```
import cv2
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```

image = plt.imread('/content/img')

# normalizing image
image = image / 255.

#reshaping image because kmeans algorithm takes 2D data as an input so converted 3D image to 2D image
image_reshaped = image.reshape(image.shape[0]*image.shape[1], image.shape[2])
print("Shape of original image {}".format(image.shape))
print("Shape of reshaped image {}".format(image_reshaped.shape))

```

```
Shape of original image (1247, 2000, 3)
Shape of reshaped image (2494000, 3)
```

```

plt.axis('off')
plt.title("original image")
plt.imshow(image)

```

```
<matplotlib.image.AxesImage at 0x7fd288d23880>
```


```
# it returns clustered pixels of an image

def kmeans_over_image(number_of_clusters, org_image, reshaped_image):

  kmeans = KMeans(n_clusters=number_of_clusters, random_state=0)
  kmeans = kmeans.fit(reshaped_image)
  image_cluster = kmeans.cluster_centers_[kmeans.labels_]

  # reshaping image size form 2D to 3D
  image_cluster = image_cluster.reshape(org_image.shape[0], org_image.shape[1], org_image.shape[2])

  return image_cluster

```

```
# for storing clustered image
clustered_images = []

# number of cluster
no_cluster = []

# for eaxmple
# forming clusters of 2,4,6,8 over input image

for cluster in range(2, 10, 2):
  # image --> original 3D image
  # image_reshaped ---> reshaped 2D image

  clustered_images.append(kmeans_over_image(cluster, image, image_reshaped))
  no_cluster.append(cluster)
```

```
#plotting images
import numpy as np
```

```
 3
 4 fig=plt.figure(figsize=(8, 8))
 5 columns = 2
 6 rows = 2
 7 iterate = 0
 8
 9 for i in range(1, columns*rows +1):
10     fig.add_subplot(rows, columns, i)
11     plt.axis("off")
12     plt.title(str(no_cluster[iterate]) + " clusters")
13     plt.imshow(clustered_images[iterate])
14     iterate += 1
15 plt.show()
```