TASK 2

import random

import math

def euclidean\_distance(point1, point2):

return math.sqrt(sum((x - y) \*\* 2 for x, y in zip(point1, point2)))

def mean(points):

num\_points = len(points)

num\_dimensions = len(points[0])

return [sum(p[dim] for p in points) / num\_points for dim in range(num\_dimensions)]

def assign\_clusters(data, centroids):

clusters = [[] for \_ in centroids]

for point in data:

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

closest\_centroid\_index = distances.index(min(distances))

clusters[closest\_centroid\_index].append(point)

return clusters

def k\_means\_clustering(data, k, max\_iterations=100, tolerance=1e-4):

# Step 1: Initialize centroids randomly from the data points

centroids = random.sample(data, k)

for iteration in range(max\_iterations):

# Step 2: Assign points to the nearest centroid

clusters = assign\_clusters(data, centroids)

# Step 3: Update centroids

new\_centroids = [mean(cluster) if cluster else centroids[i] for i, cluster in enumerate(clusters)]

# Check for convergence

if all(euclidean\_distance(c1, c2) < tolerance for c1, c2 in zip(centroids, new\_centroids)):

break

centroids = new\_centroids

return clusters, centroids

# Sample data: Customer purchase history (e.g., amount spent on different product categories)

data = [

[10, 20],

[20, 30],

[30, 40],

[100, 120],

[120, 130],

[130, 140]

]

# Number of clusters

k = 2

# Run K-means clustering

clusters, centroids = k\_means\_clustering(data, k)

print("Clusters:")

for i, cluster in enumerate(clusters):

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

print("Centroids:")

for centroid in centroids:

print(centroid)