**def** getSquaredDistance(point1, point2):

**return** round(((point2[0] **-** point1[0])**\*\***2 **+** (point2[1] **-** point1[1])**\*\***2),4)

**def** getDistanceFromPoints(centroids, datapoints):

distance\_from\_cluster **=** [] *# [cluster1\_distances, cluster2\_distances,...]*

intermediate\_result **=** []

**for** i **in** centroids:

**for** j **in** datapoints:

intermediate\_result**.**append(getSquaredDistance(i,j))

distance\_from\_cluster**.**append(intermediate\_result)

intermediate\_result **=** [] *# reset intermediate\_result as empty list*

**return** distance\_from\_cluster

**def** printResult(centroids, point\_to\_cluster\_mapping):

**for** i **in** range(len(centroids)):

print("Centroid",i,centroids[i])

**for** i **in** point\_to\_cluster\_mapping:

print("Point: ",i,"Cluster:",point\_to\_cluster\_mapping[i])

**def** kmeansclustering(centroids, datapoints):

'''

Driver code for K-Means clustering

'''

k **=** len(centroids)

distance\_from\_cluster **=** getDistanceFromPoints(centroids, datapoints) *# [cluster1\_distances, cluster2\_distances,...]*

point\_to\_cluster\_mapping **=** {} *# point -> cluster*

max\_valued\_cluster **=** 0

**for** i **in** range(len(datapoints)):

point\_to\_cluster\_mapping[i] **=** **None**

**for** cluster **in** range(len(centroids)):

**if** distance\_from\_cluster[cluster][i] **<** distance\_from\_cluster[max\_valued\_cluster][i]:

max\_valued\_cluster **=** cluster

point\_to\_cluster\_mapping[i] **=** max\_valued\_cluster

cluster\_counter **=** 0

**for** i **in** range(len(centroids)):

**for** j **in** point\_to\_cluster\_mapping:

**if** point\_to\_cluster\_mapping[j] **==** i:

centroids[i][0] **+=** datapoints[j][0]

centroids[i][1] **+=** datapoints[j][1

cluster\_counter **+=** 1

**if** cluster\_counter **!=** 0:

centroids[i][0] **=** round(centroids[i][0]**/**cluster\_counter,4)

centroids[i][1] **=** round(centroids[i][1]**/**cluster\_counter,4)

cluster\_counter **=** 0

printResult(centroids, point\_to\_cluster\_mapping)

**return** centroids

**def** kmeans\_iterator(centroids, datapoints):

old\_centroids **=** centroids

new\_centroids **=** centroids

iteration **=** 0

**while** iteration **!=** 15:

iteration **+=** 1

print("\nIteration ", iteration)

old\_centroids **=** new\_centroids

new\_centroids **=** kmeansclustering(new\_centroids, datapoints)

centroids **=** [[2,10], [5,8], [1,2]]

datapoints **=** [[2,10], [2,5], [8,4], [5,8], [7,5], [6,4], [1,2], [4,9]]

kmeans\_iterator(centroids, datapoints)