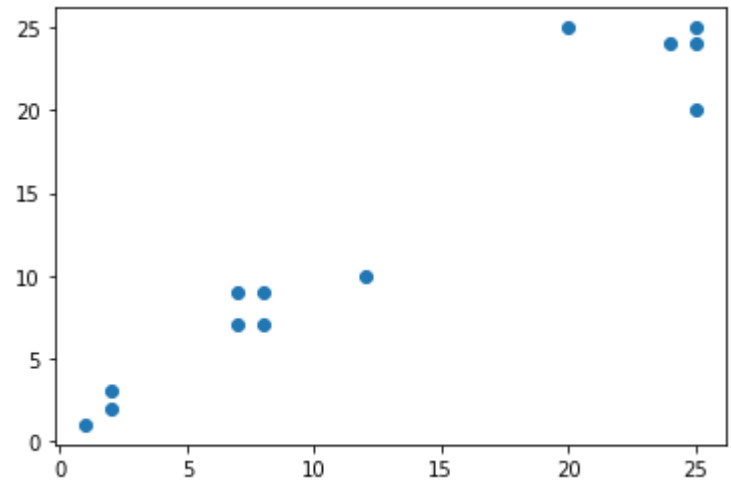
```
#anomaly detection using k means clustering



#generating the data to detect anomaly


import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist


data=np.array([[1,1],[2,2],[2,3],[8,7],[8,9],[7,9],[7,7],[12,10],[25,24],[24,24],[2


data
```

```
array([[ 1,  1],
       [ 2,  2],
       [ 2,  3],
       [ 8,  7],
       [ 8,  9],
       [ 7,  9],
       [ 7,  7],
       [12, 10],
       [25, 24],
       [24, 24],
       [25, 20],
       [20, 25],
       [25, 25]])
```

```
plt.scatter(data[:,0],data[:,1])
```

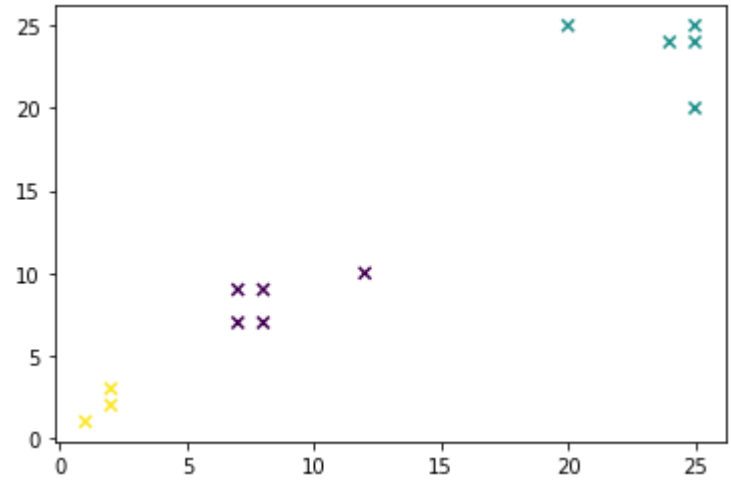
```
<matplotlib.collections.PathCollection at 0x7f707a938b50>
```




```
#k-means model with k=3
km = KMeans(n_clusters=3)
clusters=km.fit_predict(data)


clusters=km.predict(data)
```

```
plt.scatter(*zip(*data),c=clusters,marker='x')
```

```
<matplotlib.collections.PathCollection at 0x7f707a92c250>
```




```
#obtain the centers of the clusters
centroid=km.cluster_centers_
```

```
centroid
```

```
array([[ 8.4       ,  8.4       ],
       [23.8       , 23.6       ],
       [ 1.66666667,  2.        ]])
```

```
#initialise a array which will be used to reach the index
points = np.empty((0,len(data[0])),float)
```

```
points.shape
```

```
(0, 2)
```

```
#initialize an array which will be used to calculate outlier distance
distances=np.empty((0,len(data[0])),float)
```

```
distances.shape
```

```
(0, 2)
```

```
for i, center_elem in enumerate(centroid):
  distances=np.append(distances,cdist([center_elem],data[clusters==i],'euclidean'))
  points=np.append(points,data[clusters==i],axis=0)
```

```
distances
```

```
array([1.45602198, 0.72111026, 1.52315462, 1.97989899, 3.93954312,
       1.26491106, 0.4472136 , 3.79473319, 4.04969135, 1.84390889,
```

```
            1.20185043, 0.33333333, 1.05409255])
```

```
points
```

```
    array([[ 8.,  7.],
           [ 8.,  9.],
           [ 7.,  9.],
           [ 7.,  7.],
           [12., 10.],
           [25., 24.],
           [24., 24.],
           [25., 20.],
           [20., 25.],
           [25., 25.],
           [ 1.,  1.],
           [ 2.,  2.],
           [ 2.,  3.]])
```

```
threshold =80
```

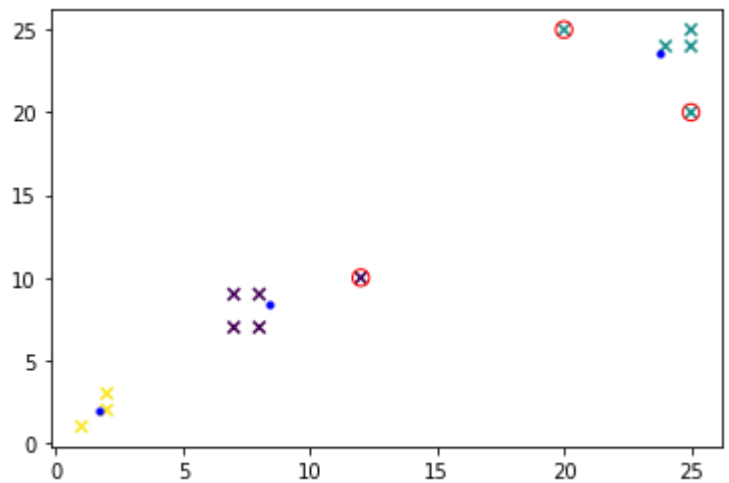
```
outliers=points[np.where(distances>np.percentile(distances,threshold))]
```

```
outliers
```

```
    array([[12., 10.],
           [25., 20.],
           [20., 25.]])
```

```
plt.scatter(*zip(*data),c=clusters,marker='x')
plt.scatter(*zip(*outliers),marker='o',facecolor='None',edgecolors='r',s=70)
plt.scatter(*zip(*centroid),marker='o',facecolor='b',edgecolors='b',s=10)
```

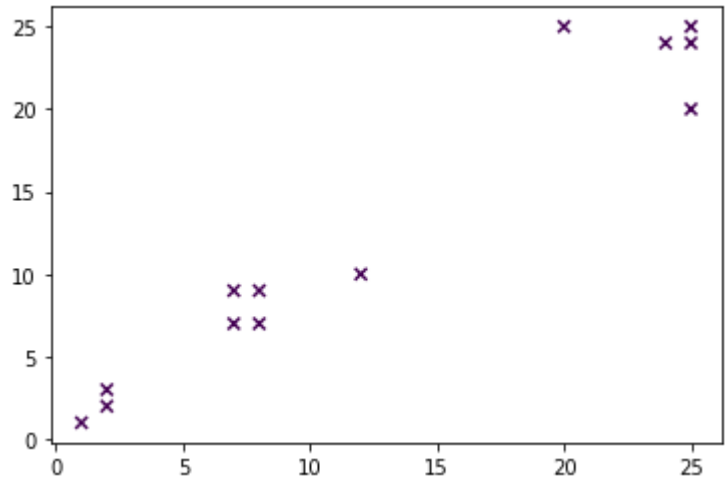
```
    <matplotlib.collections.PathCollection at 0x7f707a179950>
```




```
#similarly w k=1
km = KMeans(n_clusters=1)
clusters=km.fit_predict(data)
```

```
clusters=km.predict(data)


plt.scatter(*zip(*data),c=clusters,marker='x')
```

```
<matplotlib.collections.PathCollection at 0x7f707a0e2a50>
```




```
#obtain the centers of the clusters
centroid=km.cluster_centers_


#initialise a array which will be used to reach the index
points = np.empty((0,len(data[0])),float)


#initialize an array which will be used to calculate outlier distance
distances=np.empty((0,len(data[0])),float)


for i, center_elem in enumerate(centroid):
  distances=np.append(distances,cdist([center_elem],data[clusters==i],'euclidean'))
  points=np.append(points,data[clusters==i],axis=0)


outliers=points[np.where(distances>np.percentile(distances,threshold))]


plt.scatter(*zip(*data),c=clusters,marker='x')
plt.scatter(*zip(*outliers),marker='o',facecolor='None',edgecolors='r',s=70)
plt.scatter(*zip(*centroid),marker='o',facecolor='b',edgecolors='b',s=10)
```

```
<matplotlib.collections.PathCollection at 0x7f707a060210>
```


```
#part B : kmeans clustering on randomly generated regression data
```

