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 pit alias: import matplotlib.pyplot as plt Now the Pyplot package can be referred to as plt.

Syntax for importing matplotlib

import matplotlib.pyplot as pit

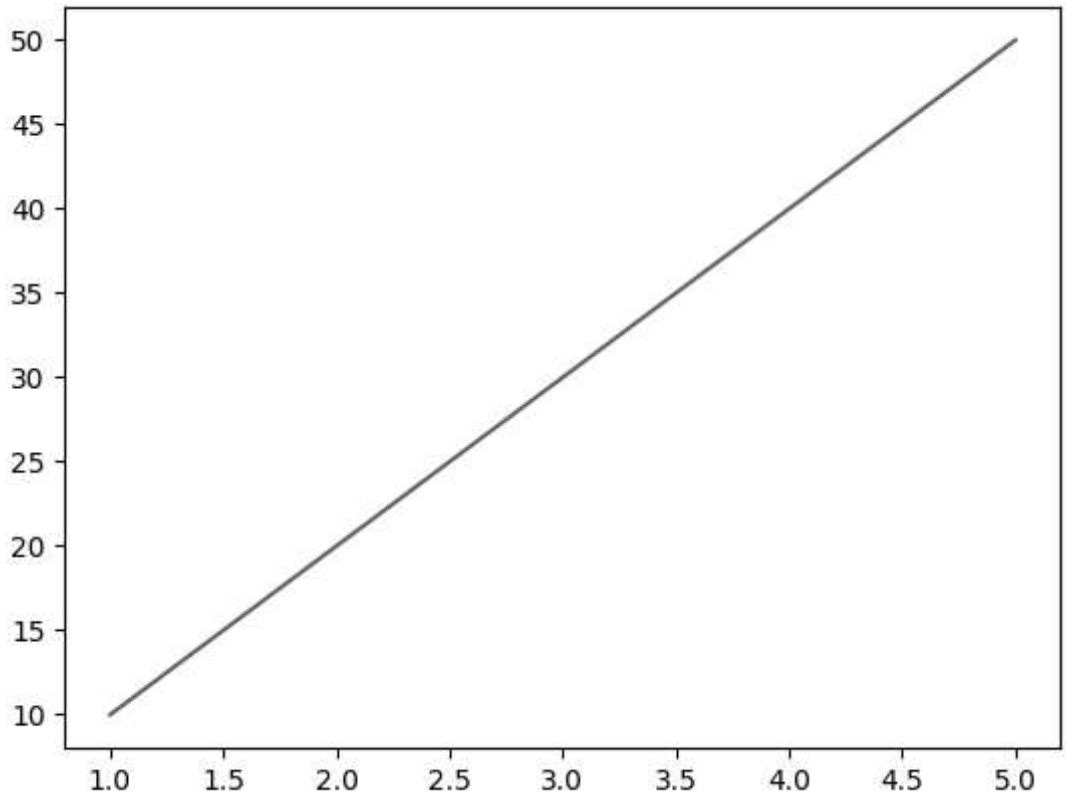
Syntax for plotting the graph

plt.plot(x,y)

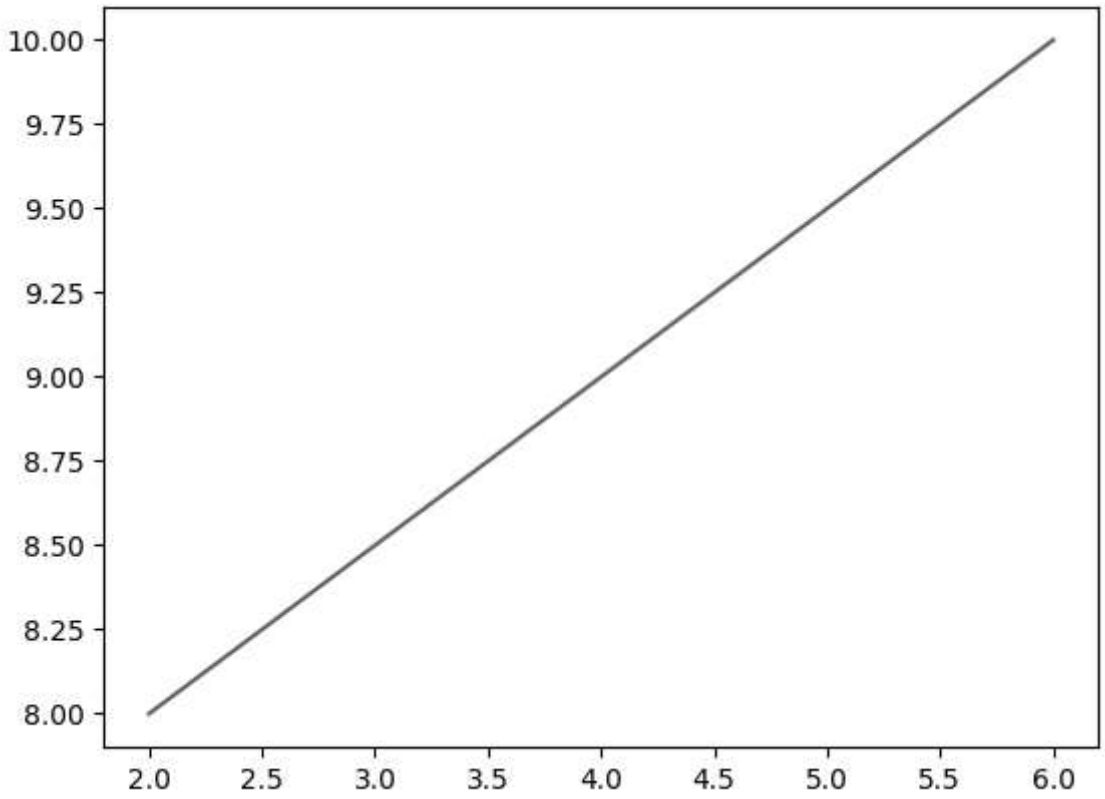
Syntax for showing the plot:

plt.show()

In [2]:
```
###Simple Line Plot
#import matplotlibrary
import matplotlib.pyplot as plt
#Giving data to variables x and y
x=[1,2,3,4,5]
y=[10,20,30,40,50]
#plotting the graph for x versus y
plt.plot(x,y)
#Show the plot
plt.show()
```



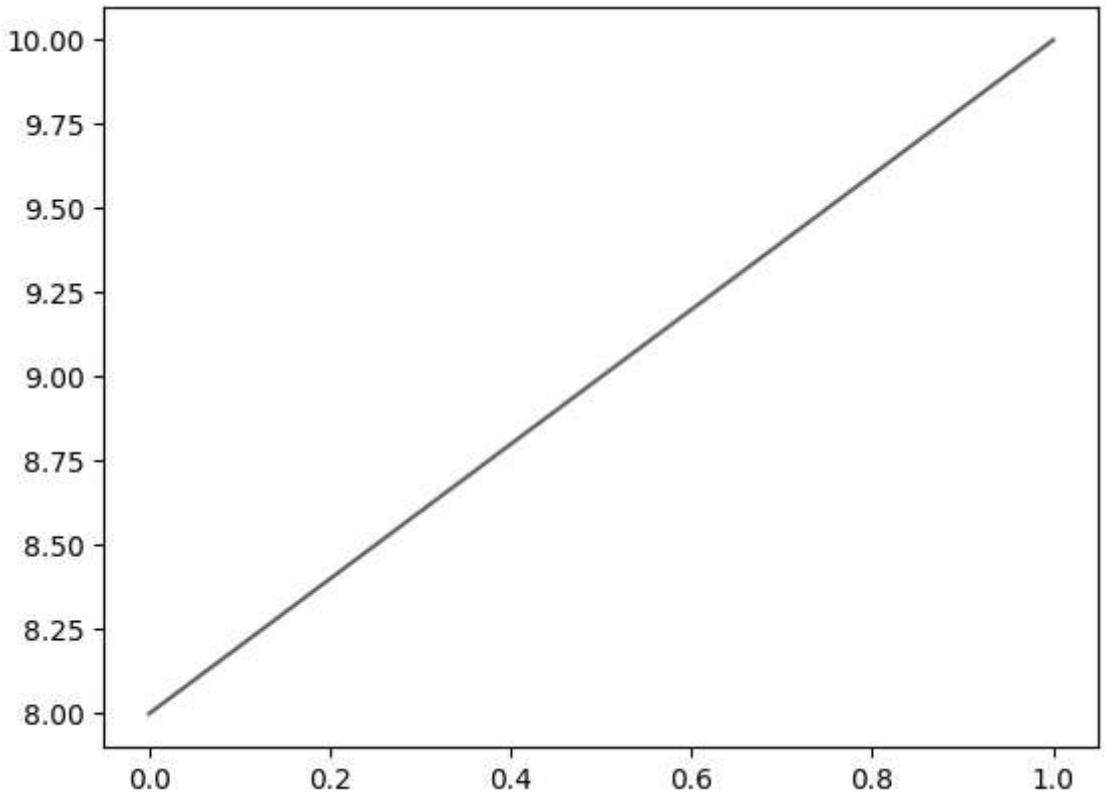
In [2]:
```
#Draw a Line in a diagram from position (2,8) to position (6,10):
#import matplotlibrary and numpy
import matplotlib.pyplot as plt
import numpy as np
#Giving data to variables x and y in arrays
xpoints=np.array([2,6])
ypoints=np.array([8,10])
#plotting the graph for x versus y
#In case of this plotting the default marker is nothing or none as nothing is specifie
plt.plot(xpoints,ypoints)
#Show the plot. The syntax of plot showing the graph is
plt.show()
```



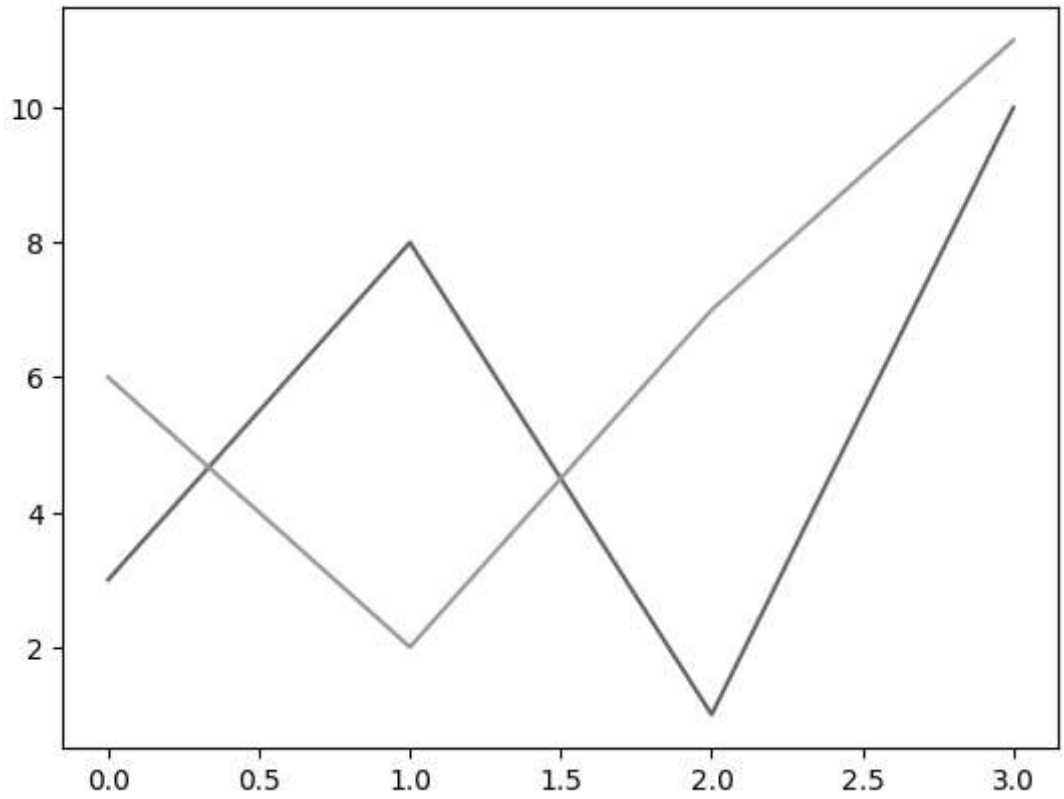
Default X-Points:

If we donot specify the points in teh x-axis, they will get default values 0,1,2,3,etc. depends on the length of y-points

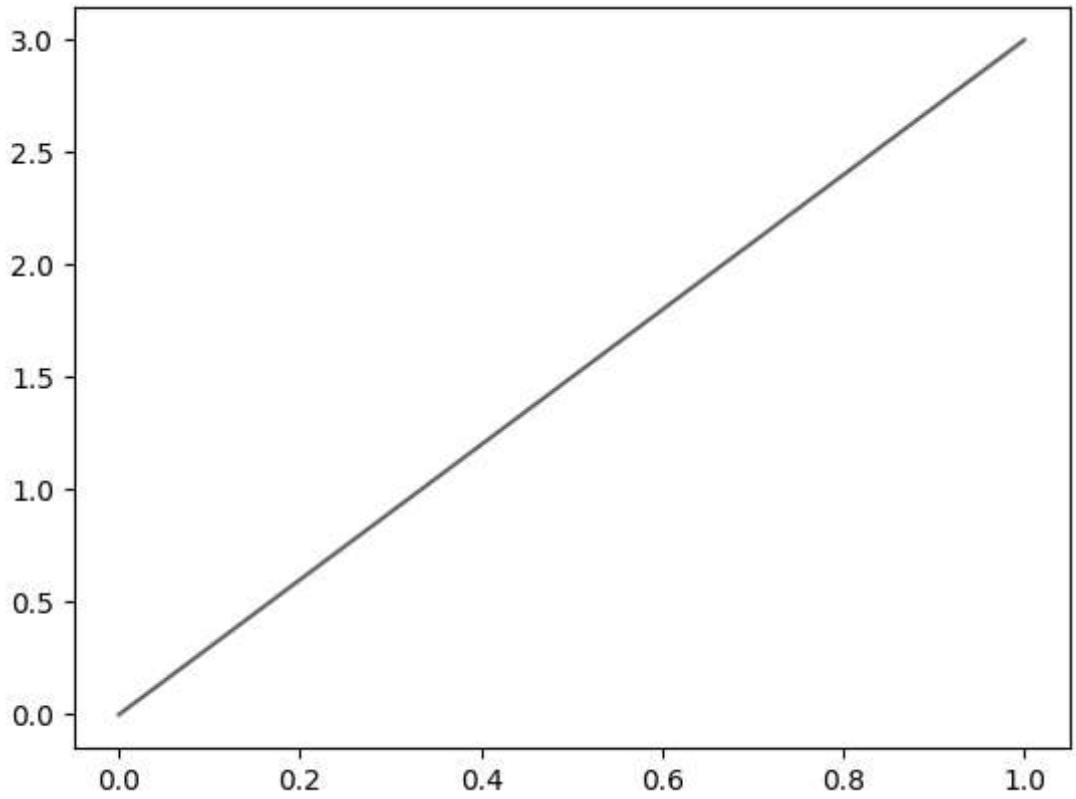
In [23]:
```
#Draw a Line in a diagram to point y axis as (8,10) with default x axis points.
#import matplotlibrary and numpy
import matplotlib.pyplot as plt
import numpy as np
#Giving data to variables x and y in arrays

ypoints=np.array([8,10])
#plotting the graph for x versus y
#In case of this plotting the default marker is nothing or none as nothing is specifie
plt.plot(ypoints)
#Show the plot. The syntax of plot showing the graph is
plt.show()
```



In [25]:
```
#Draw two Lines in a diagram with y1=[3,8,1,10] and y2=[6,2,7,11] with x points as def
#import matplotlibrary and numpy
import matplotlib.pyplot as plt
import numpy as np
#Giving data to variables y1 and y2 in arrays

y1=np.array([3,8,1,10])
y2=np.array([6,2,7,11])
#plotting the graph for x versus y
#In case of this plotting the default marker is nothing or none as nothing is specifie
plt.plot(y1)
plt.plot(y2)
#Show the plot. The syntax of plot showing the graph is
plt.show()
```



In [24]:
```
#Draw a Line in a diagram from point(0,0) i.e. origin to point(5,3)
#import matplotlibrary and numpy
import matplotlib.pyplot as plt
import numpy as np
#Giving data to variables x and y in arrays
xpoints=np.array([0,5])
ypoints=np.array([0,3])
#plotting the graph for x versus y
#In case of this plotting the default marker is nothing or none as nothing is specifie
plt.plot(ypoints)
#Show the plot. The syntax of plot showing the graph is
plt.show()
```



Plot with different markers

The matplotlib.dates module provides functions to handle markers in Matplotlib. It is used both by the marker functionality of the plot and scatter. The default marker is None (means nothing)

marker	description
"."	point
","	pixel
"o"	circle
"v"	triangle_down
"^"	triangle_up
"<"	triangle_left
">"	triangle_right
"1"	tri_down
"2"	tri_up
"3"	tri_left
"4"	tri_right
"8"	octagon
"s"	square
"p"	pentagon
"P"	plus (filled)
"*"	star
"h"	hexagon1
"H"	hexagon2
"+"	plus
"x"	x
"X"	x (filled)
"D"	diamond
"d"	thin_diamond
"|"	vline
"_"	hline
TICKLEFT	tickleft

marker	description
TICKRIGHT	tickright
TICKUP	tickup
TICKDOWN	tickdown
CARETLEFT	caretleft (centered at tip)
CARETRIGHT	caretright (centered at tip)
CARETUP	caretup (centered at tip)
CARETDOWN	caretdown (centered at tip)
CARETLEFTBASE	caretleft (centered at base)
CARETRIGHTBASE	caretright (centered at base)
CARETUPBASE	caretup (centered at base)
"None", " " or ""	nothing
'$...$'	render the string using mathtext.
verts	a list of (x, y) pairs used for Path vertices. The center of the marker is located at (0,0) and the size is normalized.
path	a *Path* instance.
(numsides, style, angle)	The marker can also be a tuple (numsides, style, angle), which will create a custom, regular symbol. **numsides:** the number of sides **style:** the style of the regular symbol: **0** a regular polygon **1** a star-like symbol **2** an asterisk **3** a circle (numsides and angle is ignored) **angle:** the angle of rotation of the symbol

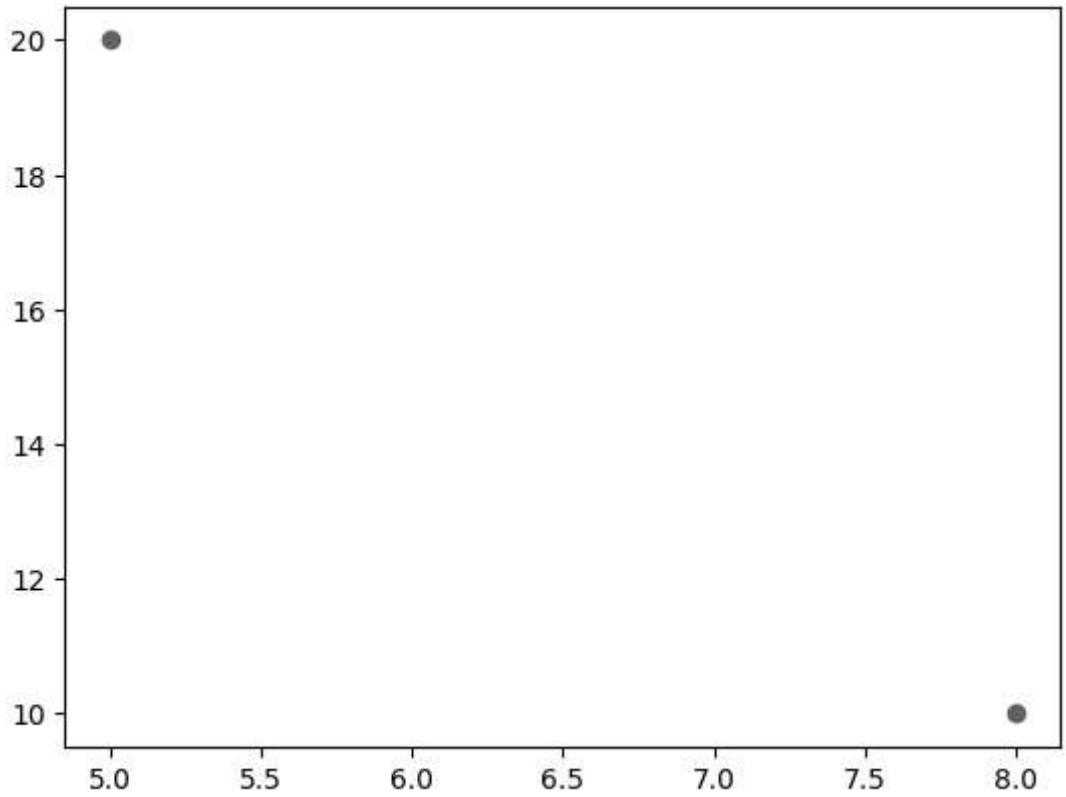
Syntax for marker is:

plt.plot(xpoints,ypoints,'o')

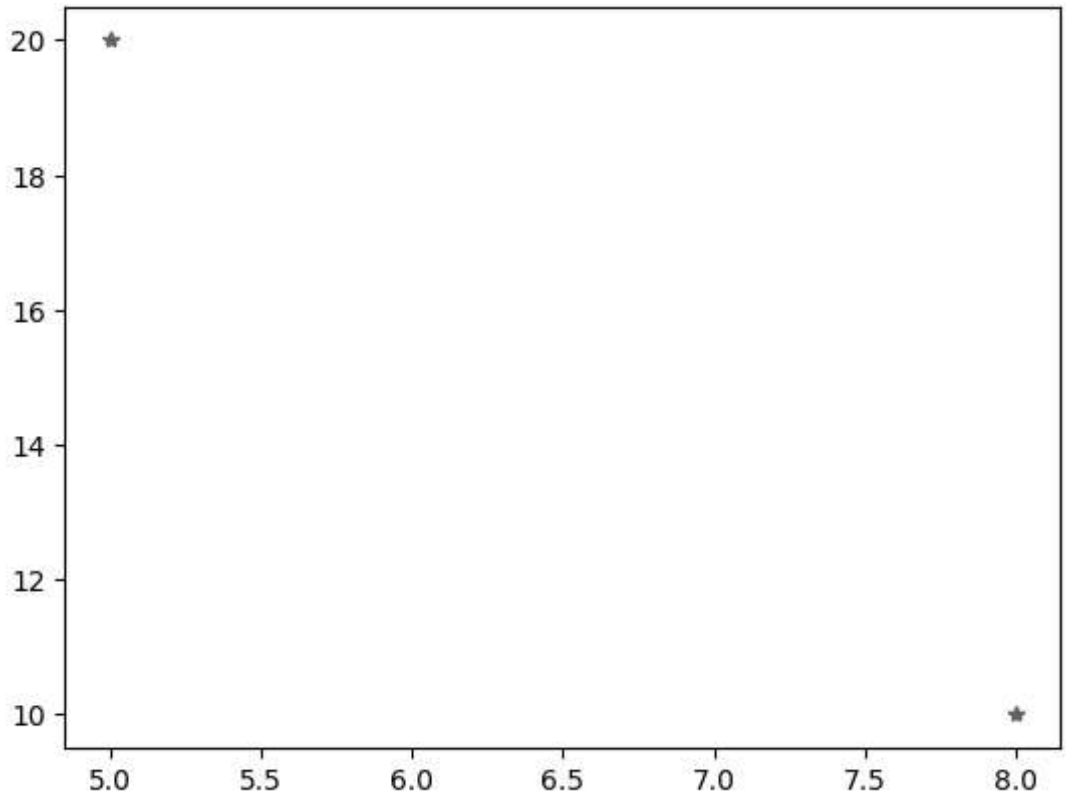
Marker size can also be defined by ms in syntax like:

plt.plot(xpoints,ypoints,'o',ms=10)

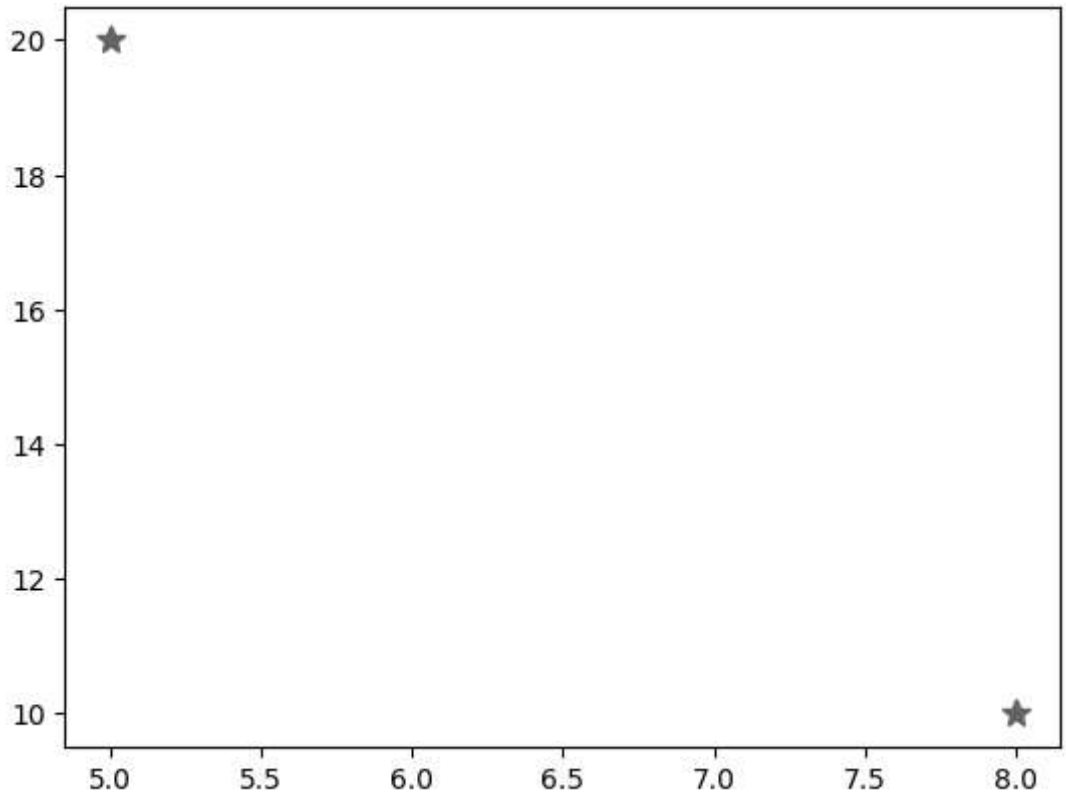
In [6]:
```
#Example:
#Draw a line in a diagram from position (8,10) to (5,20)
#import matplotlibrary and numpy
import matplotlib.pyplot as plt
import numpy as np
#Giving data to variables x and y in arrays
xpoints=np.array([8,5])
ypoints=np.array([10,20])
#plotting the graph for x versus y
plt.plot(xpoints,ypoints,'o')
#plot two points as marker is given dot and the linestyle is none
#If in a plot there is linestyle specified then it will plot.
#As in a previous plot there is a default line style as solid line and marker was not
#Show the plot. The syntax of plot showing the graph is
plt.show()
```



In [7]:
```
#Example with different marker *
#Draw a line in a diagram from position (8,10) to (5,20)
#import matplotlibrary and numpy
import matplotlib.pyplot as plt
import numpy as np
xpoints=np.array([8,5])
ypoints=np.array([10,20])
plt.plot(xpoints,ypoints,'*')
#plot two points as marker is given dot and the linestyle is none
#If in a plot there is linestyle specified then it will plot.
#As in a previous plot there is a default line style as solid line and marker was not
plt.show()
```



In [20]:
```
#Example with different marker * and marker size ms=10
#Draw a line in a diagram from position (8,10) to (5,20)
#import matplotlibrary and numpy
import matplotlib.pyplot as plt
import numpy as np
xpoints=np.array([8,5])
ypoints=np.array([10,20])
plt.plot(xpoints,ypoints,'*',ms=10)
#plot two points as marker is given dot and the linestyle is none
#If in a plot there is linestyle specified then it will plot.
#As in a previous plot there is a default line style as solid line and marker was not
plt.show()
```



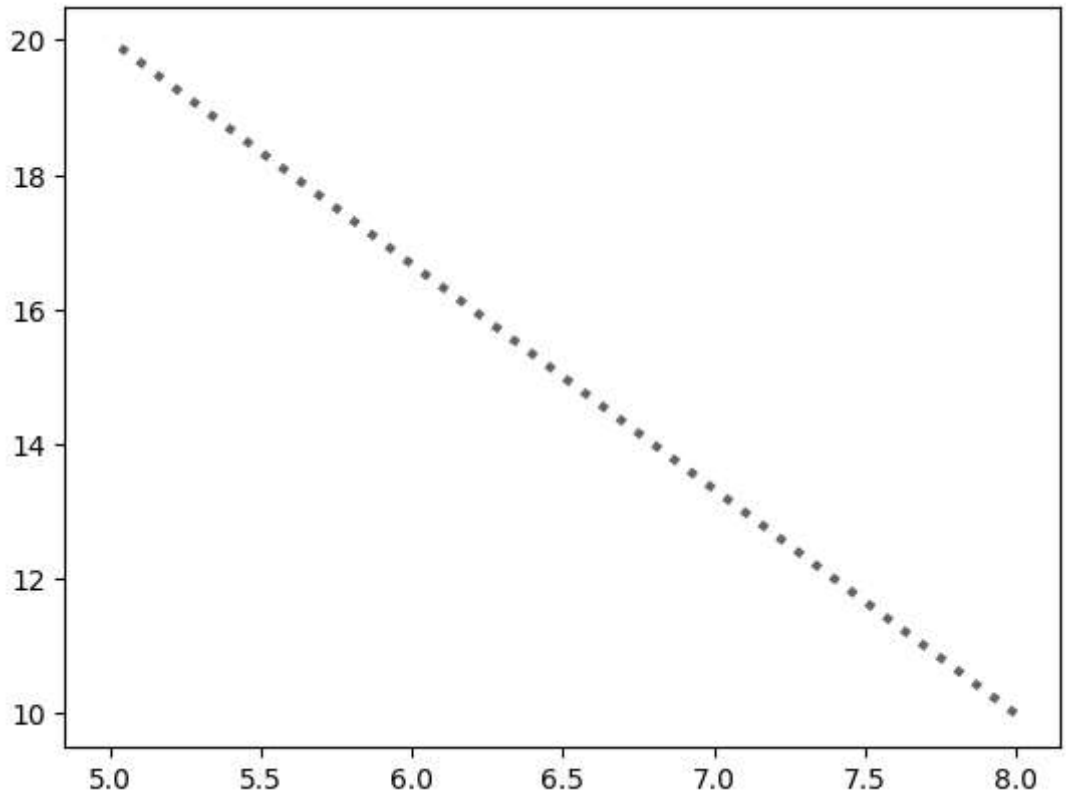
Plot Linestyle

You can choose any of these styles:

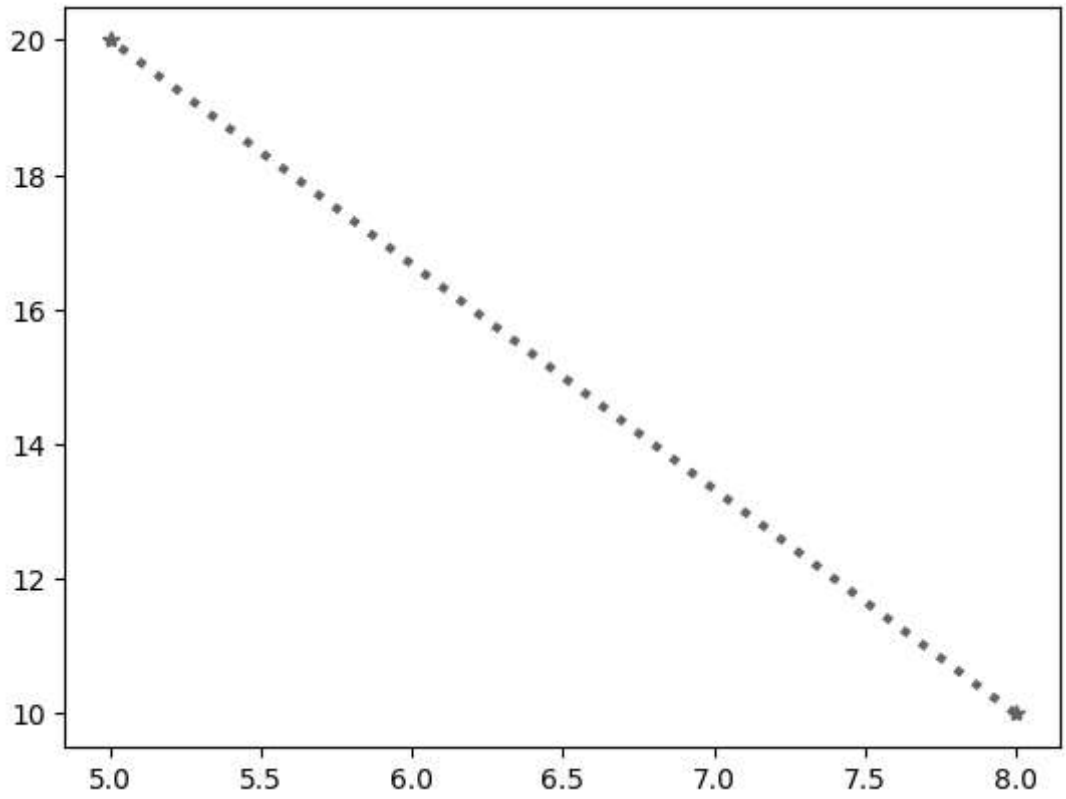
'solid' (default) '-' 'dotted' ':' 'dashed' '--' 'dashdot' '-.' 'None' '' or ' Syntax for linestyle:

plt.plot(x1,y1,linestyle=':',linewidth=3) Here Linewidth indicates the width of line whichever linestyle out of above will be selected.

