In [15]:
```
def getSquaredDistance(point1, point2):
    return round(((point2[0] - point1[0])**2 + (point2[1] - point1[1]

def getDistanceFromPoints(centroids, datapoints):
    distance_from_cluster = []     # [cluster1_distances, cluster2_di
    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 emp
    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, datapoir


    # assign each datapoint to the nearest cluster

    point_to_cluster_mapping = {}   # point -> cluster
    max_valued_cluster = 0
    for i in range(len(datapoints)):
        point_to_cluster_mapping[i] = None  # initial mapping as None
        for cluster in range(len(centroids)):
            if distance_from_cluster[cluster][i] < distance_from_clus
                max_valued_cluster = cluster
        point_to_cluster_mapping[i] = max_valued_cluster

    # compute new centroids by averaging with new points
    cluster_counter = 0       # [cluster1_new_elements_added, cluste
    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]   # x-coordinate
                centroids[i][1] += datapoints[j][1]   # y-coordinate
                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)
```

```
Iteration  1
Centroid 0 [4.0, 20.0]
Centroid 1 [7.0, 7.6]
Centroid 2 [2.0, 4.5]
Point:  0 Cluster: 0
Point:  1 Cluster: 2
Point:  2 Cluster: 1
Point:  3 Cluster: 1
Point:  4 Cluster: 1
Point:  5 Cluster: 1
Point:  6 Cluster: 2
Point:  7 Cluster: 1

Iteration  2
Centroid 0 [4.0, 20.0]
Centroid 1 [7.4, 7.52]
Centroid 2 [2.3333, 7.1667]
Point:  0 Cluster: 2
```

In [ ]: