In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
#load the iris dataset
from sklearn.datasets import load_iris
Iris = load_iris()
print(Iris)
X=Iris.data
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
```

```
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
```

```
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'targe
t_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'), 'DESCR': '.. _
iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteris
tics:**\n\n:Number of Instances: 150 (50 in each of three classes)\n:Number of Attri
butes: 4 numeric, predictive attributes and the class\n:Attribute Information:\n
- sepal length in cm\n    - sepal width in cm\n    - petal length in cm\n    - petal
width in cm\n    - class:\n            - Iris-Setosa\n            - Iris-Versicolour
\n            - Iris-Virginica\n\n:Summary Statistics:\n\n============== ==== ==== =
====== ===== ====================\n                Min  Max   Mean    SD   Class Cor
relation\n============== ==== ==== ======= ===== ====================\nsepal length:
4.3  7.9   5.84   0.83    0.7826\nsepal width:    2.0  4.4   3.05   0.43   -0.4194\n
petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\npetal width:    0.1  2.5
```

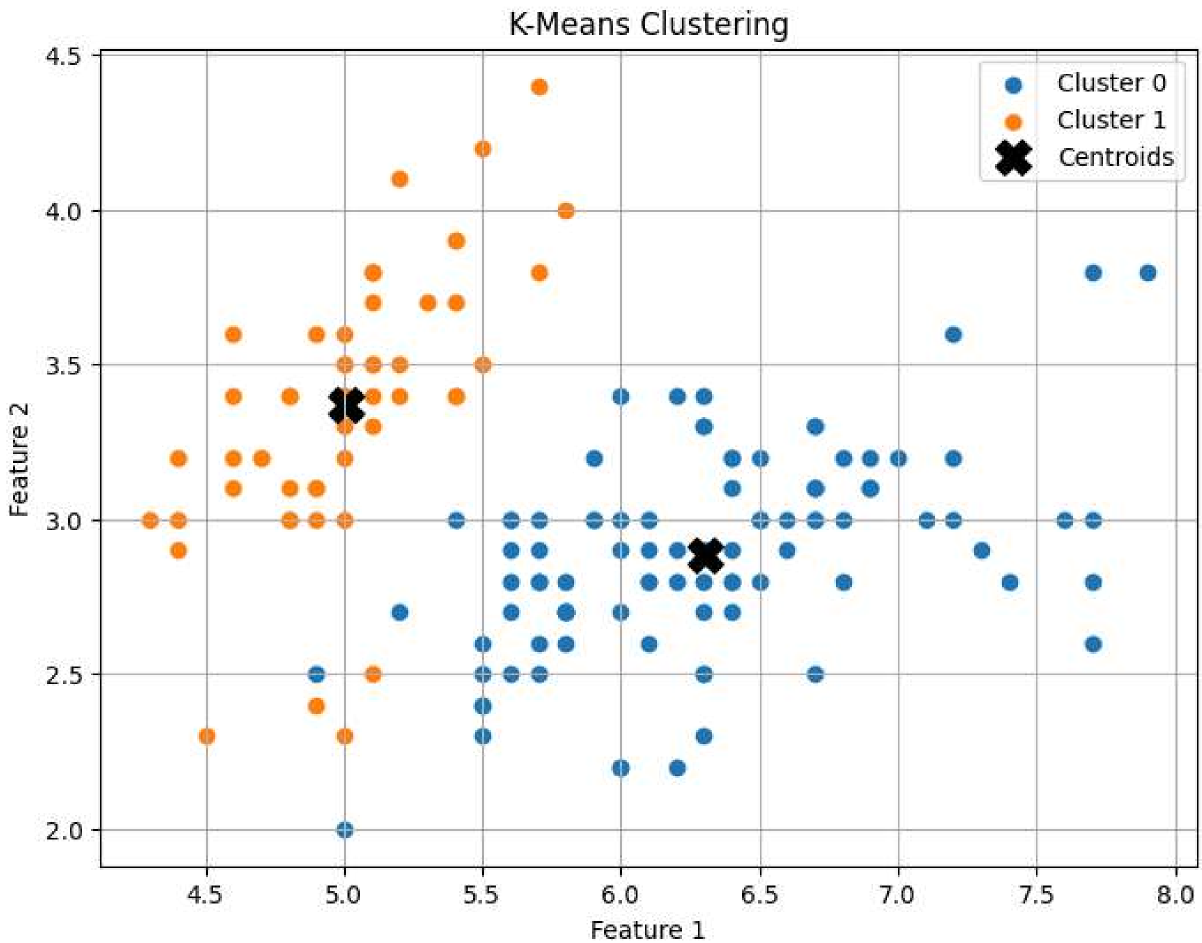
```
1.20   0.76    0.9565  (high!)\n============== ==== ==== ======= ===== =============
=======\n\n:Missing Attribute Values: None\n:Class Distribution: 33.3% for each of 3
classes.\n:Creator: R.A. Fisher\n:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.
gov)\n:Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher.
The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but no
t as in the UCI\nMachine Learning Repository, which has two wrong data points.\n\nTh
is is perhaps the best known database to be found in the\npattern recognition litera
ture.  Fisher\'s paper is a classic in the field and\nis referenced frequently to th
is day.  (See Duda & Hart, for example.)  The\ndata set contains 3 classes of 50 ins
tances each, where each class refers to a\ntype of iris plant.  One class is linearl
y separable from the other 2; the\nlatter are NOT linearly separable from each othe
r.\n\n.. dropdown:: References\n\n  - Fisher, R.A. "The use of multiple measurements
in taxonomic problems"\n    Annual Eugenics, 7, Part II, 179-188 (1936); also in "Co
ntributions to\n    Mathematical Statistics" (John Wiley, NY, 1950).\n  - Duda, R.
O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n    (Q327.D83) J
ohn Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n  - Dasarathy, B.V. (1980) "N
osing Around the Neighborhood: A New System\n    Structure and Classification Rule f
or Recognition in Partially Exposed\n    Environments".  IEEE Transactions on Patter
n Analysis and Machine\n    Intelligence, Vol. PAMI-2, No. 1, 67-71.\n  - Gates, G.
W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n    on Informatio
n Theory, May 1972, 431-433.\n  - See also: 1988 MLC Proceedings, 54-64.  Cheeseman
et al"s AUTOCLASS II\n    conceptual clustering system finds 3 classes in the dat
a.\n  - Many, many more ...\n', 'feature_names': ['sepal length (cm)', 'sepal width
(cm)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data_modul
e': 'sklearn.datasets.data'}
```

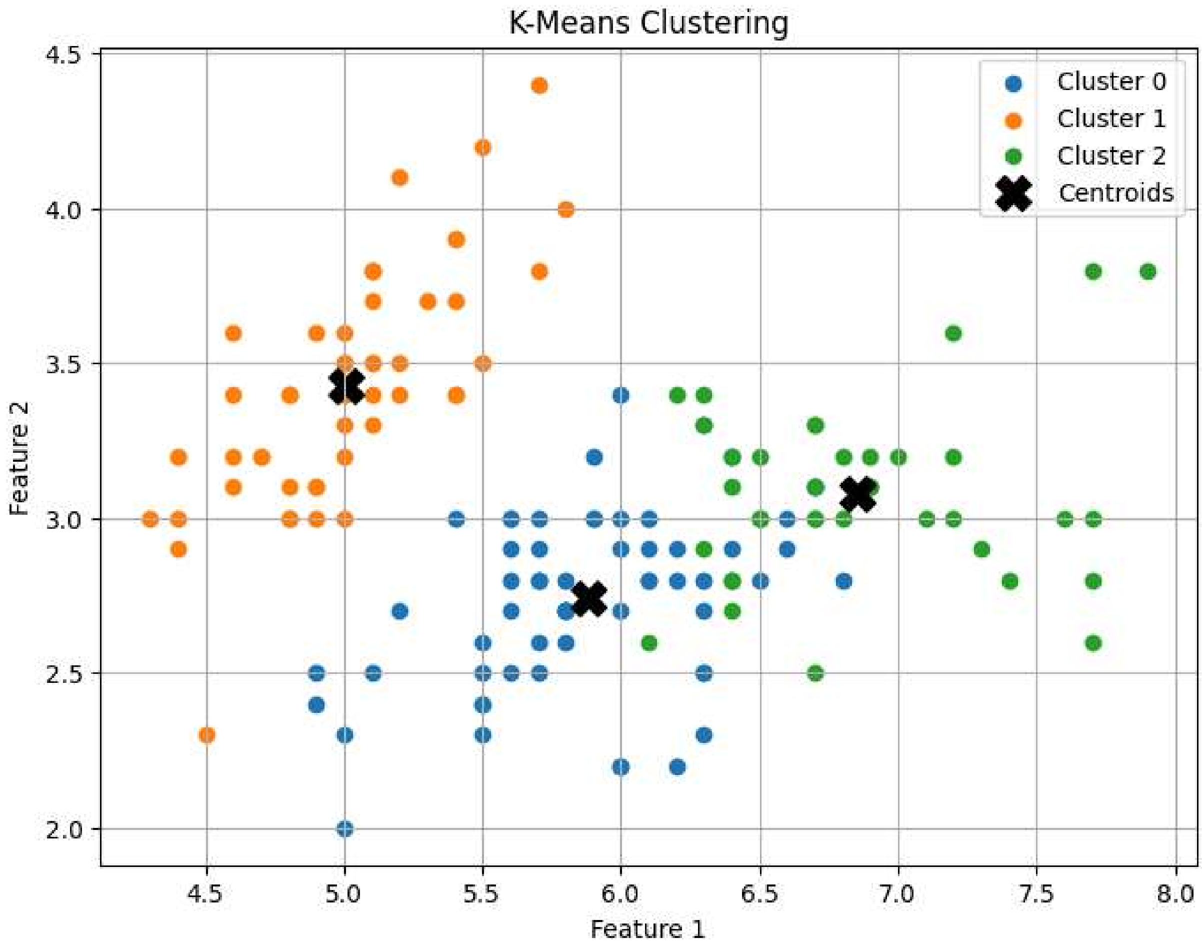

- K=3 is the best choice because the clusters are more compact and well seperated. With k=2 and k=4 the clustering is either too lax or over-segmented. Moreover, the dataset has three target classes: setosa, versicolor, and virginica, so grouping them into three clusters makes the most sense.

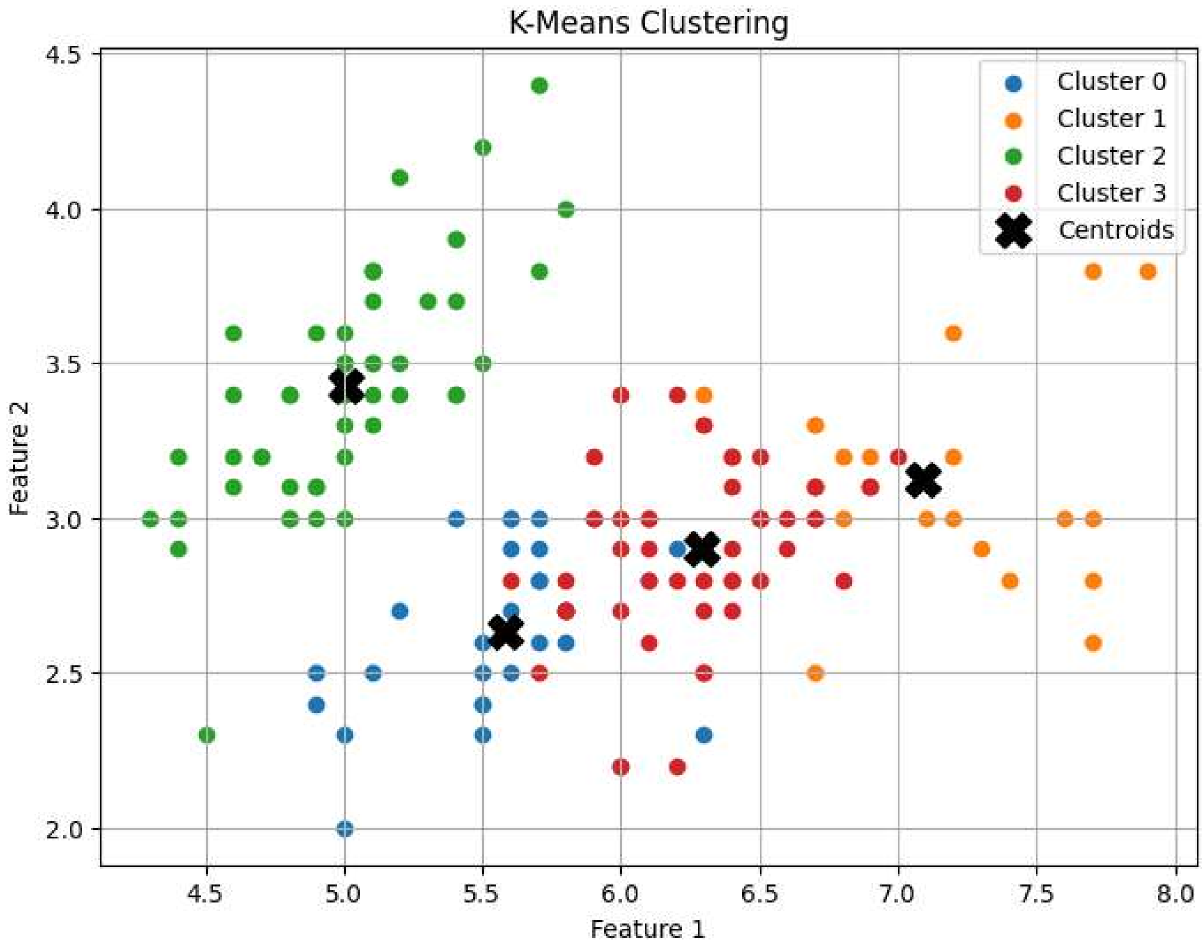
In [14]:

```python
#assign each point to the nearest centroid
def assign_clusters(X, centroids):
    clusters = np.zeros(X.shape[0], dtype=int)
    for i in range(X.shape[0]):
        #calculate the distance from each point to each centroid
        distances = np.sqrt(np.sum((centroids - X[i]) ** 2, axis=1))
        clusters[i] = np.argmin(distances)
    return clusters

#k-mean function
def kmeans(X, k, max_iterations=100):
    centroids = X[np.random.choice(X.shape[0], size=k, replace=False)]
    #assign the points to clusters
    for _ in range(max_iterations):
        clusters = assign_clusters(X, centroids)
        #update the centroids
        new_centroids = new_centroids = np.zeros((k, X.shape[1]))
        for i in range(k):
            new_centroids[i] = np.mean(X[clusters == i], axis=0)
        centroids = new_centroids
    return clusters, centroids

#display the clusters and centroids
```

```
def visualize_clusters(X, clusters, centroids):
    plt.figure(figsize=(8, 6))
    #plot each cluster with a different color
    for cluster_id in np.unique(clusters):
        plt.scatter(X[clusters == cluster_id, 0], X[clusters == cluster_id, 1], lab
    #plot centroids
    plt.scatter(centroids[:, 0], centroids[:, 1], c='black', marker='X', s=200, lab
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('K-Means Clustering')
    plt.legend()
    plt.grid(True)
    plt.show()

#run the k-means algorithm and visualize the clusters
for i in range(2,5):
    clusters, centroids = kmeans(X, k=i)
    visualize_clusters(X, clusters, centroids)
```




K-Means Clustering
Cluster 0
Cluster 1
Cluster 2
Centroids
Feature 1
Feature 2
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


K-Means Clustering
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Centroids
Feature 1
Feature 2
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
2.0
2.5
3.0
3.5
4.0
4.5