In [9]:
```
#Example: (Same example with linestyle and linewidth)
#Draw a line in a diagram from position (8,10) to (5,20)
#import matplotlibrary and numpy
import matplotlib.pyplot as plt
import numpy as np

xpoints=np.array([8,5])
ypoints=np.array([10,20])

plt.plot(xpoints,ypoints,linestyle=':',linewidth=3)
#plot two points as linestyle is dotted and marker is none.
#If in a plot there is linestyle specified then it will plot.
#As in a plot there is a default line style as solid line and marker was not specifiea

plt.show()
```



In [13]:
```
#Example: (Same example with markers, linestyle and linewidth)
#Draw a line in a diagram from position (8,10) to (5,20)
#import matplotlibrary and numpy
import matplotlib.pyplot as plt
import numpy as np

xpoints=np.array([8,5])
ypoints=np.array([10,20])

plt.plot(xpoints,ypoints,'*',linestyle=':',linewidth=3)
#plot two points as linestyle is dotted and marker is none.
#If in a plot there is linestyle specified then it will plot.
#As in a plot there is a default line style as solid line and marker was not specified

plt.show()
```



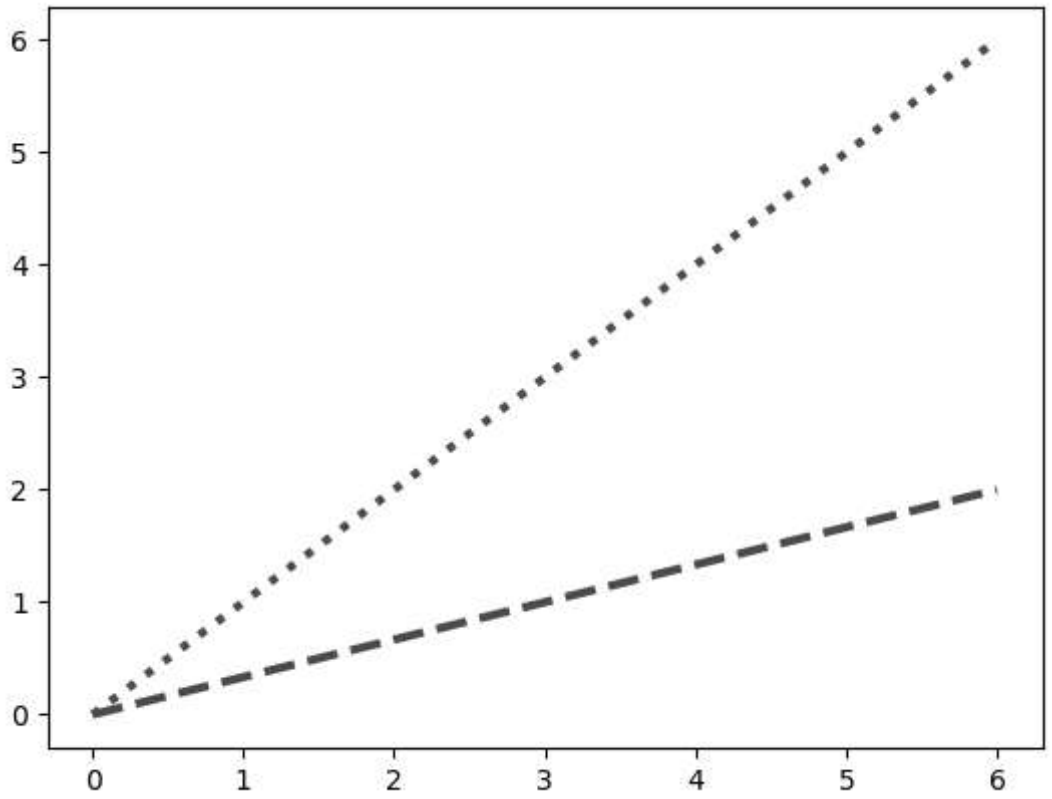
Plot with different colors

'r' Red,
'g' Green, 'b' Blue, 'c' Cyan, 'm' Magenta, 'y' Yellow, 'k' Black, 'w' White. Default color is 'b' (Blue).
Syntax for plotting graph with different color is:

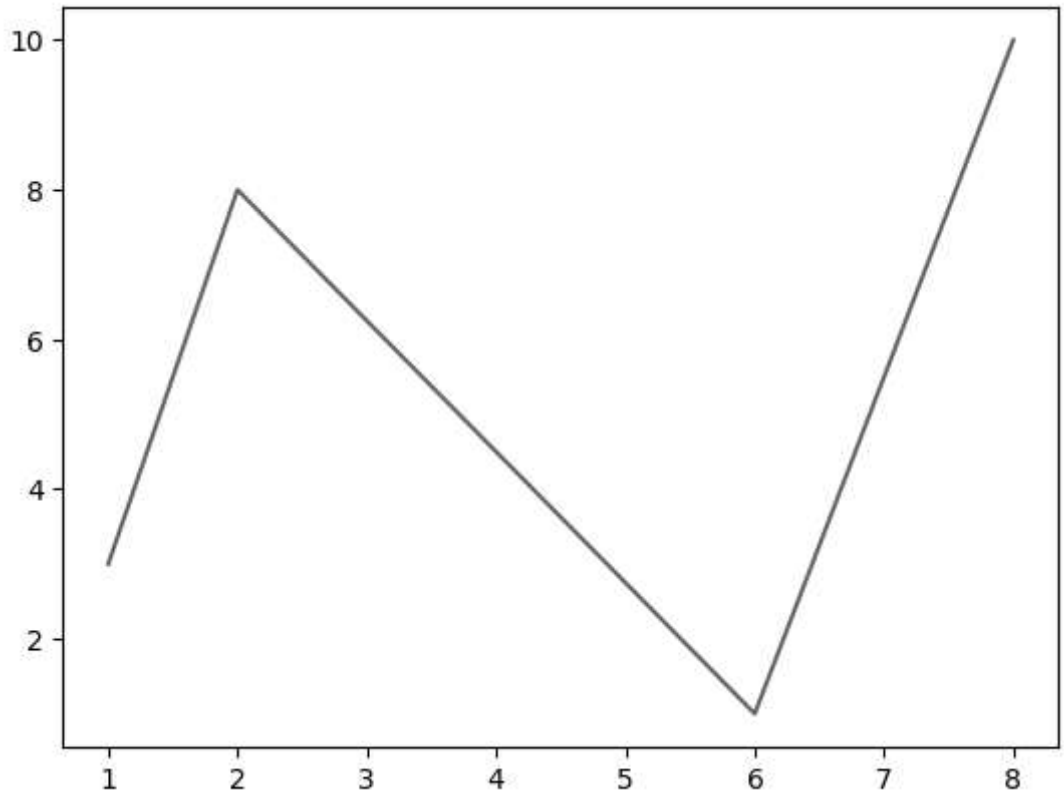
plt.plot(x,y,color="red")

The main purpose of providing color is that if in a graph more than one field is to be represented then different colors will be useful to differentiate. For example a plot for a student's marks for different five subjects and four exams.

In [19]:
```
#Example for providing different colors and multiple lines in a single plot
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.plot(x1,y1,color="red",linestyle=":",linewidth=3)
plt.plot(x1,y2,color="green",linestyle="--",linewidth=3)
plt.show()
```



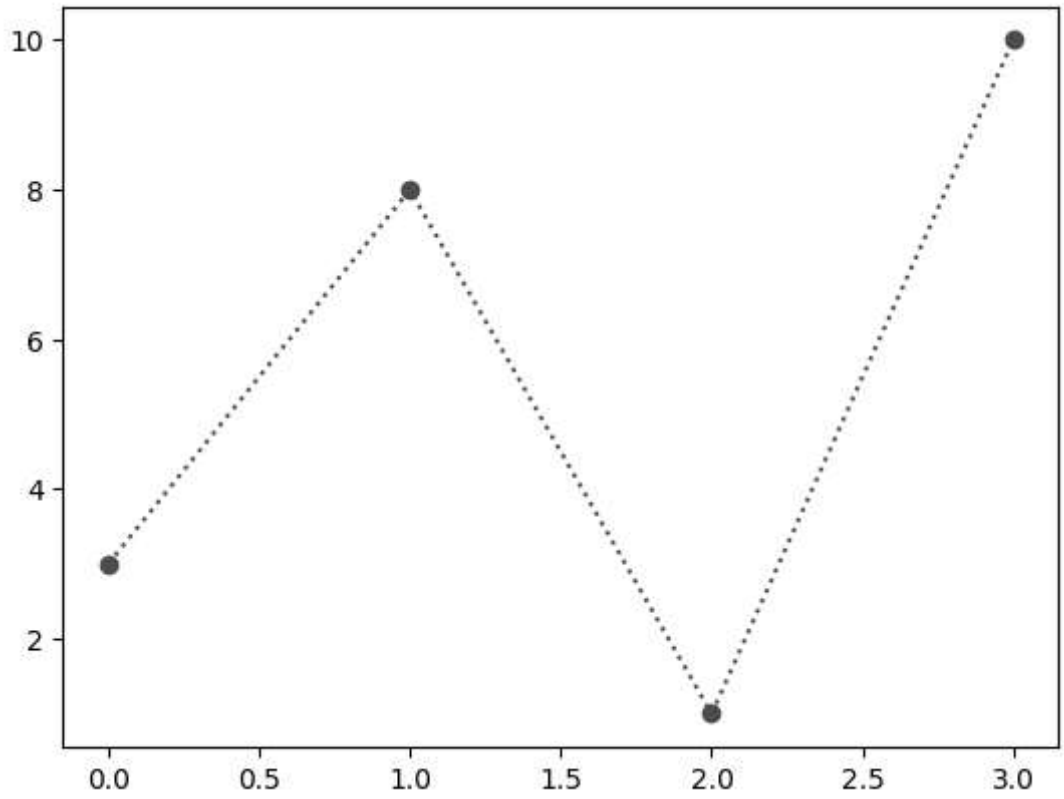
In [27]:
```
##Multiple Plots
#Draw a line in a diagram from position (1,3) to (2,8)
#then to (6,1) and finally to position (8,10)
import matplotlib.pyplot as plt
import numpy as np

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

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



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|coIor

In [30]:
```
#Example of Format strings fmt
import matplotlib.pyplot as plt
import numpy as np
ypoints= np.array([3,8,1,10])
plt.plot(ypoints,'o:r')
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. Labels provide information regarding plot i.e. plot's x axis and y axis

Syntax for x axis:

plt.xlabel("Average Pulse")

Syntax for y axis:

plt.ylabel("Calorie Burnage")

Create a title for a Plot

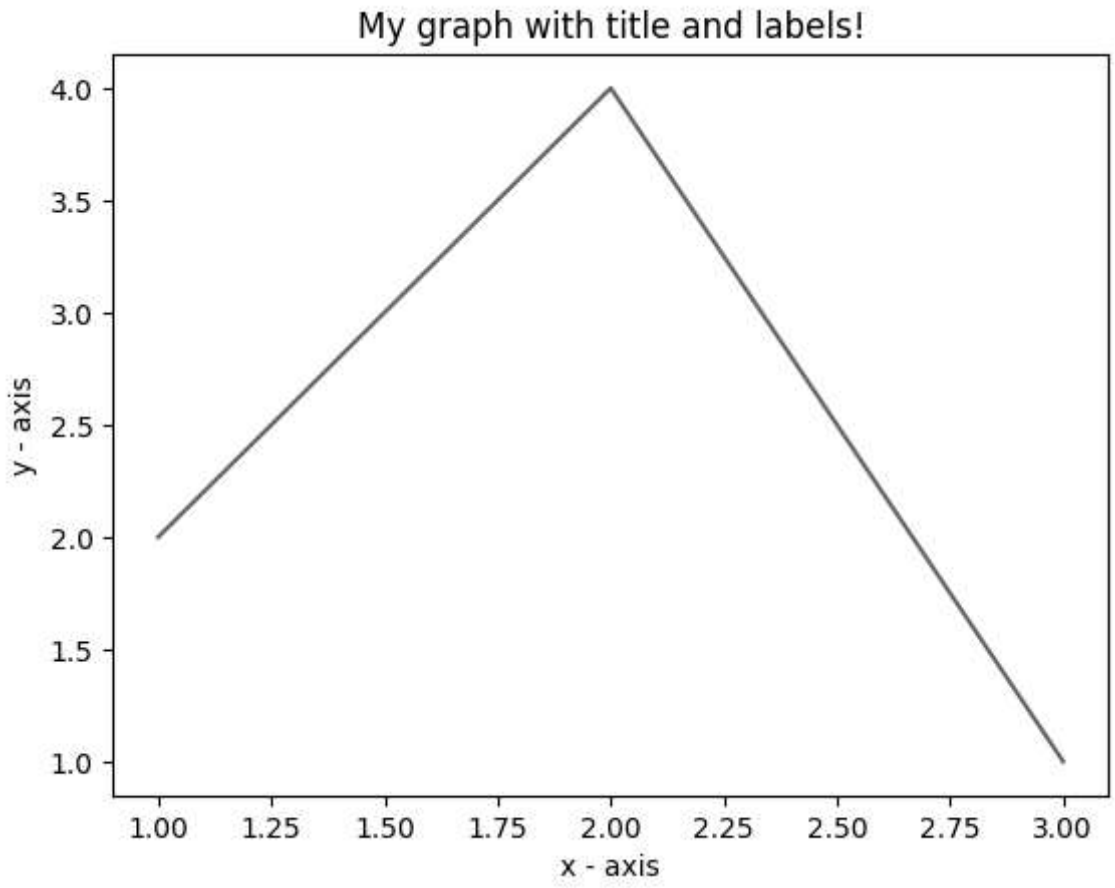
With Pyplot, you can use title for providing title to a graph

Syntax for a title of a graph:

plt.title("Sports Watch Data")

In [21]:
```
# importing the required module
import matplotlib.pyplot as plt

# x axis values
x = [1,2,3]
```

```
# corresponding y axis values
y = [2,4,1]

# plotting the points
plt.plot(x, y)

# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')

# giving a title to my graph
plt.title('My graph with title and labels!')

# function to show the plot
plt.show()
```

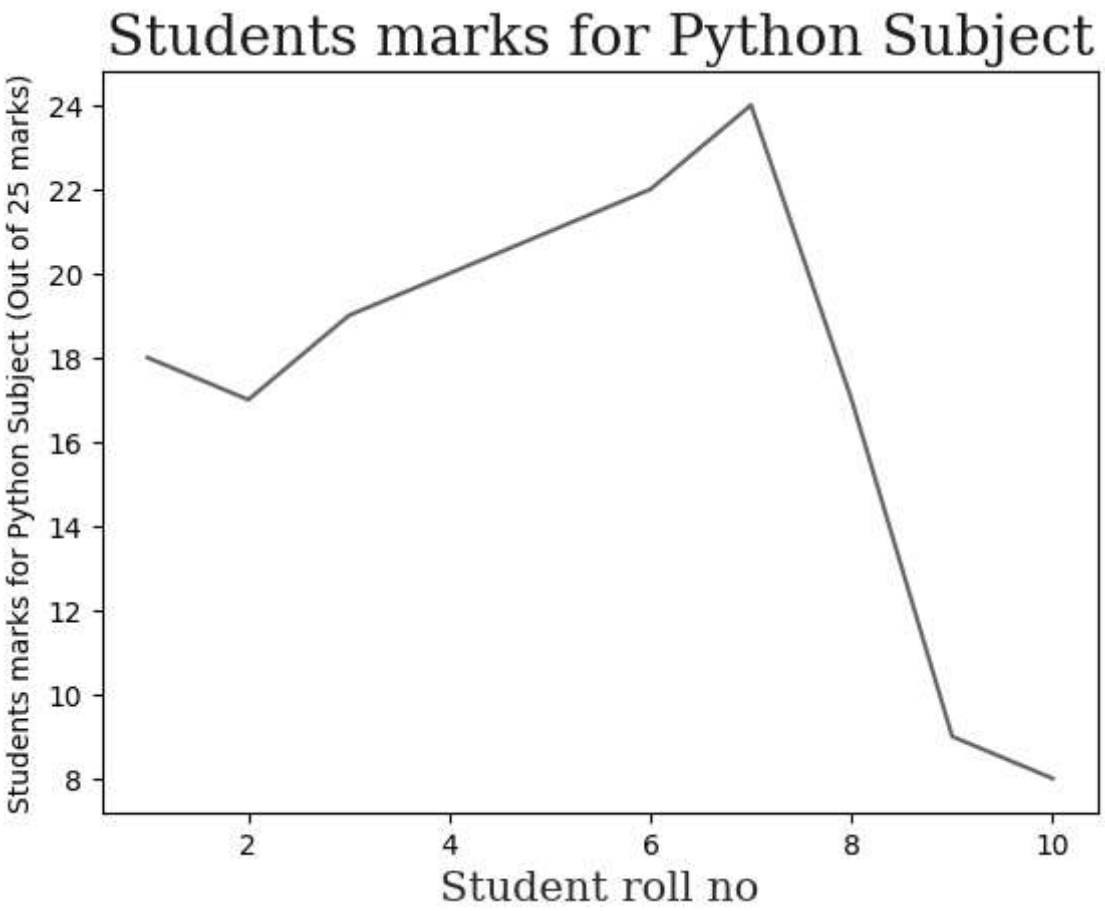


Set Font Properties for Title and Labels

This function is useful for setting font properties i.e. Title and Labels This includes various properties that needed to be set for font like font family (i.e. font type), font size, font color, font weight, etc. The font weight indicates whether it is bold or normal. The default font family is The font properties can be done in two ways:

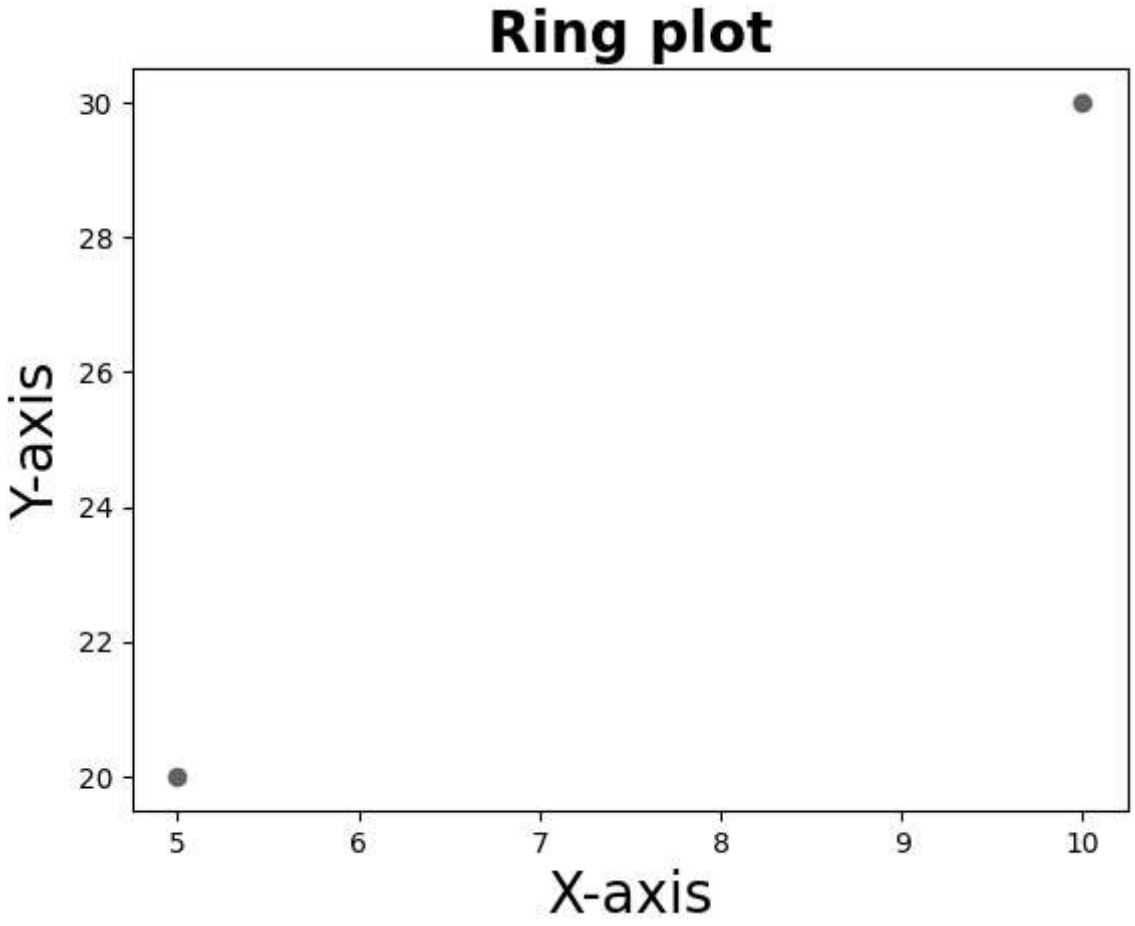
1. Using dictionary
2. Using directly in syntax with labels and titles.

In [33]:
```
#Example for setting font properties
import matplotlib.pyplot as plt
import numpy as np

x=np.array([1,2,3,4,5,6,7,8,9,10])
y=np.array([18,17,19,20,21,22,24,17,9,8])

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

plt.title("Students marks for Python Subject",fontdict=font1)
plt.xlabel("Student roll no",fontdict=font2)
plt.ylabel("Students marks for Python Subject (Out of 25 marks)")

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



In [83]:
```
#Example:
#Using directly in syntax along with labels and titles

import matplotlib.pyplot as plt
import numpy as np
x1=np.array([10,5])
y1=np.array([30,20])
#Here fontsize is defined in syntax like fontsize=20, fontweight="bold"
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()
```



In [50]:
```
import matplotlib.pyplot as plt
import numpy as np
def get_math_fontfamily(self):

        """
        Return the name of the font family used for math text.
        The default font is :rc:`mathtext.fontset`.
        """

        return self.get_math_fontfamily
        print(self.get_math_fontfamily)
```

In [52]:
```
import matplotlib.font_manager as fm
for font in fm.findSystemFonts():
    print(font)
```

C:\Windows\Fonts\cambriaz.ttf
C:\Windows\Fonts\l_10646.ttf
C:\Windows\Fonts\simsun.ttc
C:\Windows\Fonts\msjh.ttc
C:\Windows\Fonts\SitkaZ.ttc
C:\Windows\Fonts\seguisb.ttf
C:\Windows\Fonts\segoeuisl.ttf
C:\Windows\Fonts\impact.ttf
C:\Windows\Fonts\GOTHICI.TTF
C:\Windows\Fonts\gadugi.ttf
C:\Windows\Fonts\GARA.TTF
C:\Windows\Fonts\Candaraz.ttf
C:\Windows\Fonts\segoeuil.ttf
C:\Windows\Fonts\ariblk.ttf
C:\Windows\Fonts\YuGothM.ttc
C:\Windows\Fonts\corbell.ttf
C:\Windows\Fonts\segoeuiz.ttf
C:\Windows\Fonts\georgiab.ttf
C:\Windows\Fonts\constani.ttf
C:\Windows\Fonts\georgia.ttf
C:\Windows\Fonts\MTEXTRA.TTF
C:\Windows\Fonts\timesbi.ttf
C:\Windows\Fonts\segoeui.ttf
C:\Windows\Fonts\constanz.ttf
C:\Windows\Fonts\wingding.ttf
C:\Windows\Fonts\framdit.ttf
C:\Windows\Fonts\javatext.ttf
C:\Windows\Fonts\constan.ttf
C:\Windows\Fonts\trebucbi.ttf
C:\Windows\Fonts\GOTHICB.TTF
C:\Windows\Fonts\taile.ttf
C:\Windows\Fonts\GARABD.TTF
C:\Windows\Fonts\micross.ttf
C:\Windows\Fonts\calibriz.ttf
C:\Windows\Fonts\mmrtext.ttf
C:\Windows\Fonts\arialbd.ttf
C:\Windows\Fonts\Candarali.ttf
C:\Windows\Fonts\seguiemj.ttf
C:\Windows\Fonts\trebucit.ttf
C:\Windows\Fonts\Gabriola.ttf
C:\Windows\Fonts\SitkaB.ttc
C:\Windows\Fonts\CENTURY.TTF
C:\Windows\Fonts\consolai.ttf
C:\Windows\Fonts\cambria.ttc
C:\Windows\Fonts\WINGDNG2.TTF
C:\Windows\Fonts\timesi.ttf
C:\Windows\Fonts\DUBAI-LIGHT.TTF
C:\Windows\Fonts\ARIALNI.TTF
C:\Windows\Fonts\Inkfree.ttf
C:\Windows\Fonts\Candarai.ttf
C:\Windows\Fonts\MSUIGHUR.TTF
C:\Windows\Fonts\ARIALN.TTF
C:\Windows\Fonts\holomdl2.ttf
C:\Windows\Fonts\DUBAI-REGULAR.TTF
C:\Windows\Fonts\YuGothL.ttc
C:\Windows\Fonts\DUBAI-MEDIUM.TTF
C:\Windows\Fonts\times.ttf

```
C:\Windows\Fonts\msgothic.ttc
C:\Windows\Fonts\georgiaz.ttf
C:\Windows\Fonts\framd.ttf
C:\Windows\Fonts\cour.ttf
C:\Windows\Fonts\lucon.ttf
C:\Windows\Fonts\seguisli.ttf
C:\Windows\Fonts\malgun.ttf
C:\Windows\Fonts\phagspa.ttf
C:\Windows\Fonts\comicz.ttf
C:\Windows\Fonts\courbd.ttf
C:\Windows\Fonts\corbelz.ttf
C:\Windows\Fonts\LeelaUIb.ttf
C:\Windows\Fonts\mvboli.ttf
C:\Windows\Fonts\seguisym.ttf
C:\Windows\Fonts\consola.ttf
C:\Windows\Fonts\msyh.ttc
C:\Windows\Fonts\GOTHICBI.TTF
C:\Windows\Fonts\WINGDNG3.TTF
C:\Windows\Fonts\Nirmala.ttf
C:\Windows\Fonts\comic.ttf
C:\Windows\Fonts\ARIALNB.TTF
C:\Windows\Fonts\tahoma.ttf
C:\Windows\Fonts\segoeuib.ttf
C:\Windows\Fonts\segmdl2.ttf
C:\Windows\Fonts\arialbi.ttf
C:\Windows\Fonts\MSUIGHUB.TTF
C:\Windows\Fonts\comici.ttf
C:\Windows\Fonts\BSSYM7.TTF
C:\Windows\Fonts\sylfaen.ttf
C:\Windows\Fonts\ANTQUAI.TTF
C:\Windows\Fonts\BOOKOSBI.TTF
C:\Windows\Fonts\YuGothR.ttc
C:\Windows\Fonts\mmrtextb.ttf
C:\Windows\Fonts\segoesc.ttf
C:\Windows\Fonts\msyhl.ttc
C:\Windows\Fonts\seguisbi.ttf
C:\Windows\Fonts\msyi.ttf
C:\Windows\Fonts\segoepr.ttf
C:\Windows\Fonts\seguibl.ttf
C:\Windows\Fonts\calibri.ttf
C:\Windows\Fonts\ebrimabd.ttf
C:\Windows\Fonts\BOOKOSB.TTF
C:\Windows\Fonts\seguibli.ttf
C:\Windows\Fonts\Candarab.ttf
C:\Windows\Fonts\ntailu.ttf
C:\Windows\Fonts\calibrib.ttf
C:\Windows\Fonts\HATTEN.TTF
C:\Windows\Fonts\himalaya.ttf
C:\Windows\Fonts\REFSPCL.TTF
C:\Windows\Fonts\LEELAWAD.TTF
C:\Windows\Fonts\malgunsl.ttf
C:\Windows\Fonts\phagspab.ttf
C:\Windows\Fonts\BOOKOSI.TTF
C:\Windows\Fonts\corbelli.ttf
C:\Windows\Fonts\comicbd.ttf
C:\Windows\Fonts\verdanaz.ttf
C:\Windows\Fonts\ARIALNBI.TTF
```

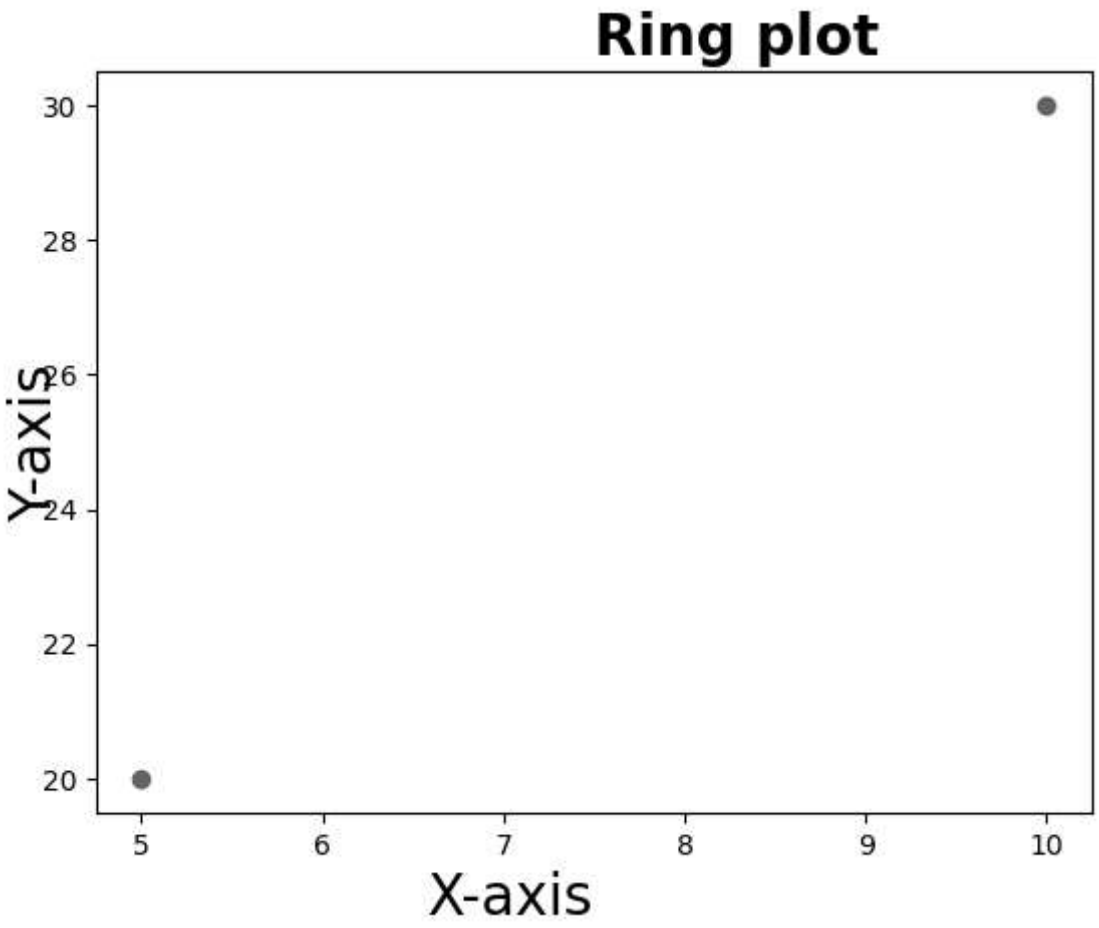
C:\Windows\Fonts\tahomabd.ttf
C:\Windows\Fonts\timesbd.ttf
C:\Windows\Fonts\trebucbd.ttf
C:\Windows\Fonts\courbi.ttf
C:\Windows\Fonts\gadugib.ttf
C:\Windows\Fonts\palai.ttf
C:\Windows\Fonts\msyhbd.ttc
C:\Windows\Fonts\NirmalaB.ttf
C:\Windows\Fonts\calibril.ttf
C:\Windows\Fonts\georgiai.ttf
C:\Windows\Fonts\segoeuii.ttf
C:\Windows\Fonts\BKANT.TTF
C:\Windows\Fonts\GOTHIC.TTF
C:\Windows\Fonts\SitkaI.ttc
C:\Windows\Fonts\ebrima.ttf
C:\Windows\Fonts\OUTLOOK.TTF
C:\Windows\Fonts\webdings.ttf
C:\Windows\Fonts\monbaiti.ttf
C:\Windows\Fonts\malgunbd.ttf
C:\Windows\Fonts\taileb.ttf
C:\Windows\Fonts\corbel.ttf
C:\Windows\Fonts\bahnschrift.ttf
C:\Windows\Fonts\palabi.ttf
C:\Windows\Fonts\calibrili.ttf
C:\Windows\Fonts\consolaz.ttf
C:\Windows\Fonts\arial.ttf
C:\Windows\Fonts\corbeli.ttf
C:\Windows\Fonts\simsunb.ttf
C:\Windows\Fonts\cambriab.ttf
C:\Windows\Fonts\corbelb.ttf
C:\Windows\Fonts\verdanai.ttf
C:\Windows\Fonts\trebuc.ttf
C:\Windows\Fonts\consolab.ttf
C:\Windows\Fonts\msjhl.ttc
C:\Windows\Fonts\seguihis.ttf
C:\Windows\Fonts\symbol.ttf
C:\Windows\Fonts\ANTQUABI.TTF
C:\Windows\Fonts\ariali.ttf
C:\Windows\Fonts\segoescb.ttf
C:\Windows\Fonts\REFSAN.TTF
C:\Windows\Fonts\constanb.ttf
C:\Windows\Fonts\segoeprb.ttf
C:\Windows\Fonts\marlett.ttf
C:\Windows\Fonts\pala.ttf
C:\Windows\Fonts\Candaral.ttf
C:\Windows\Fonts\LeelawUI.ttf
C:\Windows\Fonts\GARAIT.TTF
C:\Windows\Fonts\calibrii.ttf
C:\Windows\Fonts\cambriai.ttf
C:\Windows\Fonts\verdana.ttf
C:\Windows\Fonts\ntailub.ttf
C:\Windows\Fonts\msjhbd.ttc
C:\Windows\Fonts\LEELAWDB.TTF
C:\Windows\Fonts\verdanab.ttf
C:\Windows\Fonts\YuGothB.ttc
C:\Windows\Fonts\palab.ttf
C:\Windows\Fonts\BOOKOS.TTF

```
C:\Windows\Fonts\ANTQUAB.TTF
C:\Windows\Fonts\MTCORSVA.TTF
C:\Windows\Fonts\seguili.ttf
C:\Windows\Fonts\couri.ttf
C:\Windows\Fonts\NirmalaS.ttf
C:\Windows\Fonts\mingliub.ttc
C:\Windows\Fonts\Sitka.ttc
C:\Windows\Fonts\DUBAI-BOLD.TTF
C:\Windows\Fonts\Candara.ttf
C:\Windows\Fonts\LeelUIsl.ttf
```

Setting Font Positions

The font positions for x axis and y axis is different i.e. For x axis i.e. horizontal alignment the label can be set on For y axis i.e. the vertical alignment the label can be set on {'center', 'top', 'bottom', 'baseline', 'center_baseline'} Syntax for font position is: plt.xlabel('X-axis',fontsize=20,horizontalalignment="right") plt.ylabel('Y-axis',fontsize=20,verticalalignment="center_baseline") plt.title('Ring plot',fontsize=20,fontweight="bold",horizontalalignment="left")

In [98]:
```
#Example:
#Using directly in syntax along with labels and titles along setting it's positions

import matplotlib.pyplot as plt
import numpy as np
x1=np.array([10,5])
y1=np.array([30,20])
#Here fontsize is defined in syntax like fontsize=20, fontweight="bold"
plt.xlabel('X-axis',fontsize=20,horizontalalignment="right")
plt.ylabel('Y-axis',fontsize=20,verticalalignment="center_baseline")
plt.title('Ring plot',fontsize=20,fontweight="bold",horizontalalignment="left")
plt.plot(x1,y1,'o')
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 1st plot. plt.subplot(1,2,2) the figure has 1 row, 2 columns and this plot is the 2nd plot.

Adding Superior title of all subplots by adding function suptitle()

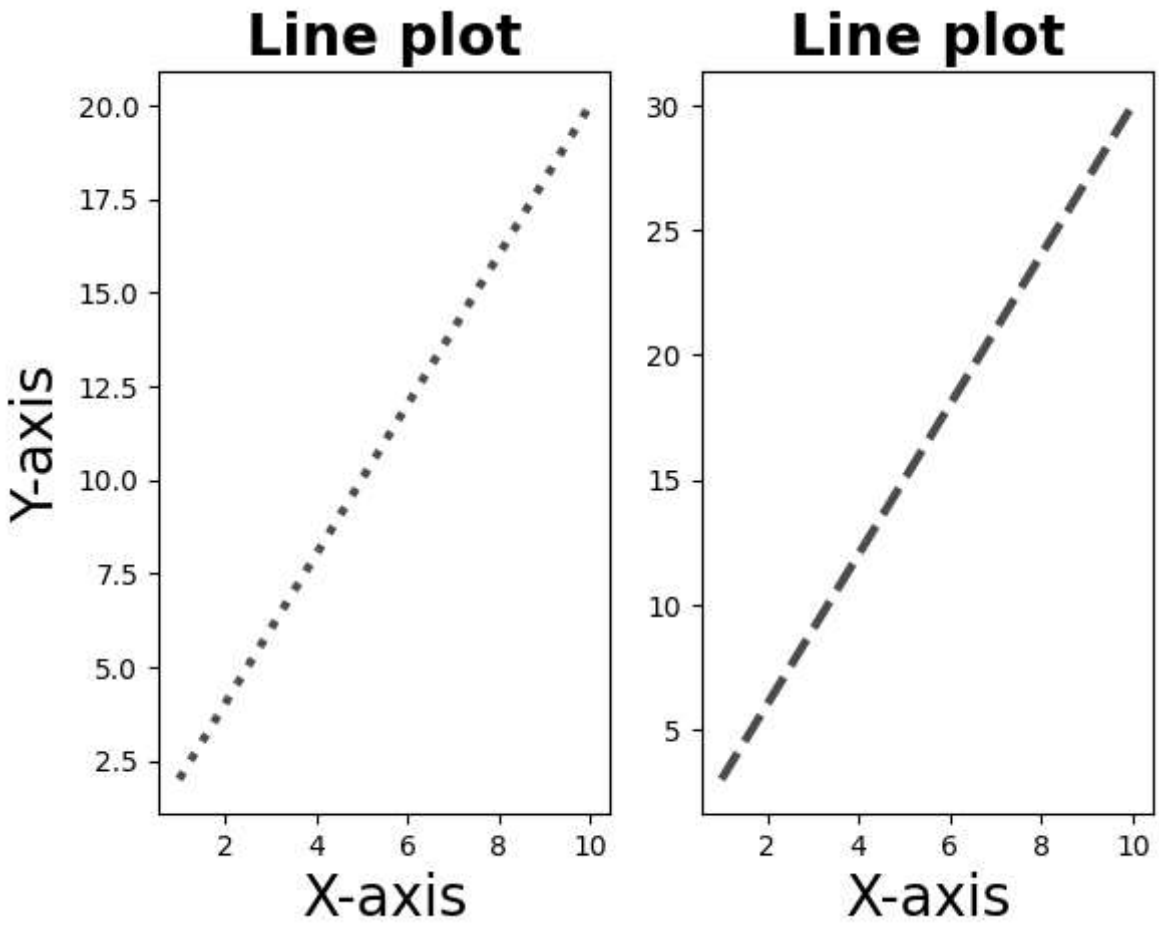
Syntax of suptitle is:

plt.suptitle("MY GRAPH")

In [101…

```
#Addtng two lines in different graph
import  matplotlib.pyplot  as  plt
import numpy as np
x=np.arange(1,11)
```

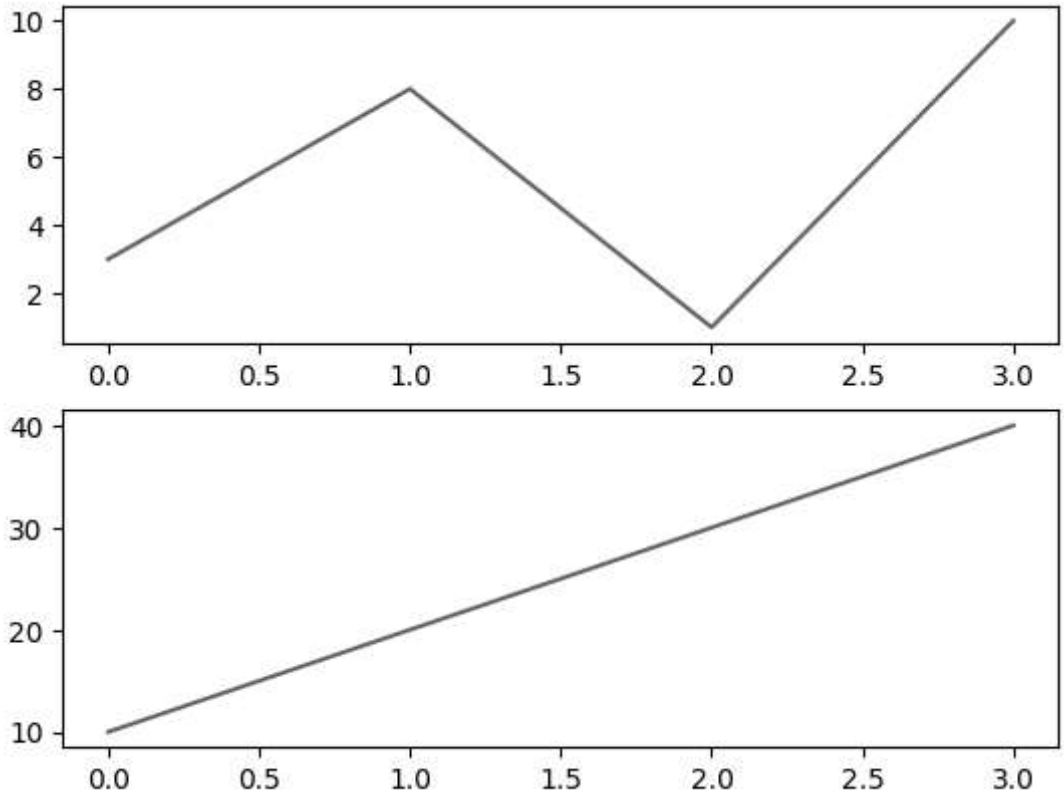
```
y1=2*x
y2=3*x

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 [102…

```
#Example 2
##if we want a figure with 2 rows an 1 column
#(meaning that the two pLots wiLL be d!spLayed 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

#p1ot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
```

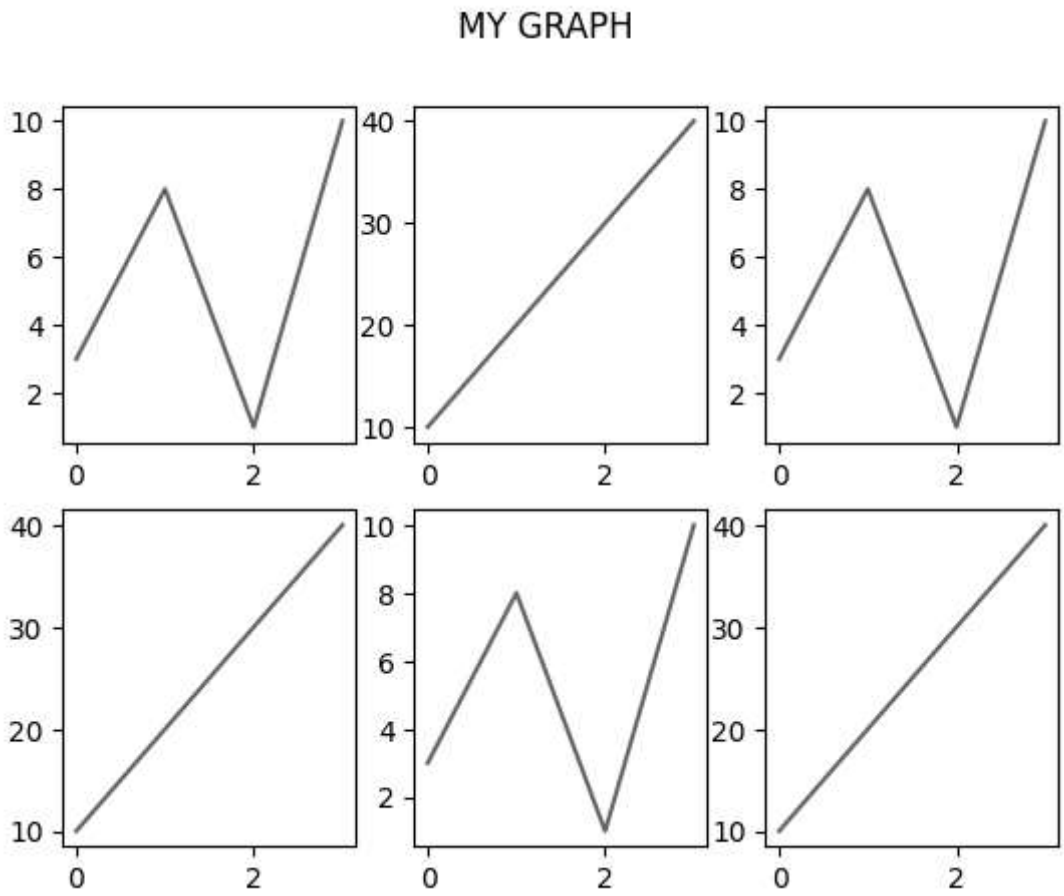
```
plt.subplot(2, 1, 1)
plt.plot(x,y)

#p1ot 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 [107…

```
#Example 3:
#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()
```



Scatter Plots

Scatter Plot:-

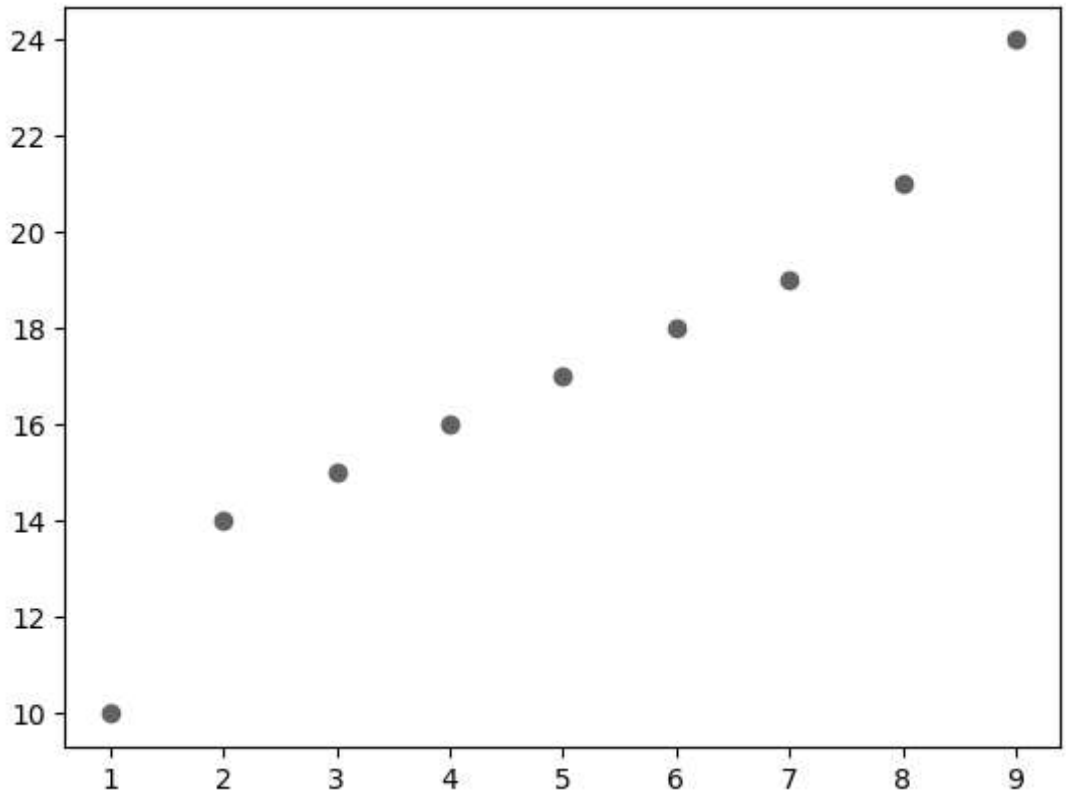
Scatter plots are used to plot data points on horizontal and vertical axis in the attempt to show how much one variable is affected by another. The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the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

Syntax of Scatter Plot: plt.scatter(x,a)

Scatter Plot takes two arguments where as an argument two variables need to written.

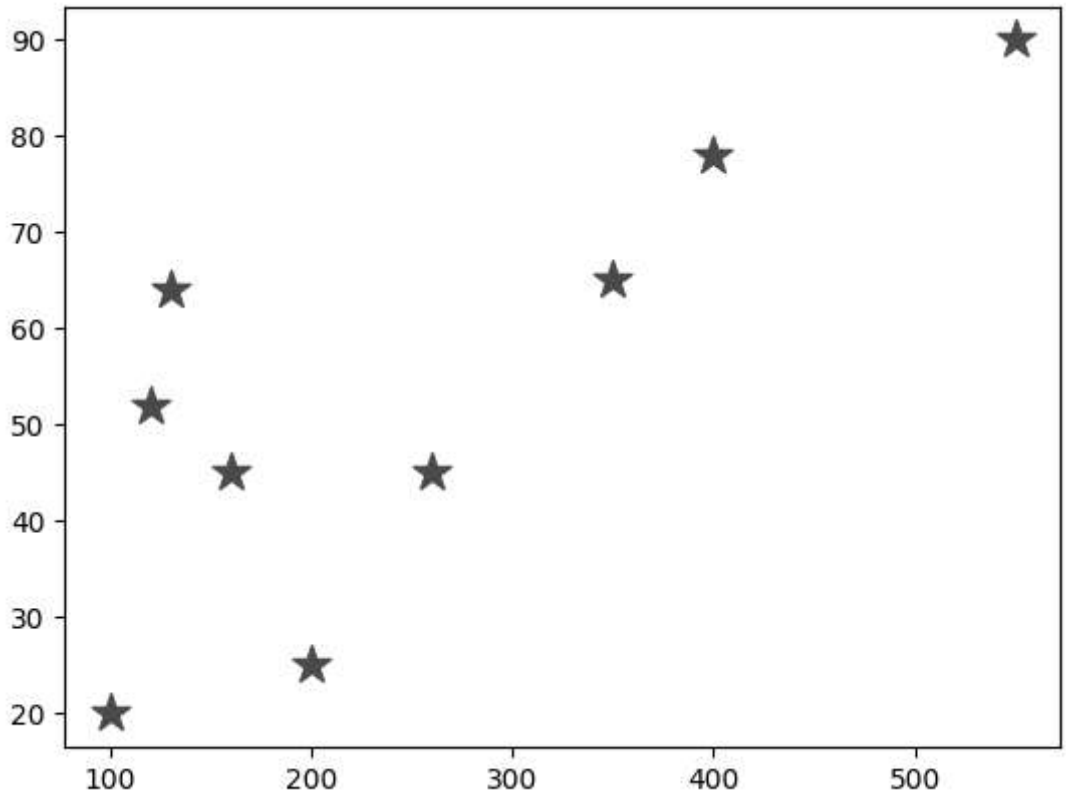
In [112…

```
#Scatter Plot
import matplotlib.pyplot as plt
import numpy as np
x=[1,2,3,4,5,6,7,8,9]
a=[10,14,15,16,17,18,19,21,24]
#For Scatter Plot define syntax of scatter
plt.scatter(x,a)
plt.show()
```



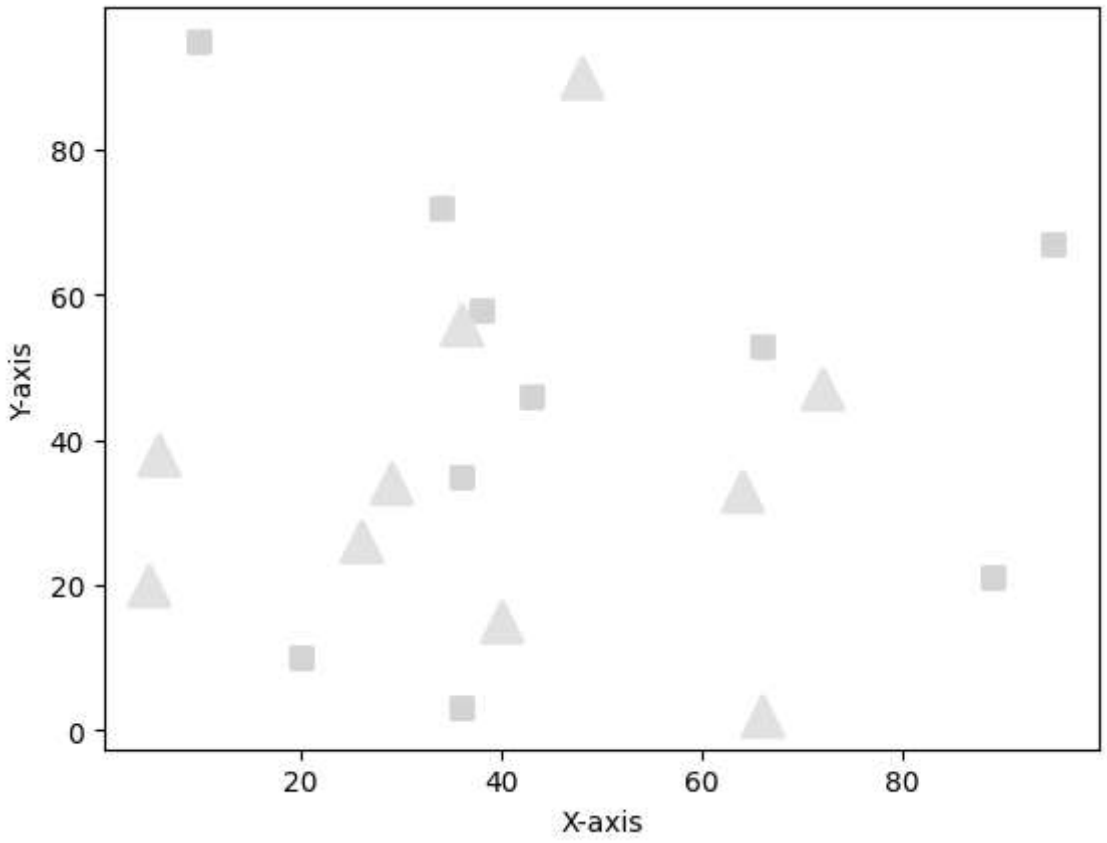
In [115…

```
#Example 2
#Scatter  PLot
x=[100,200,350,260,400,550,120,130,160]
a=[20,25,65,45,78,90,52,64,45]
#For Scatter Plot define syntax of scatter
plt.scatter(x,a,marker='*',color="green",s=200)
plt.show()
```



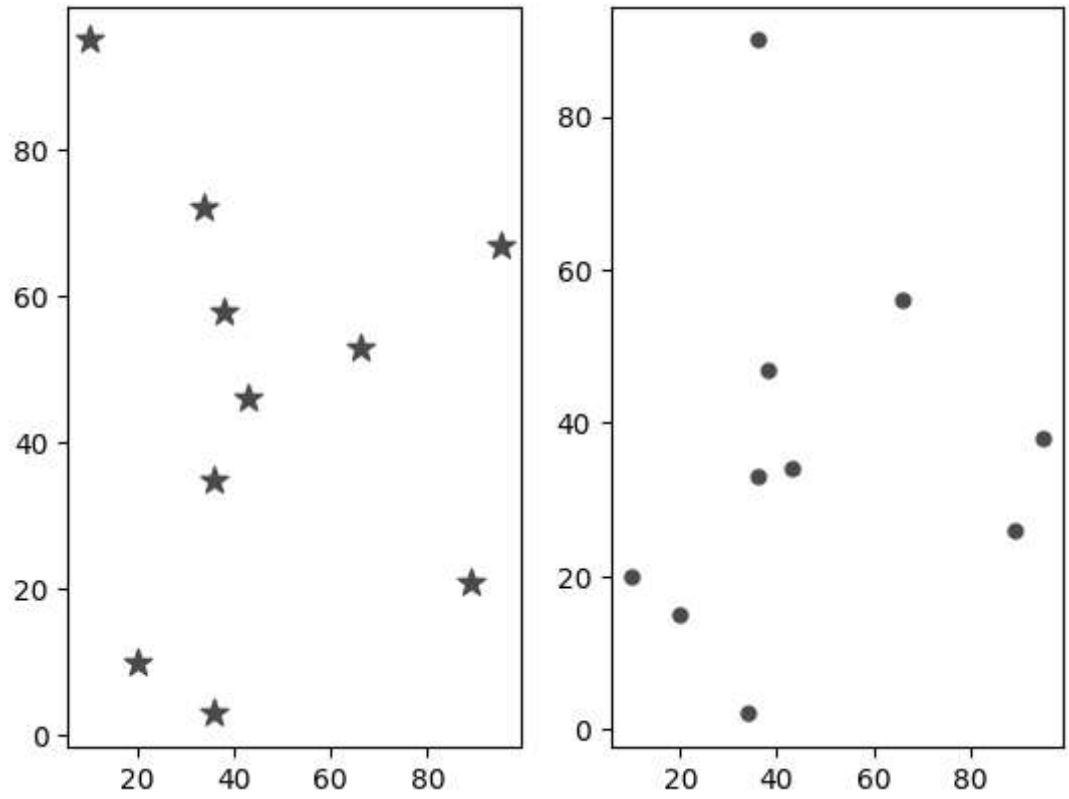
In [116…

```
#Scatter plot with different shape and colour for two datasets.
import matplotlib.pyplot as plt

# dataset-1
x1 = [89, 43, 36, 36, 95, 10,
      66, 34, 38, 20]

y1 = [21, 46, 3, 35, 67, 95,
      53, 72, 58, 10]

# dataset2
x2 = [26, 29, 48, 64, 6, 5,
      36, 66, 72, 40]

y2 = [26, 34, 90, 33, 38,
      20, 56, 2, 47, 15]

plt.scatter(x1, y1, c ="pink",linewidths = 2,marker ="s",s = 50)

plt.scatter(x2, y2, c ="yellow",linewidths = 2,marker ="^",s = 200)

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



In [117…

```
#Example 4
#Adding SubPlots in Scatter Plot

x=[89, 43, 36, 36, 95, 10,
      66, 34, 38, 20]
a=[21, 46, 3, 35, 67, 95,
      53, 72, 58, 10]
b=[26, 34, 90, 33, 38,
      20, 56, 2, 47, 15]

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()
```

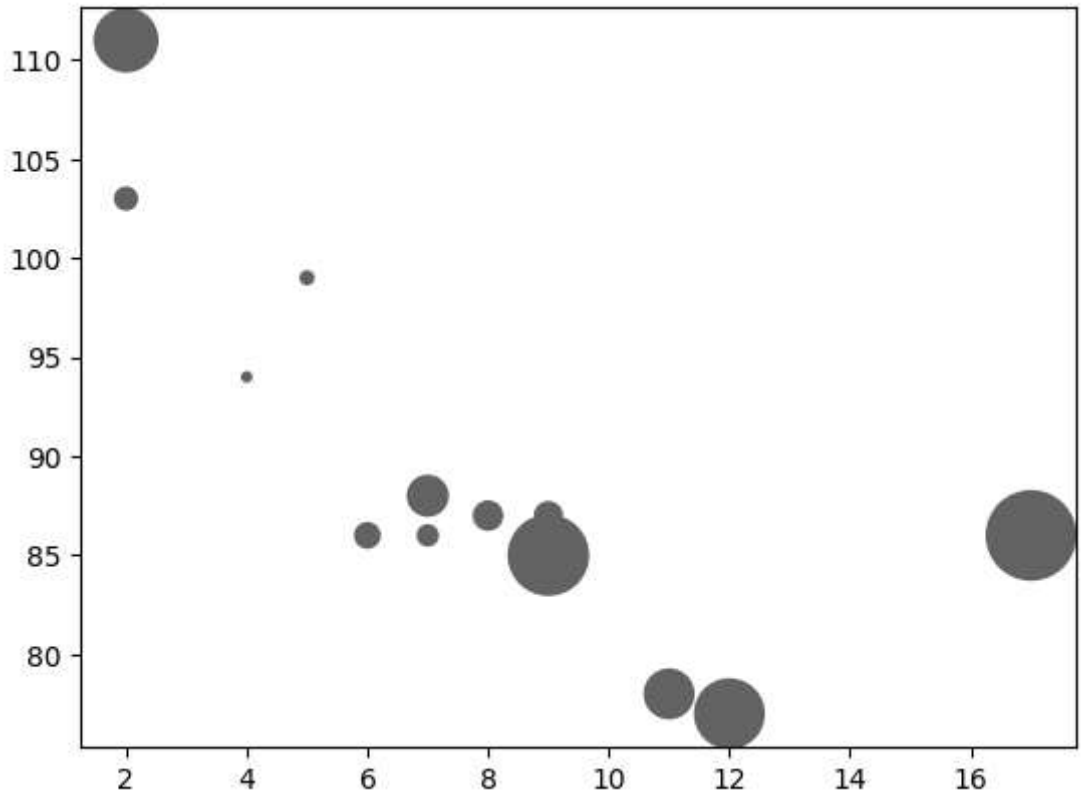


Size

You can change the size of the dots with the s argument.

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

In [1]:
```
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()
```

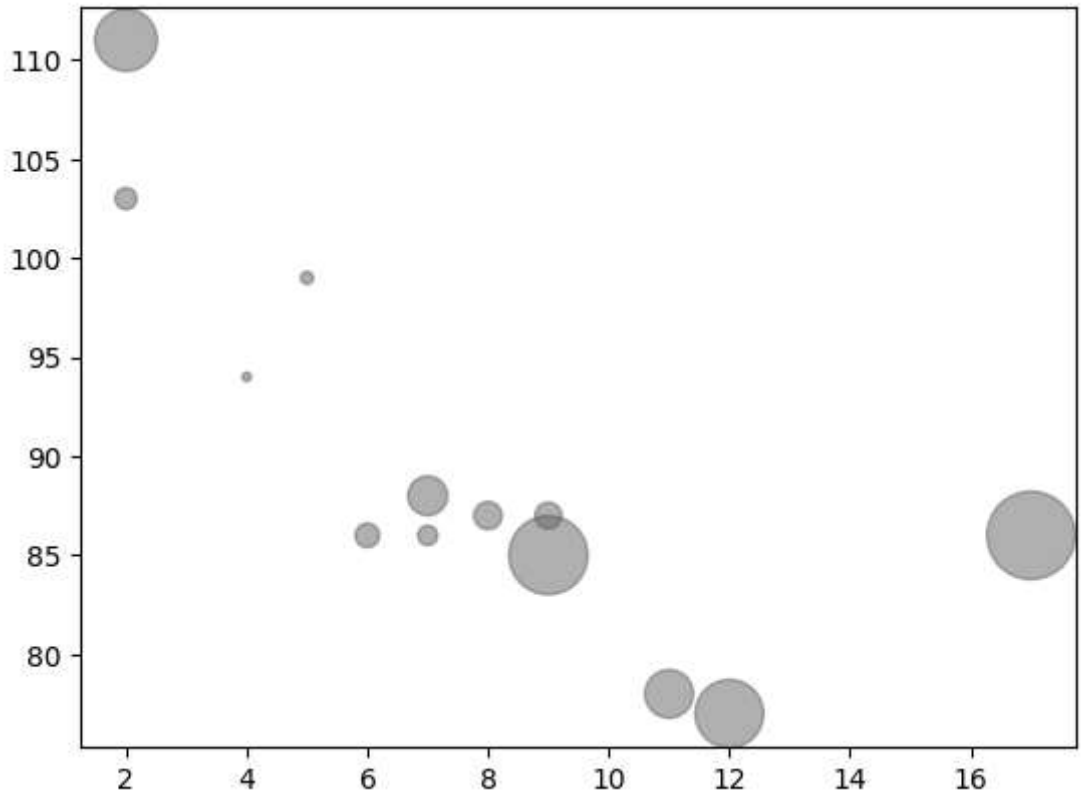


alpha in scatter

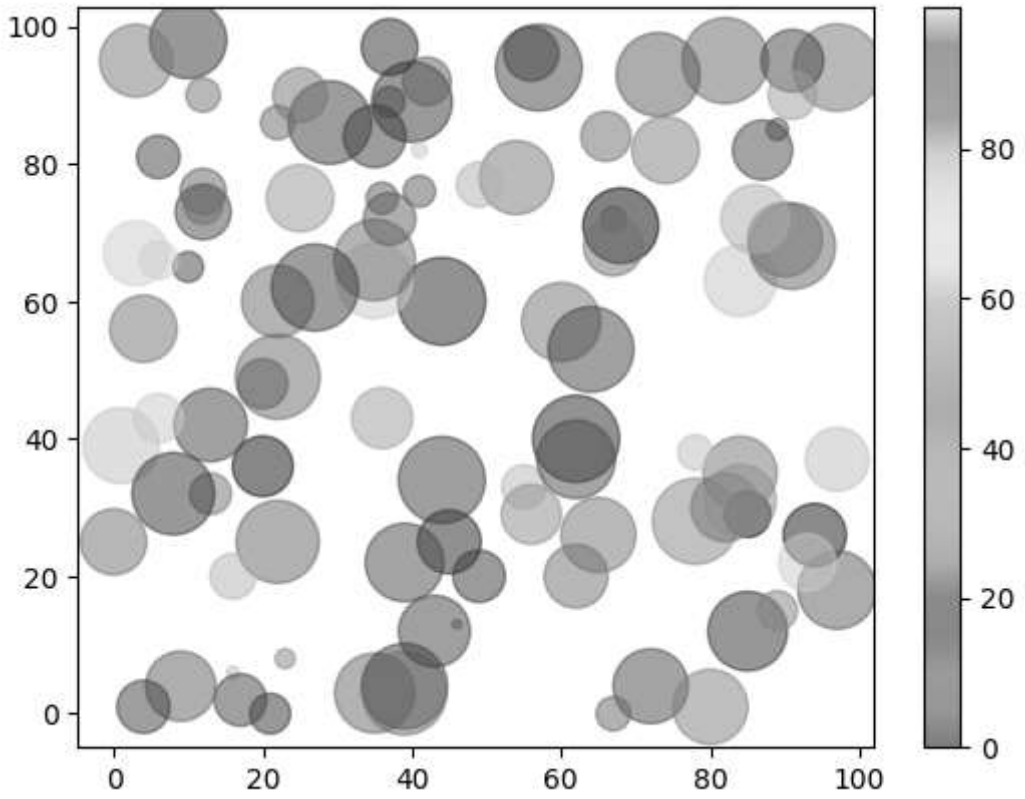
Alpha You can adjust the transparency of the dots with the alpha argument.

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

In [2]:
```
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=0.5)

plt.show()
```



In [3]:
```
import matplotlib.pyplot as plt
import numpy as np

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()
```



Bar Plots

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.

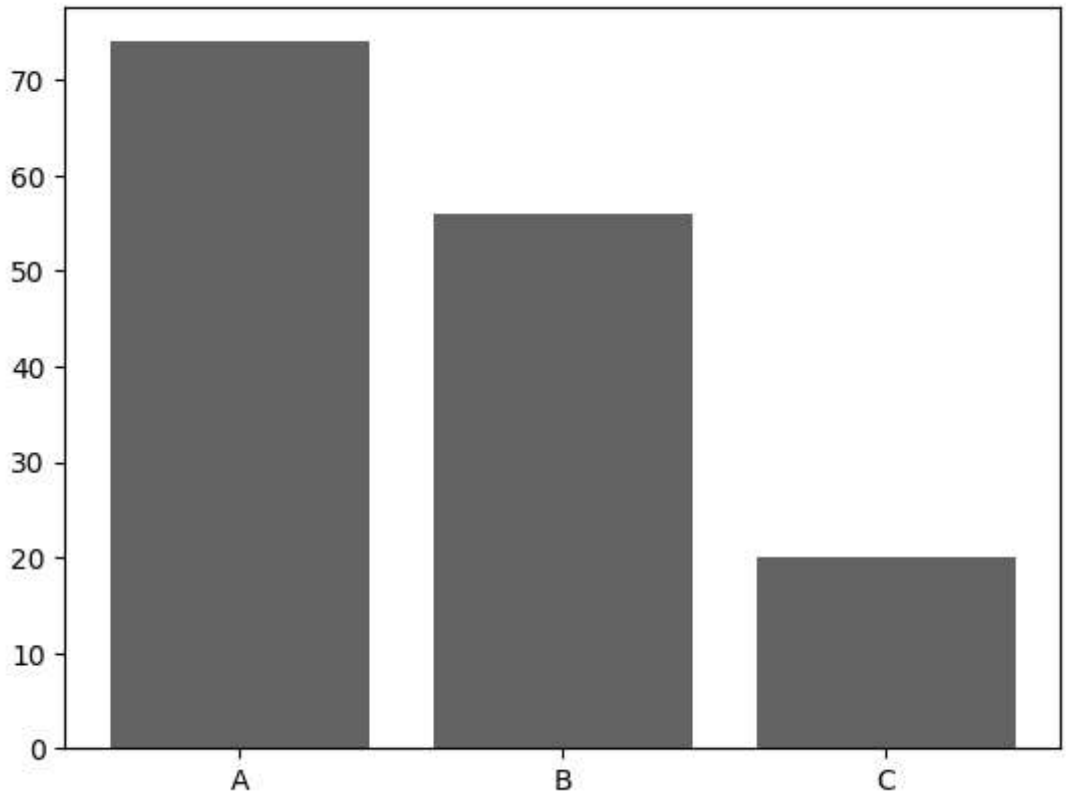
Syntax for plotting Bar Plot is: plt.bar(names,values)

Syntax for grid on: plt.grid(True)

Syntax for horizontal bar plot is :

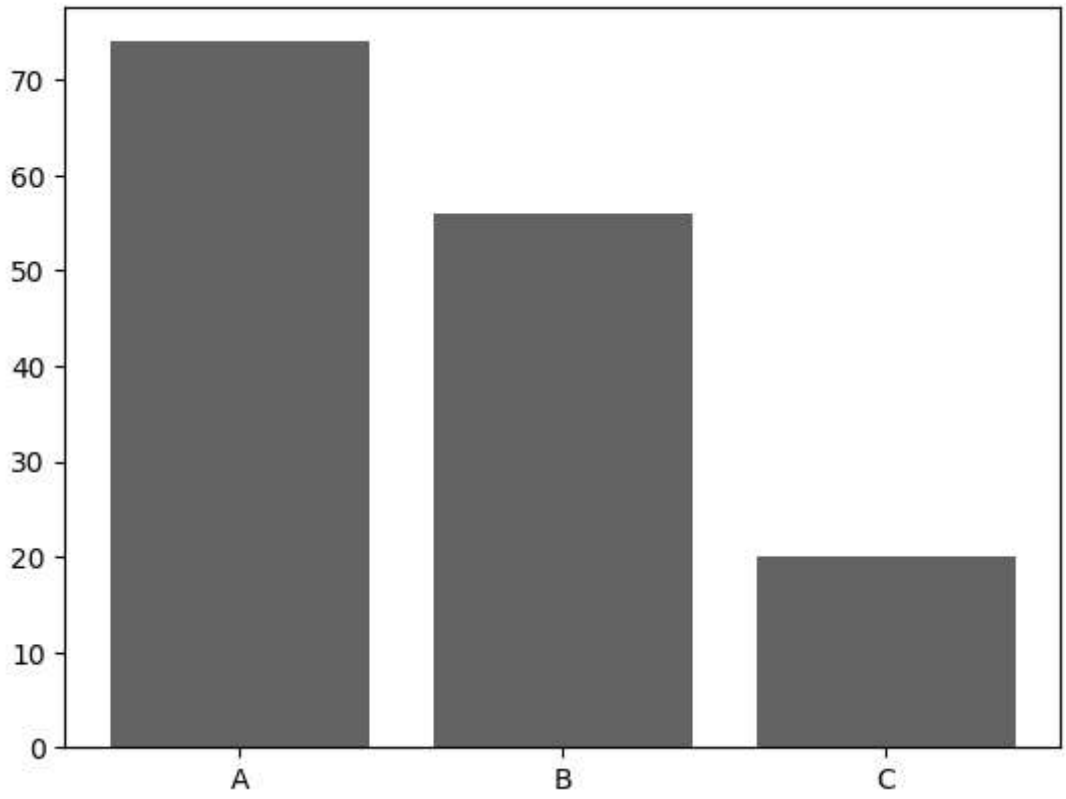
In [119…

```
#Example of bar chart with dictionary
import matplotlib.pyplot as plt
#list defined with keys and values
student={"A":74,"B":56,"C":20}
names=list(student.keys())
values=list(student.values())
#Bar Plot defined withn syntax and two arguments i.e. 2 variables
plt.bar(names,values)
plt.show()
```



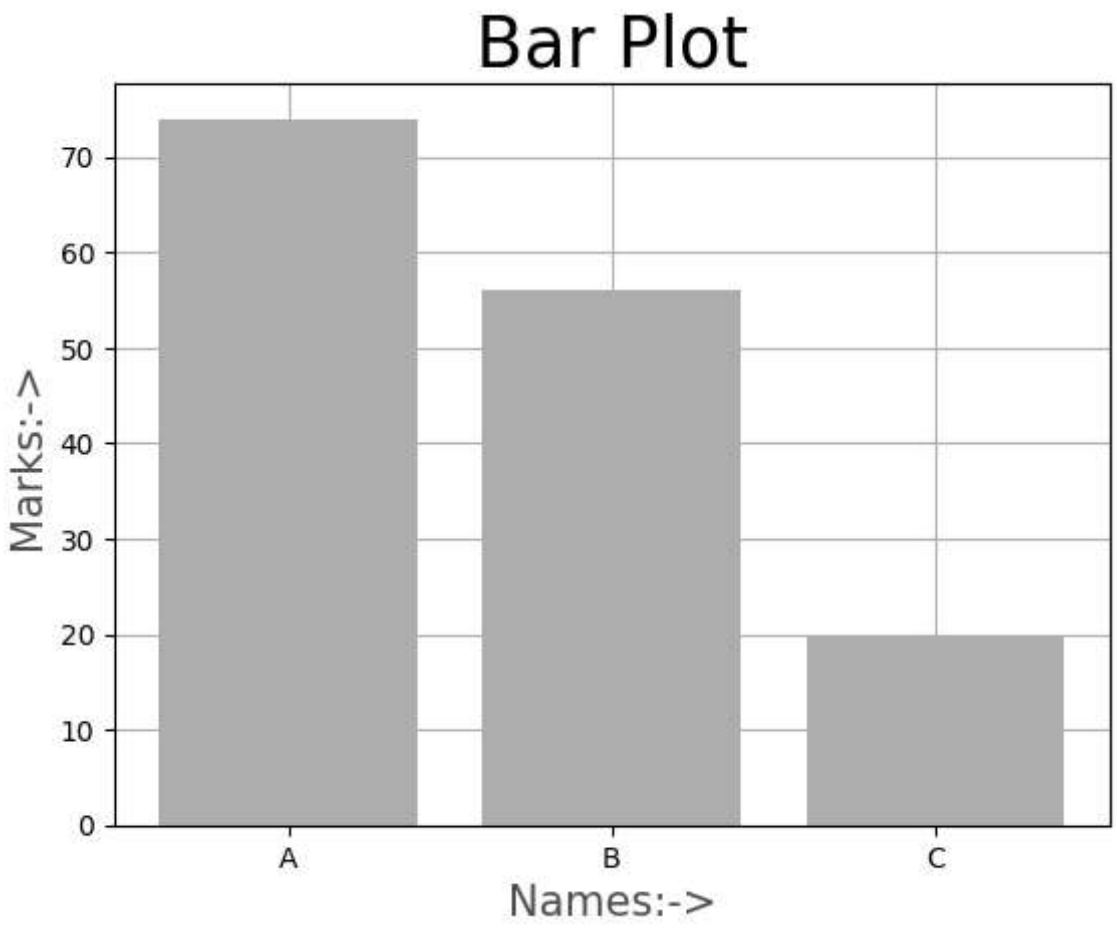
In [120…

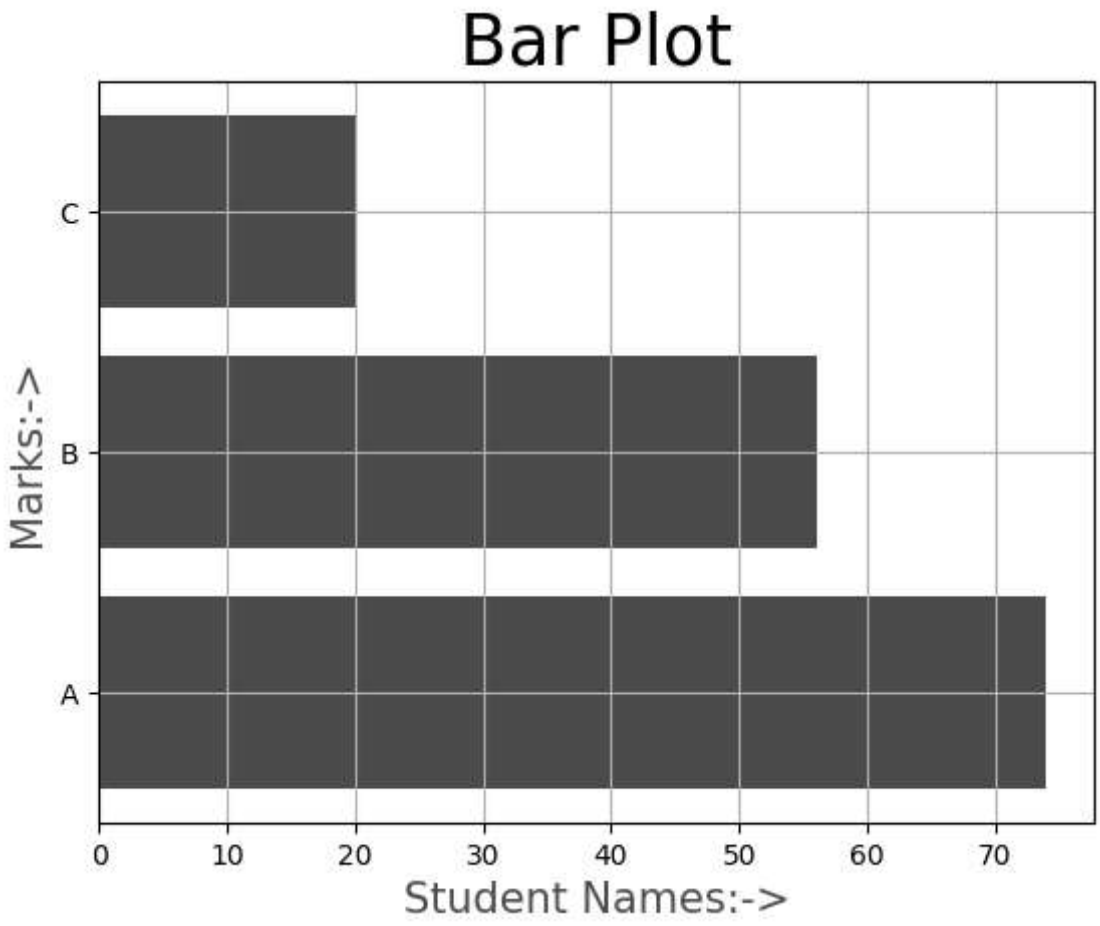
```
#Example of bar chart with list
import matplotlib.pyplot as plt

names=["A","B","C"]
values=[74,56,20]

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



In [122…

```
#Example of grid on for bar plot along with color change and fontsize different and fo
#The entire grapgh should be of orange color
import matplotlib.pyplot as plt
names=["A","B","C"]
values=[74,56,20]

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

plt.show()
```


Bar Plot
70
60
50
40
30
20
10
0
Marks:->
A
B
C
Names:->
In [123…
#Example 3:
#To plot horizontal bar plot with grid on
#Mor1zonto1 Bor plot
import matplotlib.pyplot as plt
names=["A","B","C"]
values=[74,56,20]
#Syntax for horizontal bar plot
plt.barh(names,values,color="green")
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()



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.

Syntax for histogram:

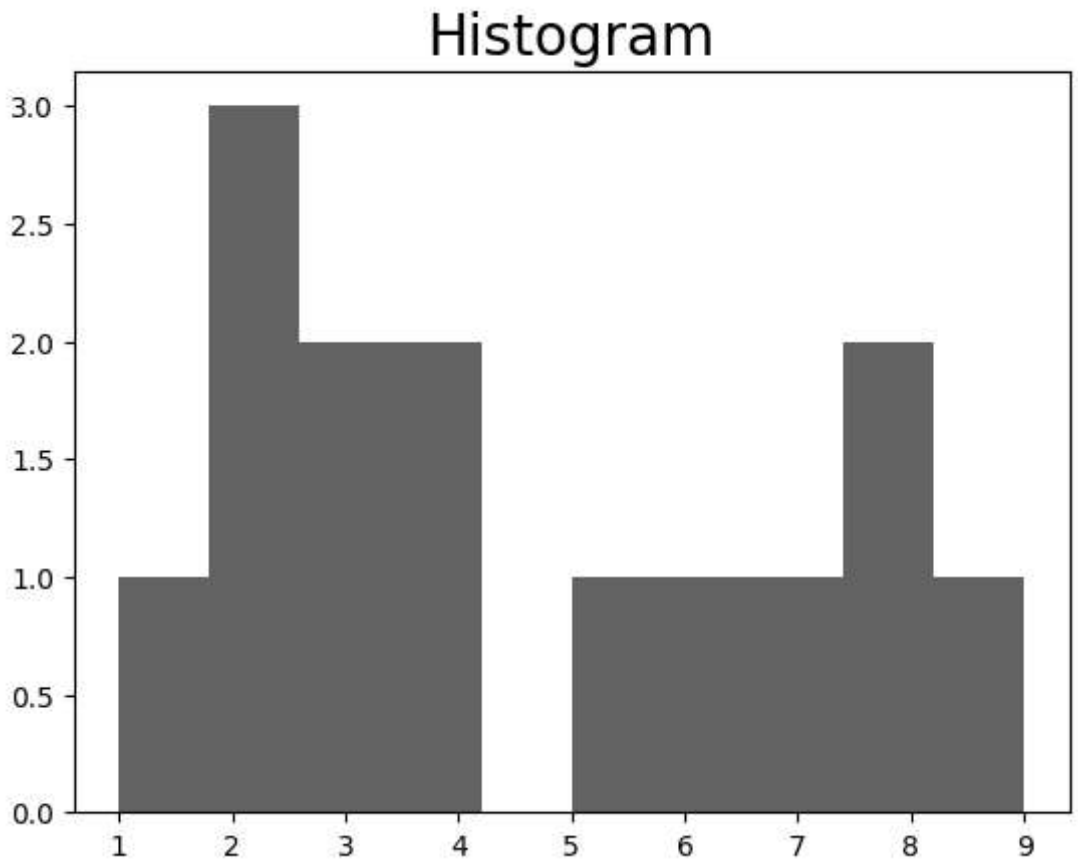
plt.hist(data)

Bin Calculation and Bin Width Calculation:

Count the number of data points. Calculate the number of bins by taking the square root of the number of data points and round up. Calculate the bin width by dividing the specification tolerance or range (USL-LSL or Max-Min value) by the # of bins. For example: data= [1,2,2,2,3,3,9,5,4,4,8,8,6,7] Now the no of datas are 14 so no. of bins= sqrt(14) no. of bins=3.74=4(to round off) and bin width=(maxvalue-minvalue)/no. of bins maxvaalue=9 minvalue=1 bin width=(9-1)/4 bin width=2 But by default bin width=1

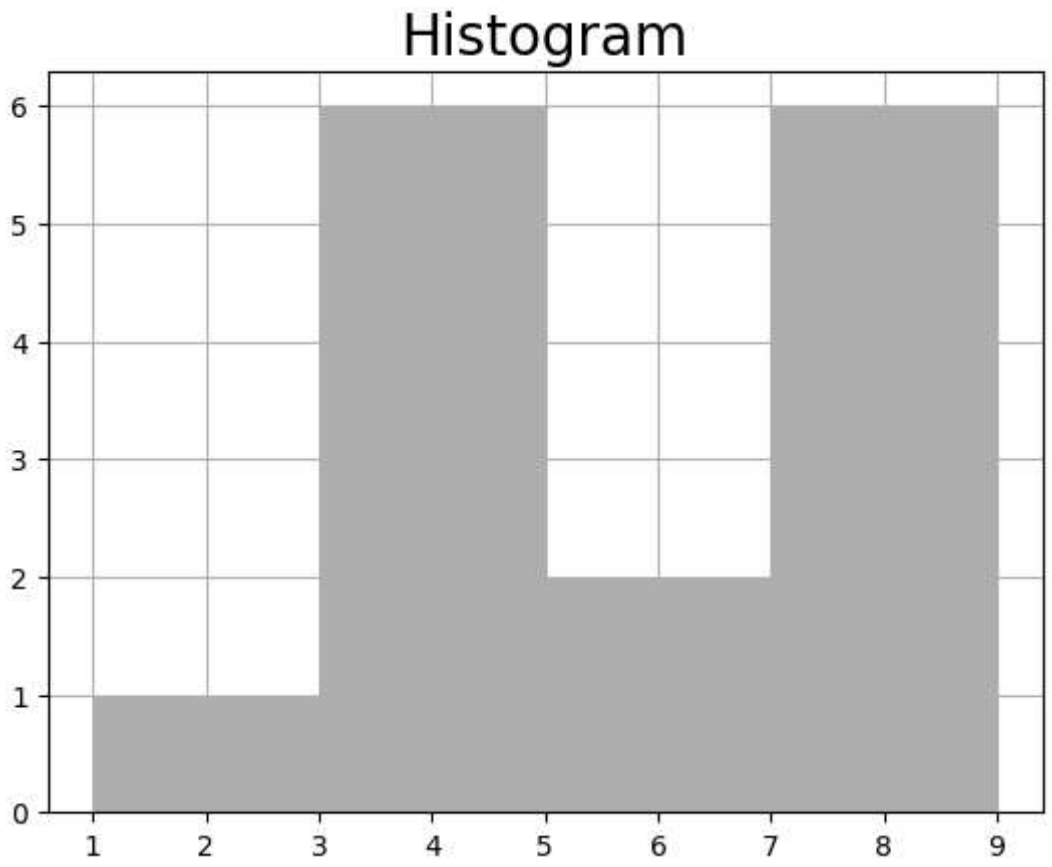
In [124…

```
#Example for histogram
import matplotlib.pyplot as plt
import numpy as np

data=[1,2,2,2,3,3,9,5,4,4,8,8,6,7]

#Syntax for histogram
plt.hist(data)
plt.title("Histogram",fontsize=20)
plt.show()
#by default bin width=1
```

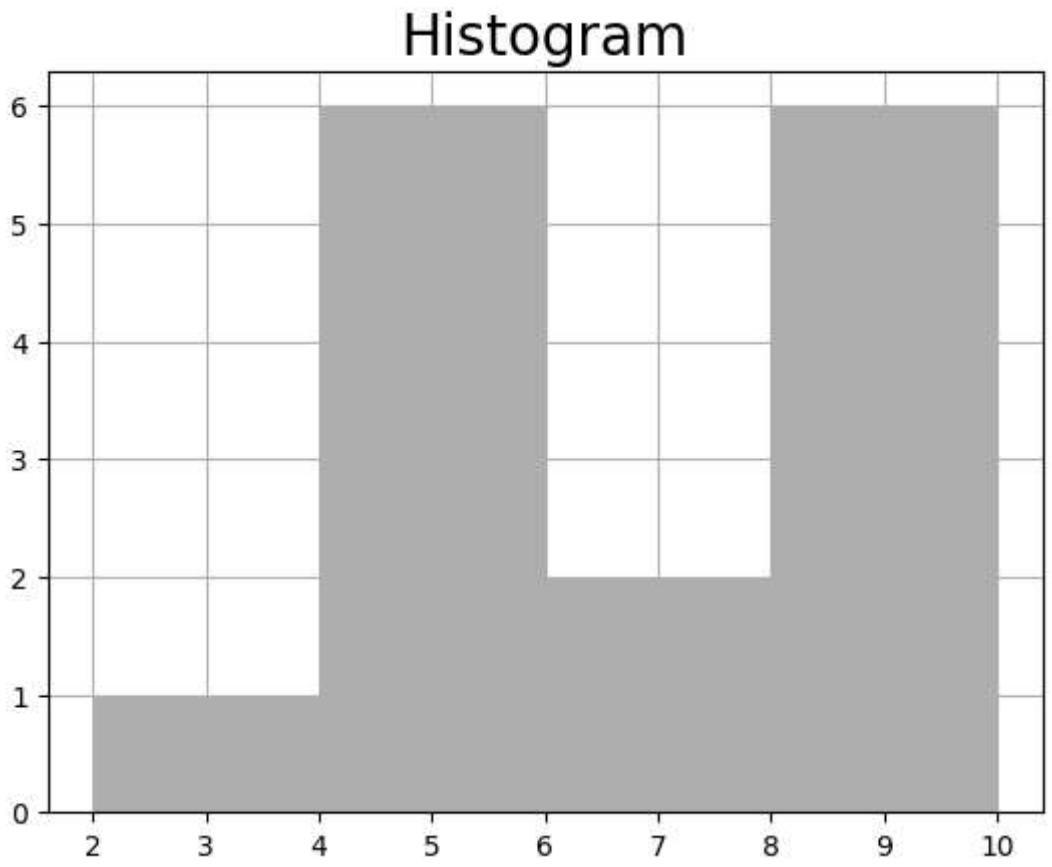


In [125…

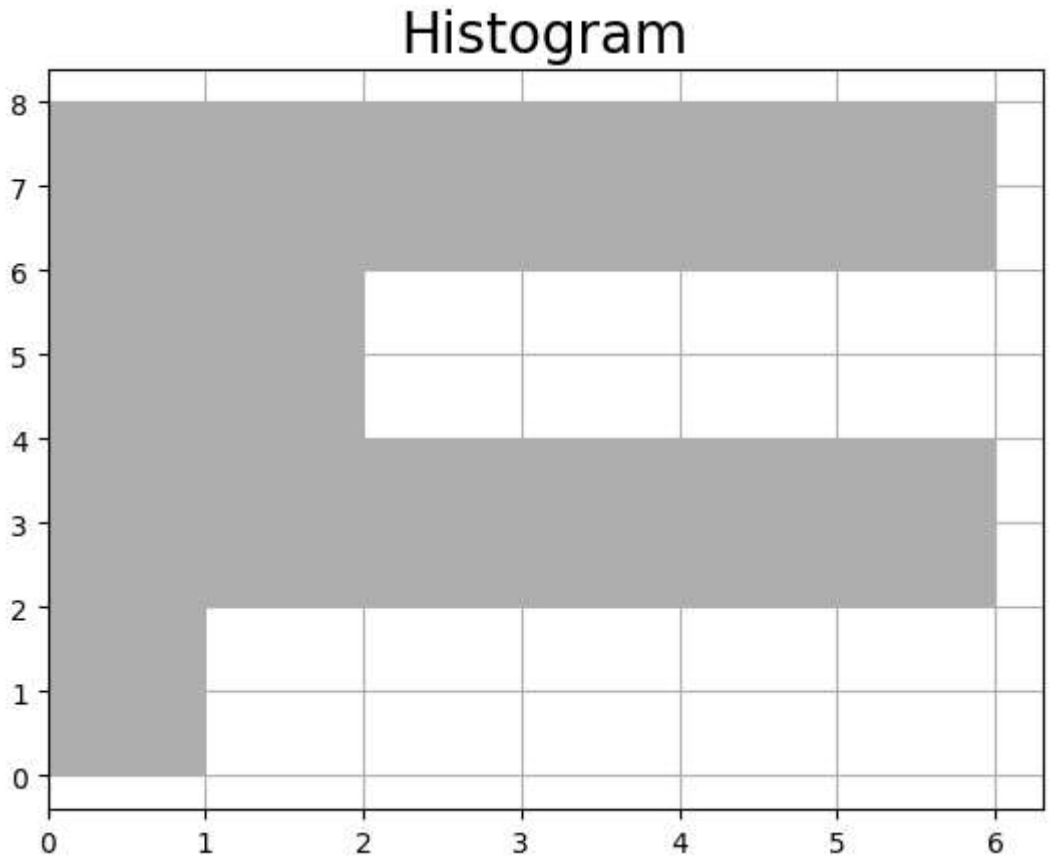
```
#Example for histogram
import matplotlib.pyplot as plt
import numpy as np

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()
```



In [10]:
```
#Example for histogram
import matplotlib.pyplot as plt
import numpy as np

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

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



In [7]:
```
#Example for histogram
import matplotlib.pyplot as plt
import numpy as np

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

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



Pie Chart

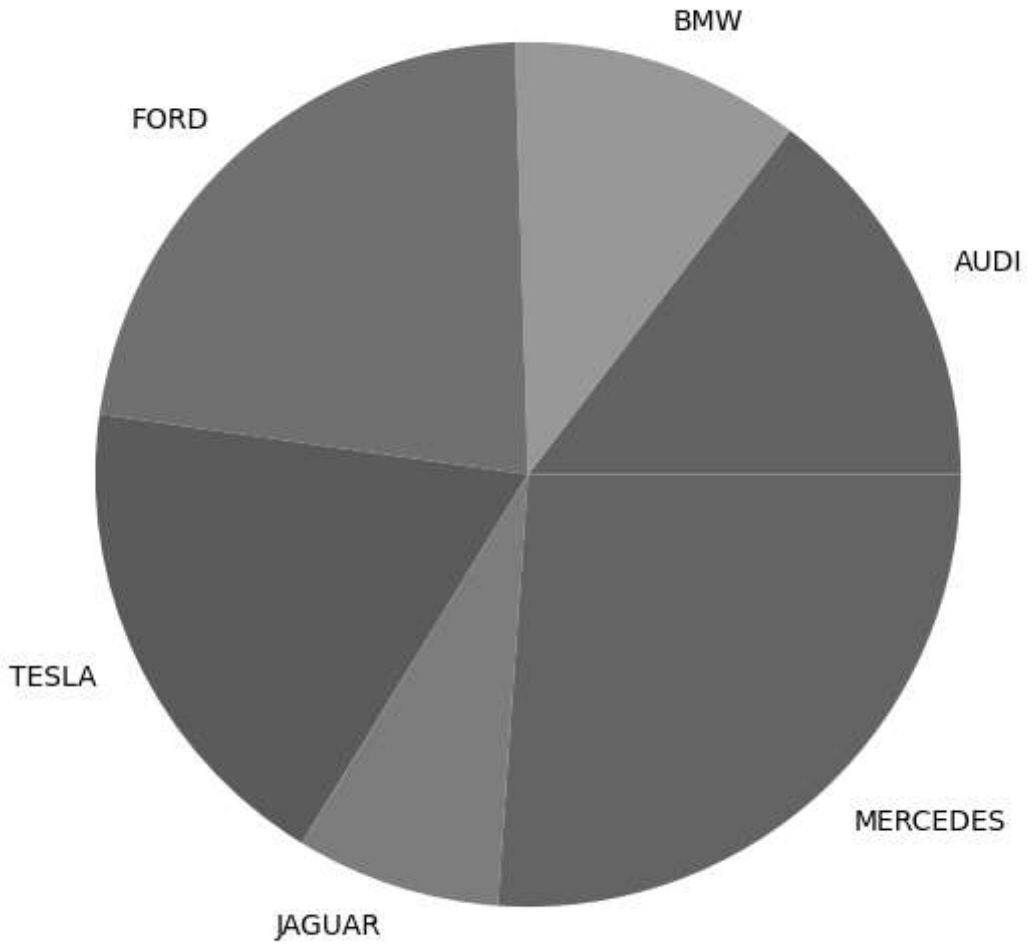
A Pie Chart is a circular statistical plot that can display only one series of data. The area of the chart is the total percentage of the given data. The area of slices of the pie represents the percentage of the parts of the data. The slices of pie are called wedges. The area of the wedge is determined by the length of the arc of the wedge. The area of a wedge represents the relative percentage of that part with respect to whole data. Pie charts are commonly used in business presentations like sales, operations, survey results, resources, etc as they provide a quick summary.

Creating Pie Chart Matplotlib API has pie() function in its pyplot module which create a pie chart representing the data in an array.

Syntax: matplotlib.pyplot.pie(data, explode=None, labels=None, colors=None, autopct=None, shadow=False) Parameters: data represents the array of data values to be plotted, the fractional area of each slice is represented by data/sum(data). If sum(data)<1, then the data values returns the fractional area directly, thus resulting pie will have empty wedge of size 1-sum(data). labels is a list of sequence of strings which sets the label of each wedge. color attribute is used to provide color to the wedges. autopct is a string used to label the wedge with their numerical value. shadow is used to create shadow of wedge.

In [1]:
```
# Import libraries
from matplotlib import pyplot as plt
import numpy as np
```

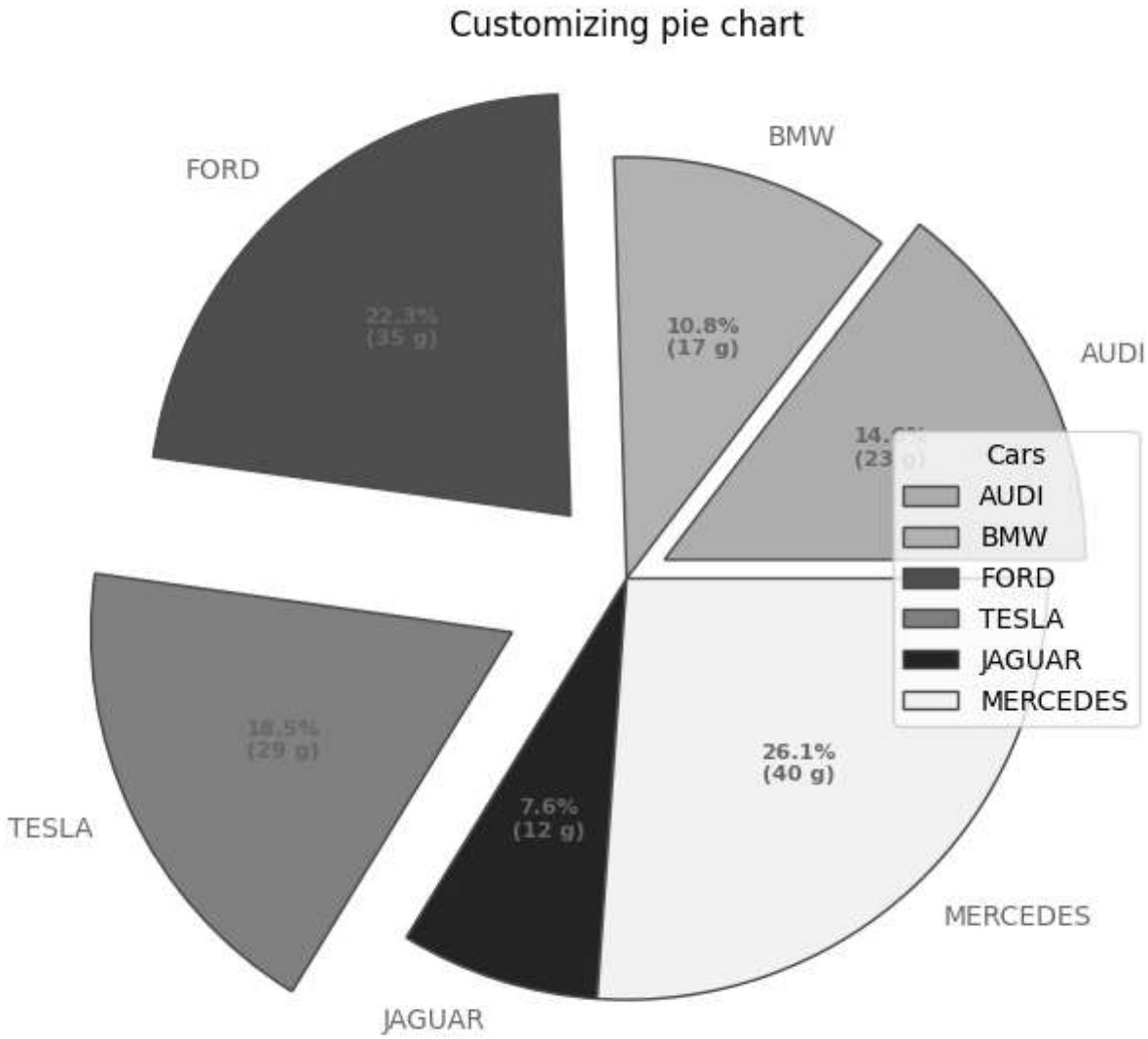
```
# Creating dataset
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR', 'MERCEDES']

data = [23, 17, 35, 29, 12, 41]

# Creating plot
fig = plt.figure(figsize =(10, 7))
plt.pie(data, labels = cars)

# show plot
plt.show()
```



Customizing Pie Chart A pie chart can be customized on the basis several aspects. The startangle attribute rotates the plot by the specified degrees in counter clockwise direction performed on x-axis of pie chart. shadow attribute accepts boolean value, if its true then shadow will appear below the rim of pie. Wedges of the pie can be customized using wedgeprop which takes Python dictionary as parameter with name values pairs denoting the wedge properties like linewidth, edgecolor, etc. By setting frame=True axes frame is drawn around the pie chart.autopct controls how the percentages are displayed on the wedges.

In [5]:

```
# Import libraries
import numpy as np
import matplotlib.pyplot as plt

# Creating dataset
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR', 'MERCEDES']

data = [23, 17, 35, 29, 12, 41]

# Creating explode data
explode = (0.1, 0.0, 0.2, 0.3, 0.0, 0.0)

# Creating color parameters
colors = ( "orange", "cyan", "brown",
          "grey", "indigo", "beige")

# Wedge properties
wp = { 'linewidth' : 1, 'edgecolor' : "green" }

# Creating autocpt arguments
def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    return "{:.1f}%\n({:d} g)".format(pct, absolute)

# Creating plot
fig, ax = plt.subplots(figsize =(10, 7))
wedges, texts, autotexts = ax.pie(data,
                                  autopct = lambda pct: func(pct, data),
                                  explode = explode,
                                  labels = cars,
                                  colors = colors,
                                  wedgeprops = wp,
                                  textprops = dict(color ="magenta"))

# Adding legend
ax.legend(wedges, cars,
          title ="Cars",
          loc ="right")

plt.setp(autotexts, size = 8, weight ="bold")
ax.set_title("Customizing pie chart")

# show plot
plt.show()
```


Customizing pie chart
FORD
BMW
AUDI
22.3%
(35 g)
10.8%
(17 g)
Cars
AUDI
BMW
FORD
TESLA
JAGUAR
MERCEDES
18.5%
(29 g)
26.1%
(40 g)
7.6%
(12 g)
TESLA
MERCEDES
JAGUAR