Matplotlib

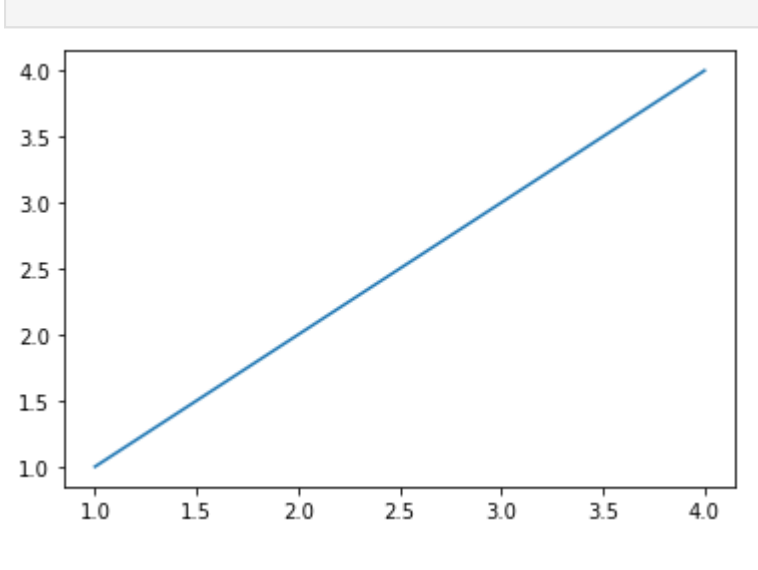
is a data visualization library in Python. The pyplot, a sublibrary of matplotlib, is a collection of functions that helps in creating a variety of charts.

In [ ]:
```
# A picture is worth a thousand words
```

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

In [31]:
```
# First import Matplotlib.pyplot library for plottin"Functions in python.ipynb"ns
import matplotlib.pyplot as plt
import numpy as np
```

In [68]:
```
plt.plot([1, 2, 3, 4],[1,2,3,4])#Plot the chart,
plt.show()  # display,,
```

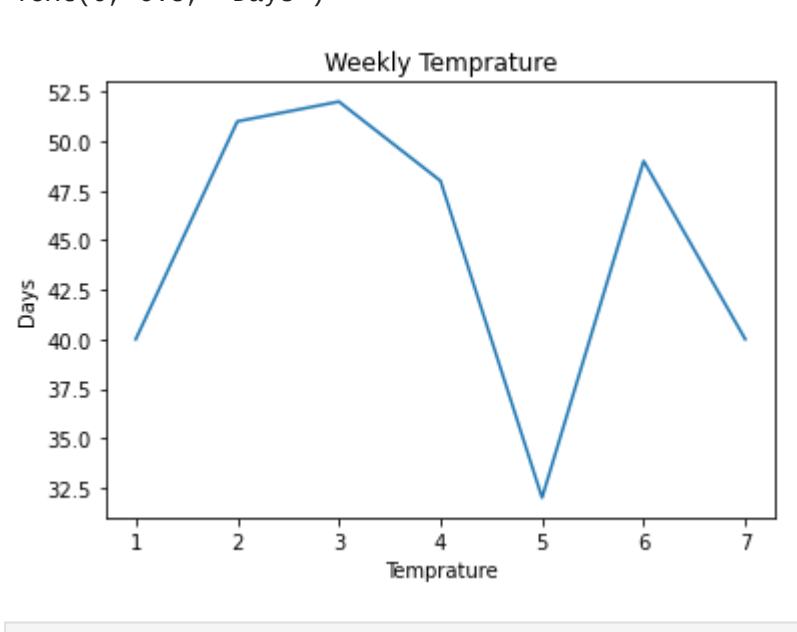


Line chart in Matplotlib

Line charts are used to represent the relation between two data X and Y on a different axis. Here we will see some of the examples of a line chart in Python :

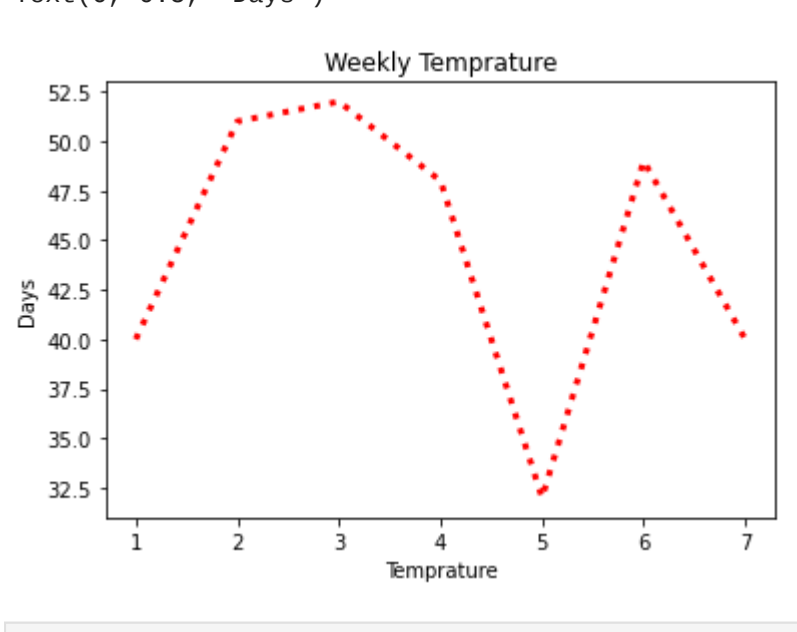
In [19]:
```
x=[1,2,3,4,5,6,7] #Week days
y=[40,51,52,48,32,49,40] # Temprature
plt.plot(x, y)
plt.title("Weekly Temprature")  # add title
plt.xlabel("Temprature")  # add X-axis label
plt.ylabel("Days")  # add Y-axis label
```

Out[19]:
```
Text(0, 0.5, 'Days')
```



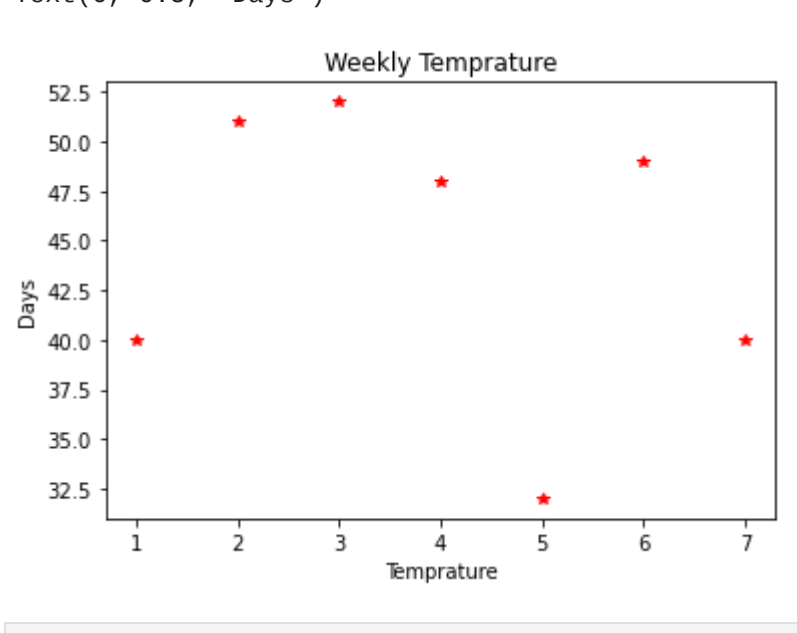
In [25]:
```
#change the color,line width and line style
plt.plot(x,y,color='red',linewidth=3,linestyle='dotted') #dashdot
plt.title("Weekly Temprature")  # add title
plt.xlabel("Temprature")  # add X-axis label
plt.ylabel("Days")  # add Y-axis label
```

Out[25]:
```
Text(0, 0.5, 'Days')
```



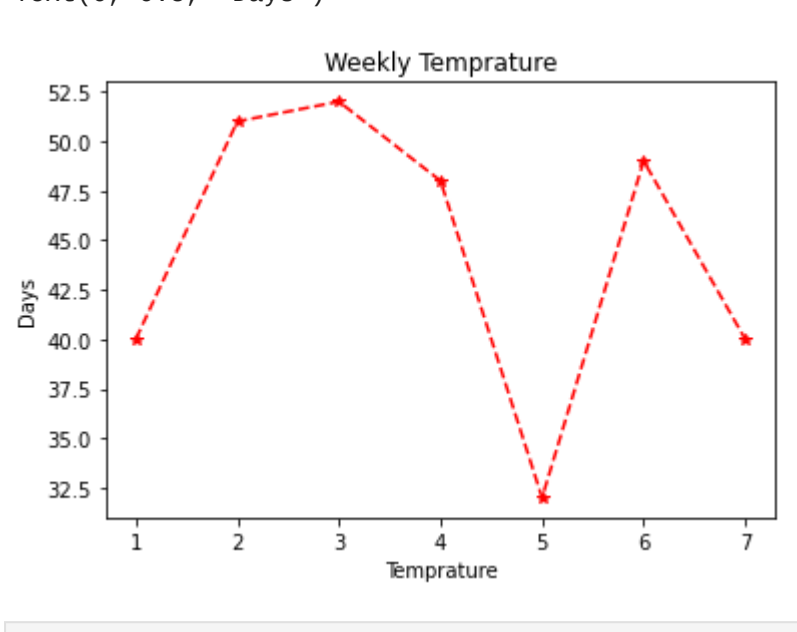
In [39]:
```
plt.plot(x,y,'r*',) #star Marker
plt.title("Weekly Temprature")  # add title
plt.xlabel("Temprature")  # add X-axis label
plt.ylabel("Days")
```

Out[39]:
```
Text(0, 0.5, 'Days')
```



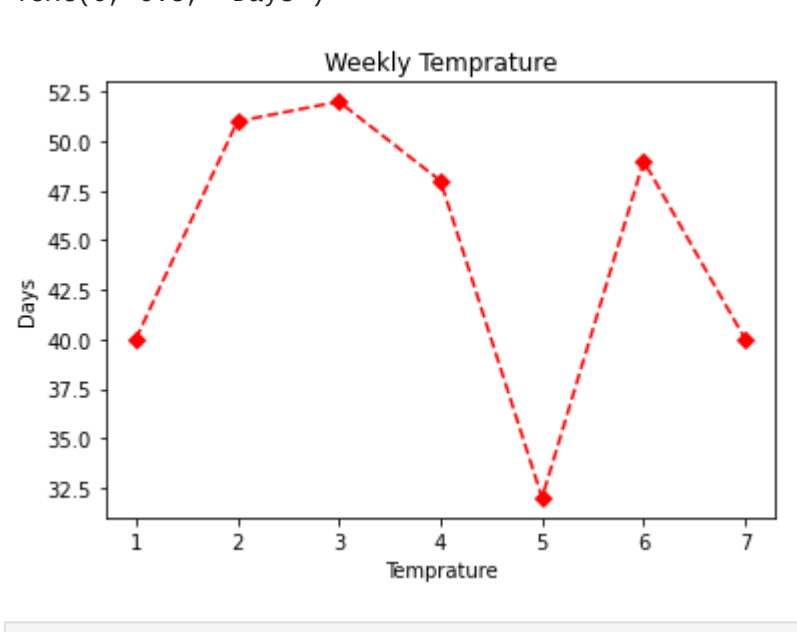
In [38]:
```
plt.plot(x,y,'r*--') #dashline '--'
plt.title("Weekly Temprature")  # add title
plt.xlabel("Temprature")  # add X-axis label
plt.ylabel("Days")
```

Out[38]:
```
Text(0, 0.5, 'Days')
```



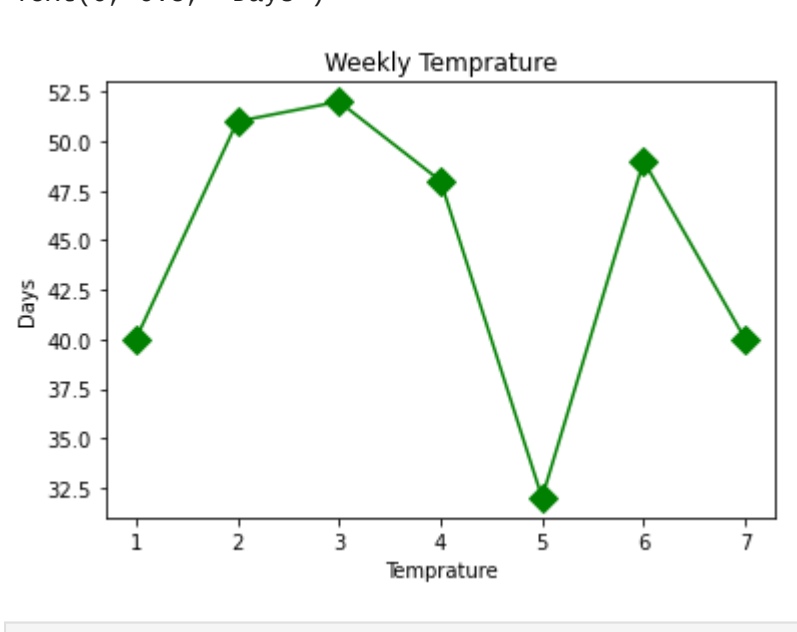
In [43]:
```
plt.plot(x,y,'rD--',) #Diamond Marker /Squre s
plt.title("Weekly Temprature")  # add title
plt.xlabel("Temprature")  # add X-axis label
plt.ylabel("Days")
```

Out[43]:
```
Text(0, 0.5, 'Days')
```



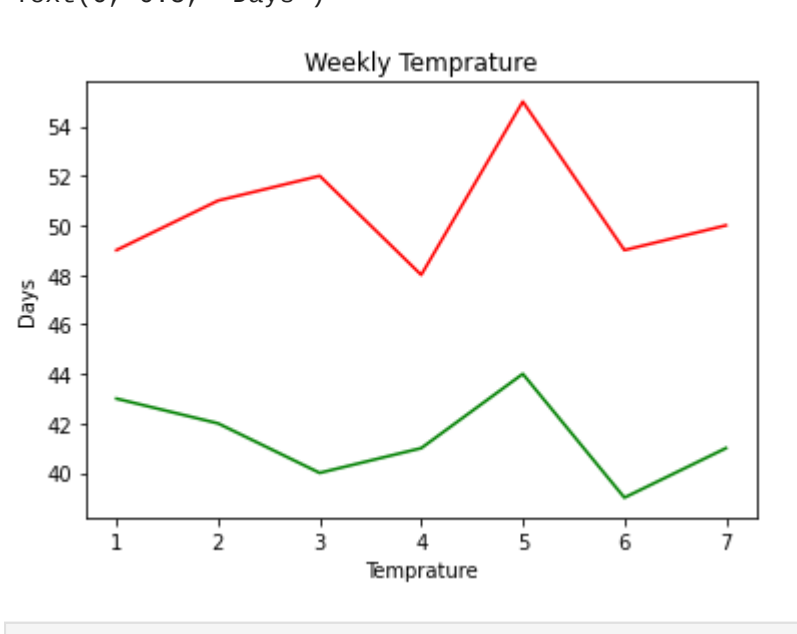
In [52]:
```
#individual argument for plot
plt.plot(x,y,color='green',marker='D',linestyle='-',markersize=10) #Diamond Marker /Squre s
plt.title("Weekly Temprature")  # add title
plt.xlabel("Temprature")  # add X-axis label
plt.ylabel("Days")
```

Out[52]:
```
Text(0, 0.5, 'Days')
```

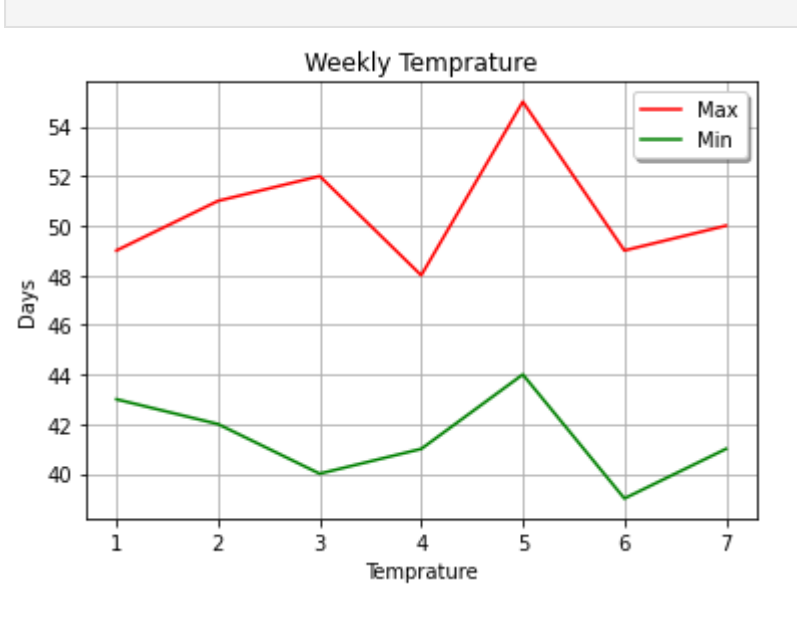


In [59]:
```
days=[1,2,3,4,5,6,7] #Week days
max_t=[49,51,52,48,55,49,50] # Temprature
min_t=[43,42,40,41,44,39,41]
plt.plot(days,max_t,color='red',)
plt.plot(days,min_t,color='green',)
plt.title("Weekly Temprature")  # add title
plt.xlabel("Temprature")  # add X-axis label
plt.ylabel("Days")
```

Out[59]:
```
Text(0, 0.5, 'Days')
```



In [67]:
```
#Create Legend and grid
days=[1,2,3,4,5,6,7] #Week days
max_t=[49,51,52,48,55,49,50] # Temprature
min_t=[43,42,40,41,44,39,41]
plt.plot(days,max_t,color='red',label='Max')
plt.plot(days,min_t,color='green',label='Min')
plt.title("Weekly Temprature")  # add title
plt.xlabel("Temprature")  # add X-axis label
plt.ylabel("Days")

plt.legend(loc='best',shadow=True) #loc=upper left
plt.grid()
```



Bar chart

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent

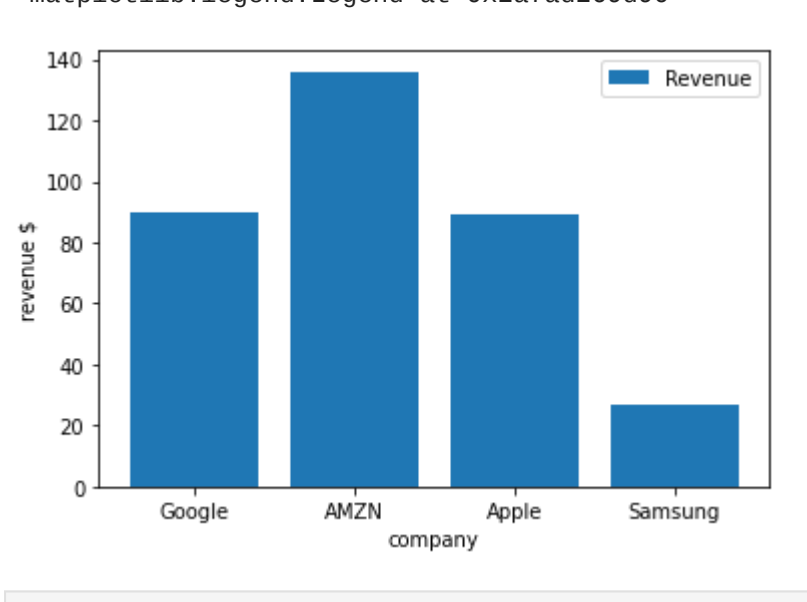
In [69]:
```
company=['Google','AMZN','Apple','Samsung']
revenue=[90,136,89,27]
```

In [74]:
```
import numpy as np
y=np.arange(len(company))
y
```

Out[74]:
```
array([0, 1, 2, 3])
```

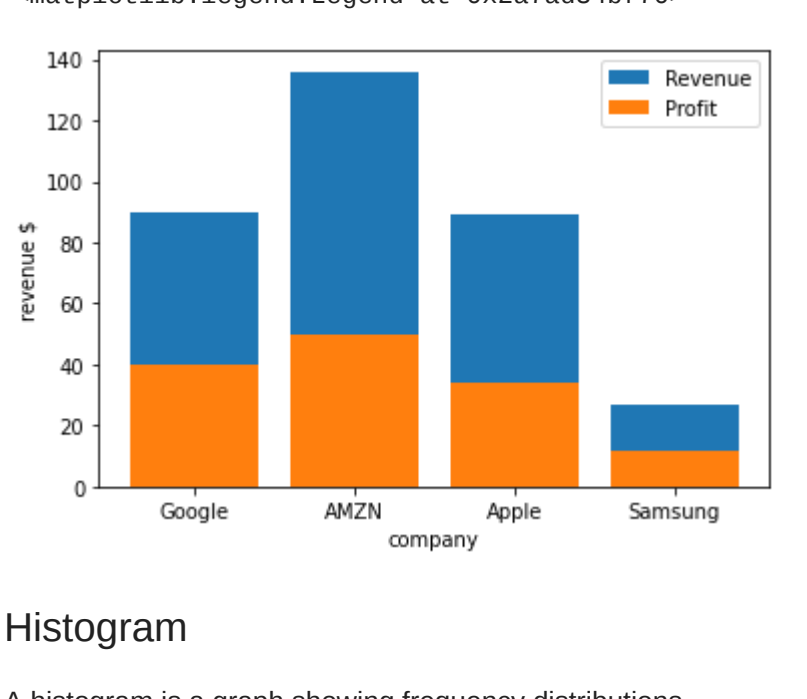
In [82]:
```
plt.xticks(y,company)
plt.bar(y,revenue,label='Revenue')
plt.ylabel('revenue $')
plt.xlabel('company')
plt.legend()
```

Out[82]:
```
<matplotlib.legend.Legend at 0x2a7ad269d90>
```



In [84]:
```
Profit=[40,50,34,12]
plt.xticks(y,company)
plt.bar(y,revenue,label='Revenue')
plt.bar(y,Profit,label='Profit')
plt.ylabel('revenue $')
plt.xlabel('company')
plt.legend()
```

Out[84]:
```
<matplotlib.legend.Legend at 0x2a7ad34bf70>
```

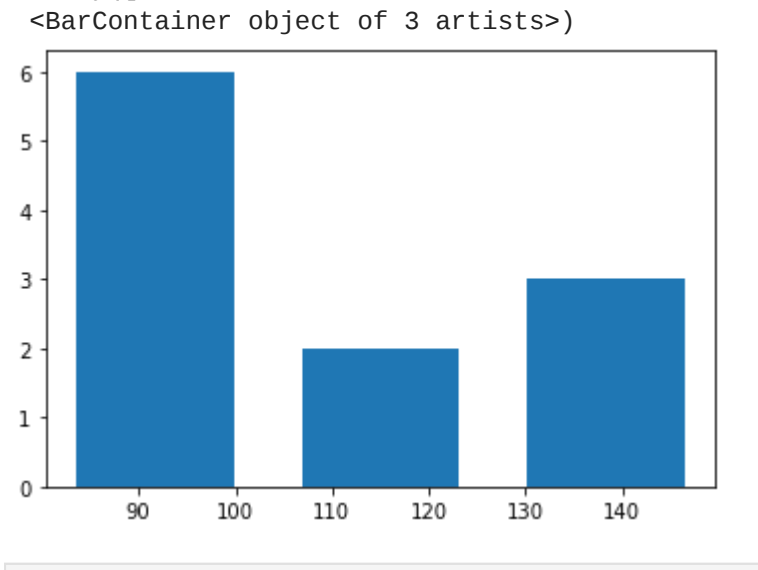


Histogram

A histogram is a graph showing frequency distributions.

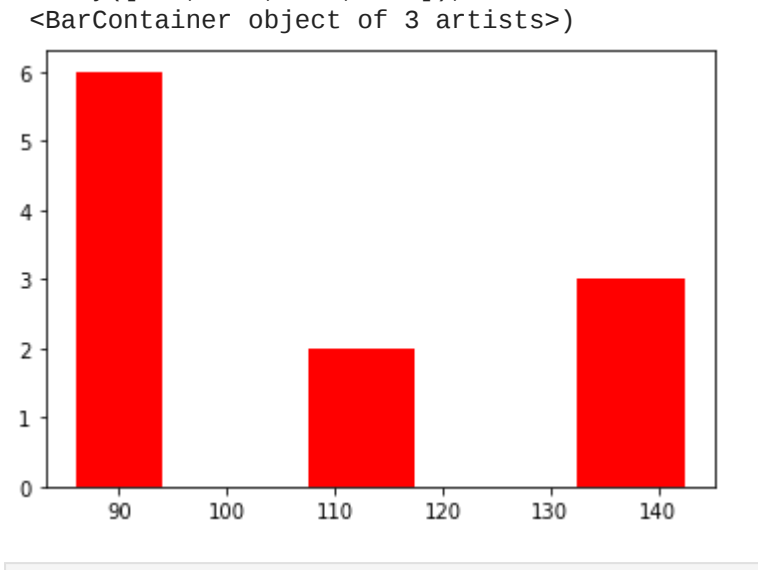
In [90]:
```
blood_suger=[113,85,150,149,88,83,93,115,80,82,129]

plt.hist(blood_suger,bins=3,rwidth=0.70)
```

Out[90]:
```
(array([6., 2., 3.]),
 array([ 80.        , 103.33333333, 126.66666667, 150.        ]),
 <BarContainer object of 3 artists>)
```



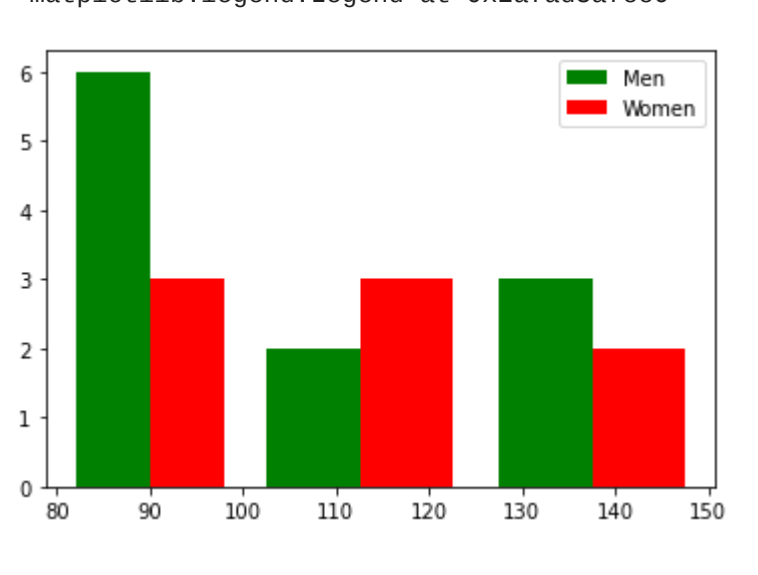
In [95]:
```
blood_suger=[113,85,150,149,88,83,93,115,80,82,129]

plt.hist(blood_suger,bins=[80,100,125,150],rwidth=0.40,color='r')
```

Out[95]:
```
(array([6., 2., 3.]),
 array([ 80, 100, 125, 150]),
 <BarContainer object of 3 artists>)
```



In [100…
```
blood_suger_man=[113,85,150,149,88,83,93,115,80,82,129]
blood_suger_women=[67,85,120,84,70,150,79,112,90,112,130]
plt.hist([blood_suger_man,blood_suger_women], bins=[80,100,125,150], color=['green','red'],label=['Men','Women'])
plt.legend()
```

Out[100…
```
<matplotlib.legend.Legend at 0x2a7ad8a7ee0>
```

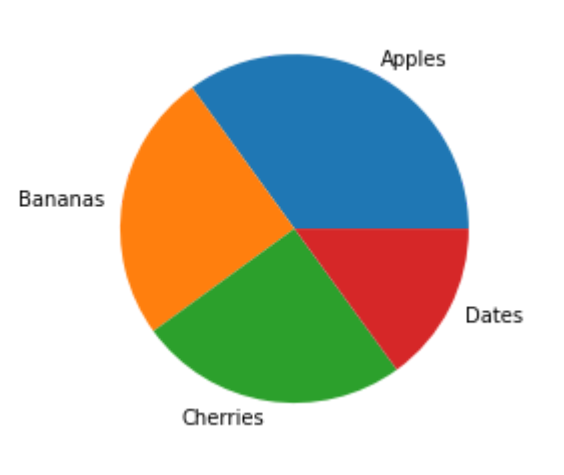


Pie Chart

A Pie Chart is a circular statistical plot that can display only one series of data.

In [3]:
```
y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

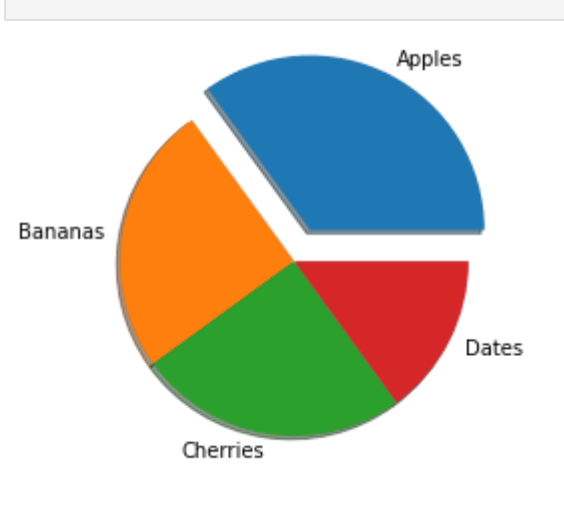
In [4]:
```
# Labels
import matplotlib.pyplot as plt
import numpy as np

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

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



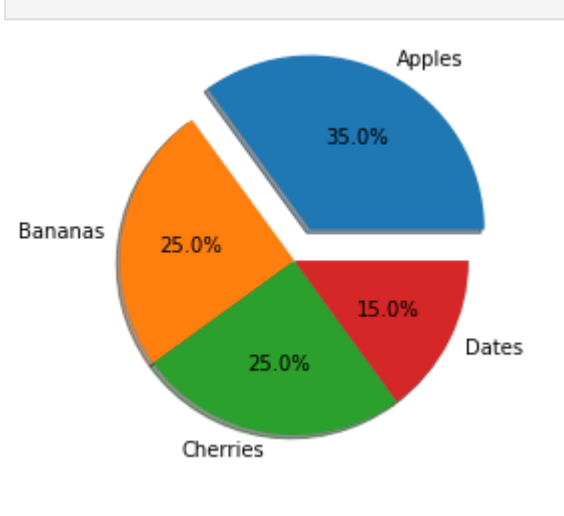
In [ ]:
```
# Start Angle
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels, startangle = 90)
plt.show()
```

In [5]:
```
# Explode
# Maybe you want one of the wedges to stand out? The explode parameter allows you to do that.
```

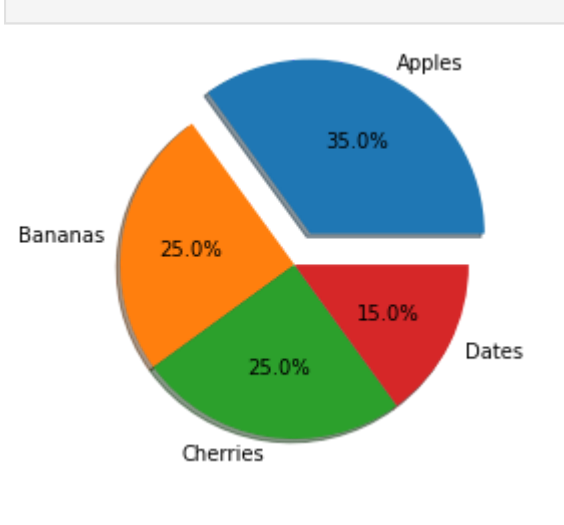
In [9]:
```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]

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



In [16]:
```
# show percentage
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]

plt.pie(y, labels = mylabels, explode = myexplode,shadow = True,autopct='%1.1f%%')
plt.show()
```



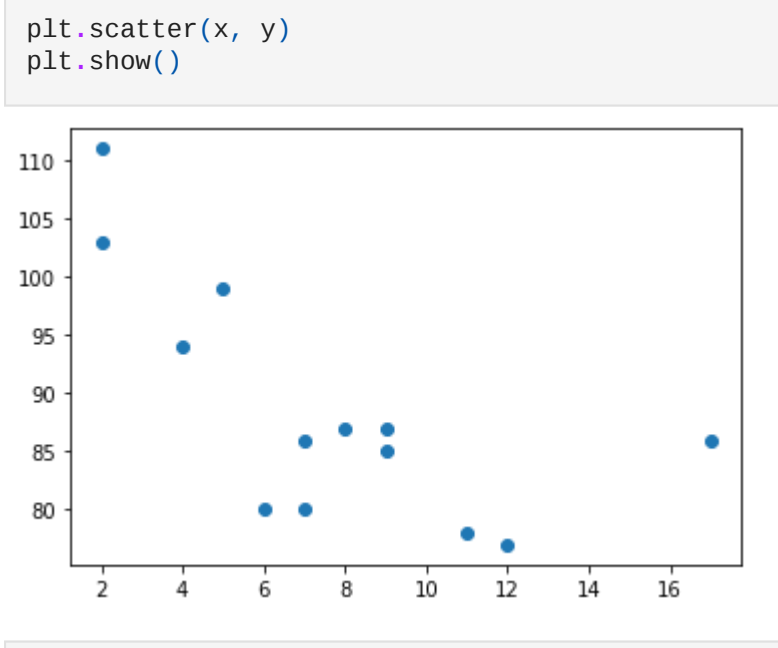
In [20]:
```
# Save Chart To a File Using savefig
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]

plt.pie(y, labels = mylabels, explode = myexplode,shadow = True,autopct='%1.1f%%')
plt.savefig('piechart.png',pad_inches=2)
```



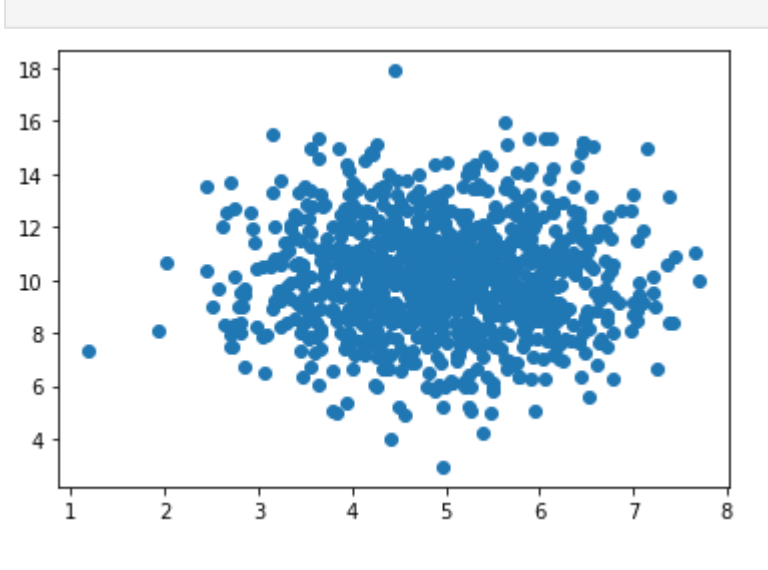
Scatter Plot

A scatter plot is a diagram where each value in the data set is represented by a dot

In [29]:
```
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,80,111,86,103,87,94,78,77,85,80]

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



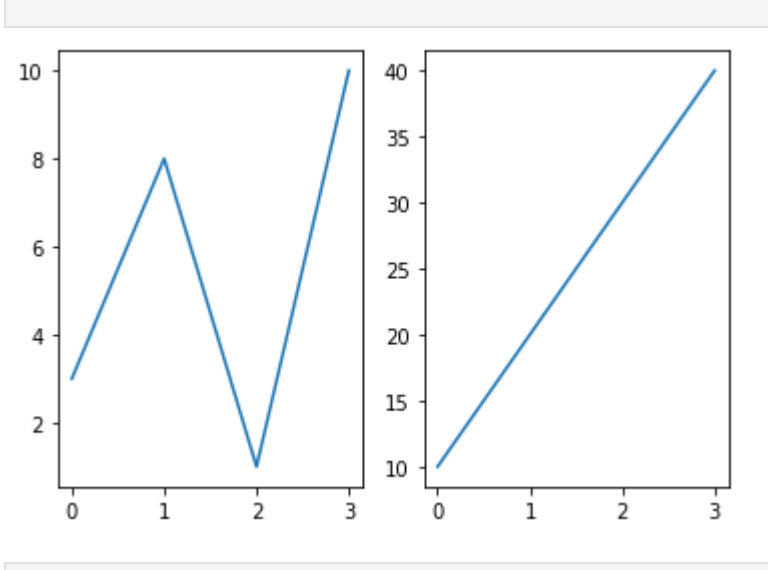
In [33]:
```
x = np.random.normal(5.0, 1.0, 1000)
y = np.random.normal(10.0, 2.0, 1000)

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



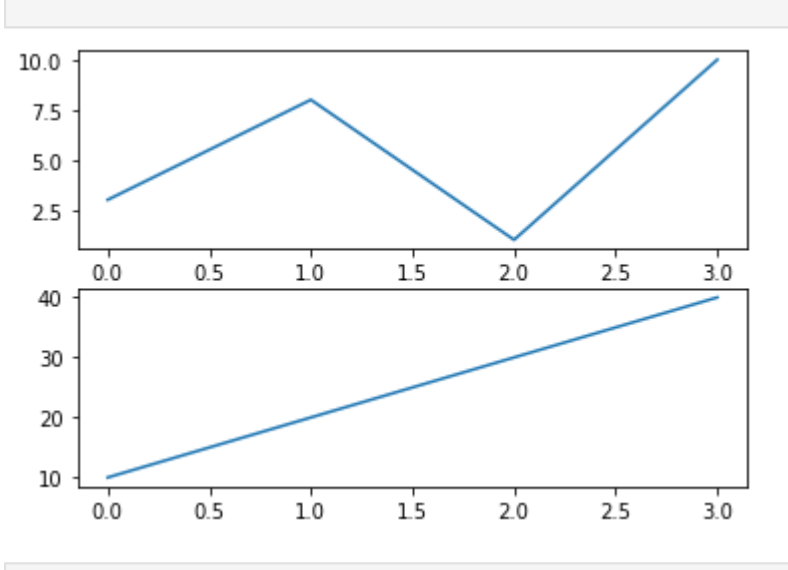
Display Multiple Plots

With the subplot() function you can draw multiple plots in one figure:

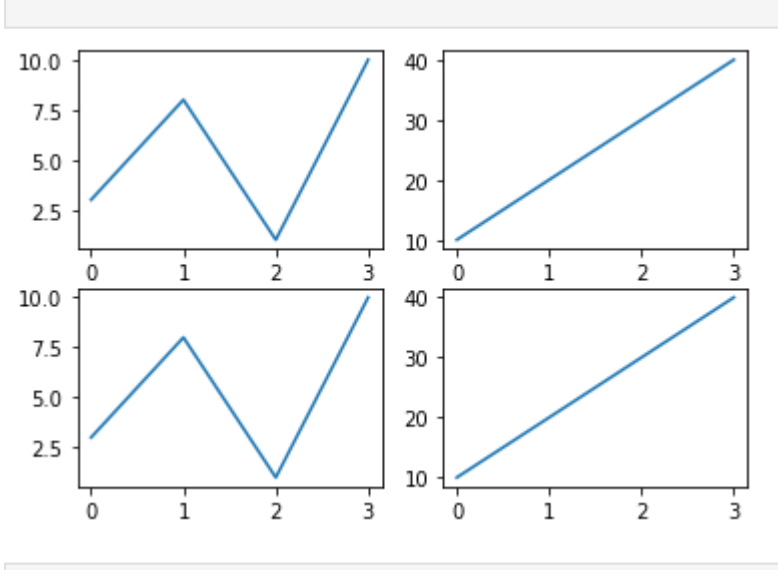
In [40]:
```
x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])

x2 = np.array([0, 1, 2, 3])
y2 = np.array([10, 20, 30, 40])
#plot 1:
plt.subplot(1, 2, 1) #Row,colunm,number of graph
plt.plot(x1,y1)

#plot 2:

plt.subplot(1, 2, 2)
plt.plot(x2,y2)

plt.show()
```



In [41]:
```
x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])

x2 = np.array([0, 1, 2, 3])
y2 = np.array([10, 20, 30, 40])
#plot 1:
plt.subplot(2, 1, 1) #Row,colunm,number of graph
plt.plot(x1,y1)

#plot 2:

plt.subplot(2, 1, 2)
plt.plot(x2,y2)

plt.show()
```



In [42]:
```
x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])

x2 = np.array([0, 1, 2, 3])
y2 = np.array([10, 20, 30, 40])
#plot 1:
plt.subplot(2, 2, 1) #Row,colunm,number of graph
plt.plot(x1,y1)

#plot 2:

plt.subplot(2, 2, 2)
plt.plot(x2,y2)
#plot 3:
plt.subplot(2, 2, 3) #Row,colunm,number of graph
plt.plot(x1,y1)

#plot 2:

plt.subplot(2, 2, 4)
plt.plot(x2,y2)

plt.show()
```



In [ ]: