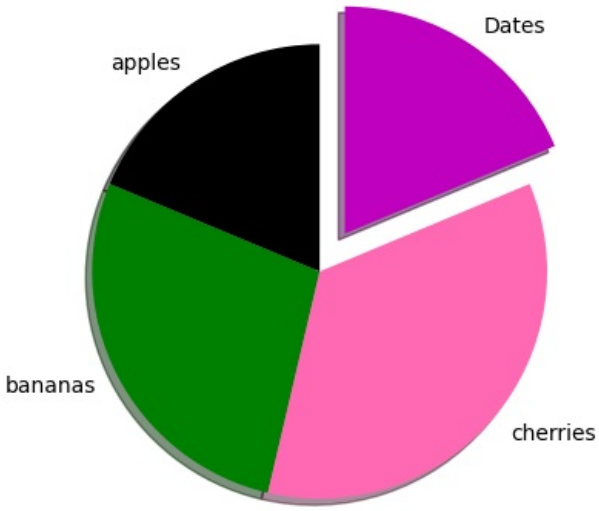


In [ ]:
```
"Colors"
```

we can set the color of each wedge with the 'colors' parameter.

The 'colors' parameter ,if specified ,must be an array with one value for each wedge.

In [358…
```
y=np.array([23,34,43,23])
mylabels=["apples","bananas","cherries","Dates"]
mycolors=["black","green","hotpink","m"]
plt.pie(y,labels=mylabels,startangle=90,explode=[0,0,0,0.2],shadow=True,colors=mycolors)
plt.show()
```

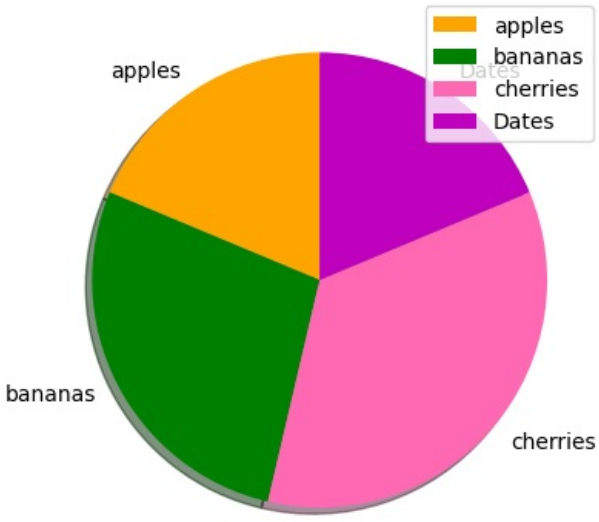


In [ ]:
```
'r' - Red
'g' - Green
'b' - Blue
'c' - Cyan
'm' - Magenta
'y' - Yellow
'k' - Black
'w' - White
```

In [ ]:
```
"Legend"
```

To add a list of explanations for each wedge,use the 'legend()' function

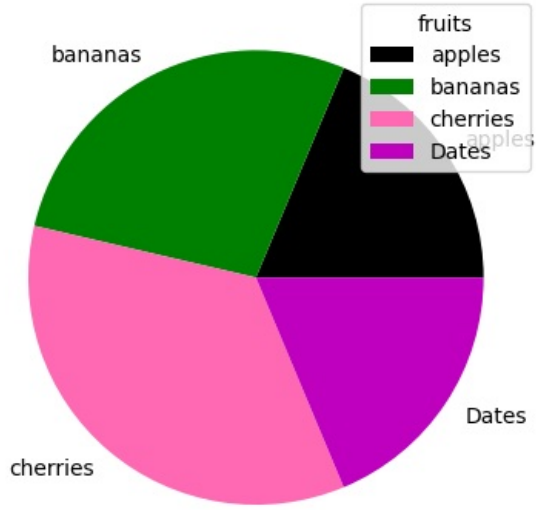
In [367…
```
y=np.array([23,34,43,23])
mylabels=["apples","bananas","cherries","Dates"]
mycolors=["orange","green","hotpink","m"]
plt.pie(y,labels=mylabels,startangle=90,shadow=True,colors=mycolors)
plt.legend()
plt.show()
```



In [ ]:
```
"Legend with header"
```

TO add the title for the 'legend' ,add the 'title' parameter to the 'legend' function

In [380…
```
y=np.array([23,34,43,23])
mylabels=["apples","bananas","cherries","Dates"]
mycolors=["black","green","hotpink","m"]
plt.pie(y,labels=mylabels,colors=mycolors)
plt.legend(title="fruits")
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: