Function for K-means and Evaluation

```
import math

def k_means(data_points, initial_centroids):


  centroids = list(initial_centroids)
  k = len(centroids)
  iteration = 0

  print(f"Starting Centroids: {centroids}")
  print(f"Data Points: {data_points}\n")

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

    clusters = [[] for _ in range(k)]

    for point in data_points:
      distances = []
      for centroid in centroids:
        dist = math.sqrt((point[0] - centroid[0])**2 + (point[1] - centroid[1])**2)
        distances.append(dist)
        closest_centroid_index = distances.index(min(distances))

      clusters[closest_centroid_index].append(point)
      print(f"Point {point} Assigned to Cluster {closest_centroid_index} with distance {min(distan

    new_centroids = []

    for i in range(k):
      cluster_points = clusters[i]

      if not cluster_points:
        new_centroids.append(centroids[i])
        print(f"Cluster {i} is empty.")
      else:
        sum_x = sum(p[0] for p in cluster_points)
        sum_y = sum(p[1] for p in cluster_points)
        mean_x = sum_x / len(cluster_points)
        mean_y = sum_y / len(cluster_points)

        new_centroid = (mean_x, mean_y)
        new_centroids.append(new_centroid)
        print(f"Cluster {i} points: {cluster_points} -> New Centroid: {new_centroid}")
```

```
        if new_centroids == centroids:
            print("convergence, stop")
            break

        centroids = new_centroids
        print("\n")

    return centroids, clusters
```

```
import numpy as np
from sklearn.metrics import silhouette_score
from scipy.spatial.distance import pdist

def evaluate_clustering(centroids, clusters):
    X = []
    labels = []
    for i, cluster_points in enumerate(clusters):
        for point in cluster_points:
            X.append(point)
            labels.append(i)

    X = np.array(X)
    labels = np.array(labels)
    centroids = np.array(centroids)

    wcss = 0
    total_dist_sum = 0
    cluster_avg_dists = {}

    for i, cluster_points in enumerate(clusters):
        if not cluster_points:
            continue
        points = np.array(cluster_points)
        centroid = centroids[i]

        dists = np.linalg.norm(points - centroid, axis=1)

        wcss += np.sum(dists ** 2)

        total_dist_sum += np.sum(dists)
        cluster_avg_dists[i] = np.mean(dists)

    if len(set(labels)) > 1 and len(X) > 1:
        sil_score = silhouette_score(X, labels)
    else:
        sil_score = -1

    if len(centroids) > 1:
```

```
        centroid_dists = pdist(centroids, metric='euclidean')
        avg_inter_cluster_dist = np.mean(centroid_dists)
        min_inter_cluster_dist = np.min(centroid_dists)
    else:
        avg_inter_cluster_dist = 0.0
        min_inter_cluster_dist = 0.0

    global_avg_dist_from_centroid = total_dist_sum / len(X) if len(X) > 0 else 0

    print("=== Clustering Evaluation Metrics ===")
    print(f"Within-Cluster Sum of Squares (WCSS): {wcss:.4f}")
    print(f"Silhouette Score: {sil_score:.4f}")
    print(f"Inter-Cluster Distance:")
    print(f"   - Average Distance: {avg_inter_cluster_dist:.4f}")
    print(f"   - Minimum Separation: {min_inter_cluster_dist:.4f}")
    print(f"Average Distance from Centroid:")
    print(f"   - Global Average: {global_avg_dist_from_centroid:.4f}")
    for i, avg in cluster_avg_dists.items():
        print(f"   - Cluster {i} Average: {avg:.4f}")

    return {
        "wcss": wcss,
        "silhouette": sil_score,
        "avg_inter_cluster": avg_inter_cluster_dist,
        "global_avg_dist": global_avg_dist_from_centroid
    }
```

T5

```
# For T5
print("T5 Answer")
all_points = [
    (1, 2), (3, 3), (2, 2), (8, 8), (6, 6), (7, 7), (-3,-3), (-2, -4), (-7,-7)
]

starting_centroids = [
    (3, 3),
    (2, 2),
    (-3,-3)
]

final_centroids, final_clusters = k_means(all_points, starting_centroids)

for i, cluster in enumerate(final_clusters):
    print(f"Cluster {i} (Centroid {final_centroids[i]}): {cluster}")

print("\n--- Evaluating T5 Results ---")
```

```
evaluate_clustering(final_centroids, final_clusters)
```

```
T5 Answer
Starting Centroids: [(3, 3), (2, 2), (-3, -3)]
Data Points: [(1, 2), (3, 3), (2, 2), (8, 8), (6, 6), (7, 7), (-3, -3), (-2, -4), (-7, -7)]

Iteration: 1
Point (1, 2) Assigned to Cluster 1 with distance 1.0
Point (3, 3) Assigned to Cluster 0 with distance 0.0
Point (2, 2) Assigned to Cluster 1 with distance 0.0
Point (8, 8) Assigned to Cluster 0 with distance 7.0710678118654755
Point (6, 6) Assigned to Cluster 0 with distance 4.242640687119285
Point (7, 7) Assigned to Cluster 0 with distance 5.656854249492381
Point (-3, -3) Assigned to Cluster 2 with distance 0.0
Point (-2, -4) Assigned to Cluster 2 with distance 1.4142135623730951
Point (-7, -7) Assigned to Cluster 2 with distance 5.656854249492381
Cluster 0 points: [(3, 3), (8, 8), (6, 6), (7, 7)] -> New Centroid: (6.0, 6.0)
Cluster 1 points: [(1, 2), (2, 2)] -> New Centroid: (1.5, 2.0)
Cluster 2 points: [(-3, -3), (-2, -4), (-7, -7)] -> New Centroid: (-4.0, -4.666666666666667)


Iteration: 2
Point (1, 2) Assigned to Cluster 1 with distance 0.5
Point (3, 3) Assigned to Cluster 1 with distance 1.8027756377319946
Point (2, 2) Assigned to Cluster 1 with distance 0.5
Point (8, 8) Assigned to Cluster 0 with distance 2.8284271247461903
Point (6, 6) Assigned to Cluster 0 with distance 0.0
Point (7, 7) Assigned to Cluster 0 with distance 1.4142135623730951
Point (-3, -3) Assigned to Cluster 2 with distance 1.9436506316151003
Point (-2, -4) Assigned to Cluster 2 with distance 2.1081851067789197
Point (-7, -7) Assigned to Cluster 2 with distance 3.8005847503304597
Cluster 0 points: [(8, 8), (6, 6), (7, 7)] -> New Centroid: (7.0, 7.0)
Cluster 1 points: [(1, 2), (3, 3), (2, 2)] -> New Centroid: (2.0, 2.3333333333333335)
Cluster 2 points: [(-3, -3), (-2, -4), (-7, -7)] -> New Centroid: (-4.0, -4.666666666666667)


Iteration: 3
Point (1, 2) Assigned to Cluster 1 with distance 1.0540925533894598
Point (3, 3) Assigned to Cluster 1 with distance 1.201850425154663
Point (2, 2) Assigned to Cluster 1 with distance 0.3333333333333335
Point (8, 8) Assigned to Cluster 0 with distance 1.4142135623730951
Point (6, 6) Assigned to Cluster 0 with distance 1.4142135623730951
Point (7, 7) Assigned to Cluster 0 with distance 0.0
Point (-3, -3) Assigned to Cluster 2 with distance 1.9436506316151003
Point (-2, -4) Assigned to Cluster 2 with distance 2.1081851067789197
Point (-7, -7) Assigned to Cluster 2 with distance 3.8005847503304597
Cluster 0 points: [(8, 8), (6, 6), (7, 7)] -> New Centroid: (7.0, 7.0)
Cluster 1 points: [(1, 2), (3, 3), (2, 2)] -> New Centroid: (2.0, 2.3333333333333335)
Cluster 2 points: [(-3, -3), (-2, -4), (-7, -7)] -> New Centroid: (-4.0, -4.666666666666667)
convergence, stop
Cluster 0 (Centroid (7.0, 7.0)): [(8, 8), (6, 6), (7, 7)]
Cluster 1 (Centroid (2.0, 2.3333333333333335)): [(1, 2), (3, 3), (2, 2)]
Cluster 2 (Centroid (-4.0, -4.666666666666667)): [(-3, -3), (-2, -4), (-7, -7)]

--- Evaluating T5 Results ---
```

```
=== Clustering Evaluation Metrics ===
Within-Cluster Sum of Squares (WCSS): 29.3333
Silhouette Score: 0.6708
Inter-Cluster Distance:
  - Average Distance: 10.6979
```

T6

```
# For T6
print("T6 Answer")

all_points = [
    (1, 2), (3, 3), (2, 2), (8, 8), (6, 6), (7, 7), (-3,-3), (-2, -4), (-7,-7)
]

starting_centroids = [
    (-3, -3),
    (2, 2),
    (-7,-7)
]

final_centroids, final_clusters = k_means(all_points, starting_centroids)

for i, cluster in enumerate(final_clusters):
    print(f"Cluster {i} (Centroid {final_centroids[i]}): {cluster}")

print("\n--- Evaluating T6 Results ---")
evaluate_clustering(final_centroids, final_clusters)
```

```
T6 Answer
Starting Centroids: [(-3, -3), (2, 2), (-7, -7)]
Data Points: [(1, 2), (3, 3), (2, 2), (8, 8), (6, 6), (7, 7), (-3, -3), (-2, -4), (-7, -7)]

Iteration: 1
Point (1, 2) Assigned to Cluster 1 with distance 1.0
Point (3, 3) Assigned to Cluster 1 with distance 1.4142135623730951
Point (2, 2) Assigned to Cluster 1 with distance 0.0
Point (8, 8) Assigned to Cluster 1 with distance 8.48528137423857
Point (6, 6) Assigned to Cluster 1 with distance 5.656854249492381
Point (7, 7) Assigned to Cluster 1 with distance 7.0710678118654755
Point (-3, -3) Assigned to Cluster 0 with distance 0.0
Point (-2, -4) Assigned to Cluster 0 with distance 1.4142135623730951
Point (-7, -7) Assigned to Cluster 2 with distance 0.0
Cluster 0 points: [(-3, -3), (-2, -4)] -> New Centroid: (-2.5, -3.5)
Cluster 1 points: [(1, 2), (3, 3), (2, 2), (8, 8), (6, 6), (7, 7)] -> New Centroid: (4.5, 4.66666666666
Cluster 2 points: [(-7, -7)] -> New Centroid: (-7.0, -7.0)


Iteration: 2
Point (1, 2) Assigned to Cluster 1 with distance 4.400126260814695
Point (3, 3) Assigned to Cluster 1 with distance 2.242270674512285
Point (2, 2) Assigned to Cluster 1 with distance 3.655285366576885
```

```
Point (8, 8) Assigned to Cluster 1 with distance 4.833333333333333
Point (6, 6) Assigned to Cluster 1 with distance 2.0069324297987157
Point (7, 7) Assigned to Cluster 1 with distance 3.419714088113865
Point (-3, -3) Assigned to Cluster 0 with distance 0.7071067811865476
Point (-2, -4) Assigned to Cluster 0 with distance 0.7071067811865476
Point (-7, -7) Assigned to Cluster 2 with distance 0.0
Cluster 0 points: [(-3, -3), (-2, -4)] -> New Centroid: (-2.5, -3.5)
Cluster 1 points: [(1, 2), (3, 3), (2, 2), (8, 8), (6, 6), (7, 7)] -> New Centroid: (4.5, 4.66666666666
Cluster 2 points: [(-7, -7)] -> New Centroid: (-7.0, -7.0)
convergence, stop
Cluster 0 (Centroid (-2.5, -3.5)): [(-3, -3), (-2, -4)]
Cluster 1 (Centroid (4.5, 4.666666666666667)): [(1, 2), (3, 3), (2, 2), (8, 8), (6, 6), (7, 7)]
Cluster 2 (Centroid (-7.0, -7.0)): [(-7, -7)]

--- Evaluating T6 Results ---
=== Clustering Evaluation Metrics ===
Within-Cluster Sum of Squares (WCSS): 77.8333
Silhouette Score: 0.5028
Inter-Cluster Distance:
  - Average Distance: 10.9462
  - Minimum Separation: 5.7009
Average Distance from Centroid:
  - Global Average: 2.4413
  - Cluster 0 Average: 0.7071
  - Cluster 1 Average: 3.4263
  - Cluster 2 Average: 0.0000
{'wcss': np.float64(77.83333333333333),
 'silhouette': np.float64(0.502846610880241),
 'avg_inter_cluster': np.float64(10.946247780867566),
 'global_avg_dist': np.float64(2.441319523946986)}
```

OT2

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

def plot_k_means_elbow(data_points, max_k):
    X = np.array(data_points)
    wcss = []
    K = range(1, max_k + 1)

    for k in K:
        kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42, n_init=10)
        kmeans.fit(X)
        wcss.append(kmeans.inertia_)

    plt.figure(figsize=(8, 4))
    plt.plot(K, wcss, 'bo-')
    plt.xlabel('k')
```

```
    plt.ylabel('WCSS')
    plt.title('Elbow Method')
    plt.show()


all_points = [
    (1, 2), (3, 3), (2, 2), (8, 8), (6, 6), (7, 7), (-3,-3), (-2, -4), (-7,-7)
]
```