**CODE:**

import random

def calculate\_distance(point1, point2):

return abs(point1 - point2)

def assign\_to\_clusters(data, centroids):

clusters = [[] for \_ in centroids]

for point in data:

distances = [calculate\_distance(point, centroid) for centroid in centroids]

closest\_cluster = distances.index(min(distances))

clusters[closest\_cluster].append(point)

return clusters

def update\_centroids(clusters):

new\_centroids = []

for cluster in clusters:

if cluster:

new\_centroid = sum(cluster) / len(cluster)

new\_centroids.append(new\_centroid)

else:

new\_centroids.append(0)

return new\_centroids

def k\_means(data, k, max\_iterations=100):

centroids = random.sample(data, k)

for iteration in range(max\_iterations):

clusters = assign\_to\_clusters(data, centroids)

new\_centroids = update\_centroids(clusters)

print(f"Iteration {iteration + 1}:")

print(f"Cluster Centroids: {centroids}")

for i, cluster in enumerate(clusters):

print(f"Cluster {i + 1} Data Points: {cluster}")

if new\_centroids == centroids:

break

centroids = new\_centroids

k = int(input("Enter the number of clusters: "))

data = list(map(float, input("Enter numbers separated by spaces: ").split()))

k\_means(data, k)

**OUTPUT:**

