You are owing a supermarket mall and through membership cards, you have some basic data about your cus
and spending score. Spending Score is something you assign to the customer based on your defined parame

```
import numpy as np #linear algebra library of Python
import pandas as pd # build on top of numpy for data analysis, data manipulation and d
import matplotlib.pyplot as plt #plotting library of Python
```

Now let's mount Google drive so that we can upload the diabetes.csv file. You can find the code in the 'Code s

```
from google.colab import drive
drive.mount('/content/gdrive')
```

First thing that we do is take a look at the shape of the dataframe (df.shape) and take a look at first 5 lines th

```
df=pd.read_csv('/content/gdrive/My Drive/Colab Notebooks/Mall_Customers.csv') #import
df.head() #shows first 5 lines including column namesdf.shape # number of rows and col
```



```
df.shape # provides # rows and # columns of the dataframe df - 200 rows and 5 columns
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, ca
```

```
X=df.iloc[:, [3,4]].values # selects columns 3 and 4 of the dataframe (2 last colums)
```

Now we are ready for K-means and we will start by finding the best k. This is the so-called elbow method. Bas
k. As k increases the cost will decrease. The for loop will fit the kmeans algorithm to our data and will compu
appended to our wcss list

Parameters kmeans:

init: 'random': random initiualization method - choose k observations (rows) at random from data for the initia

max_iter: maximum number of iterations of the k-means algorithm for a single run to converge

n_init: number of time the k-means algorithm will be run with different centroid seeds. The final results will be
inertia.

```
from sklearn.cluster import KMeans
wcss=[] # wccs is 'within cluster sum of squares'
for i in range(1,11):
    kmeans=KMeans(n_clusters=i, init='random', max_iter=300, n_init=10, random_state=4
    kmeans.fit(X) # fits the modelcompute k-means clustering
    wcss.append(kmeans.inertia_) # sum of squared distances of samples to their closes

plt.plot(range(1,11),wcss)
plt.title('The Elbow Method Graph')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Based on above graph (k vs. Cost) we suggest to take k=5 (5 centroids meaning 5 clusters)

```
model=KMeans(n_clusters=5,init='random', max_iter=300, n_init=10, random_state=42)
y_kmeans=model.fit_predict(X) # output is a vector with all samples and their cluster
```

```
plt.figure(figsize=(8,5))
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Low
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Hi
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'L
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Lo
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label =

plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend(loc=0)
```

```
plt.show()
```

X[y_kmeans == 0, 0] means outcome is cluster index 0 for column 0