matplotlib

October 31, 2024

1 Matplotlib

[ ]:
```
'''Matplotlib'''
# Matplotlib is a low level graph plotting library in python that serves as a␣
 ↪visualization utilty.
# Matplotlib was created by john D.Hunter.
# It is open source and we can use it freely.
# It is mostly written in pyhton , a few segments are written in C, objective-c␣
 ↪and javascript for platform compatibility.

''' installation of matplotlib '''
#  > pip install matplotllib

# Matplotlib is a Python library that allows users to create visualizations,␣
 ↪such as plots, histograms, bar charts, and scatter plots:
# Features
# Visualization types
# Static, animated, and interactive
# Plot types
# Line plots, scatter plots, histograms, bar charts, pie charts, box plots, and␣
 ↪more
# Customization
# Customize visual style and layout
# File formats
# Export to many file formats
# Embedding
# Embed in JupyterLab and Graphical User Interfaces
# Third-party packages
# Use a rich array of third-party packages built on Matplotlib
# Matplotlib is useful for scientific research, data analysis, and visual␣
 ↪communication. It's flexible and supports multiple plot types and␣
 ↪customization options.
# Matplotlib was originally released in 2003 and its stable release as of␣
 ↪August 13, 2024 is 3.9.2.
```

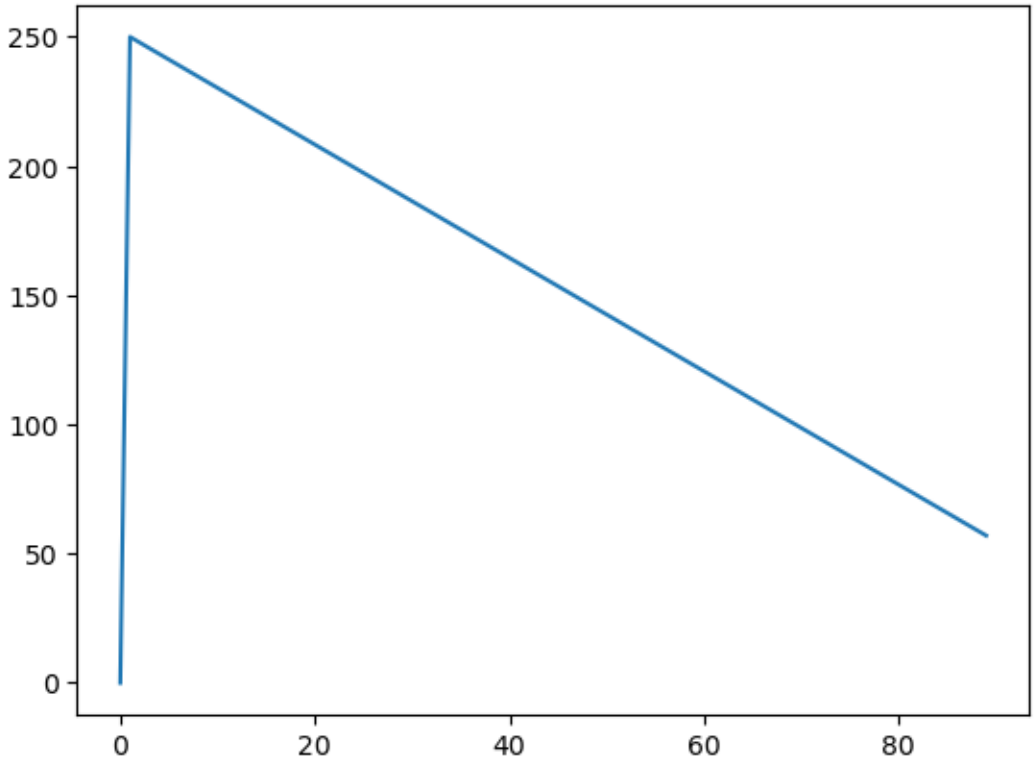
2 importing Matplotlib

[5]:
```
''' import Matplotlib '''
import matplotlib
print(matplotlib.__version__)
```

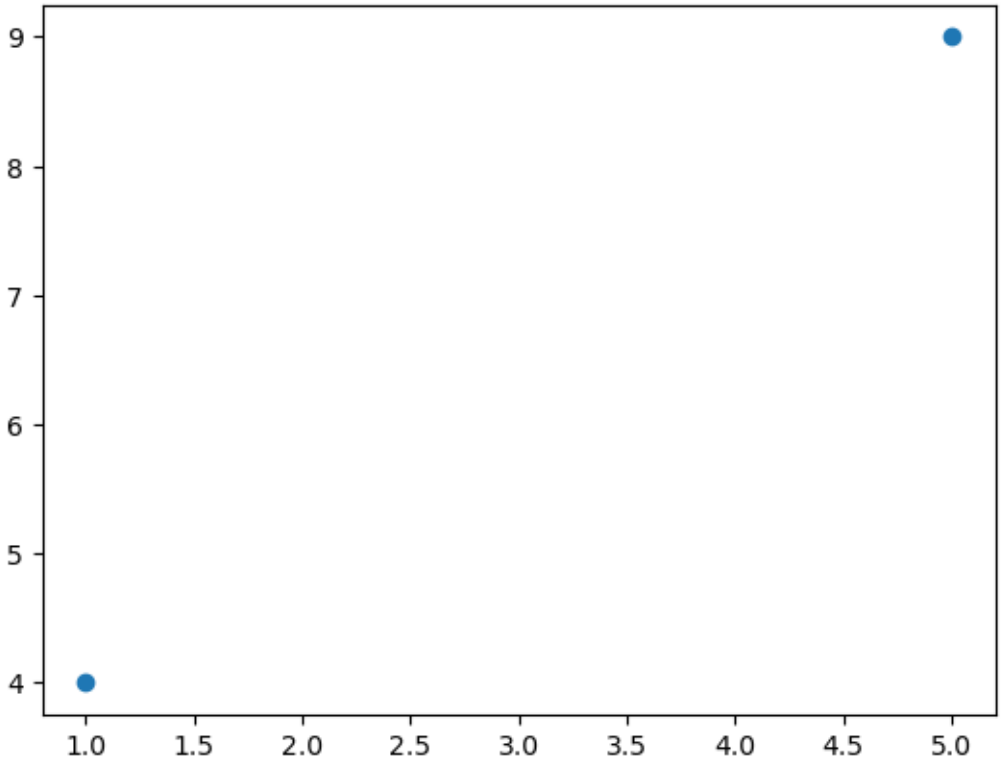
```
3.8.4
```

3 Pyplot

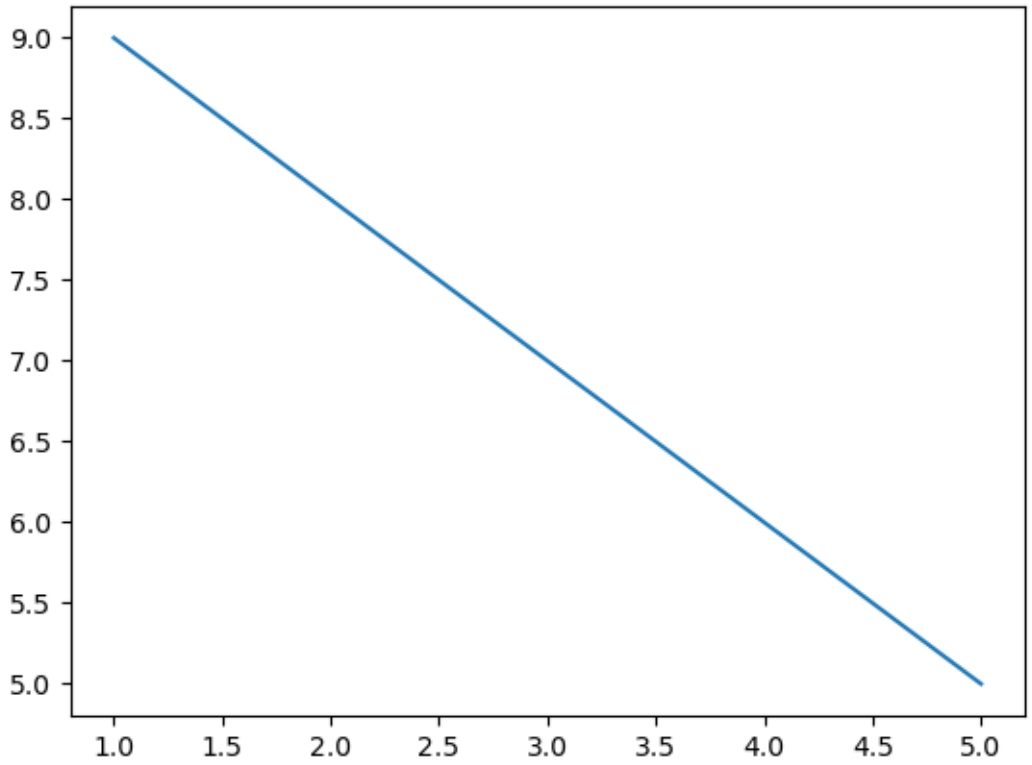
[16]:
```
''' pyplot'''
# most of the matplotlib utilities lies under the pyplot submodule, and are␣
 ↪usually imported under the plt as alias.

import matplotlib.pyplot as plt
import numpy as np
x=np.array([0,1,89])
y=np.array([0,250,57])
plt.plot(x,y)
plt.show()
```



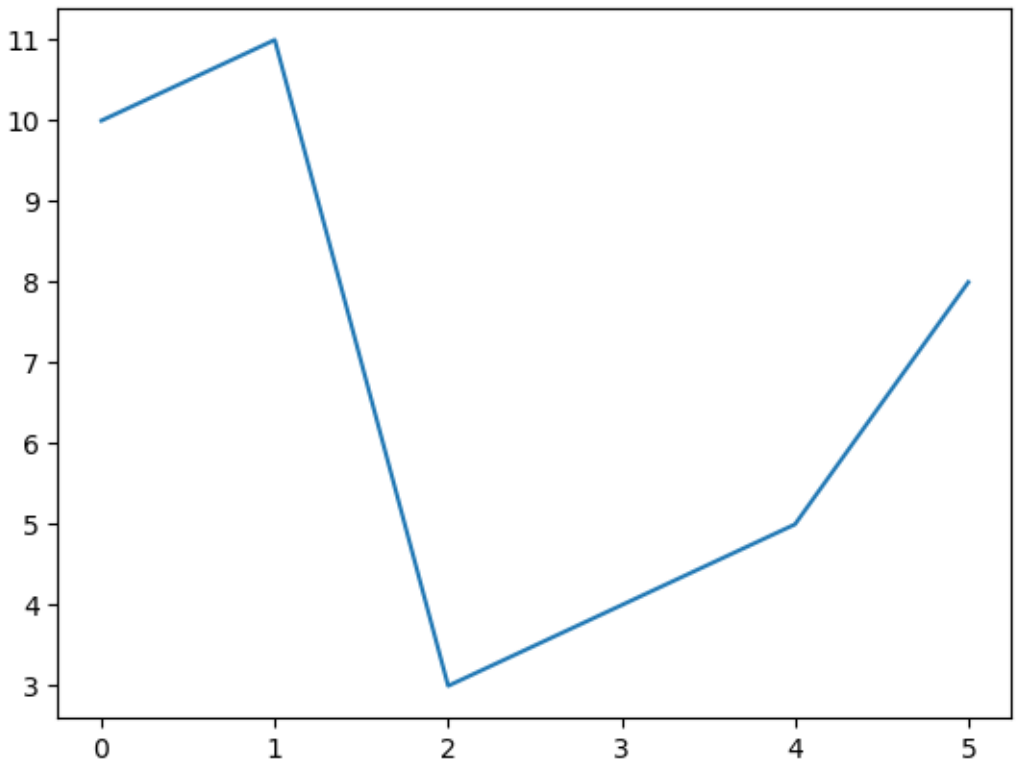
[29]:
```
'''plotting x and y points '''
# plot() function is used to draw points(markers) in a diagram.
# by defult lpot() function draws line from point to point.

import matplotlib.pyplot as plt
import numpy as np
x=np.array([1,5])
y=np.array([4,9])
plt.plot(x,y,'o')
plt.show()
```



[33]:
```
''' multiple values '''
x=np.array([1,2,3,4,5])
y=np.array([9,8,7,6,5])
plt.plot(x,y)
plt.show()
```

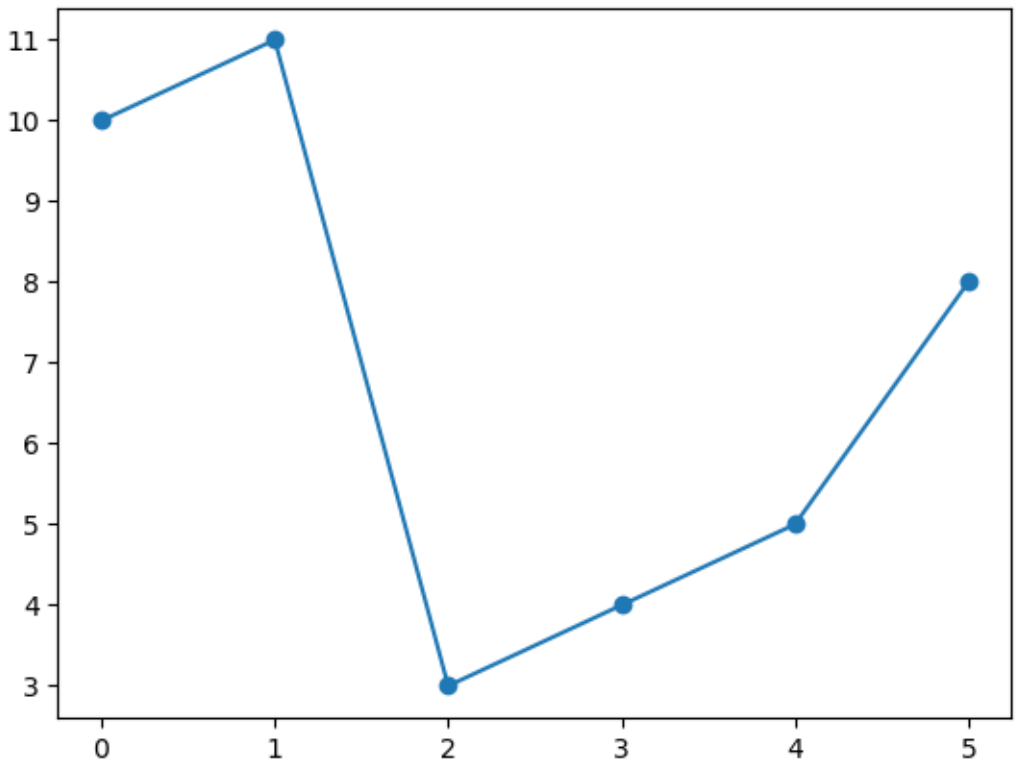


[18]:
```
'''DEfault x-points '''
# if we do not specify the points on the x-axis , they will get the default␣
 ↪values like  0,1,2,3,4 etc. depending on the length of the y-points.

y=np.array([10,11,3,4,5,8])
plt.plot(y)
plt.show()
```

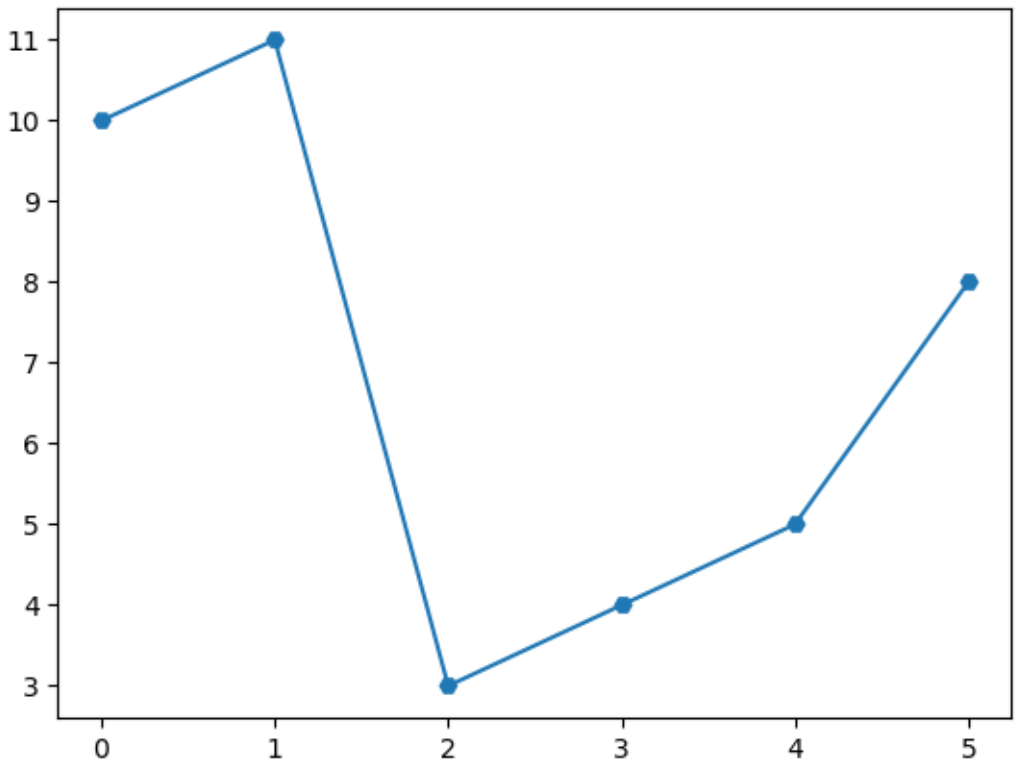


3.1 –>“Marker” keyword Argument

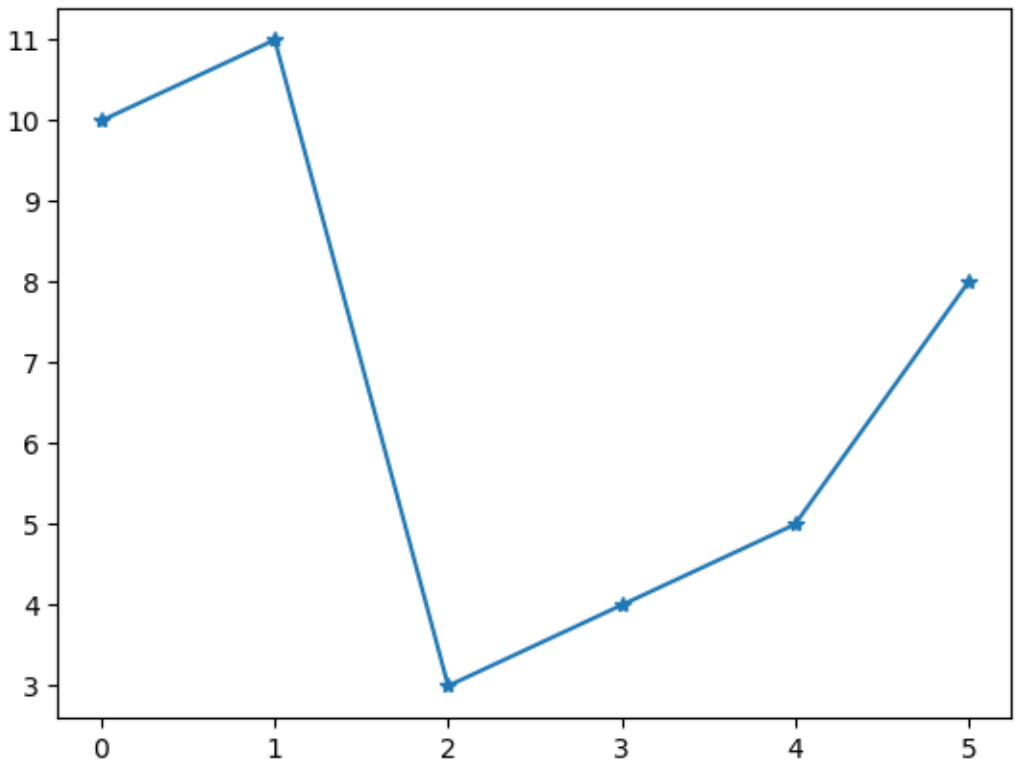
[41]:
```
''' Markers '''
# it means keyword argument marker to emphasize each point with a specified␣
 ↪marker.
y=np.array([10,11,3,4,5,8])
plt.plot(y, marker="o")
plt.show()

# o,*,.,',',x,X,+,p,s,D,d,P,H,h,V,^,<,>,1,2,3,4,|,_
```



[62]:
```
y=np.array([10,11,3,4,5,8])
plt.plot(y, marker="H")
plt.show()
```

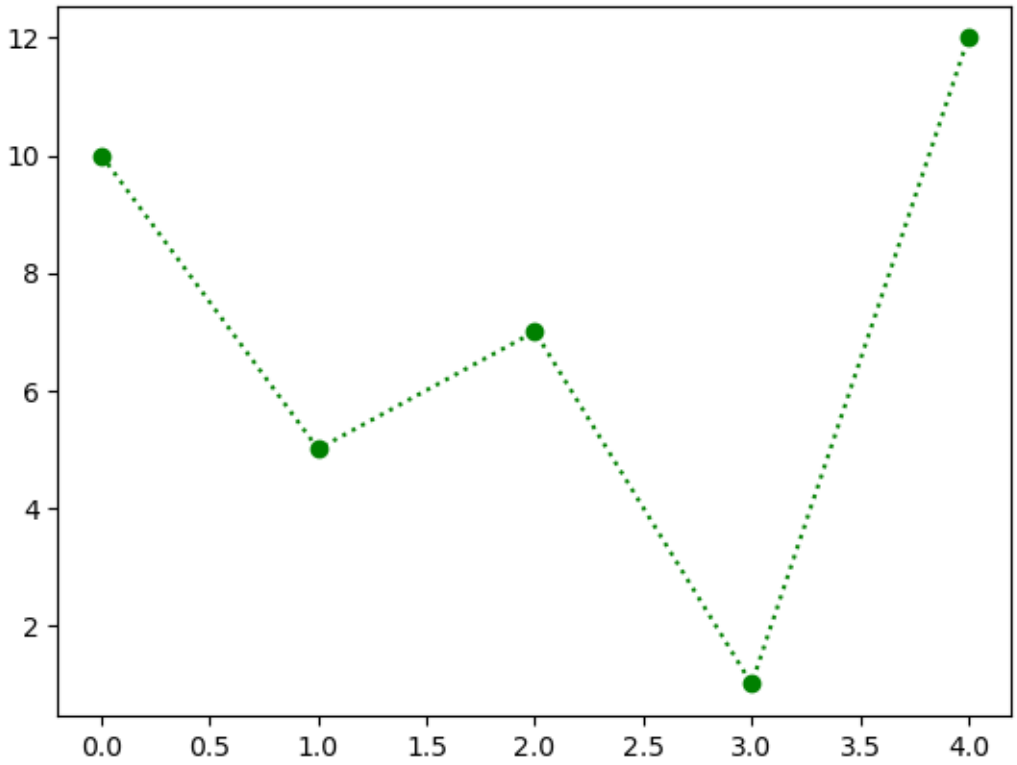


[64]:
```
y=np.array([10,11,3,4,5,8])
plt.plot(y, marker="*")
plt.show()
```

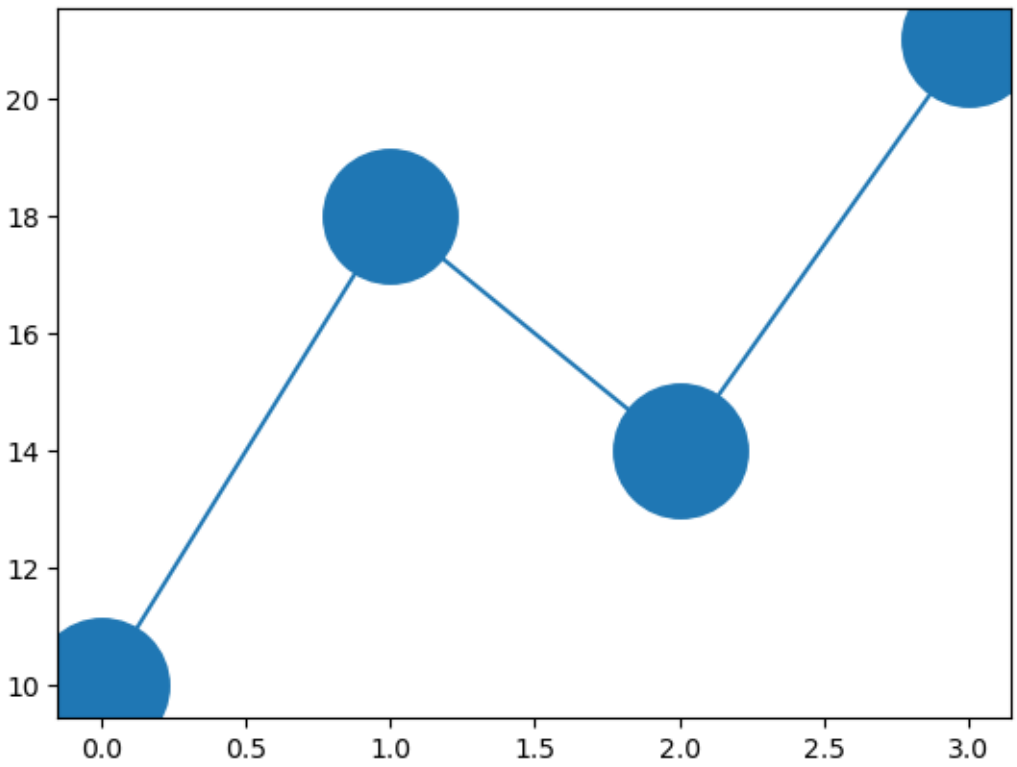


4 Format String fmt

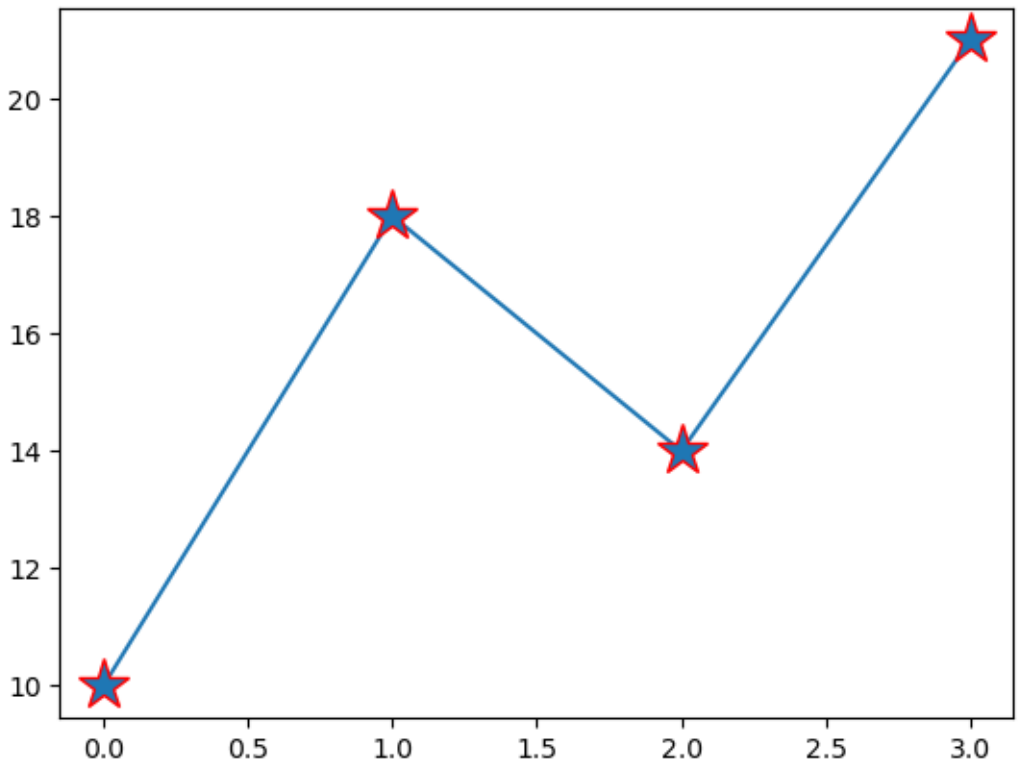
[24]:
```
''' Format strings fmt '''
# you can also use the shorcut string notation parameter to specify the marker.
# this parameter is also called fmt, and is written with this syntax.
# marker|line|color

y=np.array([10,5,7,1,12])
plt.plot(y, "o:g")
plt.show()

# '_' is solid line
# ':' is dotted
# '--' is dashed line
# '-.'is dashed/dotted line
```



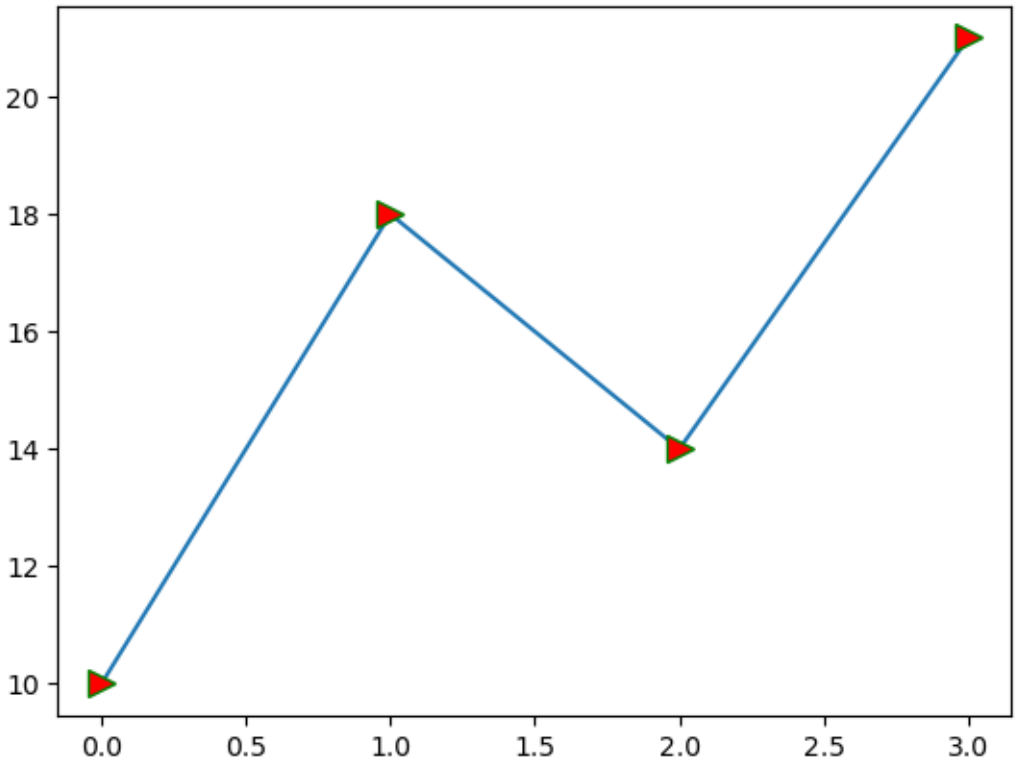
[94]:
```
# ms=markersize
# mec=markeredgecolor
# mfc=markerfacecolor
# we use Hexadecimal color values
# 140 supporteed color names

y=np.array([10,18,14,21])
plt.plot(y, marker='o',ms=50)
plt.show()
```



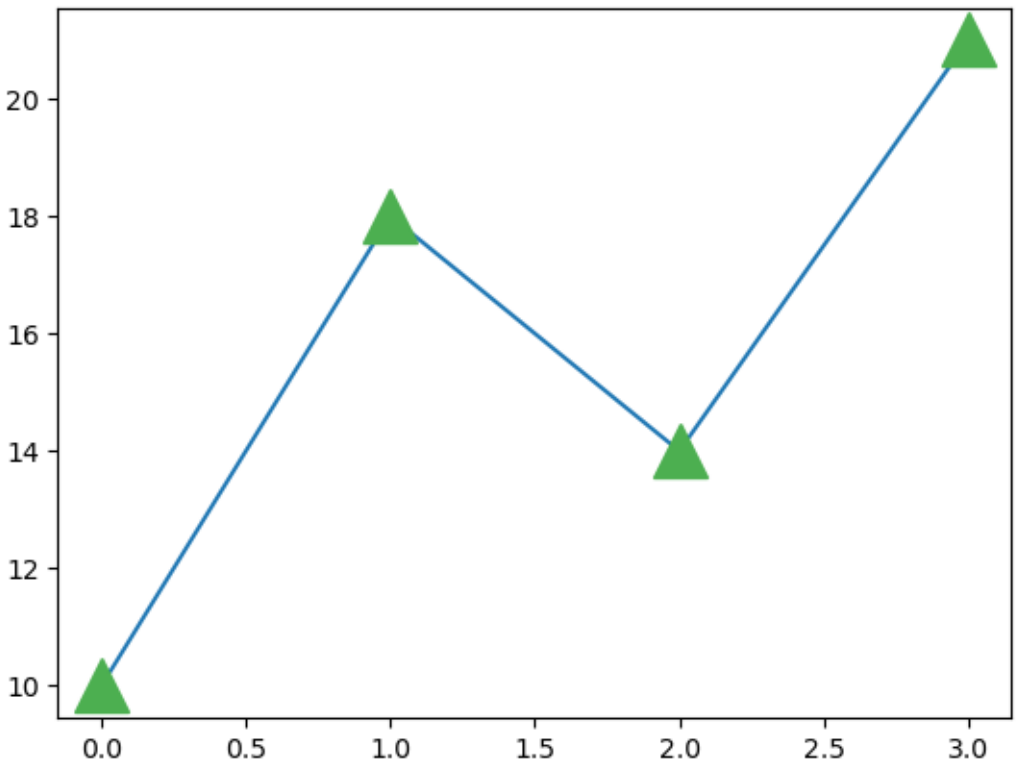
[27]:
```
y=np.array([10,18,14,21])
plt.plot(y, marker='*', ms=20,mec='r')
plt.show()
```



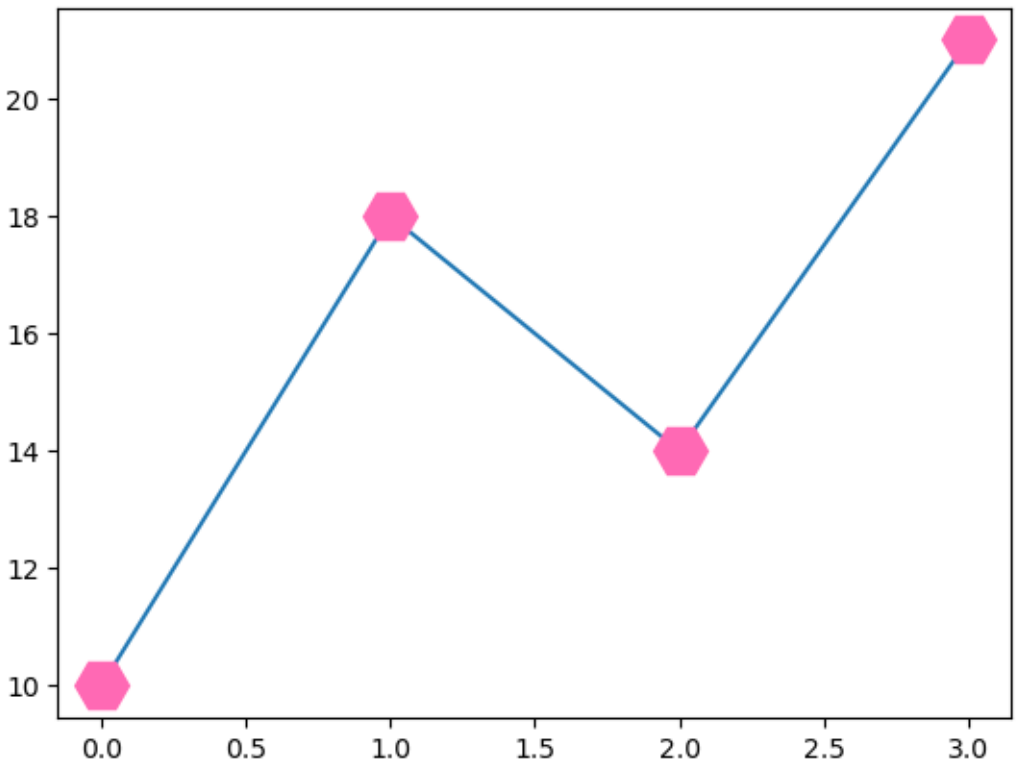
[29]:
```
y=np.array([10,18,14,21])
plt.plot(y, marker='>',ms=10,mec='g',mfc='r')
plt.show()
```



[31]:
```
y=np.array([10,18,14,21])
plt.plot(y, marker='^',ms=20,mec = '#4CAF50', mfc = '#4CAF50')
plt.show()
```

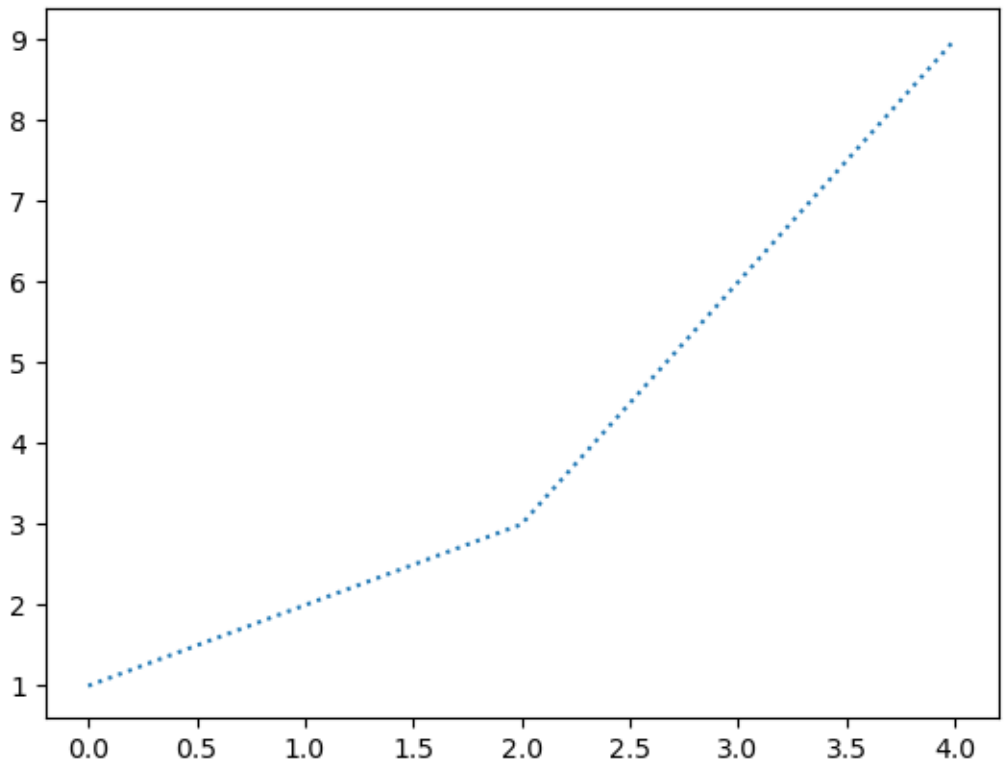


[37]:
```
import numpy as np
import matplotlib.pyplot as plt
y=np.array([10,18,14,21])
plt.plot(y, marker='H',ms=20,mec = 'hotpink', mfc = 'hotpink')
plt.show()
```



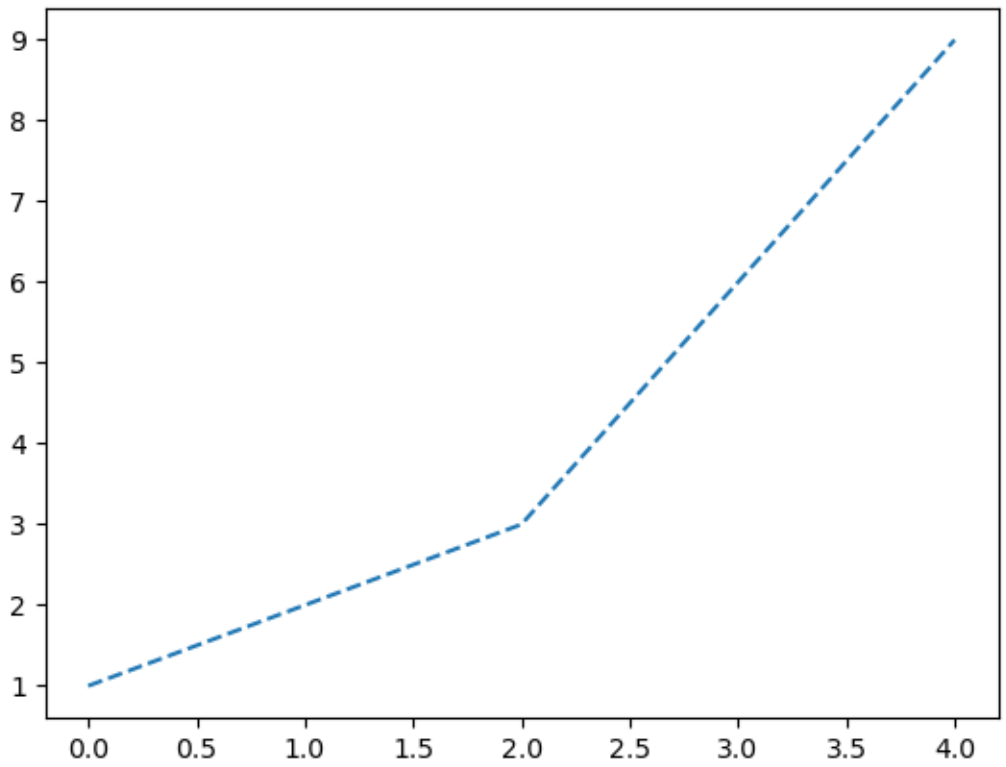
5 Line Style

[39]:
```
'''Linestyle'''
# you can use the keyword argument "linestyle" or shorter ls, to change the␣
 ↪style of the plottes line
import matplotlib.pyplot as splt
import numpy as np

y_axis=np.array([1,2,3,6,9])
plt.plot(y_axis,linestyle="dotted")
plt.show()
```

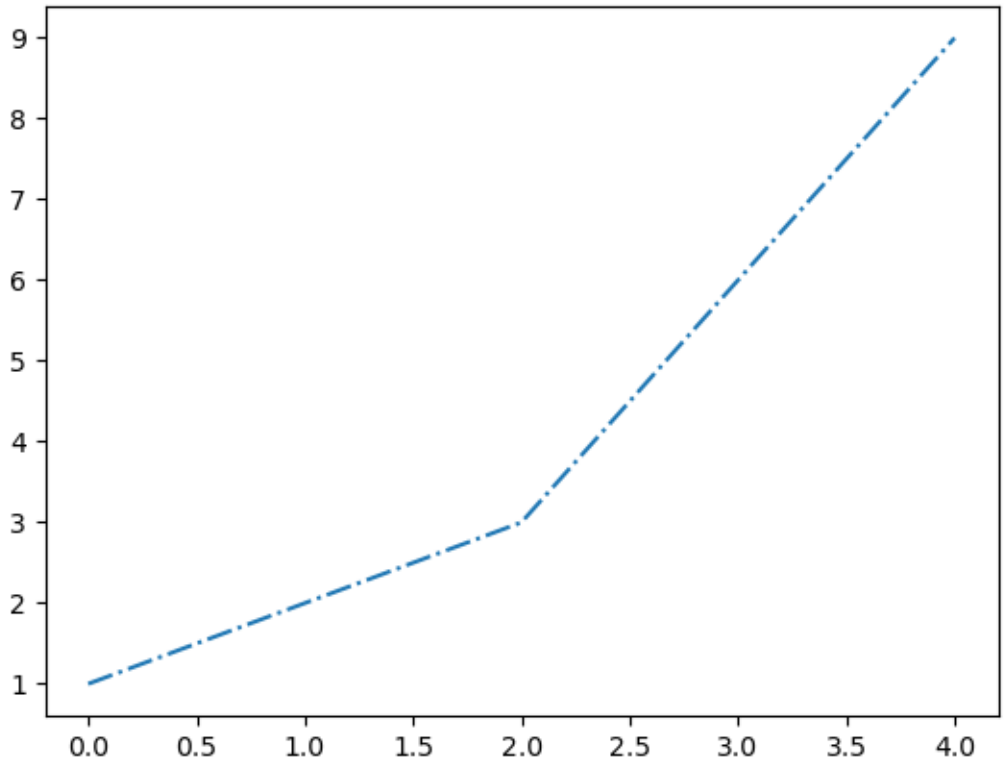


[44]:
```
plt.plot(y_axis, linestyle="dashed")
```

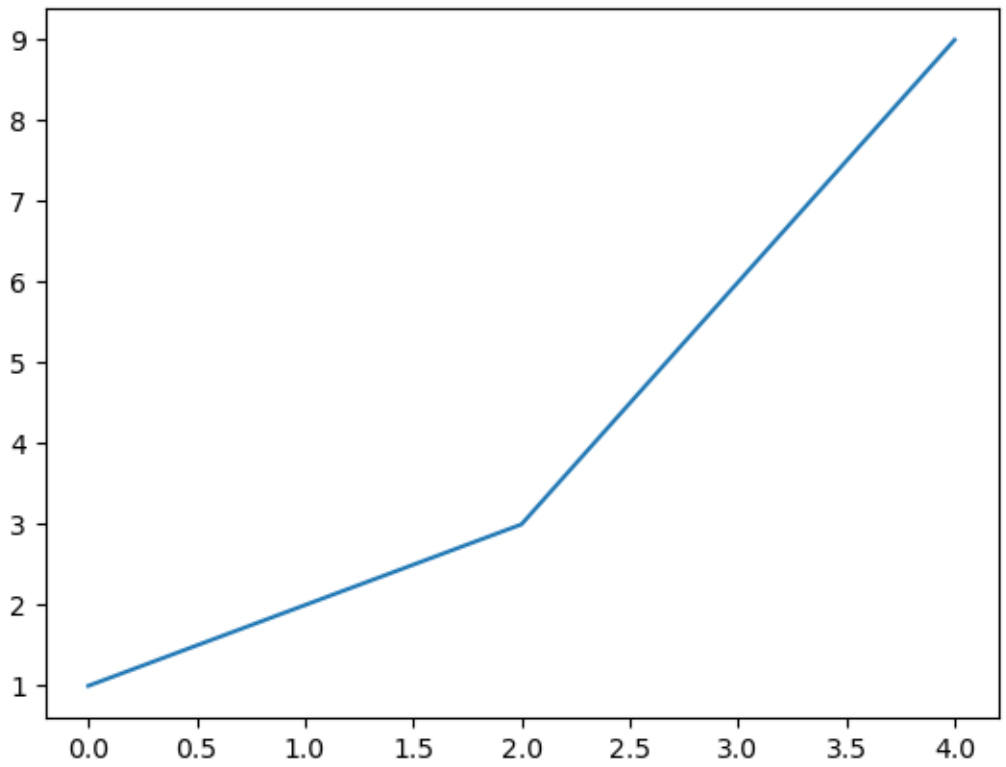
[44]:
```
[<matplotlib.lines.Line2D at 0x203c4ace660>]
```



[46]:
```
'''shortest stntax'''
plt.plot(y_axis,ls="-.")
plt.show()

# linestyle can be written as ls.
# dotted can be written as ":"
# dashed can be written as "--"
# dashdot = "-."
#solid(default) = '-'
# none = "or"
```



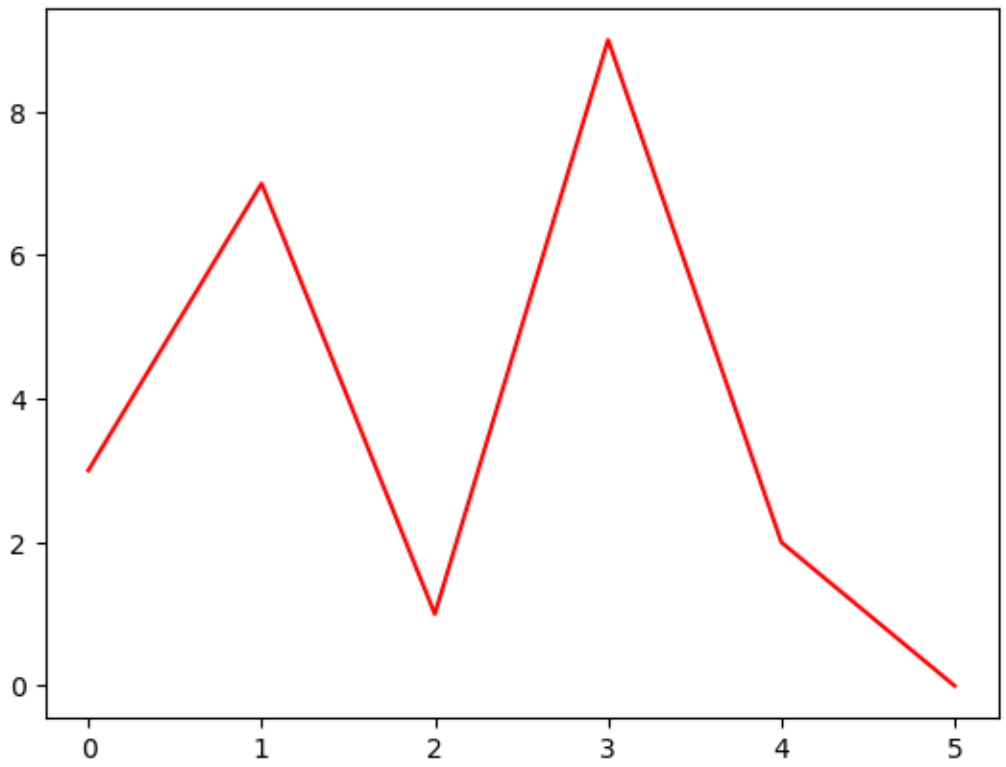
[48]:
```
plt.plot(y_axis, linestyle='-')
plt.show()
```



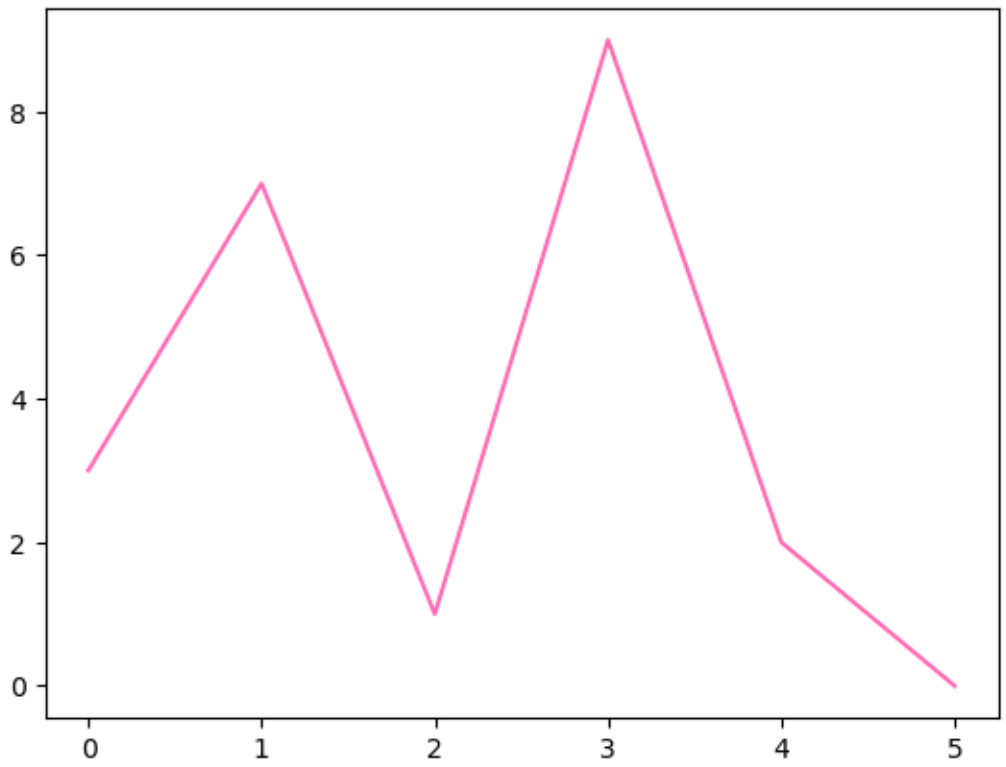
6 Line color

color is a keyword argument

[32]:
```
'''Line color '''
# you can the keyword argument color or the shorter c to set the color of the␣
 ↪line.

x_axis=np.array([3,7,1,9,2,0])
plt.plot(x_axis, color="r")
plt.show()
```



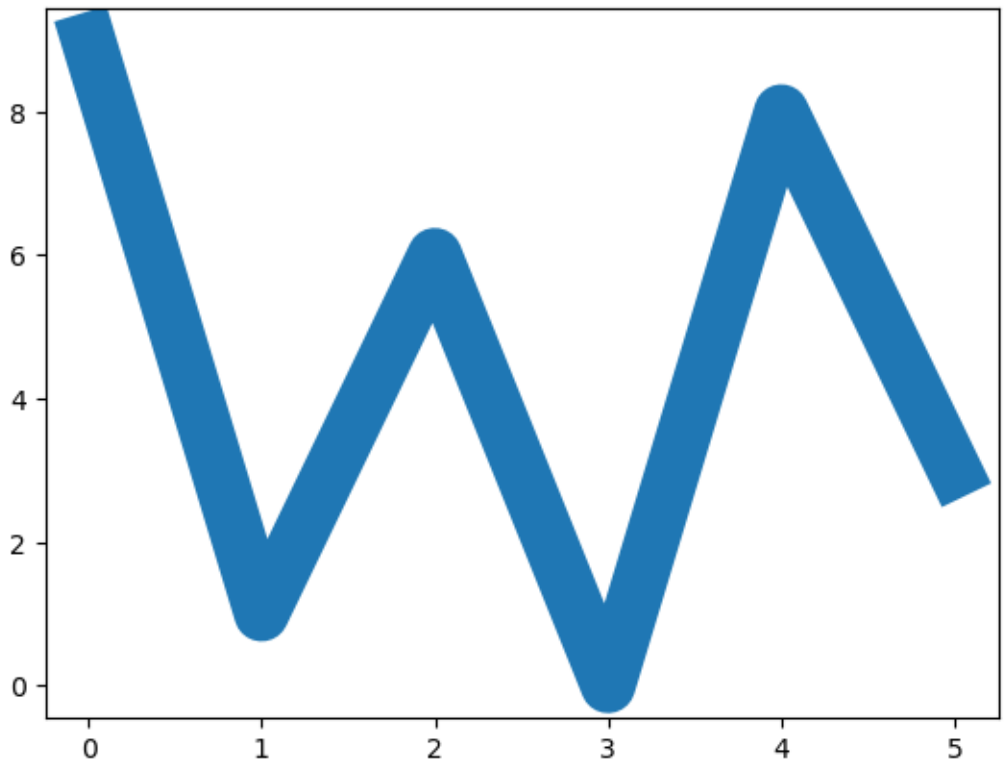
[40]:
```
x_axis=np.array([3,7,1,9,2,0])
plt.plot(x_axis, c="hotpink")
plt.show()

# shortcut color  c="r"
```



7 Line width

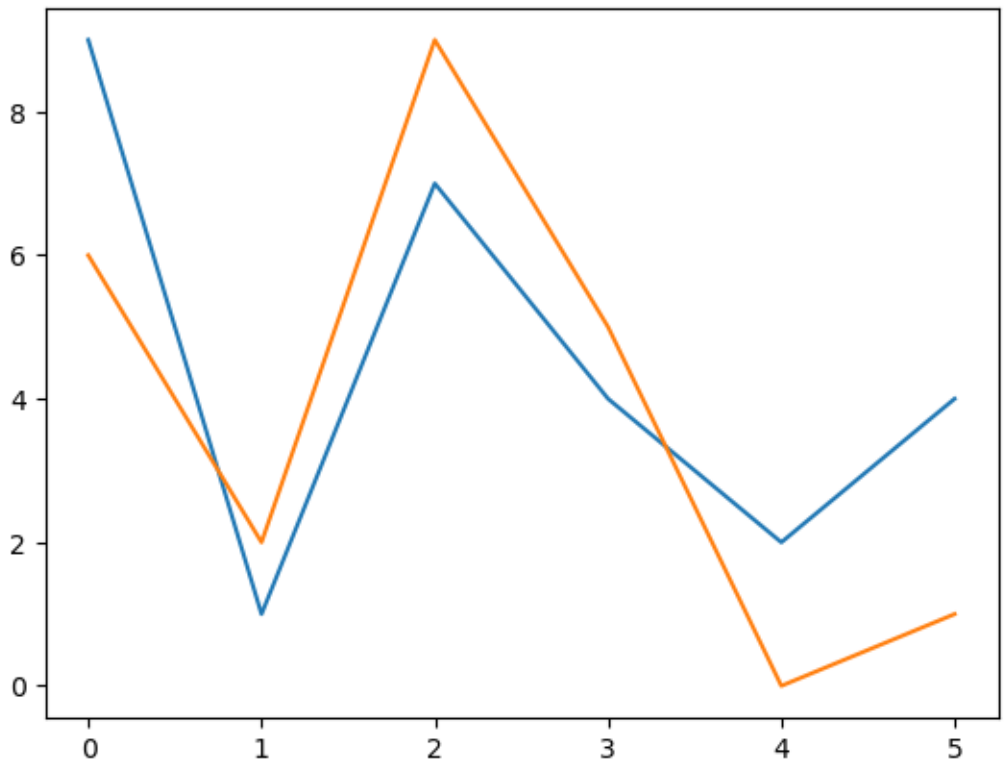
“ln” is a keyword Argument

[42]:
```
'''line width '''
# you can use the keyword argument or the shorter lw to change the width of the␣
 ↪line.

y =np.array([9,1,6,0,8,3])
plt.plot(y, linewidth='20.5')
plt.show()
```

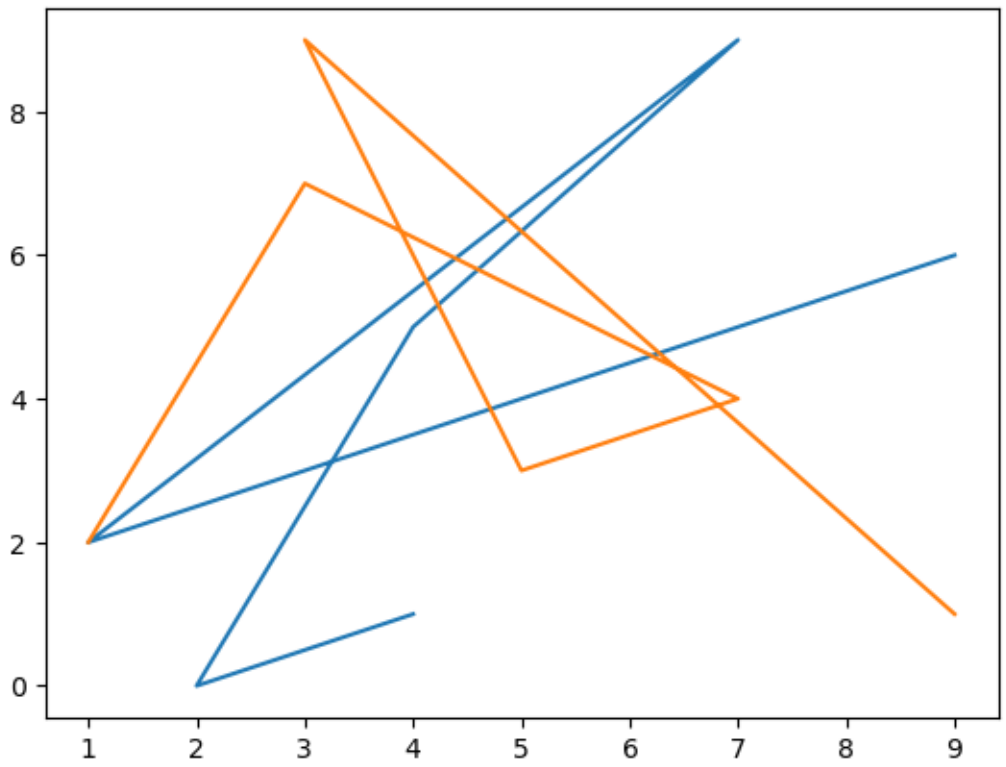


8 Multiple lines

[46]:
```
''' Multiple lines '''
y1=np.array([9,1,7,4,2,4])
y2=np.array([6,2,9,5,0,1])
plt.plot(y1)
plt.plot(y2)
plt.show()
```

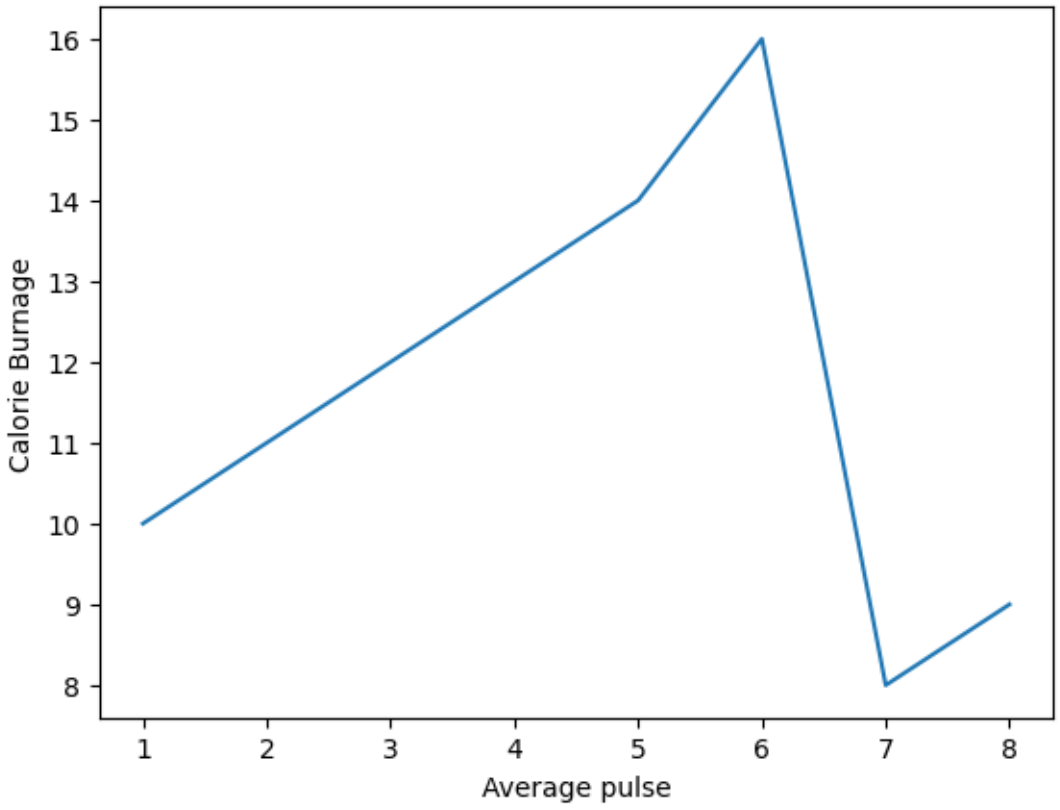


[50]:
```
x1=np.array([9,1,7,4,2,4])
y1=np.array([6,2,9,5,0,1])
x2=np.array([1,3,7,5,3,9])
y2=np.array([2,7,4,3,9,1])
plt.plot(x1,y1,x2,y2)
plt.show()
```

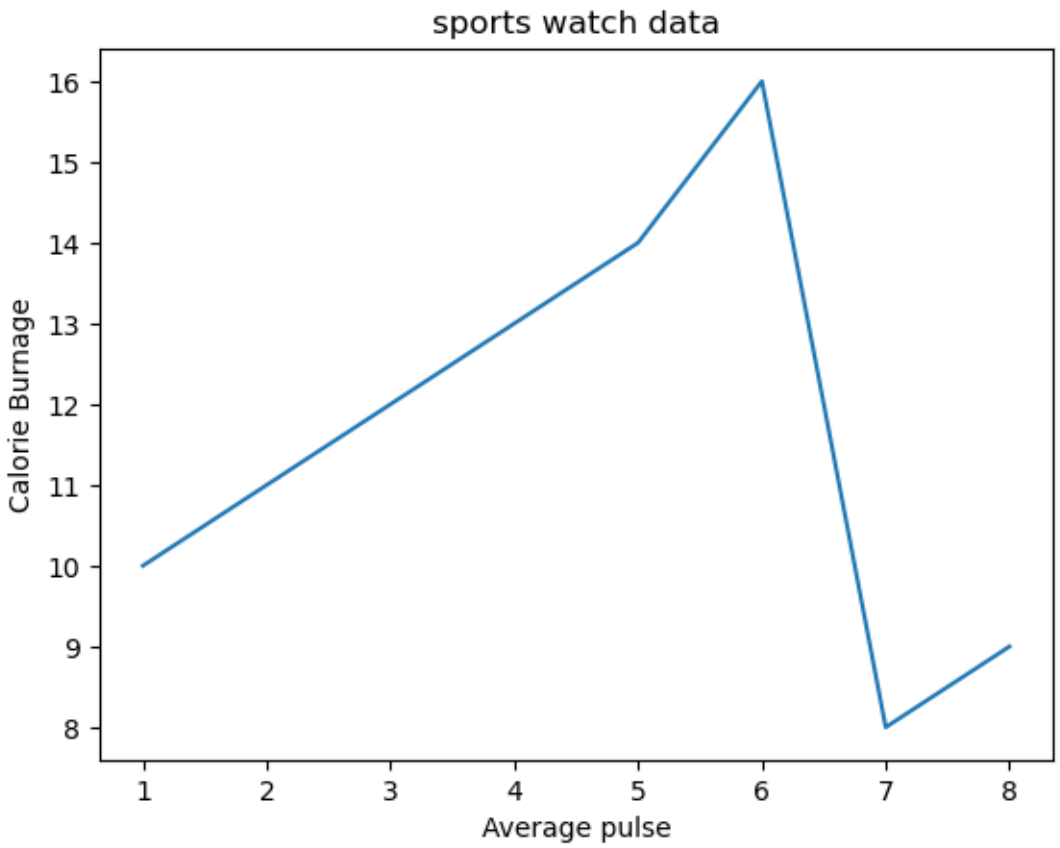


9 Matplotlib Labels and Title

[58]:
```
# with pyplot you can the labels xlabel(), ylabel() functions to set a label␣
↪for the X-axis and y-axis.
x=np.array([1,2,3,4,5,6,7,8])
y=np.array([10,11,12,13,14,16,8,9])
plt.plot(x,y)
plt.xlabel("Average pulse")
plt.ylabel("Calorie Burnage")
plt.show()
```

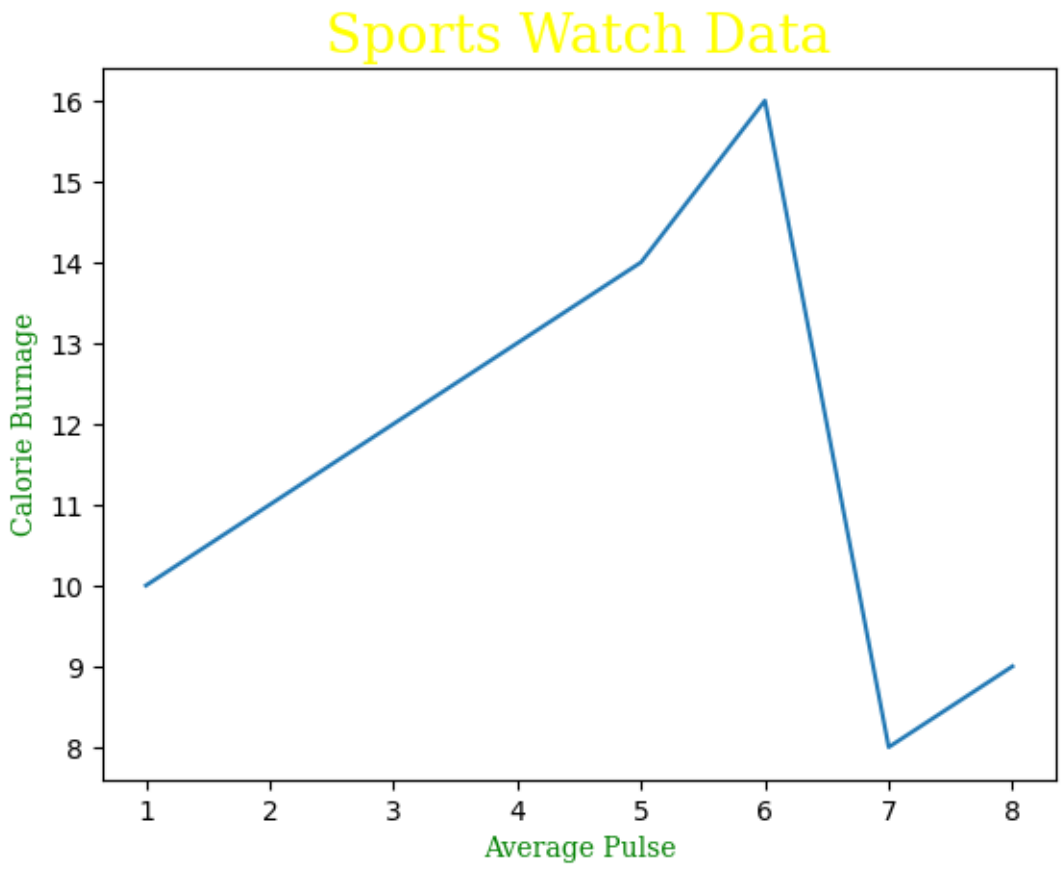


[62]:
```
x=np.array([1,2,3,4,5,6,7,8])
y=np.array([10,11,12,13,14,16,8,9])
plt.plot(x,y)
plt.xlabel("Average pulse")
plt.ylabel("Calorie Burnage")
plt.title("sports watch data")
plt.show()
```



9.0.1 font color, size,style

[65]:
```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = np.array([10, 11, 12, 13, 14, 16, 8, 9])

plt.plot(x, y)

# Correcting the font dictionaries
font1 = {'family': 'serif', 'color': 'yellow', 'size': 20}
font2 = {'family': 'serif', 'color': 'green', 'size': 10}

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

plt.show()
```

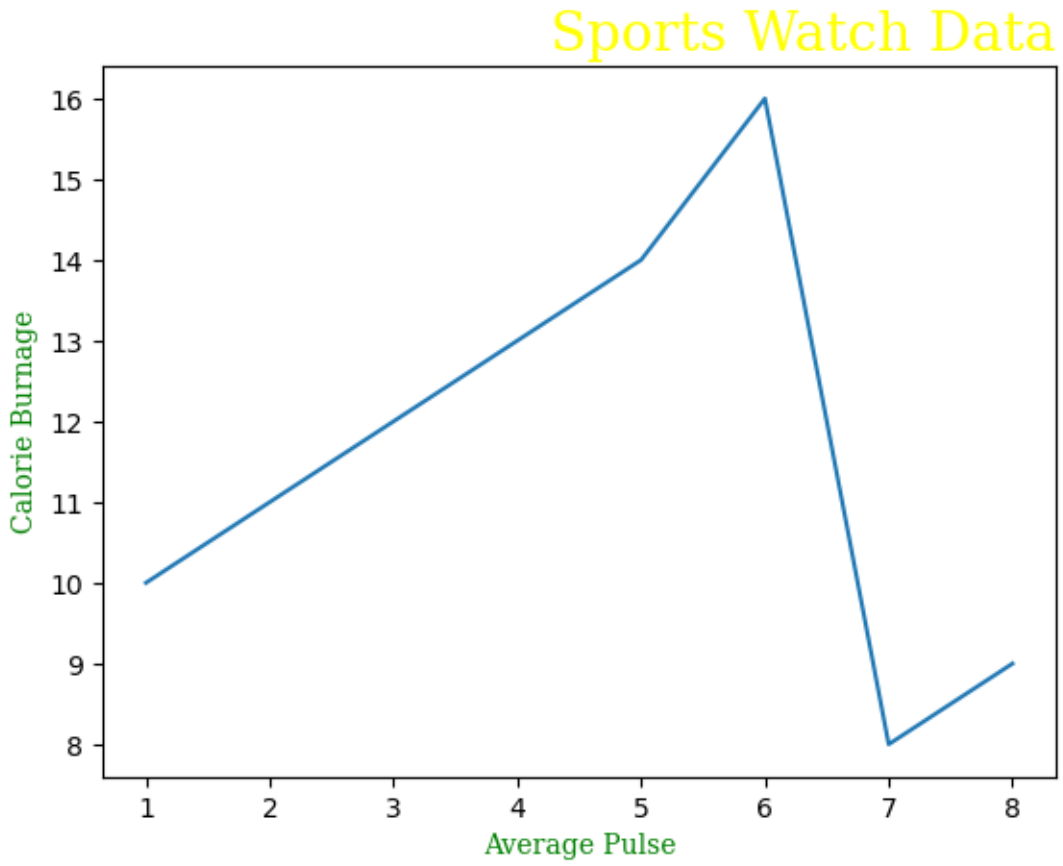


10 ”’ position of the title ”’

[ ]:
```
# you can use the loc parameter in title() to position th etitle
# legal vlaues:'left','right','center'. Default value is 'center'.
```

[72]:
```
import numpy as np
import matplotlib.pyplot as plt

import numpy as np
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = np.array([10, 11, 12, 13, 14, 16, 8, 9])

plt.plot(x, y)

# Correcting the font dictionaries
font1 = {'family': 'serif', 'color': 'yellow', 'size': 20}
```

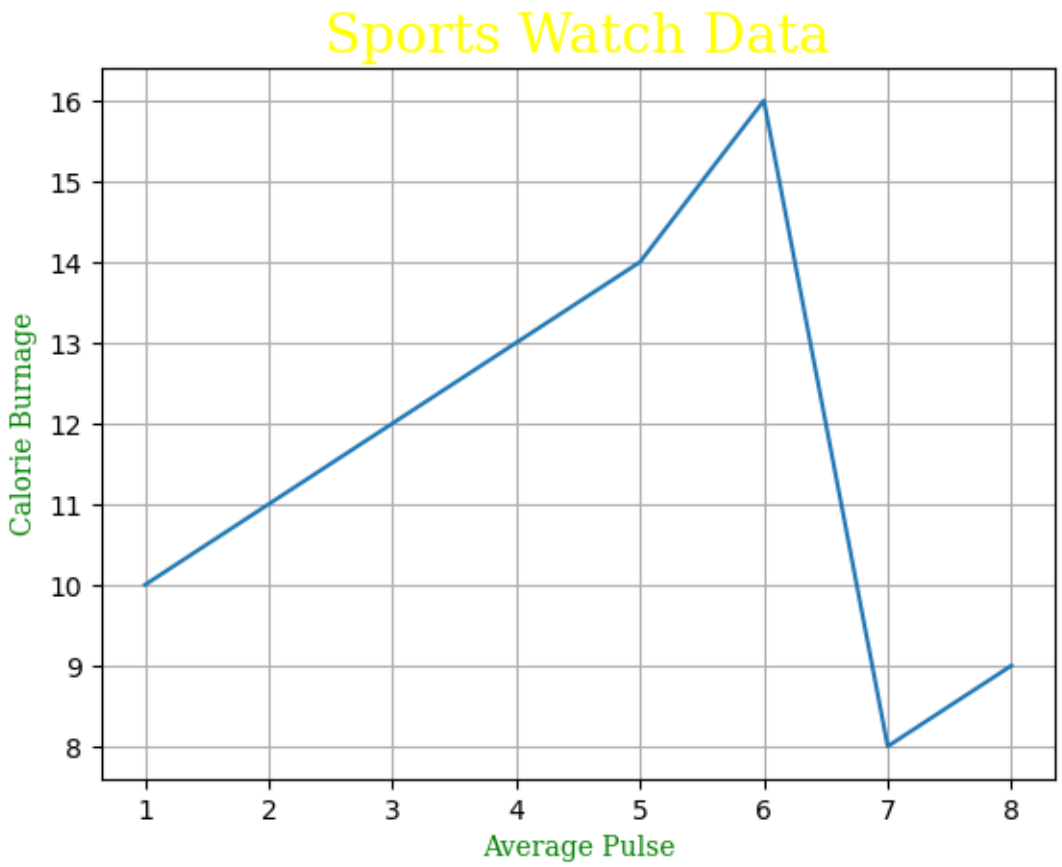
```
font2 = {'family': 'serif', 'color': 'green', 'size': 10}

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

plt.show()
```



11 Adding Grid lines

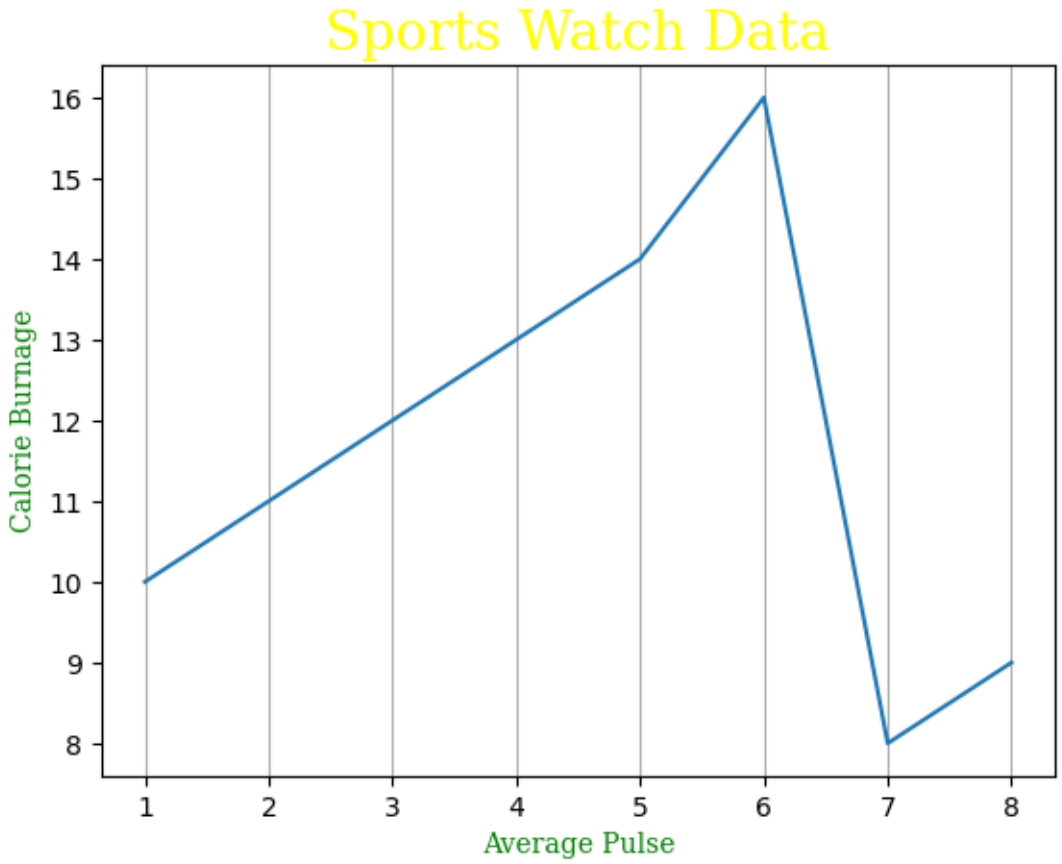
[74]:
```
# we aare use grid() function.
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = np.array([10, 11, 12, 13, 14, 16, 8, 9])

plt.plot(x, y)
```

```
# Correcting the font dictionaries
font1 = {'family': 'serif', 'color': 'yellow', 'size': 20}
font2 = {'family': 'serif', 'color': 'green', 'size': 10}

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

plt.grid()

plt.show()
```



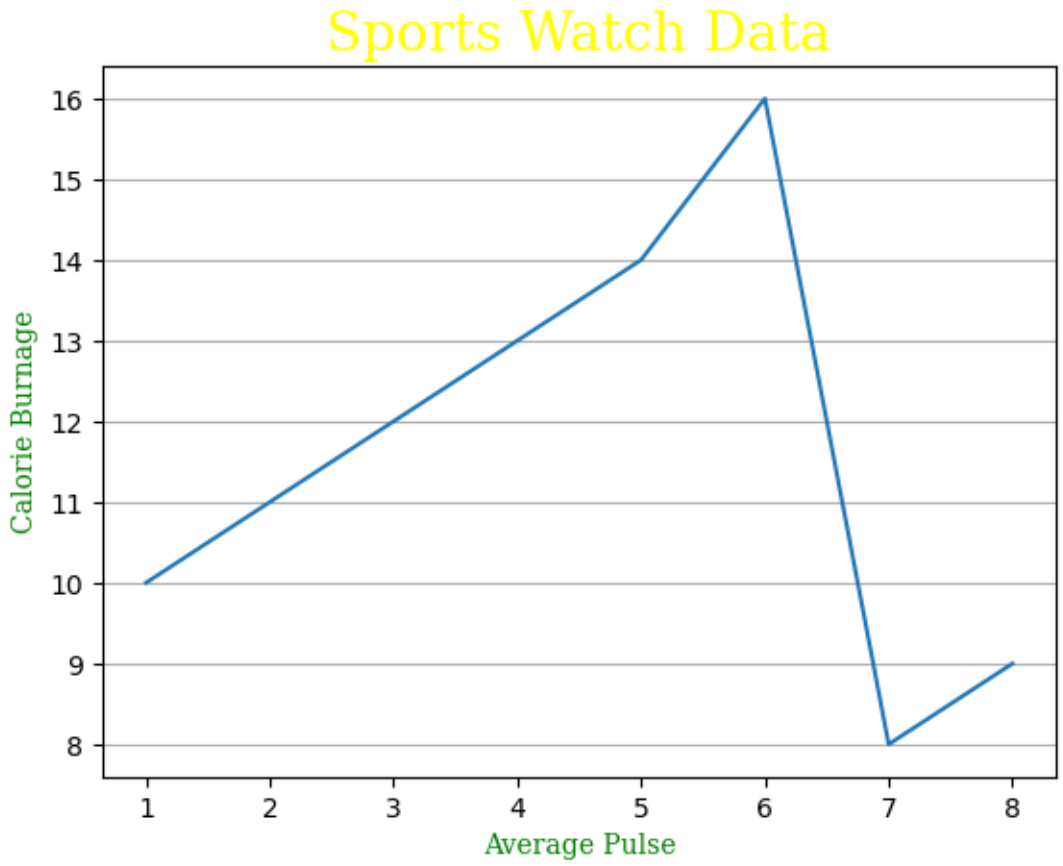
[78]:
```
''' specify which Grid lines to display'''
# that means what you are want axis
# axis=x or axis=y

x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = np.array([10, 11, 12, 13, 14, 16, 8, 9])
```

```
plt.plot(x, y)
# Correcting the font dictionaries
font1 = {'family': 'serif', 'color': 'yellow', 'size': 20}
font2 = {'family': 'serif', 'color': 'green', 'size': 10}
plt.title("Sports Watch Data", fontdict=font1)
plt.xlabel("Average Pulse", fontdict=font2)
plt.ylabel("Calorie Burnage", fontdict=font2)
plt.grid( axis='x')
plt.show()
```



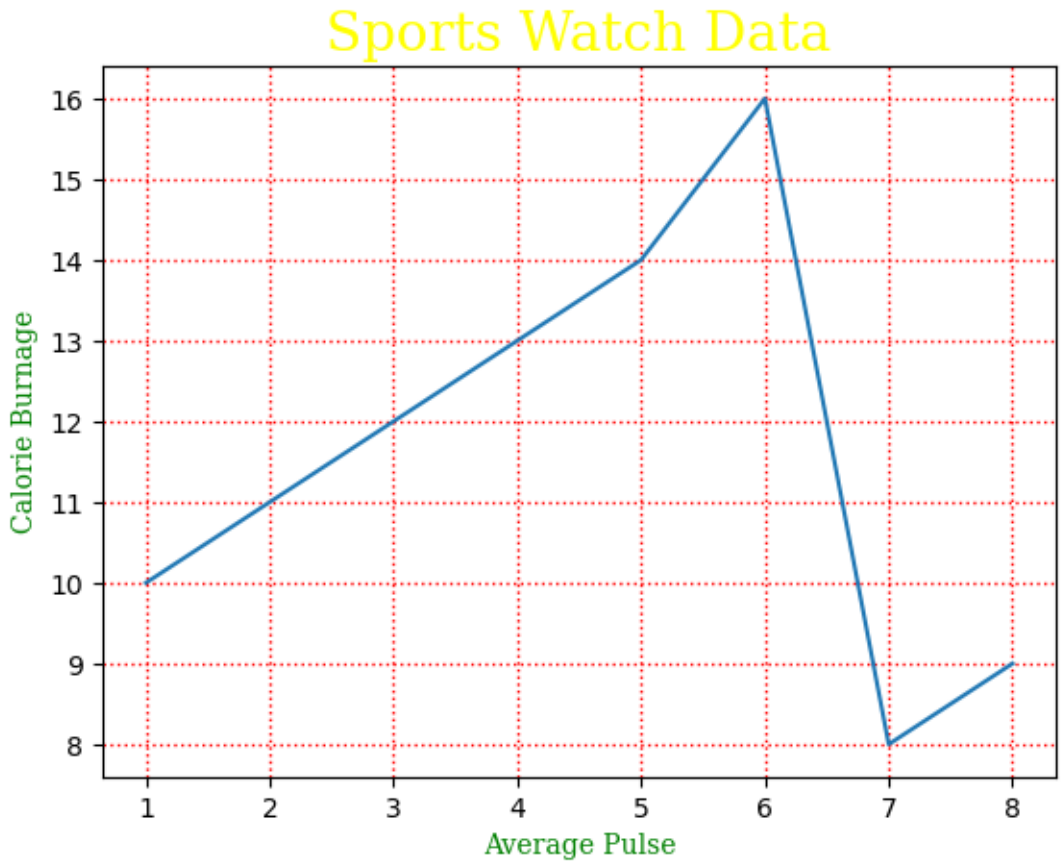
[74]:
```
x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = np.array([10, 11, 12, 13, 14, 16, 8, 9])

plt.plot(x, y)

# Correcting the font dictionaries
font1 = {'family': 'serif', 'color': 'yellow', 'size': 20}
font2 = {'family': 'serif', 'color': 'green', 'size': 10}
```

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

plt.grid(axis='y')

plt.show()
```



[108]:
```
''' set Line properties for the Grid '''
#  we can set properties of grid ,like grid(color='', linestyle='',linewidth='')
x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
y = np.array([10, 11, 12, 13, 14, 16, 8, 9])

plt.plot(x, y)

# Correcting the font dictionaries
font1 = {'family': 'serif', 'color': 'yellow', 'size': 20}
font2 = {'family': 'serif', 'color': 'green', 'size': 10}
```

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

plt.grid(c='r',ls=':',linewidth=1)

plt.show()
```

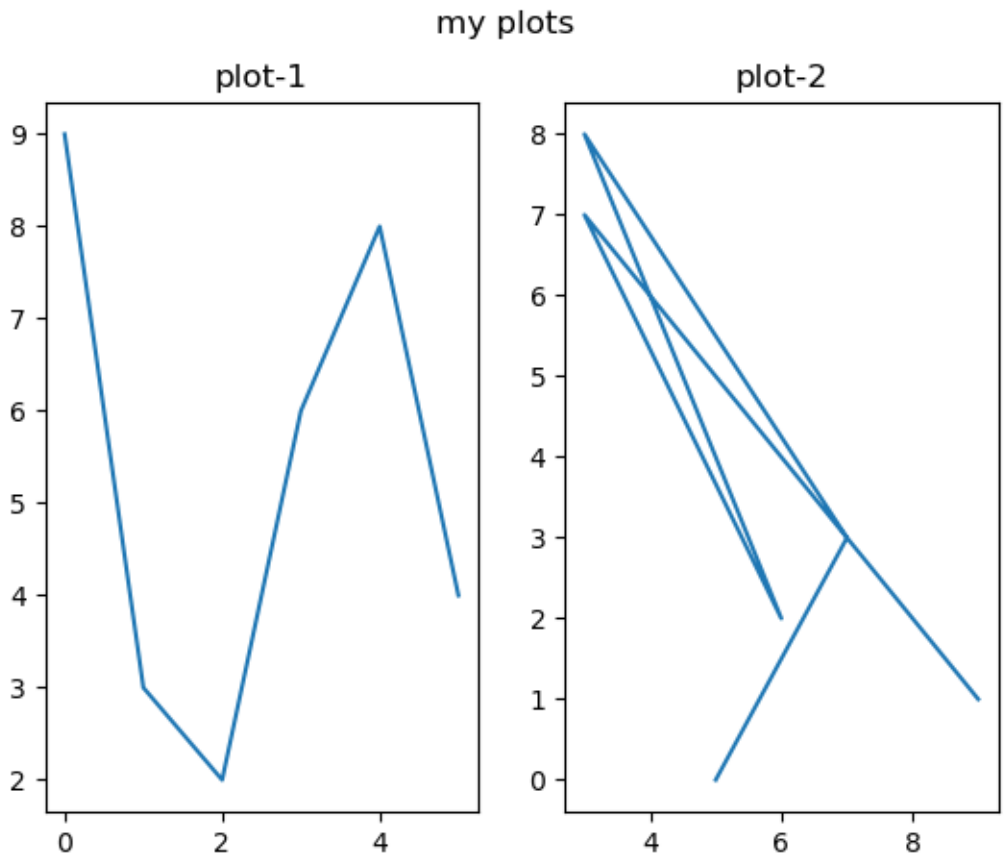


12 Subplot

[148]:
```
'''subplot'''
# it means by using this subplot() function , we can draw multiple plots in one␣
 ↪fiure.

# plot 1:
x=np.array([0,1,2,3,4,5])
y=np.array([9,3,2,6,8,4])
plt.subplot(1,2,1)
```

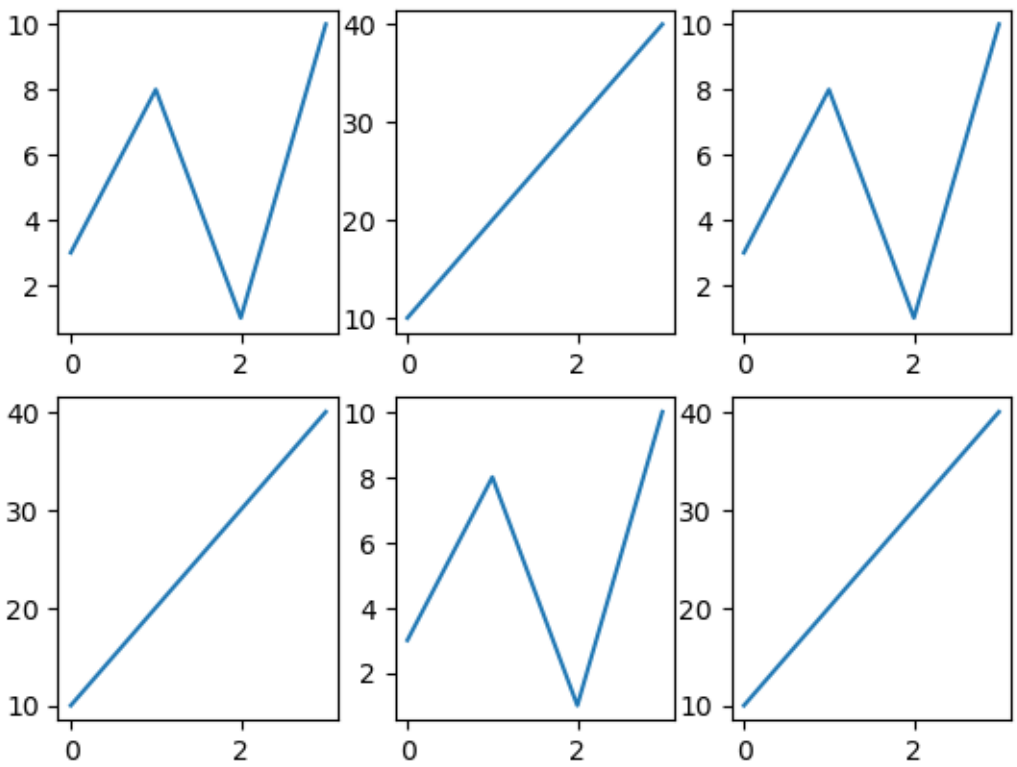
```
plt.plot(x,y)
plt.title("plot-1")

# plot 2
x=np.array([9,3,6,3,7,5])
y=np.array([1,7,2,8,3,0])
plt.subplot(1,2,2)
plt.plot(x,y)
plt.title("plot-2")
plt.suptitle("my plots")
plt.show
```

[148]: <function matplotlib.pyplot.show(close=None, block=None)>



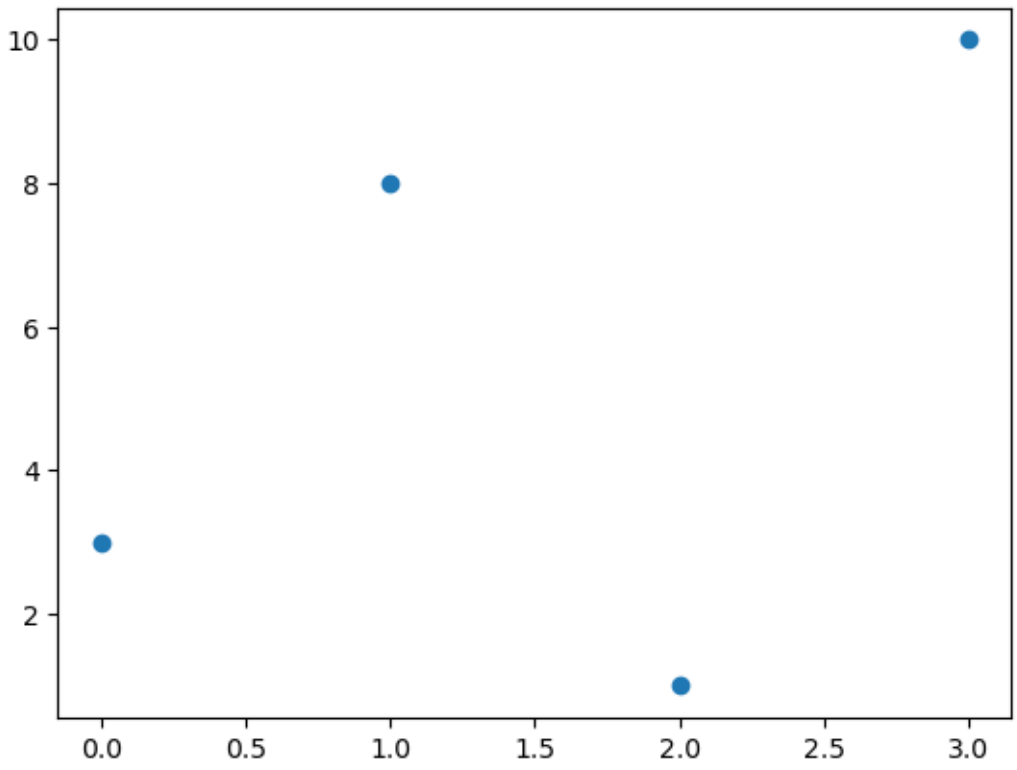
[144]:
```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

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.show()
```



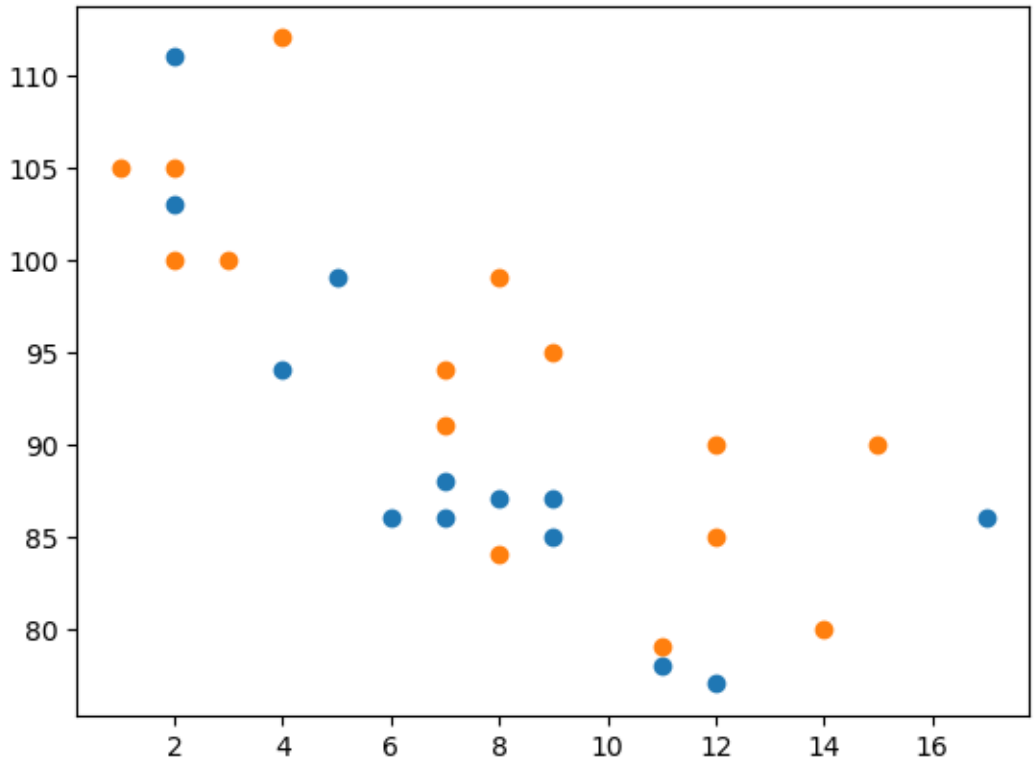
13 Scatter plot

[150]:
```
''' Scatter plots'''
# we can use scatter()
# the main purpose is relation between two variable, like two axis.

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

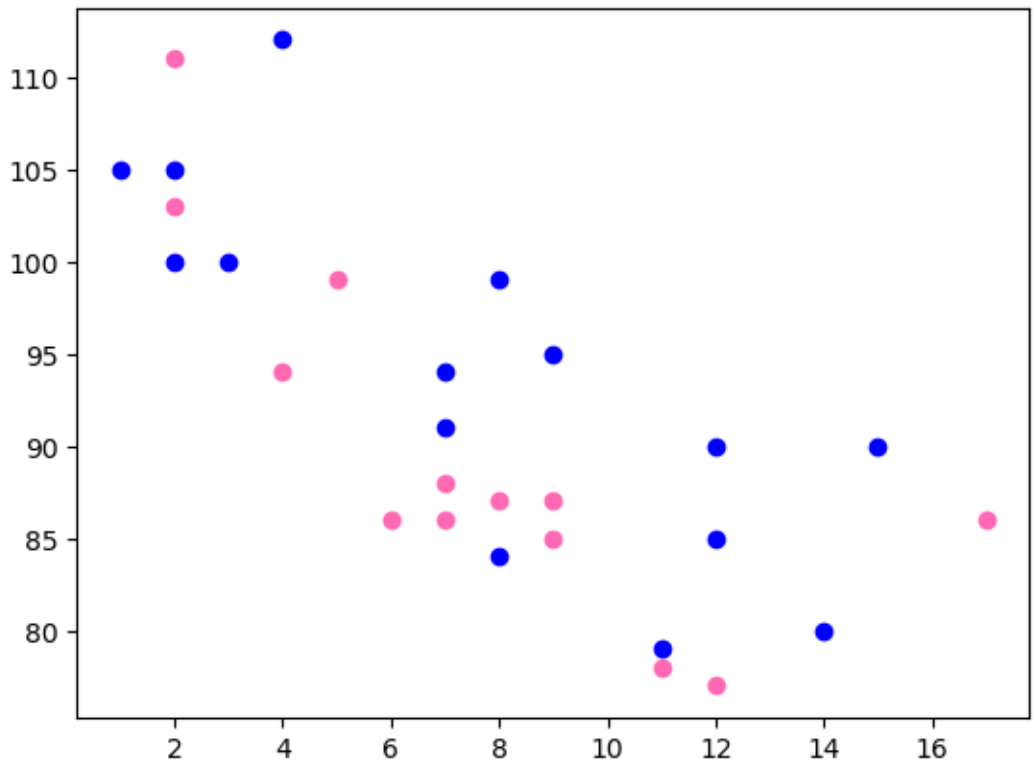


13.0.1 compare the Two plots

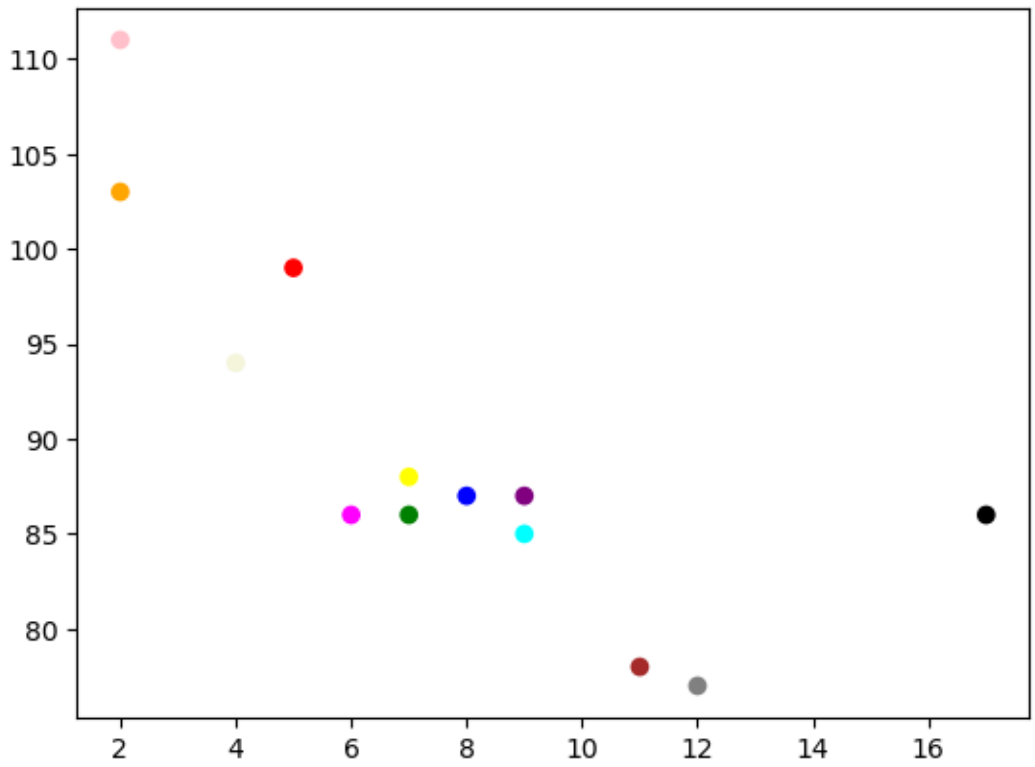
[82]:
```
''' compare plots'''
# as usall you will take two plots then default color is blue, orange.

#day one, the age and speed of 13 cars:
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)

#day two, the age and speed of 15 cars:
x1 = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y2 = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x1, y2)

plt.show()
```



[156]:
```
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 = 'hotpink')

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 = 'blue')

plt.show()
```

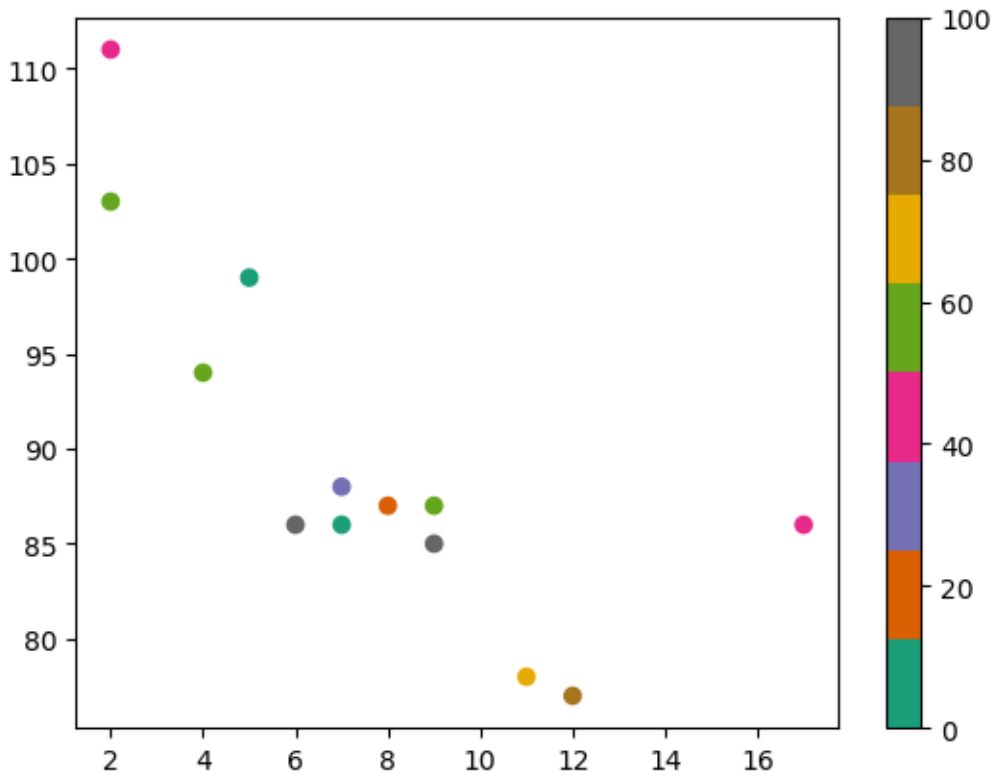


[85]:
```
''' we can change the edge h dot color'''
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors_data = np.
 ↪array(["red","green","blue","yellow","pink","black","orange","purple","beige","brown","gray

plt.scatter(x,y, c=colors_data)

plt.show()
```

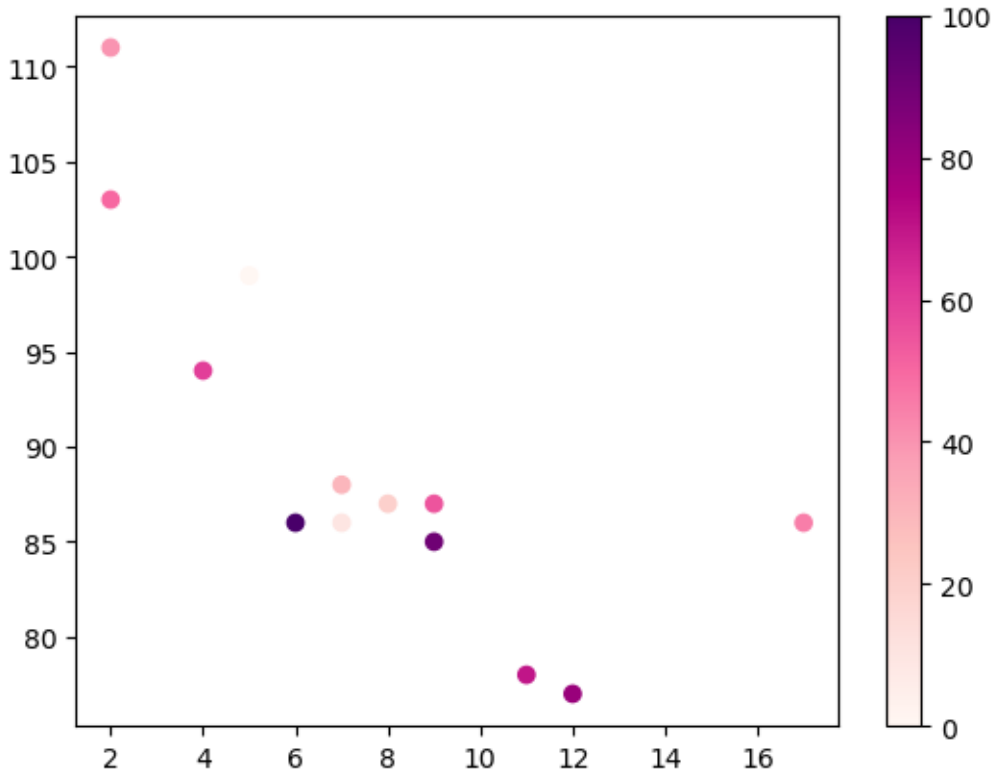


14 color map

[171]:
```
''' color map'''
# number of available colormaps.
# a colormap is like a list of colors, where each color has a value that ranges␣
 ↪from 0 to 100.
# we can use "cmap" keyword.

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, c=colors, cmap='Dark2')
plt.colorbar()
plt.show()
```

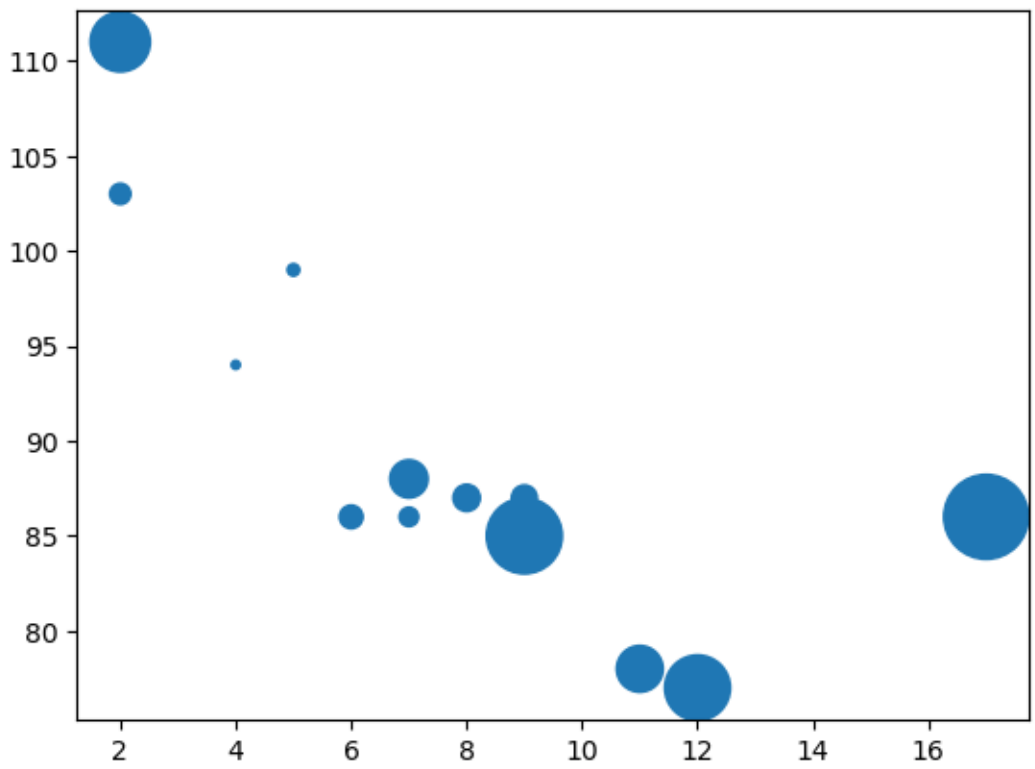


[175]:
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

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



14.0.1 Size of the bubble

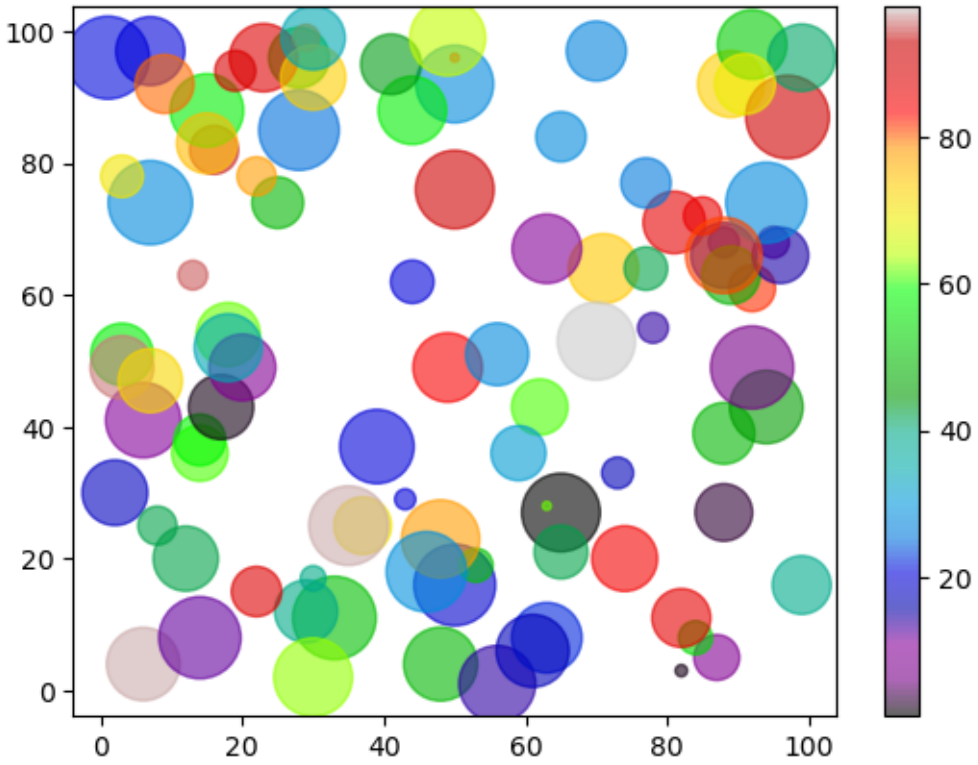
[179]:
```
''' size'''
# we can change the size of the dot argument is "s".

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()
```



15 Alpha

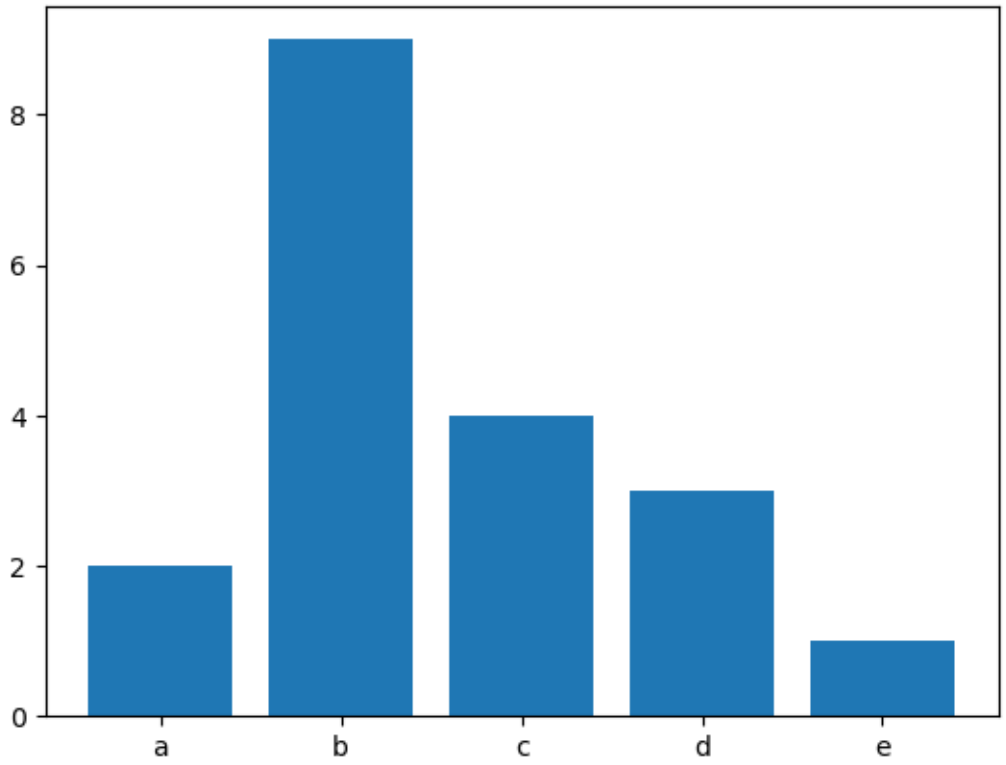
[ ]:
```
'''Alpha'''
# you can adjist th e transparancy of the dots with the alpha arguments.

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)
plt.show()
```

[191]:
```
''' combine color size and Alpha '''
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=0.6,cmap='nipy_spectral')
plt.colorbar()
plt.show()
```

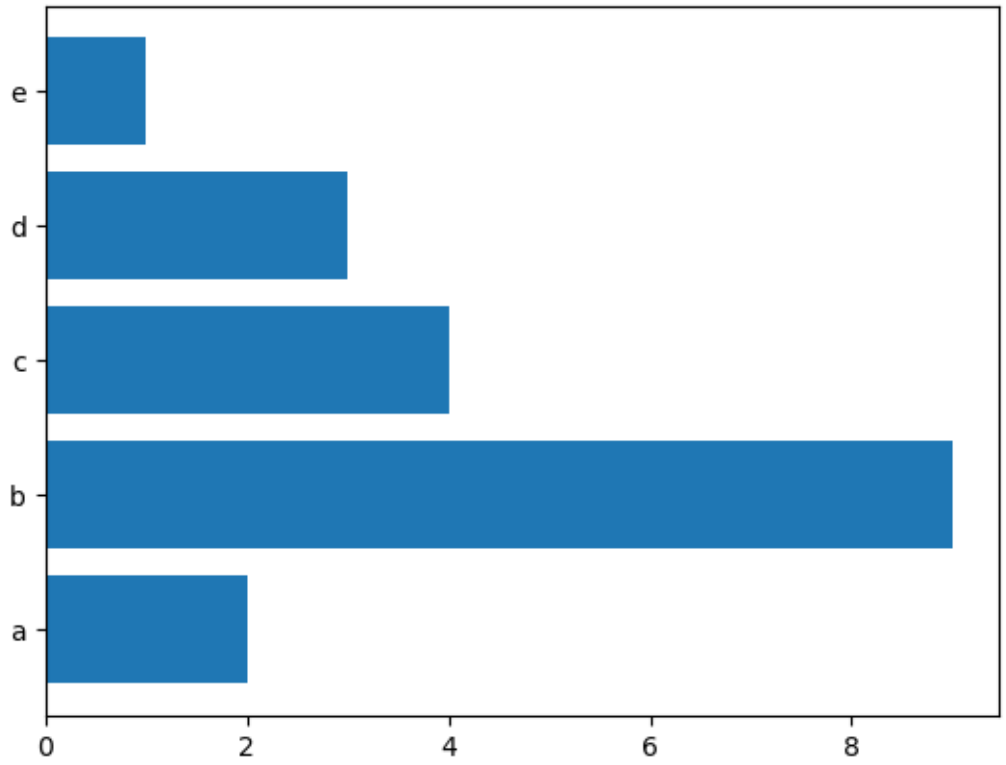


16 Bar plots

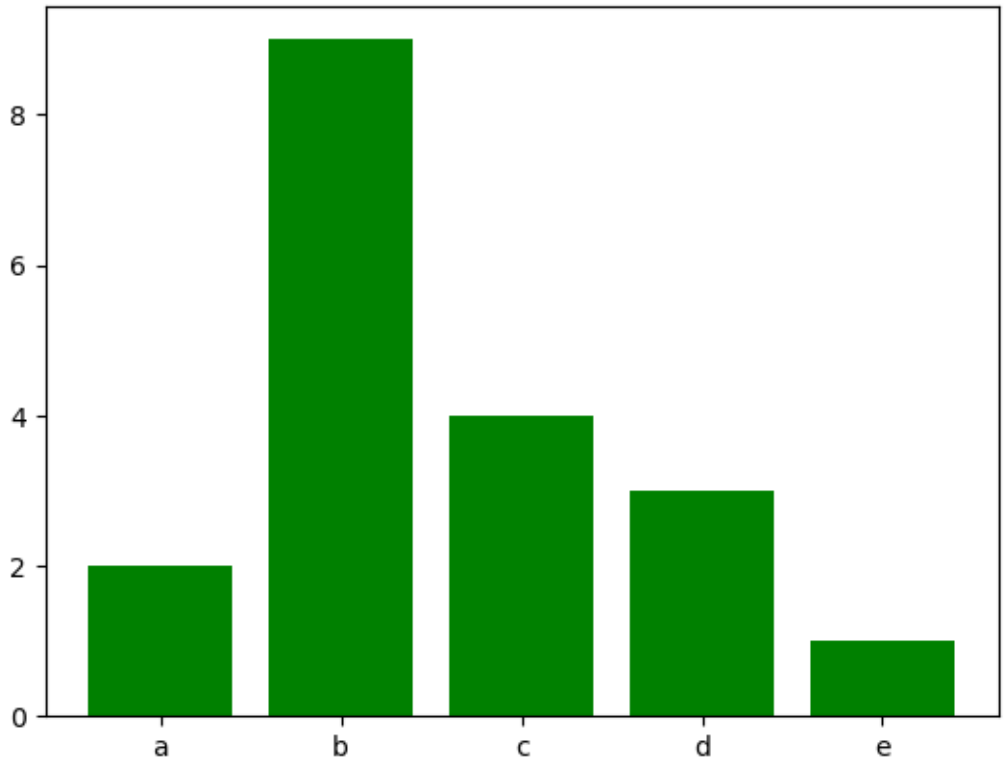
[195]:
```
''' Bar plots '''
# we can use bar()

x=np.array(['a','b','c','d','e'])
y=np.array([2,9,4,3,1])
plt.bar(x,y)
plt.show()
```



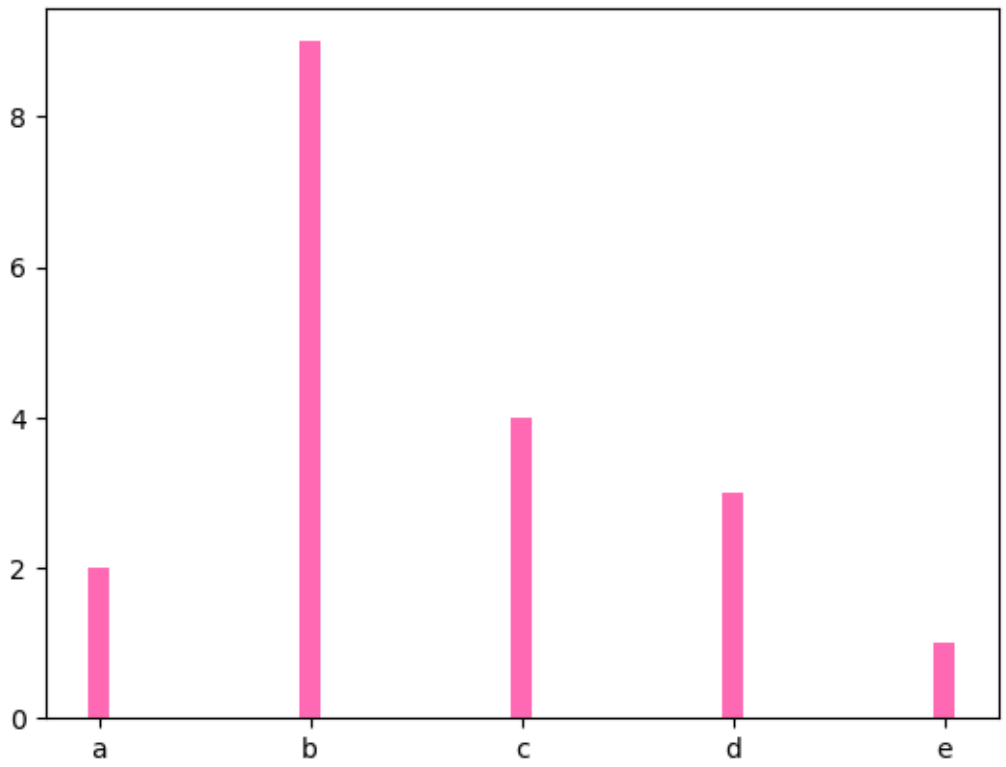
[202]:
```
''' horizontal bar'''
x=np.array(['a','b','c','d','e'])
y=np.array([2,9,4,3,1])
plt.barh(x,y)
plt.show()
```



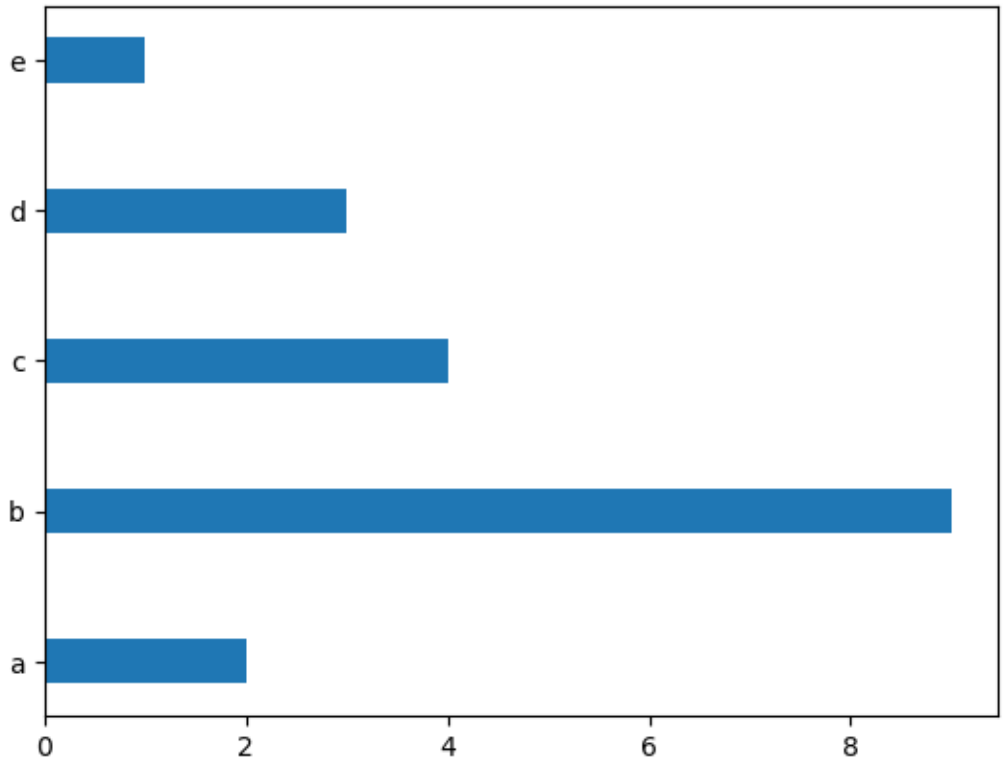
[208]:
```
'''color '''
x=np.array(['a','b','c','d','e'])
y=np.array([2,9,4,3,1])
plt.bar(x,y, color='green')
plt.show()
```



[210]:
```
'''width'''

x=np.array(['a','b','c','d','e'])
y=np.array([2,9,4,3,1])
plt.bar(x,y, color='hotpink', width=0.1)
plt.show()
```



[222]:
```
''' height'''
# in horizontal bars use height instead of width.
x=np.array(['a','b','c','d','e'])
y=np.array([2,9,4,3,1])
plt.barh(x,y, height=0.3)
plt.show()
```



17 Histogram plots

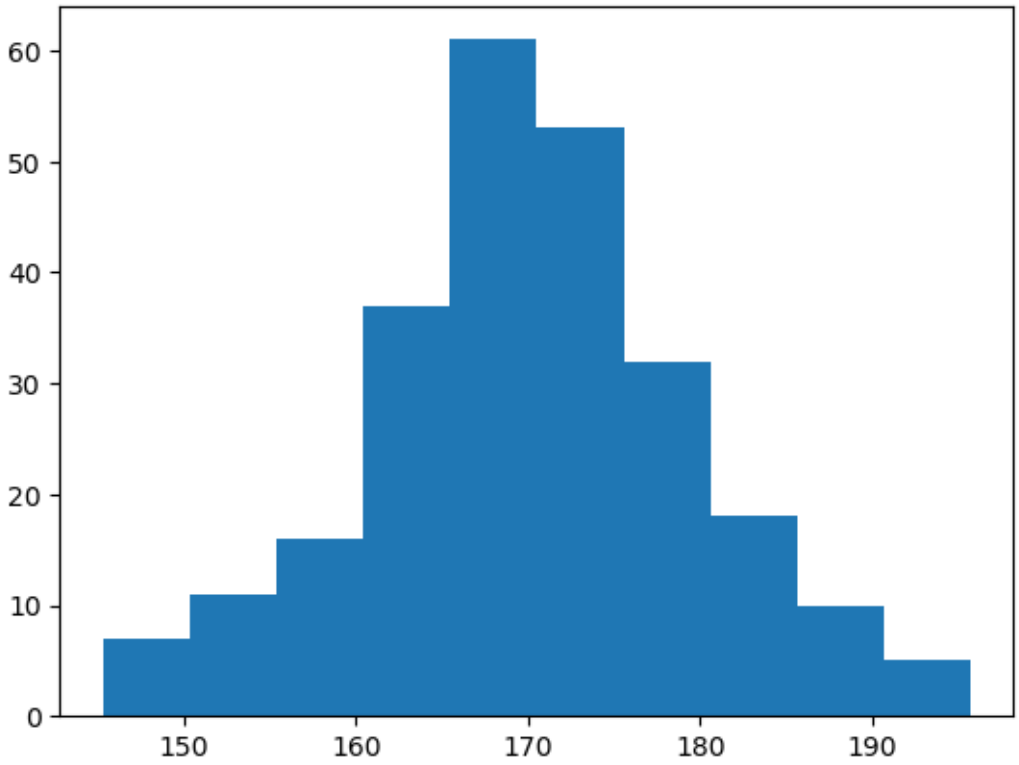
[ ]:
```
''' Histogram'''
# historam shows frequenncy distributions.
# it is showing the no.of observations within each given interval.
```

[268]:
```
import numpy as np
x=np.random.normal(170,10,250)
print(x)
```

```
[173.35454769 182.2195513  154.81002182 161.15208905 163.93920915
 155.22535904 172.5417989  167.98997797 177.04080358 169.4380675
 182.13499882 175.09153608 168.69474209 167.01951764 203.65006828
 151.56209537 177.60930646 163.1818719  166.50218046 165.7897992
 173.76466544 151.05839031 183.9650502  189.77341747 172.84644566
 155.78679238 166.76648635 178.36453629 187.44904432 173.7656665
 176.44469179 178.50810231 178.1206178  184.5127868  167.65427857
 178.02491337 175.0597713  159.61187879 181.83442311 162.3973634
 163.63156609 180.11065701 165.30825784 179.48670748 168.19296324
 180.90112629 167.39877675 185.1616318  156.01852438 170.11239474
 177.72568359 174.08992575 170.43861047 173.73026654 153.54989648
```

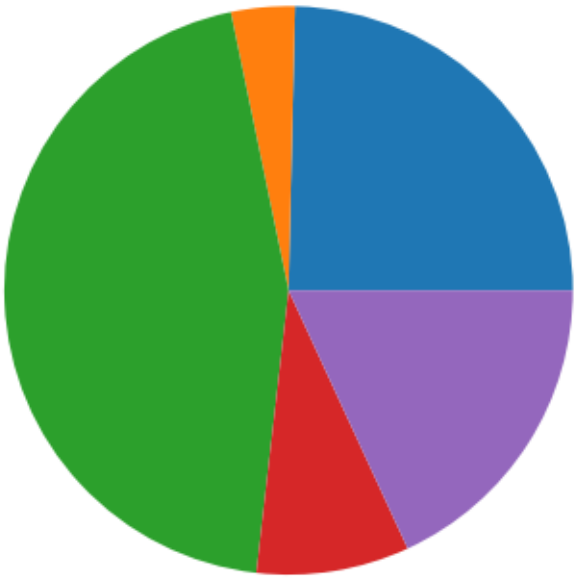
```
164.33355109 171.06448631 161.36426415 185.82862431 182.4412245
164.56836179 182.48192084 172.77471048 163.1866919  167.81949833
177.18320634 163.84920894 162.0798979  178.99822237 185.20099148
166.24414526 163.16408055 164.08911587 165.97381218 167.00430302
169.76223747 160.68447681 175.13356838 174.24786442 169.10968157
179.0947522  161.59221606 171.9006594  162.37497519 166.70595023
174.74335884 166.99978891 177.69666892 139.54913547 156.46009634
164.85041457 164.49952674 178.81960524 166.00186003 175.17278541
171.14883778 166.66405727 144.53612535 178.40640589 167.61761513
188.26362743 166.73039486 163.2023266  170.07355415 159.46084063
189.59989666 182.33826631 185.08455722 179.54280295 142.27702654
196.49539208 187.83599566 176.29479324 171.89432361 167.2088429
164.0721766  173.6285799  158.37689045 179.92579795 177.02675561
179.37795337 179.26906826 166.81703037 155.74366152 178.65791497
172.45437894 184.31433382 167.25982446 189.69779712 178.68290215
166.452709   176.21470672 167.11839687 165.2068503  166.38288817
154.96310859 167.32599972 173.19446259 171.73697316 183.58171659
164.32428581 163.80516684 173.96670378 161.89414688 182.40912986
174.40327859 157.39090841 160.96090879 149.07754276 168.36597324
181.47538283 153.73067706 165.20861127 177.29330432 170.87006427
165.37450644 154.86073373 195.31979394 164.79347938 169.27152036
175.73317062 176.742684   161.47487013 174.53745216 168.50500124
168.33947884 163.70166475 189.06327733 172.77372342 164.19333205
164.66581477 186.37914574 166.97545985 165.20614911 165.26203371
165.60740564 155.72297121 160.74923378 156.49245701 165.76848296
163.03218563 186.42145754 184.33250005 180.51815788 179.28372746
168.96943273 173.22379351 159.25954699 169.40924522 176.99668665
185.75245053 182.12514453 174.05514693 167.70381578 160.6278828
174.72364883 150.68272345 181.11930552 171.98157609 179.28984669
173.04876967 182.80466055 190.37371948 165.60365676 164.14898729
164.52250172 162.85985538 183.14673136 149.78703327 183.23415573
168.3076972  179.78442677 178.2539005  166.65095373 167.46128348
179.48878076 169.98986535 157.60124565 174.90434155 157.89868376
174.27953823 162.36769351 173.65129192 164.22735864 165.46899957
181.33581972 188.50842182 182.93845386 184.93685863 173.32240185
176.52312319 181.46016379 171.66456282 171.77155715 184.62015157
178.10450984 148.82273213 184.11700832 181.85433894 174.83375434
185.38727016 172.38646074 173.92186191 155.19016485 170.19836796
169.81444773 188.82588523 180.00918414 172.18867508 182.77814579]
```

[108]:
```
x=np.random.normal(170,10,250)
plt.hist(x)
plt.show()
```

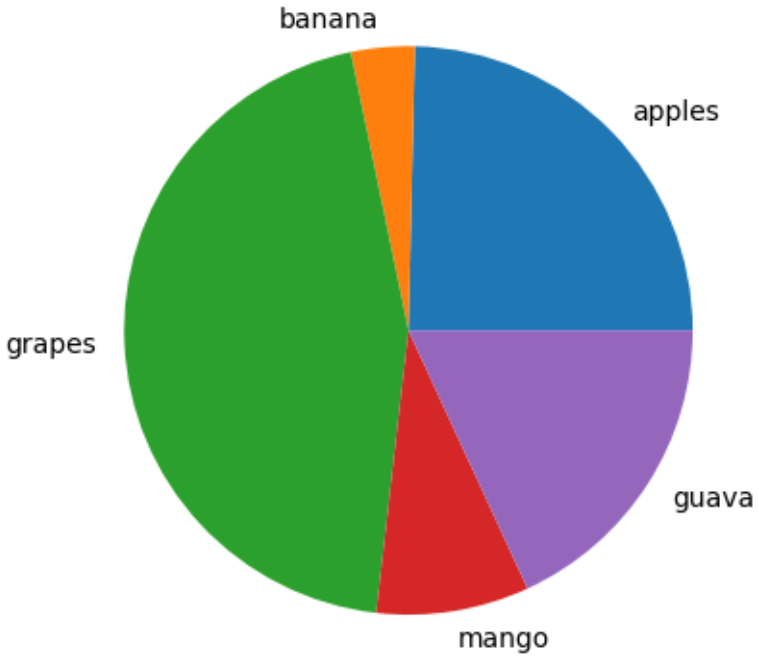


18 Pie charts

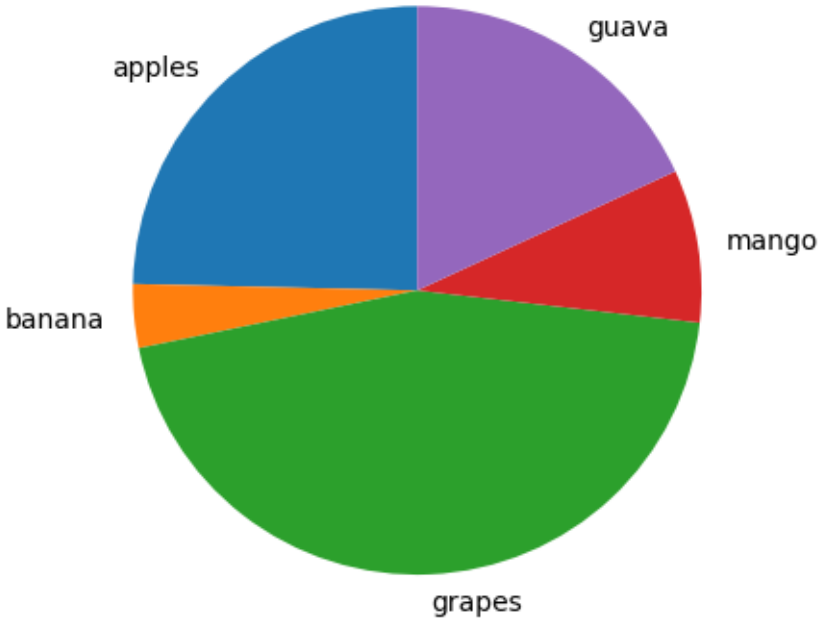
[279]:
```
'''pie charts '''
# wwe can use pie() function.

y=np.array([34,5,62,12,25])
plt.pie(y)
plt.show()
```



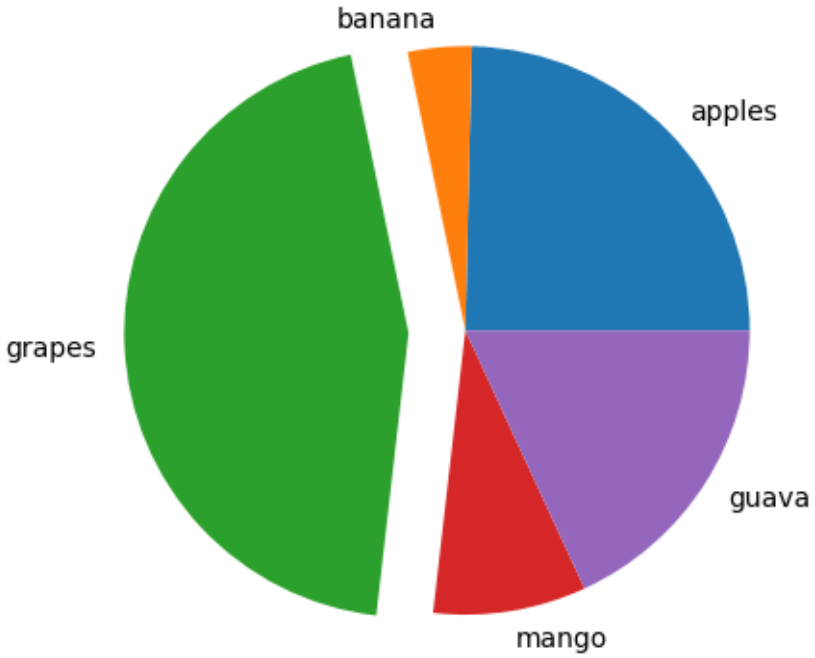
[286]:
```
'''labels'''
# we can use label argument
y=np.array([34,5,62,12,25])
mylabels=["apples","banana","grapes","mango","guava"]
plt.pie(y, labels=mylabels)
plt.show()
```



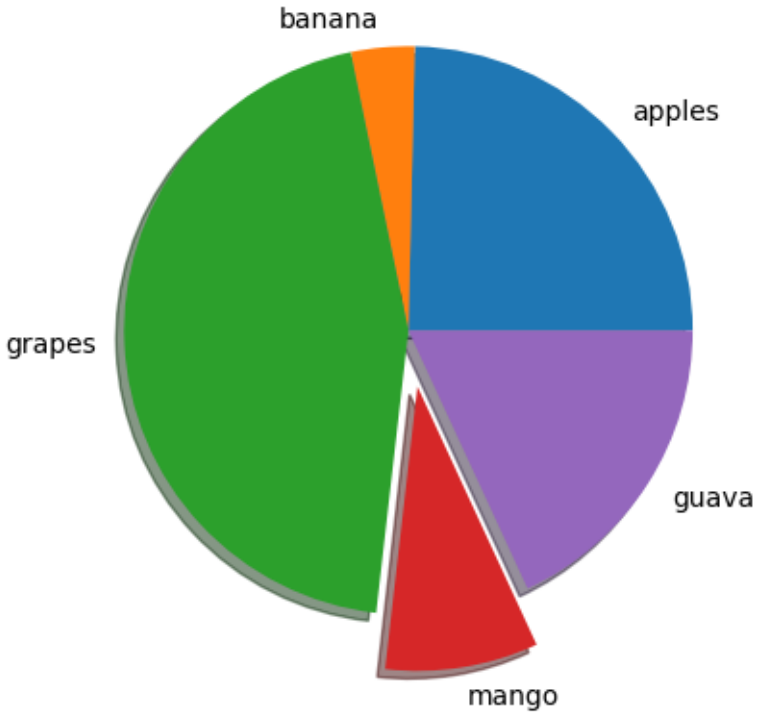
[288]:
```
''' Start Angle '''
# we can changes the default start angle is at the x-axis , but you change the start angle by specifying "startangle" parameter.
# the startangle parametr is defined with an agle in degrees , default angle is 0.

y=np.array([34,5,62,12,25])
mylabels=["apples","banana","grapes","mango","guava"]
plt.pie(y, labels=mylabels, startangle=90)
plt.show()
```



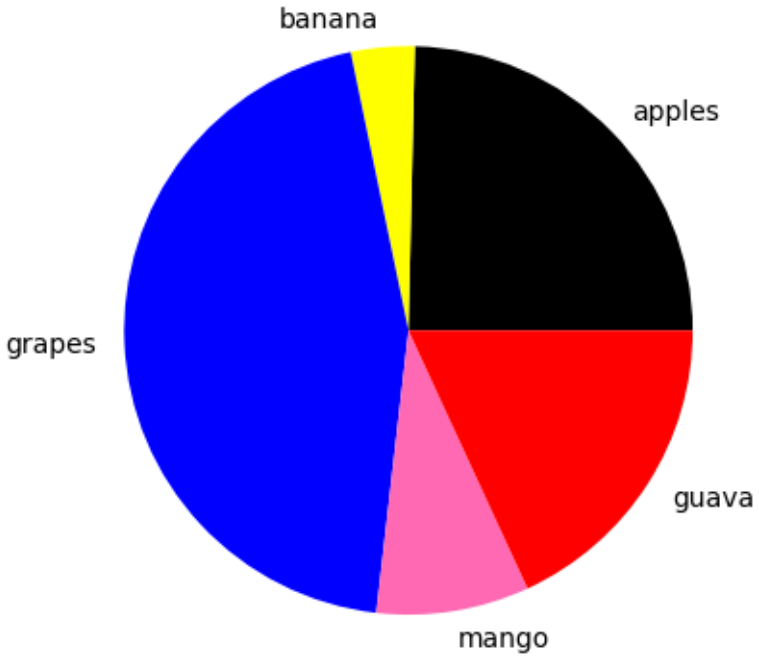
[294]:
```
''' Explode '''
# each edge splits away from pie chart.
# we can use explode.

y=np.array([34,5,62,12,25])
mylabels=["apples","banana","grapes","mango","guava"]
myexplode=[0,0,0.2,0,0]
plt.pie(y, labels=mylabels,explode=myexplode)
plt.show()
```



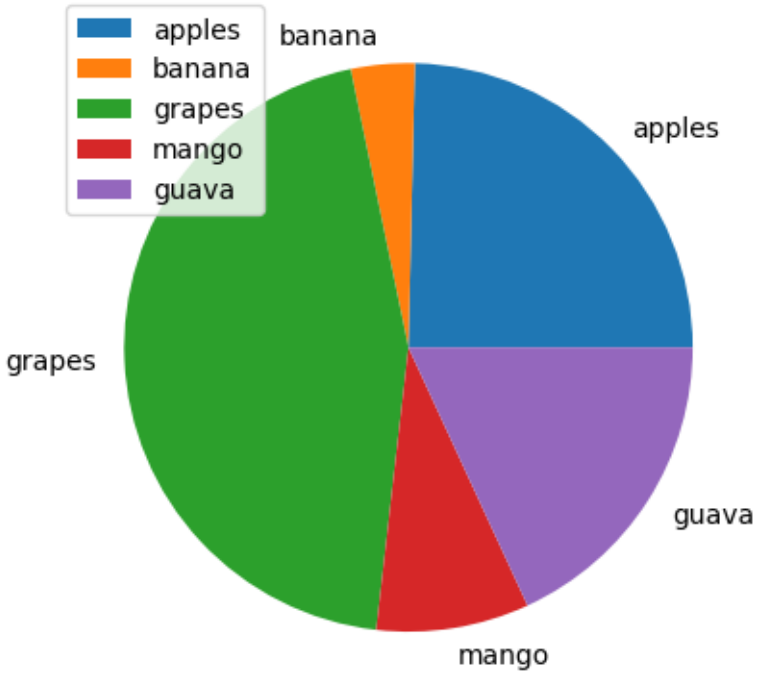
[299]:
```
''' Shadow '''
# add a shadow to the pie chart setting the shadows parameters to True.

y=np.array([34,5,62,12,25])
mylabels=["apples","banana","grapes","mango","guava"]
myexplode=[0,0,0,0.2,0]
plt.pie(y, labels=mylabels, explode=myexplode, shadow=True )
plt.show()
```



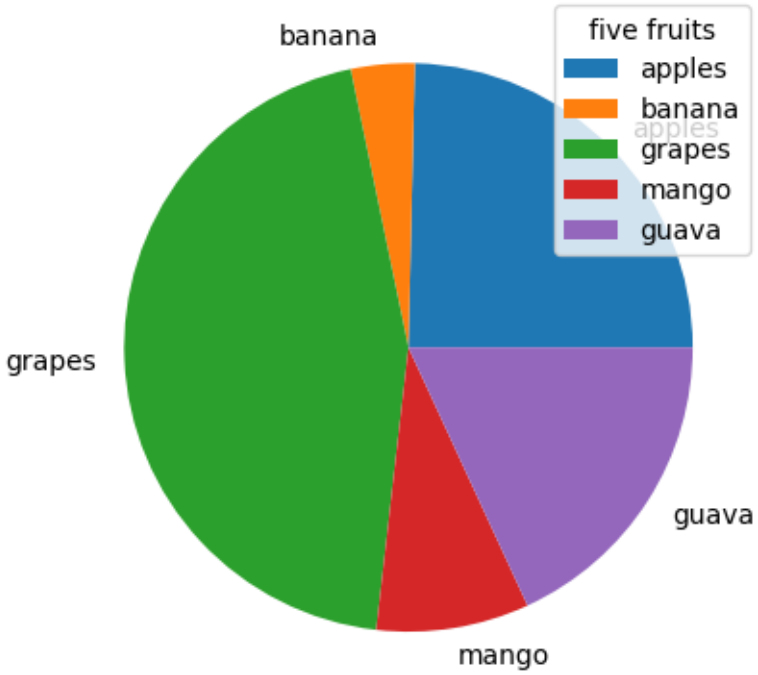
[315]:
```
''' colors'''
# we can set color of each wedge with th ecolor parameter.
# we can use color parameter.

y=np.array([34,5,62,12,25])
mylabels=["apples","banana","grapes","mango","guava"]
mycolor=["black","yellow","blue","hotpink","red"]
plt.pie(y, labels=mylabels, colors=mycolor)
plt.show()
```



[317]:
```
'''Legends'''
# add a list of explanation for each wedge
# use the legend() function.

y=np.array([34,5,62,12,25])
mylabels=["apples","banana","grapes","mango","guava"]
plt.pie(y, labels=mylabels)
plt.legend()
plt.show()
```



[321]:
```
y=np.array([34,5,62,12,25])
mylabels=["apples","banana","grapes","mango","guava"]
plt.pie(y, labels=mylabels)
plt.legend(title="five fruits")
plt.show()
```


banana
five fruits
apples
banana
grapes
mango
guava
grapes
guava
mango