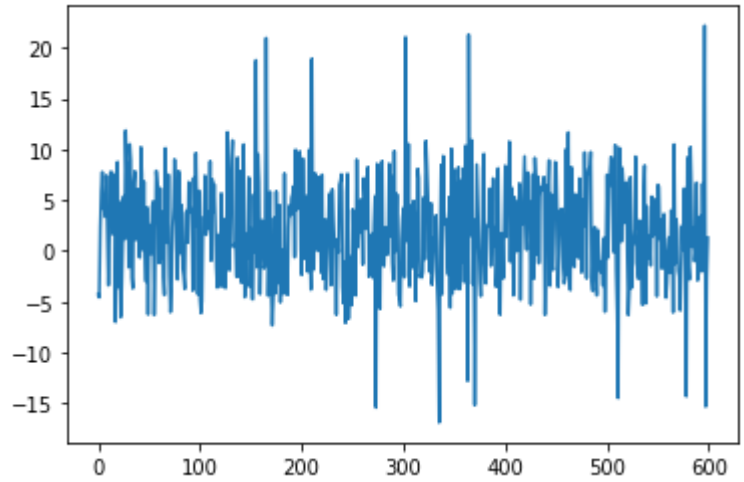
```
from sklearn.cluster import KMeans
from numpy import sqrt, array,random,argsort
from sklearn.preprocessing import scale
import matplotlib.pyplot as plt
```

```
#generating the data
random.seed(121)
def makeData(N):
  x=[]
  for i in range(N):
    a=i/1000 + random.uniform(-3,2)
    r=random.uniform(-5,10)
    if(r>=9.9):
      r+=10
    elif(r<(-4.8)):
      r-=10
    x.append([a+r])
  return array(x)


x=makeData(600)


x.shape
```

```
(600, 1)
```

```
plt.plot(x)
```

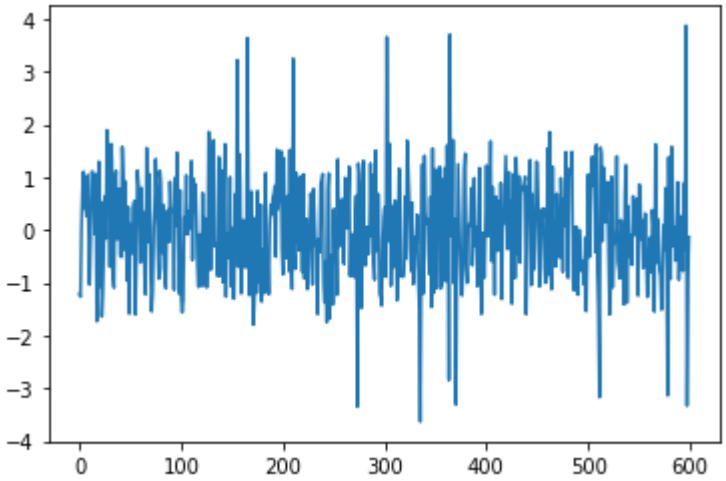
```
[<matplotlib.lines.Line2D at 0x7f9ce8c412d0>]
```




```
#scale the data
x=scale(x)
```

```
plt.plot(x) #the data has been scaled and standardized as you can see from the rang
```

```
[<matplotlib.lines.Line2D at 0x7f9ce8733650>]
```




```
kmeans=KMeans(n_clusters=1).fit(x)


#determine the centroid of the data cluster
center=kmeans.cluster_centers_


center
```

```
array([[-1.19811568e-17]])
```

```
#determine the distance of the point from the center of the clustr
distance=sqrt((x-center)**2)


#sorting the distance
order_index=argsort(distance,axis=0)
indexes=order_index[-5:]#5points w max distance from the center
values=x[indexes]


values
```

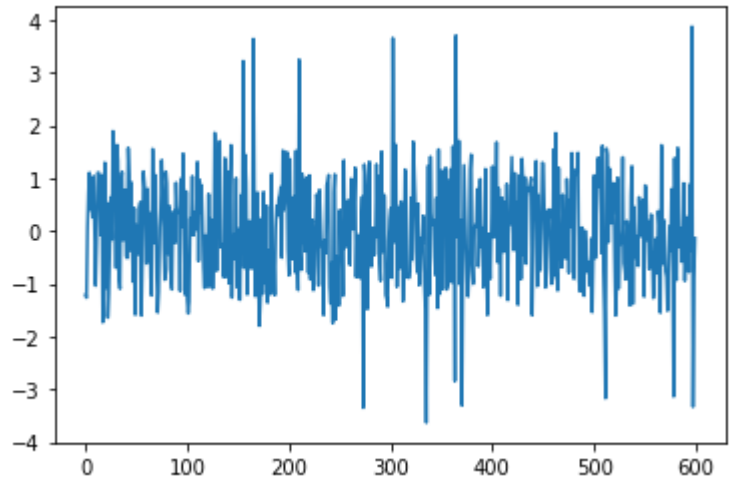
```
array([[[-3.63432676]],

       [[ 3.64141157]],

       [[ 3.65996024]],

       [[ 3.70682974]],

       [[ 3.87702328]]])


indexes
```

```
array([[335],
       [165],
```

```
            [302],
            [364],
            [596]])
```

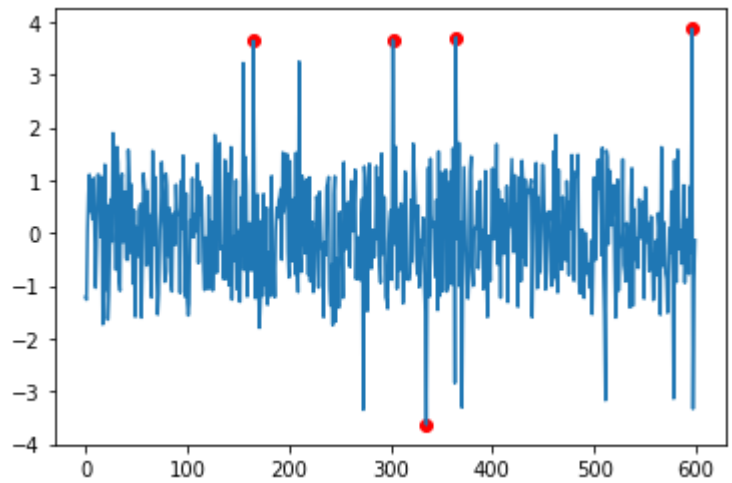
```
plt.plot(x)
```

```
[<matplotlib.lines.Line2D at 0x7f9ce8736bd0>]
```



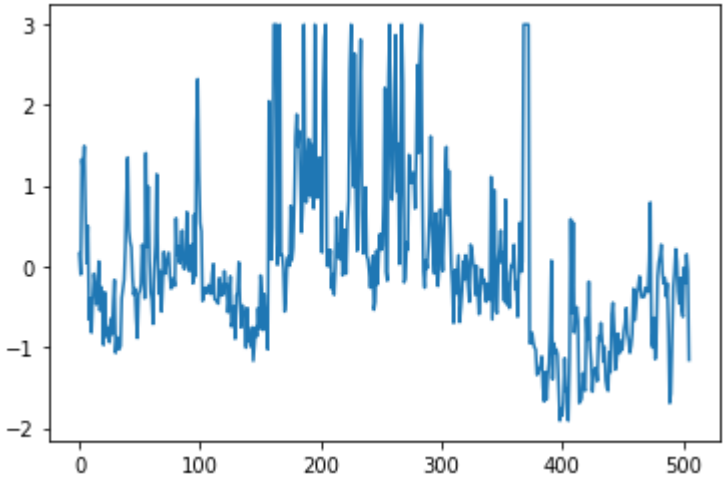
```
plt.plot(x)
plt.scatter(indexes,values,color='r')
```

```
<matplotlib.collections.PathCollection at 0x7f9ce85adb90>
```



```
#PART C :applying it on boston housing dataset
from sklearn.datasets import load_boston



boston=load_boston()
y=boston.target
y=y.reshape(y.shape[0],1)
y=scale(y)



y_ax=range(y.shape[0])
plt.plot(y)
```

```
[<matplotlib.lines.Line2D at 0x7f9ce79e3490>]
```



```
kmeans=KMeans(n_clusters=1).fit(y)


#determine the centroid of the data cluster
center=kmeans.cluster_centers_


#determine the distance of the point from the center of the clustr
distance=sqrt((y-center)**2)


#sorting the distance
order_index=argsort(distance,axis=0)
indexes=order_index[-8:]#5points w max distance from the center
values=y[indexes]
```

```
plt.plot(y)
plt.scatter(indexes,values,color='r')
```

```
<matplotlib.collections.PathCollection at 0x7f9ce77cdc10>
```