Pandas Series

Series are very similar to nd arrays: the main difference is we can provide custom index labels then and then operations you perform on series automatically align the data based on labels

create series

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
my_series = pd.Series(data= [2,3,4,5], index=['a','b','c','d'])
my_series
```

Out[2]:

```
a    2
b    3
c    4
d    5
dtype: int64
```

In [3]:

```
my_series["a"]
```

Out[3]:

```
2
```

In [4]:

```
my_series[1]
```

Out[4]:

```
3
```

In [5]:

```
my_series[1:3]
```

Out[5]:

```
b    3
c    4
dtype: int64
```

In [6]:

```
my_series+my_series
```

Out[6]:

```
a     4
b     6
c     8
d    10
dtype: int64
```

In [7]:

```
my_series+my_series
```

Out[7]:

```
a     4
b     6
c     8
d    10
dtype: int64
```

In [8]:

```
my_dict={"x":2,"a":5,"b":4,"c":8}
my_series2=pd.Series(my_dict)
my_series2
```

Out[8]:
```
x    2
a    5
b    4
c    8
dtype: int64
```

In [9]:
```
my_series1=pd.Series(data= [2,3,4,5], index=['x','b','c','d'])
```

In [10]:
```
my_series1
```

Out[10]:
```
x    2
b    3
c    4
d    5
dtype: int64
```

In [11]:
```
my_series
my_series2
```

Out[11]:
```
x    2
a    5
b    4
c    8
dtype: int64
```

In [12]:
```
my_series
```

Out[12]:
```
a    2
b    3
c    4
d    5
dtype: int64
```

In [13]:
```
my_series+my_series2
```

Out[13]:
```
a     7.0
b     7.0
c    12.0
d     NaN
x     NaN
dtype: float64
```

In [14]:
```
np.mean(my_series)
```

Out[14]:
```
3.5
```

In [15]:
```
my_dict={"name":["x","y","z"],"age":np.array([10,15,20]),"weight":(75,123,135),"hei
        index=[1,2,3]),"gender":"M"}
```

In [16]:
```
my_dict
```

Out[16]:
```
{'name': ['x', 'y', 'z'],
 'age': array([10, 15, 20]),
 'weight': (75, 123, 135),
 'height': 1    4.5
 2    6.0
 3    5.5
 dtype: float64,
 'gender': 'M'}
```

In [17]:
```
df=pd.DataFrame(my_dict)#convert dictionary to DataFrame
```

In [18]:
```
df
```

Out[18]:

	name	age	weight	height	gender
1	x	10	75	4.5	M
2	y	15	123	6.0	M
3	z	20	135	5.5	M

In [19]:

```
type(df)
```

Out[19]:

```
pandas.core.frame.DataFrame
```

In [20]:

```
df["weight"]
```

Out[20]:

```
1     75
2    123
3    135
Name: weight, dtype: int64
```

In [21]:

```
df.weight
```

Out[21]:

```
1     75
2    123
3    135
Name: weight, dtype: int64
```

In [22]:

```
df.describe()
```

Out[22]:

	age	weight	height
count	3.0	3.000000	3.000000
mean	15.0	111.000000	5.333333
std	5.0	31.749016	0.763763
min	10.0	75.000000	4.500000
25%	12.5	99.000000	5.000000
50%	15.0	123.000000	5.500000
75%	17.5	129.000000	5.750000
max	20.0	135.000000	6.000000

why Data visualization?

Data visualization allows us to quickly interpret the data and adjust differnt variables to see their effect

What is Data Visualization?

Data visualization is the presentation of data in a pictorial or graphical format.

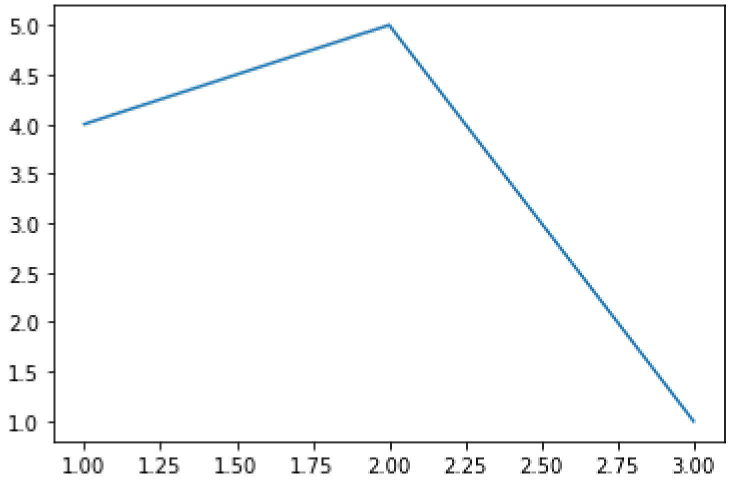
what is matplotlib?

types of plot

1.bar graph 2.histograms 3.scatter plot 4.pie plot 5.hexagonal bin plot 6.area plot

In [23]:

```
from matplotlib import pyplot as plt
plt.plot([1,2,3],[4,5,1])
plt.show()
```

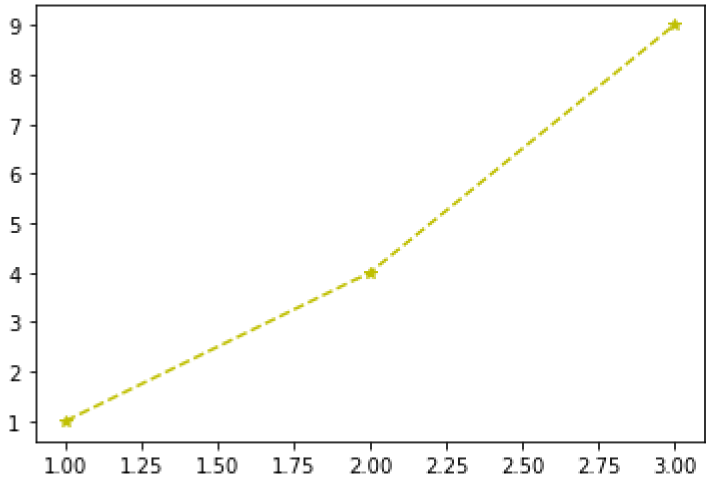


In [24]:

```
x=np.array([1,2,3])
y=x * x
plt.plot(x,y,'y--*')
```

Out[24]:

```
[<matplotlib.lines.Line2D at 0x1f6cdf5ae50>]
```

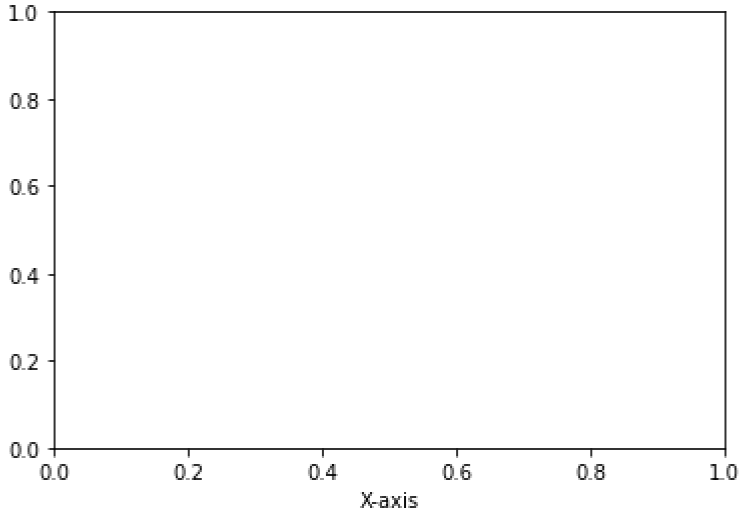


In [25]:

```
plt.xlabel("X-axis")
```

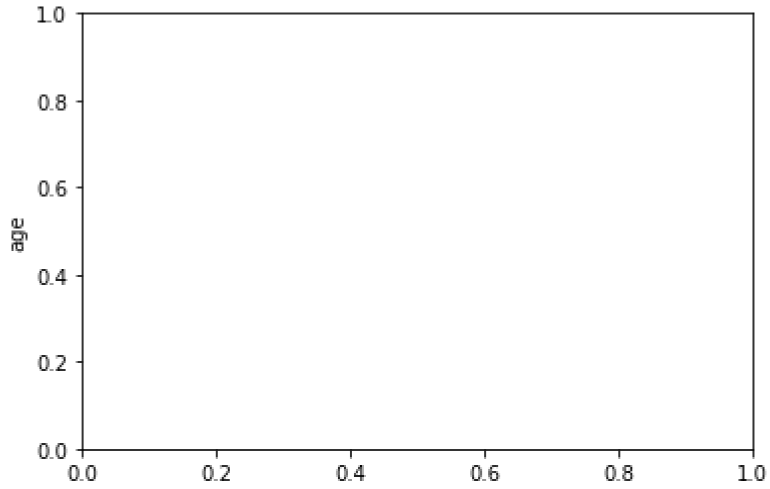
Out[25]:

```
Text(0.5, 0, 'X-axis')
```



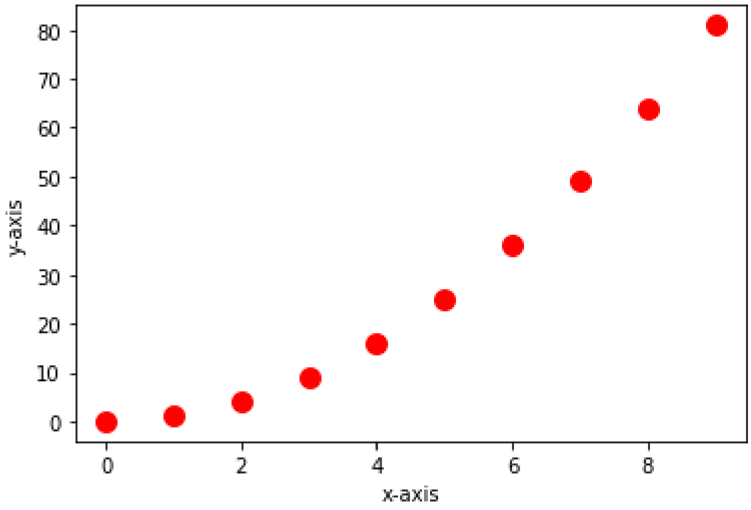
In [26]:
```
plt.ylabel("age")
```

Out[26]:
```
Text(0, 0.5, 'age')
```



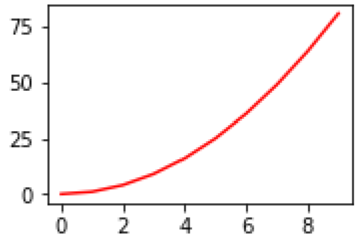
In [27]:
```
x=np.arange(0,10)
y=x*x
```

In [28]:
```
plt.scatter(x,y,c='r',linewidths=5)
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.savefig("img.jpg")
```



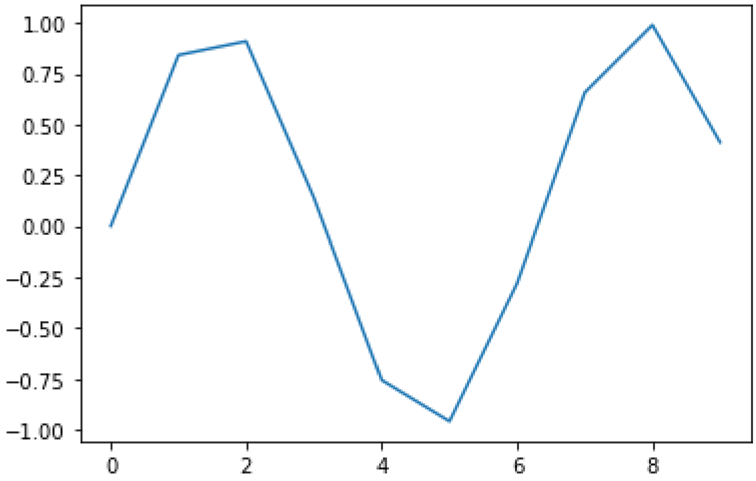
In [29]:
```
plt.subplot(2,2,1)
plt.plot(x,y,'r')
```

Out[29]:
```
[<matplotlib.lines.Line2D at 0x1f6ce099ca0>]
```



In [30]:
```
#sin curve
x=np.arange(0,10)
y=np.sin(x)
plt.plot(x,y)
```

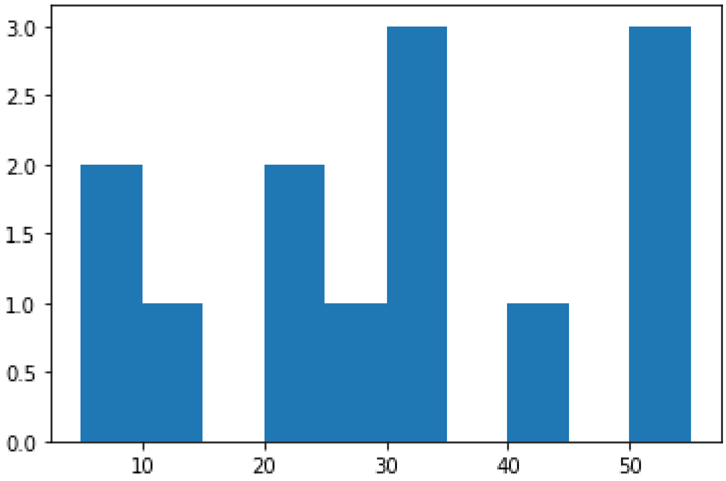
Out[30]:
```
[<matplotlib.lines.Line2D at 0x1f6ce0f5eb0>]
```



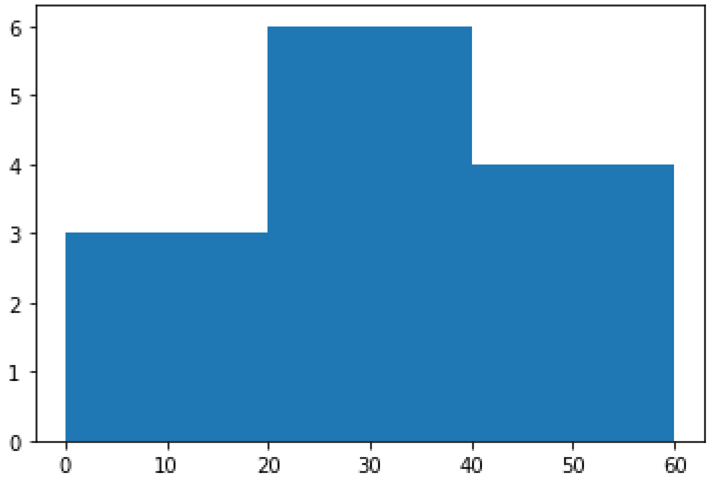
