In [6]:

```
import random
import math
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np
%matplotlib inline

def pointGenerator(l,h):
    a=[]
    a.append(random.randint(l,h))
    a.append(random.randint(l,h))
    return a

def euclideanDist(p,c):
    return math.sqrt((p[0]-c[0])**2 + (p[1]-c[1])**2)

def calcCentroid(List):
    sm1 = sm2 =0
    for i in List:
        sm1 += i[0]
        sm2 += i[1]
    return [float(sm1)/len(List),float(sm2)/len(List)]



def K_means(centroids,randomPoints):
    while True:
        cluster=[] #nested list storing the co-ordiates closest to
the respective inedexed centroid
        for i in range(0,k):
            cluster.append([])

        #Calculating euclidean distance of each co-ordinate with al
l the centroids and finding the closest centroids
        for x in randomPoints:
            mn = 999
            i=0
            index =0
            for y in centroids:
                dist = euclideanDist(x,y)
                if mn >= dist:
                    mn = dist
                    index = i
                i += 1

            #appending the co-ordinate to a list representing centr
oid closest to the point
            cluster[index].append(x)
        centroids1 =[]
        #Calculating the mean of the co-ordinates or the centroid of the cluster created
        for i in range(0,len(centroids)):
            if len(cluster[i]) is not 0:
                centroids1.append(calcCentroid(cluster[i]))
            else:
                centroids1.append(centroids[i])

        #Condition to check whether to continue looking for a new centroid or not
        if centroids != centroids1:
            centroids = centroids1
        else:
            break
    return cluster


def plot(cluster):
    #converting the data set in a format compatible for scatter plotting


    for i in range(0,k+1):
        X.append([])
        Y.append([])


    i=0
    for x in cluster:
        for y in x:
            XX.append(y[0])
            YY.append(y[1])
            X[i].append(y[0])
            Y[i].append(y[1])
        i +=1





#lists for random co-ordinates,centroids and the clustered co-ordinates
randomPoints=[]
centroids=[]
cluster=[]

#Range within which the random points are to be chosen
l=0
h=100

#setting the value for k
```

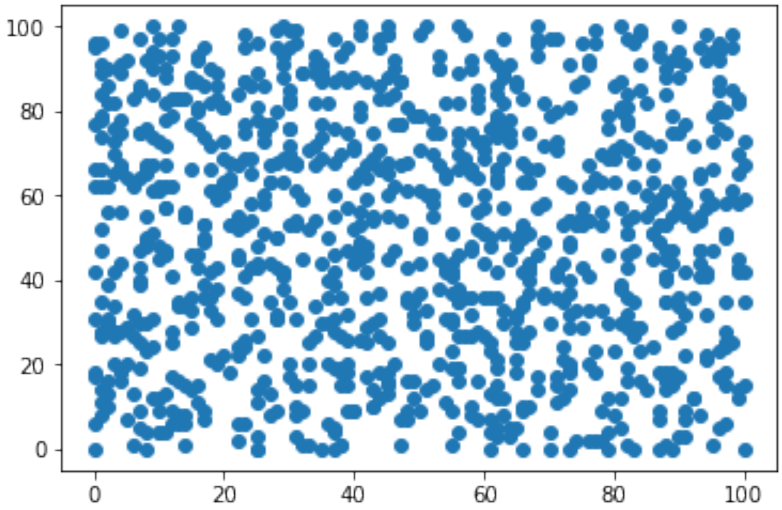
```
k=3


#generating random co-ordinated and creating a list for it
for i in range(1,1000):
    randomPoints.append(pointGenerator(l,h))


#Starting with a list of random K-centroids,as is done in k_means a
lgorithm
for i in range(0,k):
    centroids.append(pointGenerator(l,h))


#calling the k_means functions which returns a nested list containi
ng the lists of clusteres co-ordinates
cluster = K_means(centroids,randomPoints)


#X will the lists containing x-cordinates of points of same cluster
grouped together,similarly Y too
X=[]
Y=[]

#XX and YY  will contain the whole x-cordinates and y-cordinates re
spectively
XX=[]
YY=[]

plot(cluster)
```

In [7]:

```
#Unclustered dataset
plt.scatter(XX,YY)
```

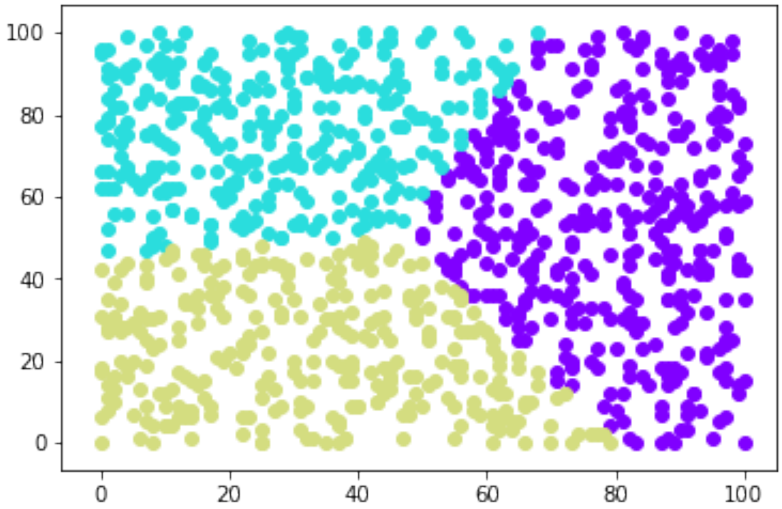
Out[7]:

```
<matplotlib.collections.PathCollection at 0x11a72fc90>
```



In [8]:

```
import itertools
colors = iter(cm.rainbow(np.linspace(0, 1, len(Y))))
for i in range(0,k):
    plt.scatter(X[i],Y[i], color=next(colors))
```

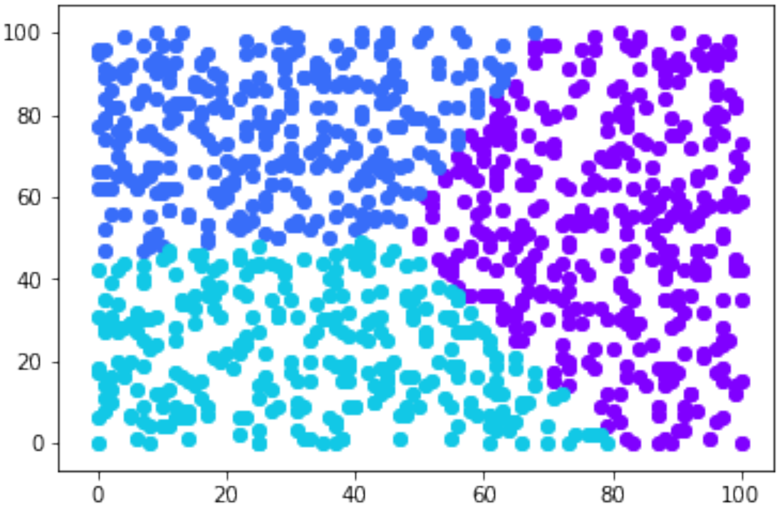


In [10]:

```
print("Enter a co-ordinate :")
x,y = raw_input().split()
x,y =[int(x),int(y)]
new =[x,y]
randomPoints.append(new)
cluster = K_means(centroids,randomPoints)
plot(cluster)
```

```
Enter a co-ordinate :
5 6
```

In [11]:
```
import itertools
colors = iter(cm.rainbow(np.linspace(0, 1, len(Y))))
for i in range(0,k):
    plt.scatter(X[i],Y[i], color=next(colors))
```



In [ ]: