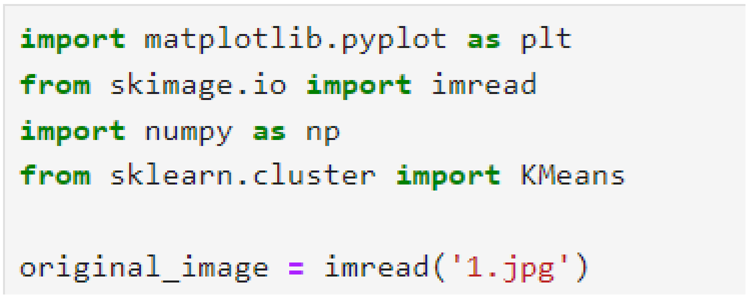
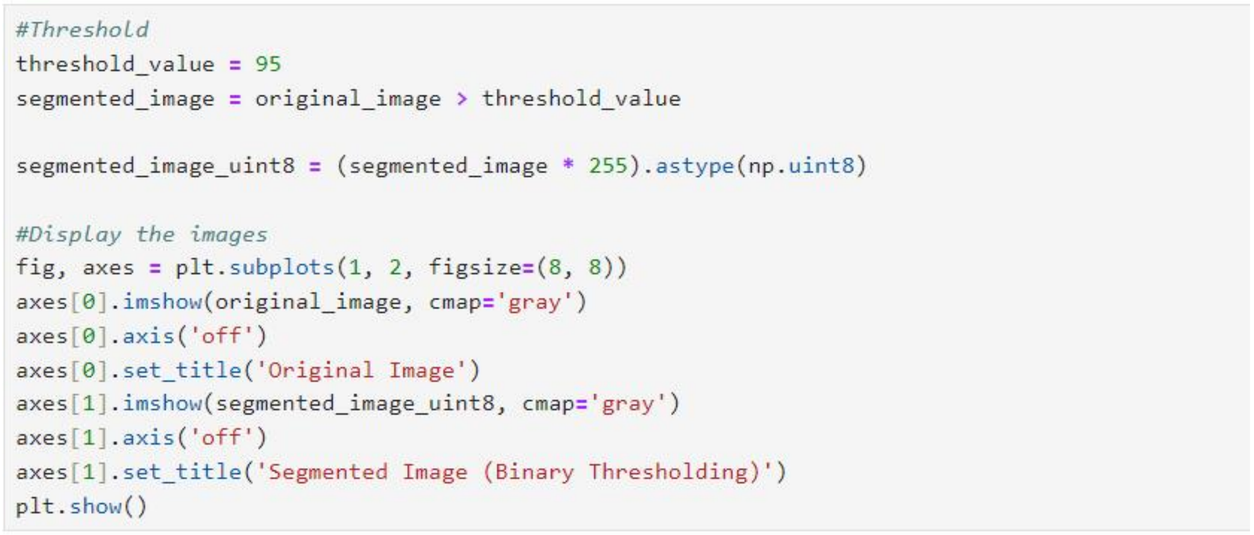
At the beginning, I've imported the necessary libraries for reading and showing the images, and for two segmentation methods (Binary thresholding, and K-means).

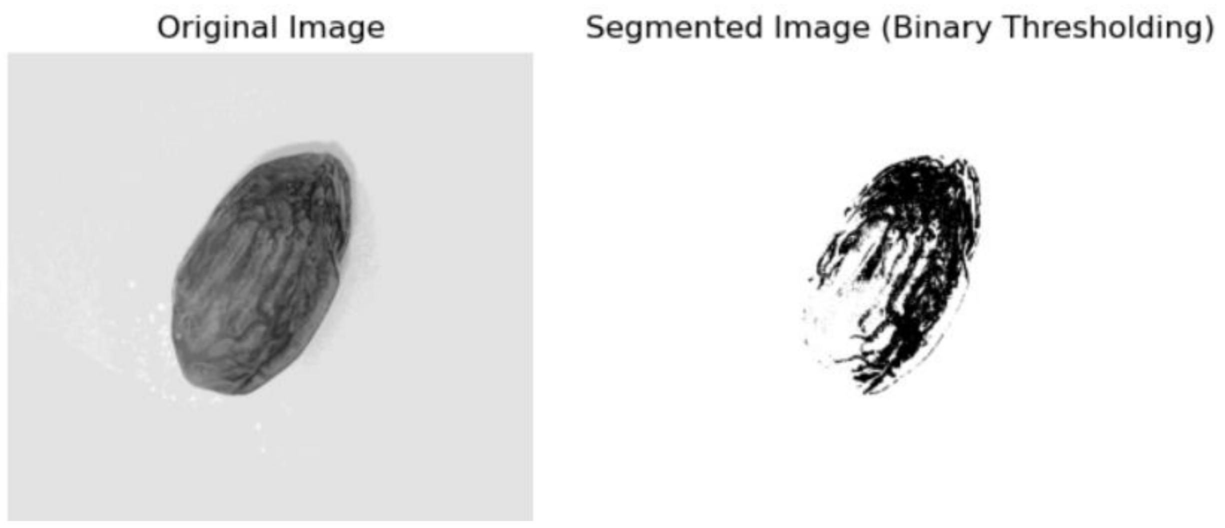


```
import matplotlib.pyplot as plt
from skimage.io import imread
import numpy as np
from sklearn.cluster import KMeans

original_image = imread('1.jpg')
```

First, applying the first method (**Binary thresholding**)



```
#Threshold
threshold_value = 95
segmented_image = original_image > threshold_value

segmented_image_uint8 = (segmented_image * 255).astype(np.uint8)

#Display the images
fig, axes = plt.subplots(1, 2, figsize=(8, 8))
axes[0].imshow(original_image, cmap='gray')
axes[0].axis('off')
axes[0].set_title('Original Image')
axes[1].imshow(segmented_image_uint8, cmap='gray')
axes[1].axis('off')
axes[1].set_title('Segmented Image (Binary Thresholding)')
plt.show()
```



Basically, the code consists of 3 lines only, the rest are for showing the images

```
#Threshold
threshold_value = 95
segmented_image = original_image > threshold_value

segmented_image_uint8 = (segmented_image * 255).astype(np.uint8)
```

Here I’ve chosen 95 as a value for threshold, this will classify which pixel will be a background and which as foreground.

Then, if the pixel value > 95, it will be True, (Foreground)

And if the pixel value <= 95, it will be False, (Background)

Then the Boolean array will become an integer array by converting:

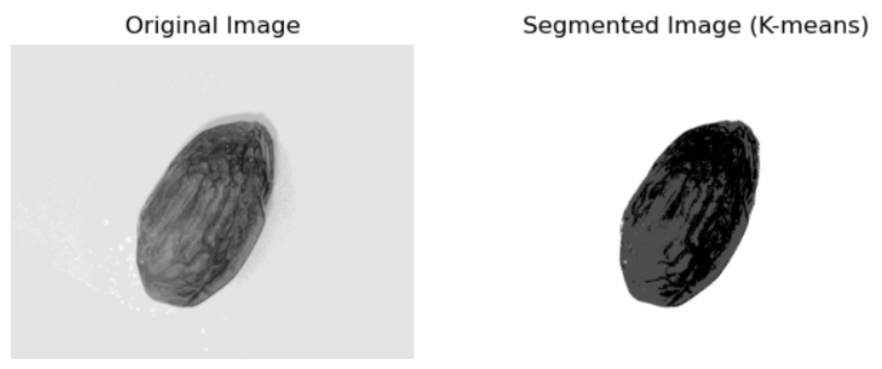
True = 255 “White”, False= 0 “Black”

Finally, the original image and the segmented image will be shown

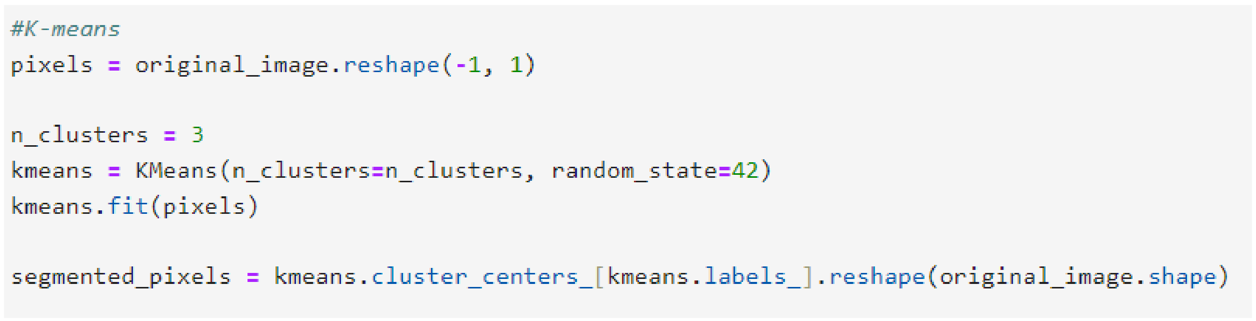
---

Second, applying the first method (**K-means**)

```
#K-means
pixels = original_image.reshape(-1, 1)

n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
kmeans.fit(pixels)

segmented_pixels = kmeans.cluster_centers_[kmeans.labels_].reshape(original_image.shape)

# Display the images
fig, axes = plt.subplots(1, 2, figsize=(8, 8))
axes[0].imshow(original_image, cmap='gray')
axes[0].axis('off')
axes[0].set_title('Original Image')
axes[1].imshow(segmented_pixels, cmap='gray')
axes[1].axis('off')
axes[1].set_title('Segmented Image (K-means)')
plt.show()
```



Like the first method, the last part was for showing the images, so let's talk about the important part in the code.



```
#K-means
pixels = original_image.reshape(-1, 1)

n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
kmeans.fit(pixels)

segmented_pixels = kmeans.cluster_centers_[kmeans.labels_].reshape(original_image.shape)
```

At first, I changed the image to be as a 2D array with one column, which is basically like

a 1D array. It is still technically 2D-Array, that's good for separating the data and the K-means expecting a 2D-Array.

Then I specified that I want to segment the image into 3 distinct clusters.

I've searched through google and I found out that putting a random_state is very important for controlling the randomness of clustering.

Then, each pixel is replaced by the color of the cluster it belongs to.

Finally, the original image and the segmented image will be shown