histograms

In [31]:
```
a=np.array([22,32,31,5,43,11,51,5,31,22,55,27,55])
plt.hist(a)
```

Out[31]:
```
(array([2., 1., 0., 2., 1., 3., 0., 1., 0., 3.]),
 array([ 5., 10., 15., 20., 25., 30., 35., 40., 45., 50., 55.]),
 <BarContainer object of 10 artists>)
```



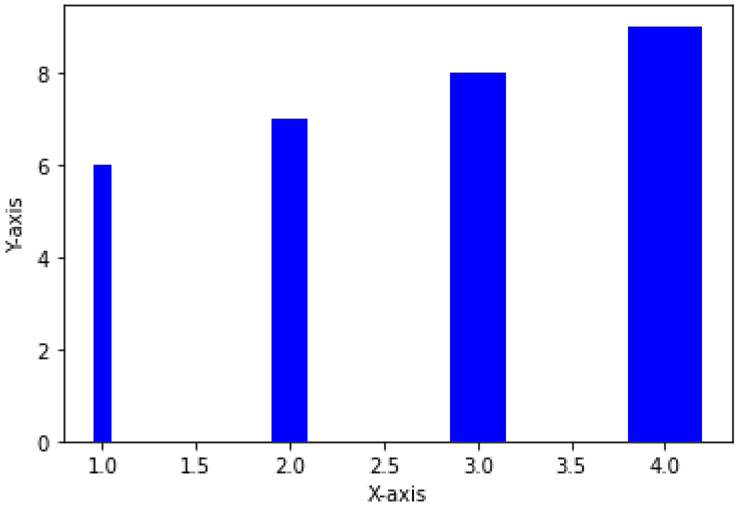
In [32]:
```
a=np.array([22,32,31,5,43,11,51,5,31,22,55,27,55])
bins=[0,20,40,60]
plt.hist(a,bins)
plt.show()
```



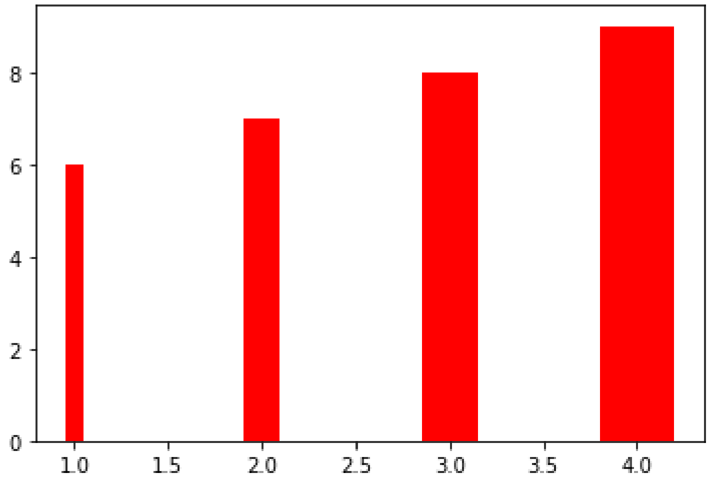
Bar graph

In [33]:
```
x=[1,2,3,4]
y=[6,7,8,9]
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.bar(x,y,width=[0.1,0.2,0.3,0.4],color='b')
```

Out[33]:
```
<BarContainer object of 4 artists>
```



In [34]:
```
plt.bar(x,y,width=[0.1,0.2,0.3,0.4],color='r')
plt.show()
```



```
In [35]: dataframe=pd.read_csv("I:\ADC_LAB\employee.csv")
```

```
In [36]: dataframe
```

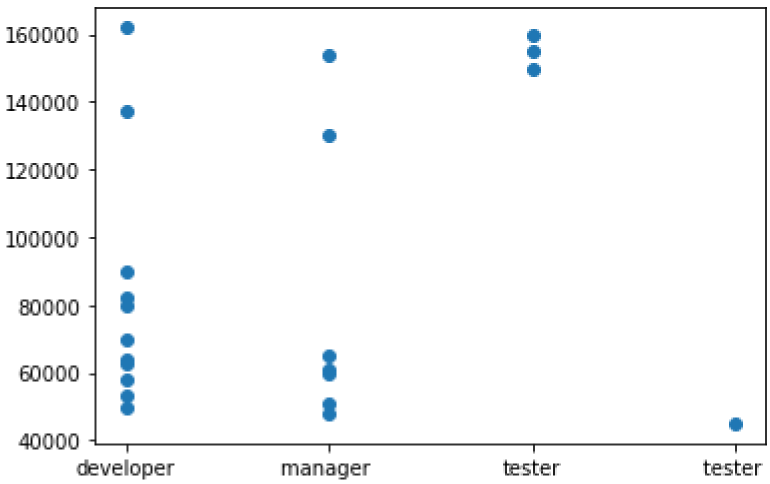
Out[36]:

	Name	Age	salary
0	developer	27	70000
1	developer	29	90000
2	manager	29	61000
3	manager	28	60000
4	tester	42	150000
5	tester	39	155000
6	tester	41	160000
7	developer	38	162000
8	manager	36	154000
9	manager	35	130000
10	developer	37	137000
11	tester	26	45000
12	manager	27	48000
13	manager	28	51000
14	developer	29	49500
15	developer	32	53000
16	manager	40	65000
17	developer	41	63000
18	developer	43	64000
19	developer	39	80000
20	developer	41	82000
21	developer	39	58000

```
In [37]: from matplotlib import pyplot as pl
         pl.scatter(dataframe['Name'],dataframe['salary'])
```

Out[37]:

```
<matplotlib.collections.PathCollection at 0x1f6ce329370>
```



In [38]:

```
dataframe.iloc[:5]
```

Out[38]:

	Name	Age	salary
0	developer	27	70000
1	developer	29	90000
2	manager	29	61000
3	manager	28	60000
4	tester	42	150000

In [39]:

```
slices=[27-30,35-40,41-50]
```

In [40]:

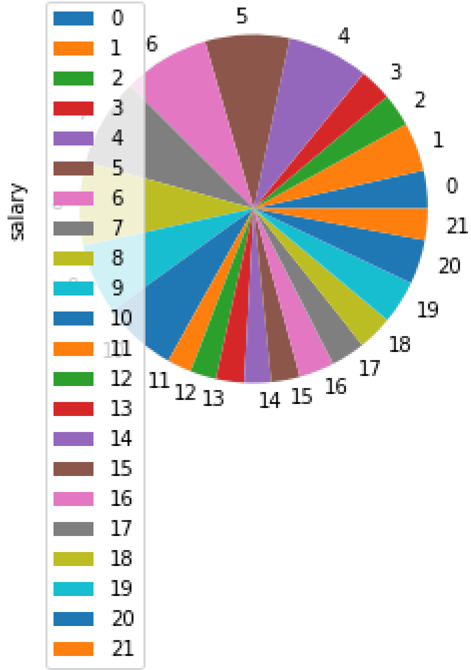
```
salary=[60000-80000,81000-100000,100001-150000]
cols=['g','r','b']
```

In [41]:

```
dataframe.plot.pie(y='salary')
```

Out[41]:

```
<AxesSubplot:ylabel='salary'>
```



In [42]:

```
dataframe=pd.read_excel("I:\AIML\Elective.xlsx")
```

In [43]:

```
dataframe
```

Out[43]:

	Timestamp	Name	Division	Roll No. (MCA2022XXX)	Elective	Elec
0	2023-03-28 14:40:57.467	Namrata Baviskar	A	MCA 20220005	Internet of Things	Digital Marketing & Business Analytics
1	2023-03-28 14:44:58.366	Ajay Thorat	B	MCA2022134	Internet of Things	Natural Language Processing
2	2023-03-28 14:46:44.953	Vishal Vijay Shewale	B	MCA2022122	Internet of Things	Digital Marketing & Business Analytics
3	2023-03-28 14:48:08.043	Eshaan Gupta	B	085	Internet of Things	Natural Language Processing
4	2023-03-28 14:52:01.278	DIPESH MUKUND SURYWANSHI	A	MCA2022063	Internet of Things	Natural Language Processing
...	...	...	...	...	...	...
105	2023-03-31 12:22:00.546	Atul Vishwakarma	B	MCA2022136	Internet of Things	Digital Marketing & Business Analytics
106	2023-03-31 12:23:09.627	NEHAL Tawade	A	MCA2022064	Internet of Things	Natural Language Processing
107	2023-03-31 13:40:51.275	Sushmita giri	B	82	Internet of Things	Digital Marketing & Business Analytics
108	2023-03-31 13:40:53.099	Siddhi Darde	B	MCA2022076	Internet of Things	Digital Marketing & Business Analytics
109	2023-03-31 21:50:25.914	Namrata Baviskar	A	MCA20220005	Internet of Things	Natural Language Processing

110 rows × 6 columns

Classification

In [44]:

```
import numpy as np
import pandas as pd
```

In [ ]:

```
from sklearn.datasets import load_iris
```

In [45]:

```
from sklearn.datasets import load_iris
```

In [46]:

```
iris=load_iris()
```

In [47]:

```
iris.feature_names
```

Out[47]:

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [48]:

```
iris.target_names
```

Out[48]:

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [49]:

```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
```

In [50]:

```
df.head()
```

Out[50]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [51]:

```
df['target']=iris.target
```

In [52]:

```
df.head()
```

Out[52]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [53]:

```
df
```

Out[53]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [54]:

```
df[df.target==0].head()
```

Out[54]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [55]:

```
df0=df[:50]
```

In [56]:

```
df0
```

Out[56]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0
10	5.4	3.7	1.5	0.2	0
11	4.8	3.4	1.6	0.2	0
12	4.8	3.0	1.4	0.1	0
13	4.3	3.0	1.1	0.1	0
14	5.8	4.0	1.2	0.2	0
15	5.7	4.4	1.5	0.4	0
16	5.4	3.9	1.3	0.4	0
17	5.1	3.5	1.4	0.3	0
18	5.7	3.8	1.7	0.3	0
19	5.1	3.8	1.5	0.3	0
20	5.4	3.4	1.7	0.2	0
21	5.1	3.7	1.5	0.4	0
22	4.6	3.6	1.0	0.2	0
23	5.1	3.3	1.7	0.5	0
24	4.8	3.4	1.9	0.2	0
25	5.0	3.0	1.6	0.2	0
26	5.0	3.4	1.6	0.4	0
27	5.2	3.5	1.5	0.2	0
28	5.2	3.4	1.4	0.2	0
29	4.7	3.2	1.6	0.2	0
30	4.8	3.1	1.6	0.2	0
31	5.4	3.4	1.5	0.4	0
32	5.2	4.1	1.5	0.1	0
33	5.5	4.2	1.4	0.2	0
34	4.9	3.1	1.5	0.2	0
35	5.0	3.2	1.2	0.2	0

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
36	5.5	3.5	1.3	0.2	0
37	4.9	3.6	1.4	0.1	0
38	4.4	3.0	1.3	0.2	0
39	5.1	3.4	1.5	0.2	0
40	5.0	3.5	1.3	0.3	0
41	4.5	2.3	1.3	0.3	0
42	4.4	3.2	1.3	0.2	0
43	5.0	3.5	1.6	0.6	0
44	5.1	3.8	1.9	0.4	0
45	4.8	3.0	1.4	0.3	0
46	5.1	3.8	1.6	0.2	0
47	4.6	3.2	1.4	0.2	0
48	5.3	3.7	1.5	0.2	0
49	5.0	3.3	1.4	0.2	0

In [57]:

```
df[45:55]
```

Out[57]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
45	4.8	3.0	1.4	0.3	0
46	5.1	3.8	1.6	0.2	0
47	4.6	3.2	1.4	0.2	0
48	5.3	3.7	1.5	0.2	0
49	5.0	3.3	1.4	0.2	0
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

In [58]:

```
df['flower-type']=df.target.apply(lambda x:iris.target_names[x])
```

In [59]:

```
df
```

Out[59]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower-type
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2	virginica
146	6.3	2.5	5.0	1.9	2	virginica
147	6.5	3.0	5.2	2.0	2	virginica
148	6.2	3.4	5.4	2.3	2	virginica
149	5.9	3.0	5.1	1.8	2	virginica

150 rows × 6 columns

In [60]:

```
import matplotlib.pyplot as plt
```

In [61]:

```
%matplotlib inline
```

df0=df[:50] df1=df[50:100] df2=df[100:] plt.xlabel('sepal length') plt.ylabel('sepal width')

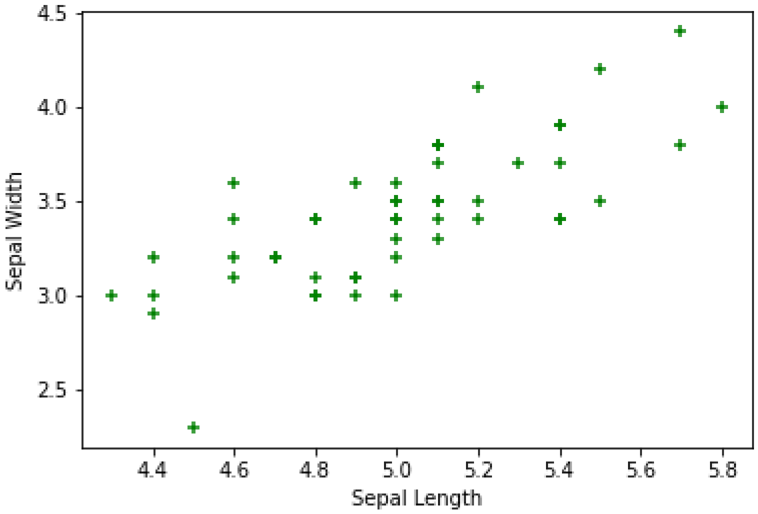
Sepal length vs Sepal Width (Setosa vs Versicolor)

In [62]:

```
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker=
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='
plt.scatter(df2['sepal length (cm)'], df2['sepal width (cm)'],color="yellow",marker
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [62], in <cell line: 4>()
      2 plt.ylabel('Sepal Width')
      3 plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="gree
n",marker='+')
----> 4 plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blu
e",marker='.')
      5 plt.scatter(df2['sepal length (cm)'], df2['sepal width (cm)'],color="yello
w",marker='*')

NameError: name 'df1' is not defined
```



Train test split

In [ ]:

```
from sklearn.model_selection import train_test_split
```

In [ ]:

```
X = df.drop(['target','flower-type'], axis='columns')
y = df.target
```

In [ ]:

```
X
```

In [ ]:

```
y
```

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [ ]:

```
len(X_train)
```

In [ ]:

```
len(X_test)
```

In [64]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
```

In [65]:

```
knn.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [65], in <cell line: 1>()
----> 1 knn.fit(X_train, y_train)

NameError: name 'X_train' is not defined
```

In [ ]:

```
#accuracy of the model
knn.score(X_test, y_test)
```

In [ ]:

```
knn.predict([[4.8,3.0,1.5,0.3]])
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

In [ ]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

In [ ]:

In [ ]: