> Name: Suyash Katkam

> Class: TE9-B-25

> Subject: DWM

> EXPERIMENT NO. 4

> Title: Implementation of Clustering algorithm (K-means/Agglomerative)

```
import random
import matplotlib.pyplot as plt

def k_means_clustering():
    # Step 1: Accept user input
    data = list(map(float, input("Enter numbers separated by spaces: ").split()))
    k = int(input("Enter the number of clusters (k): "))

    # Step 2: Initialize cluster means randomly
    means = random.sample(data, k)
    print(f"Initial means: {means}")

    iteration = 0
    while True:
        iteration += 1
        print(f"Iteration {iteration}:")

        # Step 3: Assign each data point to the nearest mean
        clusters = {i: [] for i in range(k)}
        for point in data:
            distances = [abs(point - mean) for mean in means]
            cluster_index = distances.index(min(distances))
            clusters[cluster_index].append(point)

        # Step 4: Calculate new means
        new_means = []
        for i in range(k):
            if clusters[i]:
                new_means.append(sum(clusters[i]) / len(clusters[i]))
            else:
                new_means.append(means[i])  # Keep the same mean if the cluster is empty

        print(f"Clusters: {clusters}")
        print(f"Updated means: {new_means}")

        # Step 5: Check for exact match in means
        if new_means == means:
            print("Exact same means achieved. Clustering complete.")
            break
        means = new_means

    print("Final clusters:")
    for i in range(k):
        print(f"Cluster {i + 1}: {clusters[i]}")

    # Visualization
    plt.figure(figsize=(8, 6))
    for i, cluster in clusters.items():
        plt.scatter(cluster, [i + 1] * len(cluster), label=f'Cluster {i + 1}')
    plt.scatter(means, range(1, k + 1), color='red', marker='x', label='Means', s=100)
    plt.title("K-Means Clustering")
    plt.xlabel("Data Points")
    plt.ylabel("Clusters")
    plt.legend()
    plt.grid()
    plt.show()

if __name__ == "__main__":
    k_means_clustering()
```

```
Enter numbers separated by spaces: 15 30 45 60 75 90 105
Enter the number of clusters (k): 4
Initial means: [45.0, 60.0, 15.0, 105.0]
Iteration 1:
Clusters: {0: [30.0, 45.0], 1: [60.0, 75.0], 2: [15.0], 3: [90.0, 105.0]}
Updated means: [37.5, 67.5, 15.0, 97.5]
Iteration 2:
Clusters: {0: [30.0, 45.0], 1: [60.0, 75.0], 2: [15.0], 3: [90.0, 105.0]}
Updated means: [37.5, 67.5, 15.0, 97.5]
Exact same means achieved. Clustering complete.
Final clusters:
Cluster 1: [30.0, 45.0]
Cluster 2: [60.0, 75.0]
Cluster 3: [15.0]
Cluster 4: [90.0, 105.0]
```