K means Clustering

K means clustering is a very vast topic and is a part of Unsupervised Learning.
It's use in real life can be explained by a real life example:
Dominoes wants to open a new outlet in your city. They already have two outlets in the city. They want to decide where to open it. They have the geographical locations of all their previous customers. This problem can be solved using clustering.
They will pinpoint all the locations of their customers on the map. These locations will become the data points on an XY plane. They will form clusters and then they will decide which cluster needs fulfillment based on the distance of the data points from the cluster center. The final cluster center will become the new location of their store.

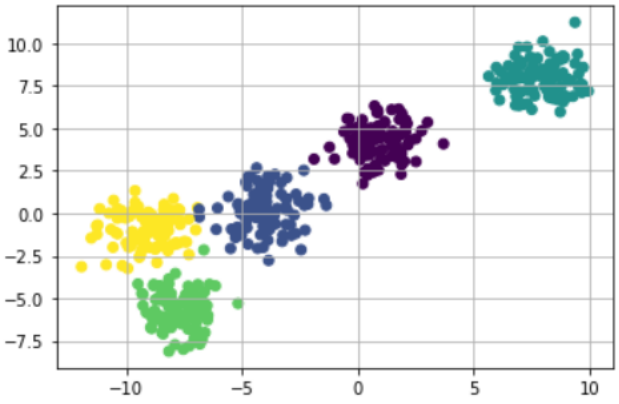
https://www.geeksforgeeks.org/k-means-clustering-introduction/
https://www.geeksforgeeks.org/ml-k-means-algorithm/

These two articles of GFG are more than sufficient to tackle the problem of clustering.
The added issue of implying K means using scikit learn should be tackled at the end of the video lecture.

Dataset Preperation

```
X,y = make_blobs(n_samples = 500,n_features = 2,centers = 5,random_state = 3)
```

```
plt.figure(0)
plt.grid(True)
plt.scatter(X[:,0],X[:,1],c=y)
plt.show()
```



```
km = KMeans(n_clusters = 5)
km.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_it
       n_clusters=5, n_init=10, n_jobs=None, precompute_distar
       random_state=None, tol=0.0001, verbose=0)
```

```
centers = km.cluster_centers_
labels = km.labels_
```

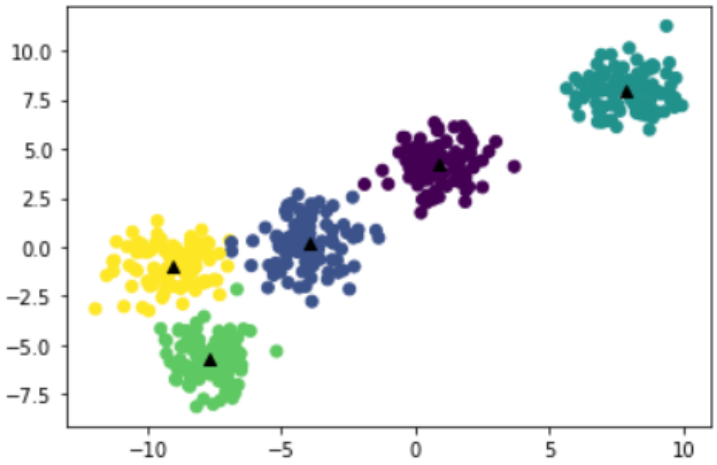
```
print(centers)
```

```
[[ 7.8649674   7.97988109]
 [-7.6726196  -5.69211059]
 [-3.97953526  0.19458336]
 [ 0.88641907  4.19441853]
 [-9.0671623  -0.96740759]]
```

```
plt.scatter(X[:,0],X[:,1],c=y)

plt.scatter(centers[:,0],centers[:,1],color = 'black',marker = '^')

plt.show()
```



K means algorithm can also be used to perform image segmentation.
The example is as follows.
----------X----------X
To be noted, the below given example implies some use of the library OpenCV to convert the colour scheme of an image to RGB. This example is entirely optional and can be chosen to be ignored by the tutor, based on the response of the class.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

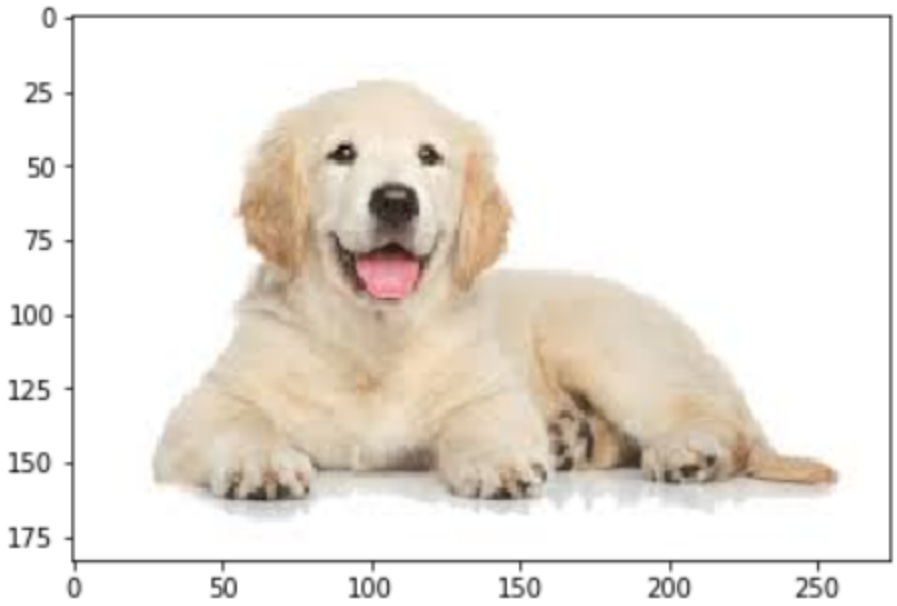
```
img = cv2.imread('dog.png')
```

```
original_shape = img.shape
print(original_shape)  #3 DEnotes the c
```

```
(183, 275, 3)
```

```
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)  #We convert t
```

```
plt.imshow(img)
plt.show()
```



```
#Now you flatten the pixels into a linear array
pixel_vals = img.reshape(-1,3)
```

```
print(pixel_vals.shape)
```

```
(50325, 3)
```

```
from sklearn.cluster import KMeans
```

```
dominant_colors = 10

km = KMeans(n_clusters = dominant_colors)
km.fit(pixel_vals)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=10, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```
centers = km.cluster_centers_
#shows the RGB values of the most dominant colour forms
```

```
centers = np.array(centers,dtype = 'int')
print(centers)
```

```
[[254 254 254]
 [220 203 180]
 [143 121 102]
 [227 214 195]
 [197 170 144]
 [ 66  50  39]
 [235 227 216]
 [106  86  72]
 [209 188 163]
 [175 148 122]]
```

Plot all these colours

```
i = 1
plt.figure(0,figsize=(8,2))
colors = []
for each_color in centers:
    plt.subplot(1,10,i)
    plt.axis("off")
    colors.append(each_color)
    i+=1
    #color switch
    a = np.zeros((100,100,3), dtype = 'int')
    a[:,:,:] = each_color
    plt.imshow(a)
plt.show()
```