- Dataset can be found at [Mall Customer Segmentation Data (https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python)](https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python)
- More about K-Means clustering at [KMeans (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)

In [1]:
```
import numpy as np
import pandas as pd

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

In [2]:
```
df = pd.read_csv("Mall_Customers.csv")
df.head()
```

Out[2]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

What does the dataset contain ?

The data is about customers like Customer ID, age, gender, annual income and spending score. Spending Score is assigned to the customer based on a defined parameters like customer behavior and purchasing data.

In [3]:
```
df.drop(columns=["CustomerID"], inplace=True)
df['Gender'] = df['Gender'].apply(lambda x: 1 if x == "Male" else 0)
df.head()
```

Out[3]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15	39
1	1	21	15	81
2	0	20	16	6
3	0	23	16	77
4	0	31	17	40

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Gender                  200 non-null    int64
 1   Age                     200 non-null    int64
 2   Annual Income (k$)      200 non-null    int64
 3   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4)
memory usage: 6.4 KB
```

What is the algorithm

```
K-Means clustering is an unsupervised algorithm used to divide the data into `K` number of clusters based on
the distance of the each point from the mean (could be arithmetric mean) from the centroid of the cluster. In
other words, the algorithm uses the distance between points as a measure of similarity, based on k averages
(i.e. means)
```

How does it work

```
The idea behind k-Means is that, we want to add k new points to the data we have.
Each one of those points — called a Centroid — will be going around trying to center itself in the middle of
one of the k clusters we have. Once those points stop moving, our clustering algorithm stops.

1. Choose random K points from the data and mark them as centroid of each cluster (K clusters).
2. For rest of the points: Assign each point to the cluster whose centroid is the closest. The closest
centroid is the one whose distance to the point is closer than any other centroid.
3. Once we assign all the points with a cluster, we calcuate the mean of all the points in a cluster and
choose that mean point as the new centroid. Hence we choose K new centroids.
4. Follow the same process (step 2 and 3) until given number of iterations or until the centroids converge
```

Advantages and Disadvantges of the algorithm

```
Advantages:
* Faster than other clustering algorithms even though the data is large (if K is small)
* Produces tighter clusters than other clustering algorithms

Disadvantages:
* It is often difficult to estimate the best K-value
* Doesn't work well with different clusters sizes and densities
* Different initialization of the centroids lead to different clusters
```

How is it performed on the dataset

In [5]:
```
X = df.values
```

In [6]:
```
X.shape
```

Out[6]:
```
(200, 4)
```

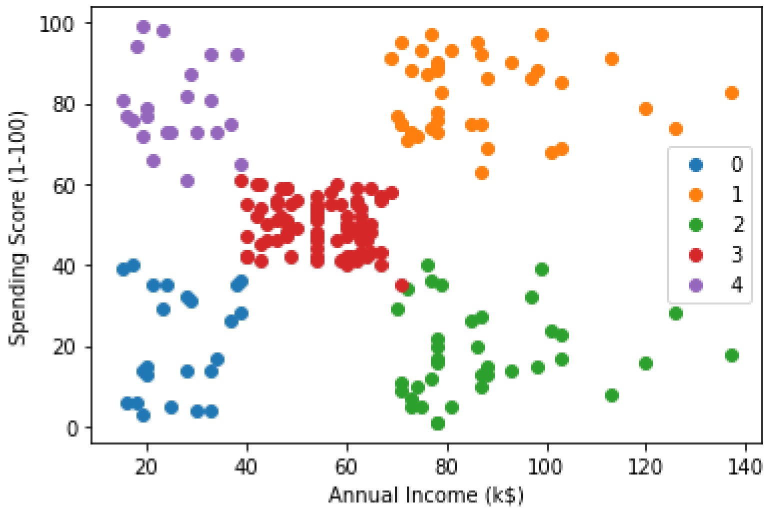
In [7]:
```
algo = KMeans(n_clusters=5, random_state=42)
algo.fit(X)
labels = algo.labels_
centroids = algo.cluster_centers_
df['Clusters'] = labels
```

In [8]:
```
df.head()
```

Out[8]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Clusters
0	1	19	15	39	0
1	1	21	15	81	4
2	0	20	16	6	0
3	0	23	16	77	4
4	0	31	17	40	0

In [9]:
```
for i in np.unique(labels):
    label = df['Clusters']
    dx = df[label == i]
    plt.scatter(dx.iloc[:, 2], dx.iloc[:, 3], label=i)
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show();
```



Summary

- K-Means clustering with 5 clusters is found to be optimal
- Annual Income and Spending Score are found to be the important features
- Fitting the model with just these 2 features might result in better clusters

In [ ]: