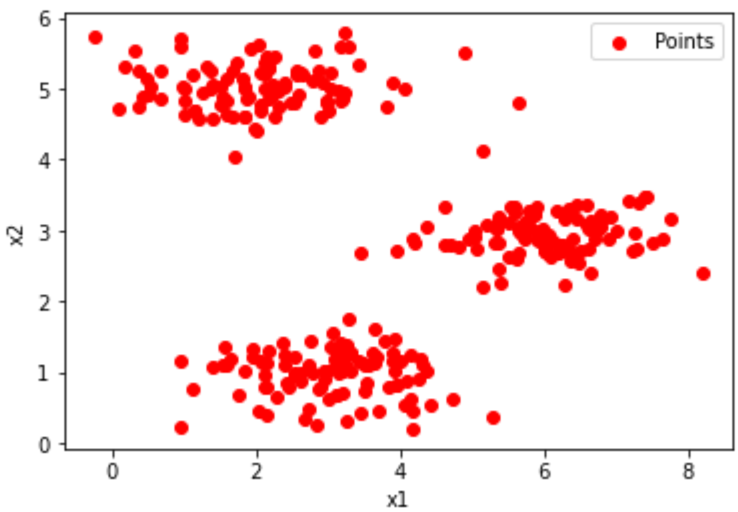
LAB-10

Kmeans_data.csv file contains 300x2 data , to read the csv file pandas library is used.The dataframe created is then converted to numpy array X.To visualize the dataset , scatter plot is plotted.As shown in figure, we can see that the scatter plot has around 3 clusters.To find the optimal number of clusters ,elbow method is used.Before that,algorithm was implemented from scratch for different values of k =[1,2,3,4].

In [ ]:
```
#question 1
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.linalg as la
import math
from scipy.fftpack import fft,fftfreq
from scipy.linalg import toeplitz
from matplotlib import animation
import random
df = pd.read_csv('kmeans_data.csv',header=None)
X = df.iloc[:,:]
X=np.array(X)
plt.scatter(X[:,0],X[:,1],c='red',alpha=1,label='Points')
plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```



Implemented for k=1

In [ ]:
```
k=1
rnd=np.random.randint(300, size=(k))
centroid=np.zeros((k,2))
for i in range(0,k):
  centroid[i]=X[rnd[i]]
print(centroid)
```

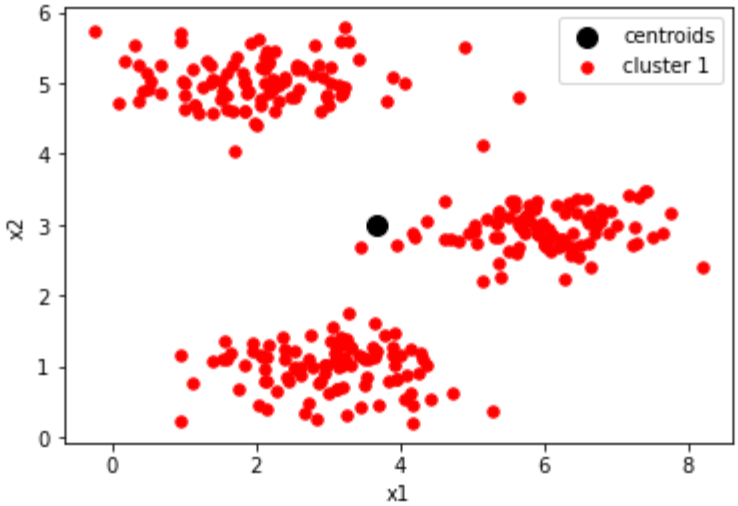
```
[[3.62202931 1.28643763]]
```

In [ ]:
```
def dist(X1, X2):
    return(sum((X1 - X2)**2)**0.5)
for j in range(1):
  cl=[]
  for i in range(k):
    cl.append([])
  for i in range(X.shape[0]):
    dis=[]
    for z in range(k):
      dis.append(dist(X[i],centroid[z]))
    ind=dis.index(min(dis))
    cl[ind].append(X[i])
  for i in range(k):
    centroid[i]=np.average(cl[i],axis=0)
print(centroid)
```

```
[[3.68437558 2.9871008 ]]
```

In [ ]:
```
colors =["r", "g", "c", "b", "k"]
plt.scatter(centroid[:,0],centroid[:,1],s=100,c='k',alpha=1,label='centroids')
for i in range(k):
  color = colors[i]
  for j in range(len(cl[i])):
    if j==0:
      plt.scatter(cl[i][j][0], cl[i][j][1], color = color,s = 30,label='cluster
'+str(i+1))
    else :
       plt.scatter(cl[i][j][0], cl[i][j][1], color = color,s = 30)

plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```



Implemented for k=2

In [91]:
```
k=2
rnd=np.random.randint(300, size=(k))
centroid=np.zeros((k,2))
for i in range(0,k):
  centroid[i]=X[rnd[i]]
print(centroid)
```

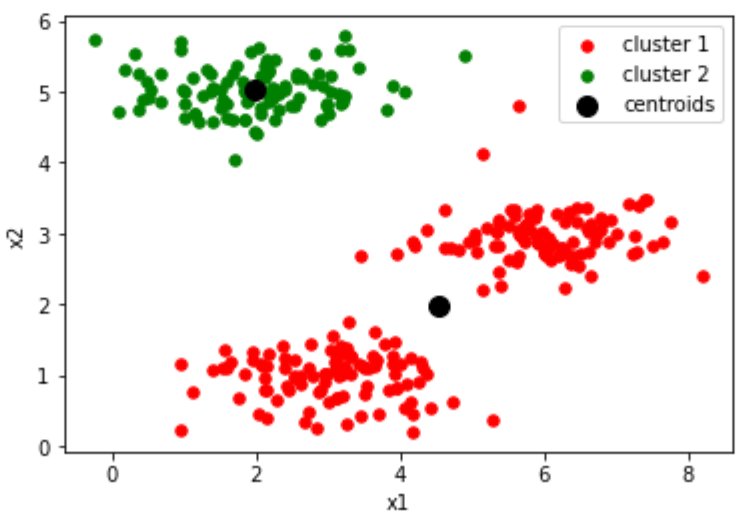
```
[[5.29239452 0.36873298]
 [3.96162465 2.72025046]]
```

In [92]:
```
def dist(X1, X2):
    return(sum((X1 - X2)**2)**0.5)
for j in range(100):
  cl=[]
  for i in range(k):
    cl.append([])
  for i in range(X.shape[0]):
    dis=[]
    for z in range(k):
      dis.append(dist(X[i],centroid[z]))
    ind=dis.index(min(dis))
    cl[ind].append((X[i]))
  for i in range(k):
    centroid[i]=np.average(cl[i],axis=0)
print(centroid)
```

```
[[4.52205549 1.9806849 ]
 [1.98363152 5.03043004]]
```

In [93]:
```
colors =["r", "g", "c", "b", "k"]

for i in range(k):
  color = colors[i]
  for j in range(len(cl[i])):
    if j==0:
      plt.scatter(cl[i][j][0], cl[i][j][1], color = color,s = 30,label='cluster
'+str(i+1))
    else :
       plt.scatter(cl[i][j][0], cl[i][j][1], color = color,s = 30)
plt.scatter(centroid[:,0],centroid[:,1],s=100,c='k',alpha=1,label='centroids')
plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```



Implemented for k=3

In [94]:
```
k=3
rnd=np.random.randint(300, size=(k))
centroid=np.zeros((k,2))
for i in range(0,k):
  centroid[i]=X[rnd[i]]
print(centroid)
```

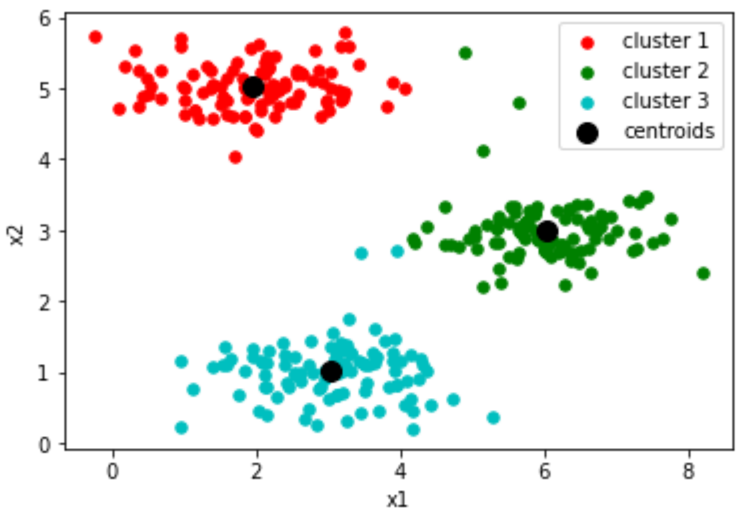
```
[[2.20321658 4.94516379]
 [6.11768055 2.85475655]
 [5.36708111 3.19502552]]
```

In [95]:
```
def dist(X1, X2):
    return(sum((X1 - X2)**2)**0.5)
for j in range(100):
  cl=[]
  for i in range(k):
    cl.append([])
  for i in range(X.shape[0]):
    dis=[]
    for z in range(k):
      dis.append(dist(X[i],centroid[z]))
    ind=dis.index(min(dis))
    cl[ind].append(X[i])
  for i in range(k):
    centroid[i]=np.average(cl[i],axis=0)
print(centroid)
```

```
[[1.95399466 5.02557006]
 [6.03366736 3.00052511]
 [3.04367119 1.01541041]]
```

In [96]:
```
colors =["r", "g", "c", "b", "k"]

for i in range(k):
  color = colors[i]
  for j in range(len(cl[i])):
    if j==0:
      plt.scatter(cl[i][j][0], cl[i][j][1], color = color,s = 30,label='cluster
'+str(i+1))
    else :
       plt.scatter(cl[i][j][0], cl[i][j][1], color = color,s = 30)
plt.scatter(centroid[:,0],centroid[:,1],s=100,c='k',alpha=1,label='centroids')
plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```



Implemented for k=4

In [97]:
```
k=4
rnd=np.random.randint(300, size=(k))
centroid=np.zeros((k,2))
for i in range(0,k):
  centroid[i]=X[rnd[i]]
print(centroid)
```

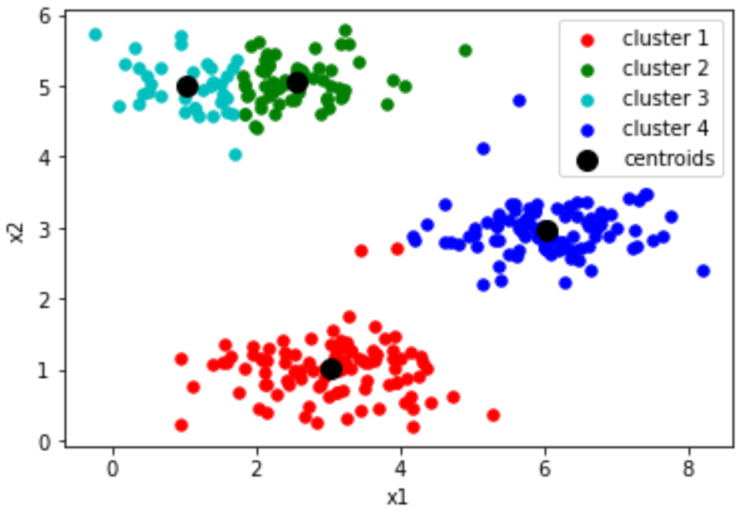
```
[[1.57449255 1.34811126]
 [1.60493227 5.13663139]
 [1.20099981 4.57829763]
 [2.15520661 0.80696562]]
```

In [98]:
```
def dist(X1, X2):
    return(sum((X1 - X2)**2)**0.5)
for j in range(100):
  cl=[]
  for i in range(k):
    cl.append([])
  for i in range(X.shape[0]):
    dis=[]
    for z in range(k):
      dis.append(dist(X[i],centroid[z]))
    ind=dis.index(min(dis))
    cl[ind].append(X[i])
  for i in range(k):
    centroid[i]=np.average(cl[i],axis=0)
print(centroid)
```

```
[[3.04367119 1.01541041]
 [2.57620478 5.04891014]
 [1.0323955  5.00076462]
 [6.04523932 2.97521013]]
```

In [99]:
```
colors =["r", "g", "c", "b", "k"]

for i in range(k):
  color = colors[i]
  for j in range(len(cl[i])):
    if j==0:
      plt.scatter(cl[i][j][0], cl[i][j][1], color = color,s = 30,label='cluster
'+str(i+1))
    else :
       plt.scatter(cl[i][j][0], cl[i][j][1], color = color,s = 30)
plt.scatter(centroid[:,0],centroid[:,1],s=100,c='k',alpha=1,label='centroids')
plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```



Elbow method for optimal value of K

As shown in figure elbow method is been used to find optimal value of k.As we can see after value of k=3 , change in inertia becomes very less.So , k=3 can be considered as elbow.so,3 will be the optimal value.

In [113]:

```
from sklearn.cluster import KMeans
elbow=[]
for i in range(1,10):
  kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = None)
  kmeans.fit(X)
  elbow.append(kmeans.inertia_)
x=np.linspace(1,9,9,dtype=int)
plt.plot(x,elbow)
plt.xlabel("k")
plt.ylabel("inertia")
plt.show()
```