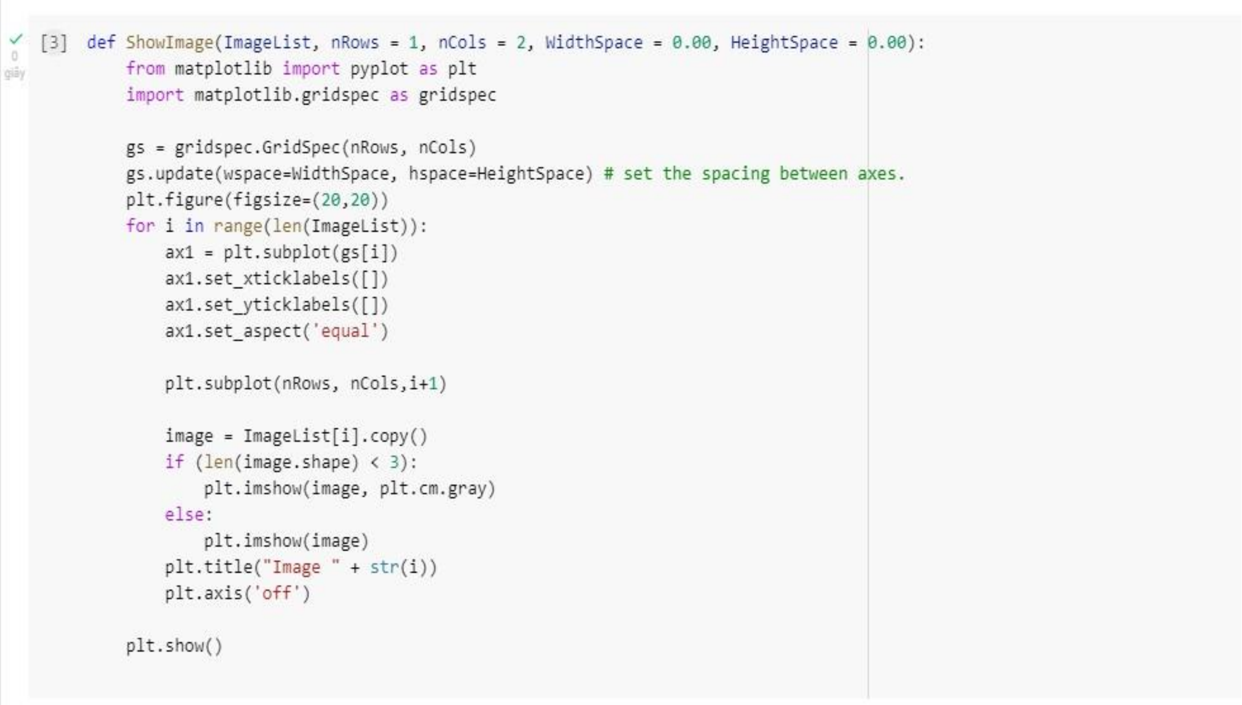
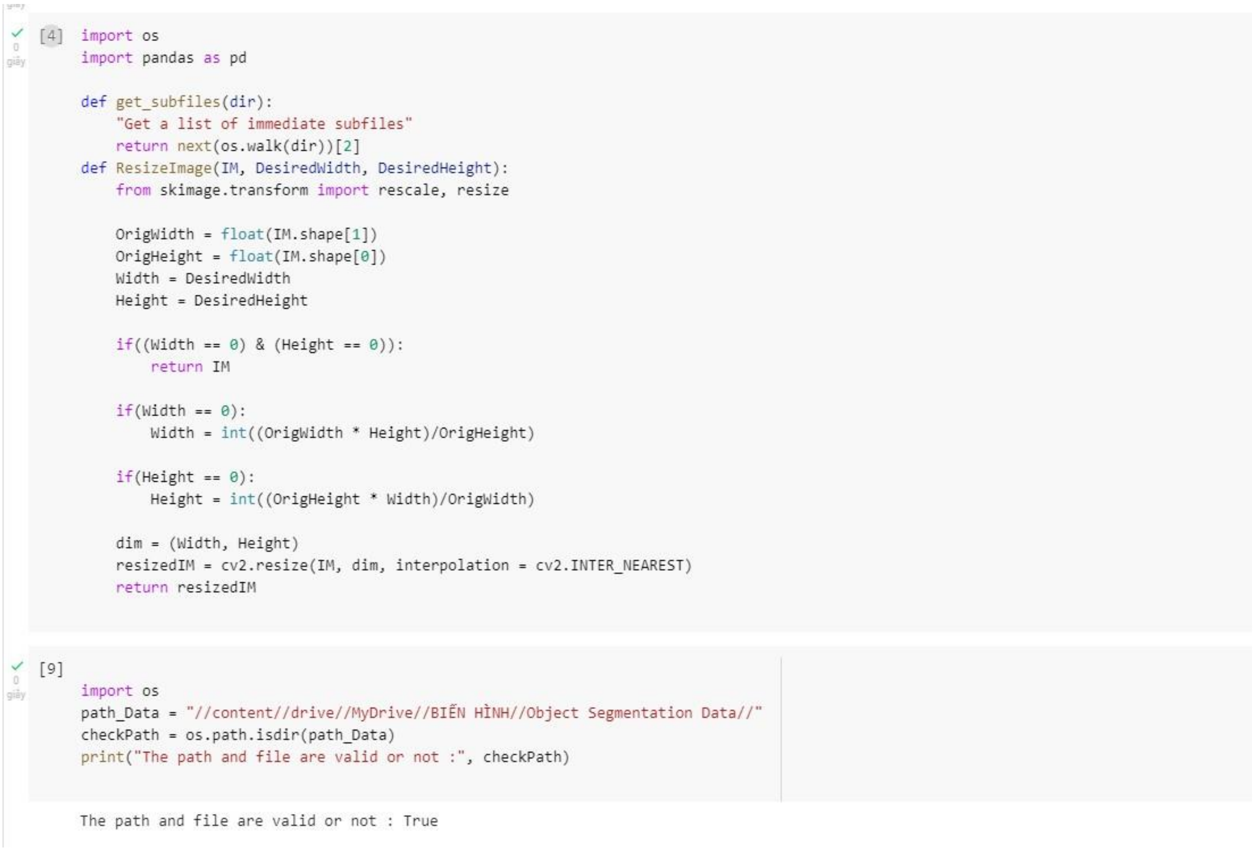


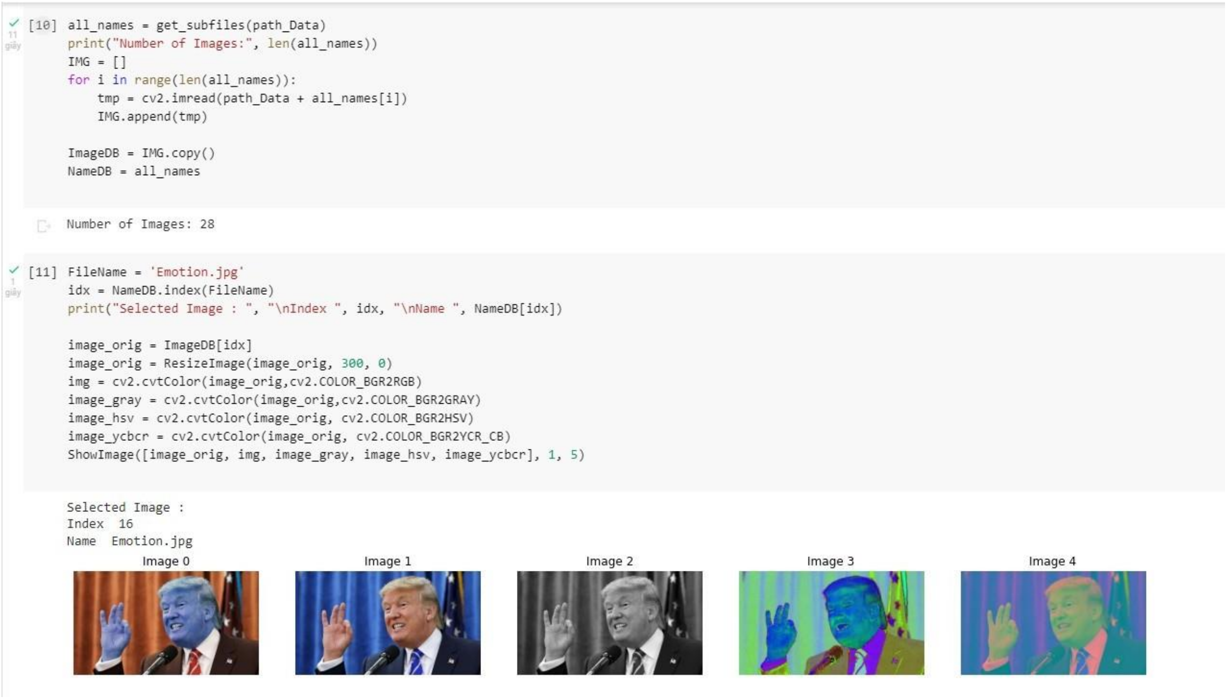
```
def ShowImage(ImageList, nRows = 1, nCols = 2, WidthSpace = 0.00, HeightSpace = 0.00):
    from matplotlib import pyplot as plt
    import matplotlib.gridspec as gridspec

    gs = gridspec.GridSpec(nRows, nCols)
    gs.update(wspace=WidthSpace, hspace=HeightSpace) # set the spacing between axes.
    plt.figure(figsize=(20,20))
    for i in range(len(ImageList)):
        ax1 = plt.subplot(gs[i])
        ax1.set_xticklabels([])
        ax1.set_yticklabels([])
        ax1.set_aspect('equal')

        plt.subplot(nRows, nCols,i+1)

        image = ImageList[i].copy()
        if (len(image.shape) < 3):
            plt.imshow(image, plt.cm.gray)
        else:
            plt.imshow(image)
        plt.title("Image " + str(i))
        plt.axis('off')

    plt.show()
```



```
[4]  import os
     import pandas as pd

     def get_subfiles(dir):
         "Get a list of immediate subfiles"
         return next(os.walk(dir))[2]
     def ResizeImage(IM, DesiredWidth, DesiredHeight):
         from skimage.transform import rescale, resize

         OrigWidth = float(IM.shape[1])
         OrigHeight = float(IM.shape[0])
         Width = DesiredWidth
         Height = DesiredHeight

         if((Width == 0) & (Height == 0)):
             return IM

         if(Width == 0):
             Width = int((OrigWidth * Height)/OrigHeight)

         if(Height == 0):
             Height = int((OrigHeight * Width)/OrigWidth)

         dim = (Width, Height)
         resizedIM = cv2.resize(IM, dim, interpolation = cv2.INTER_NEAREST)
         return resizedIM
```

```
[9]
     import os
     path_Data = "//content//drive//MyDrive//BIẾN HÌNH//Object Segmentation Data//"
     checkPath = os.path.isdir(path_Data)
     print("The path and file are valid or not :", checkPath)
```

```
     The path and file are valid or not : True
```

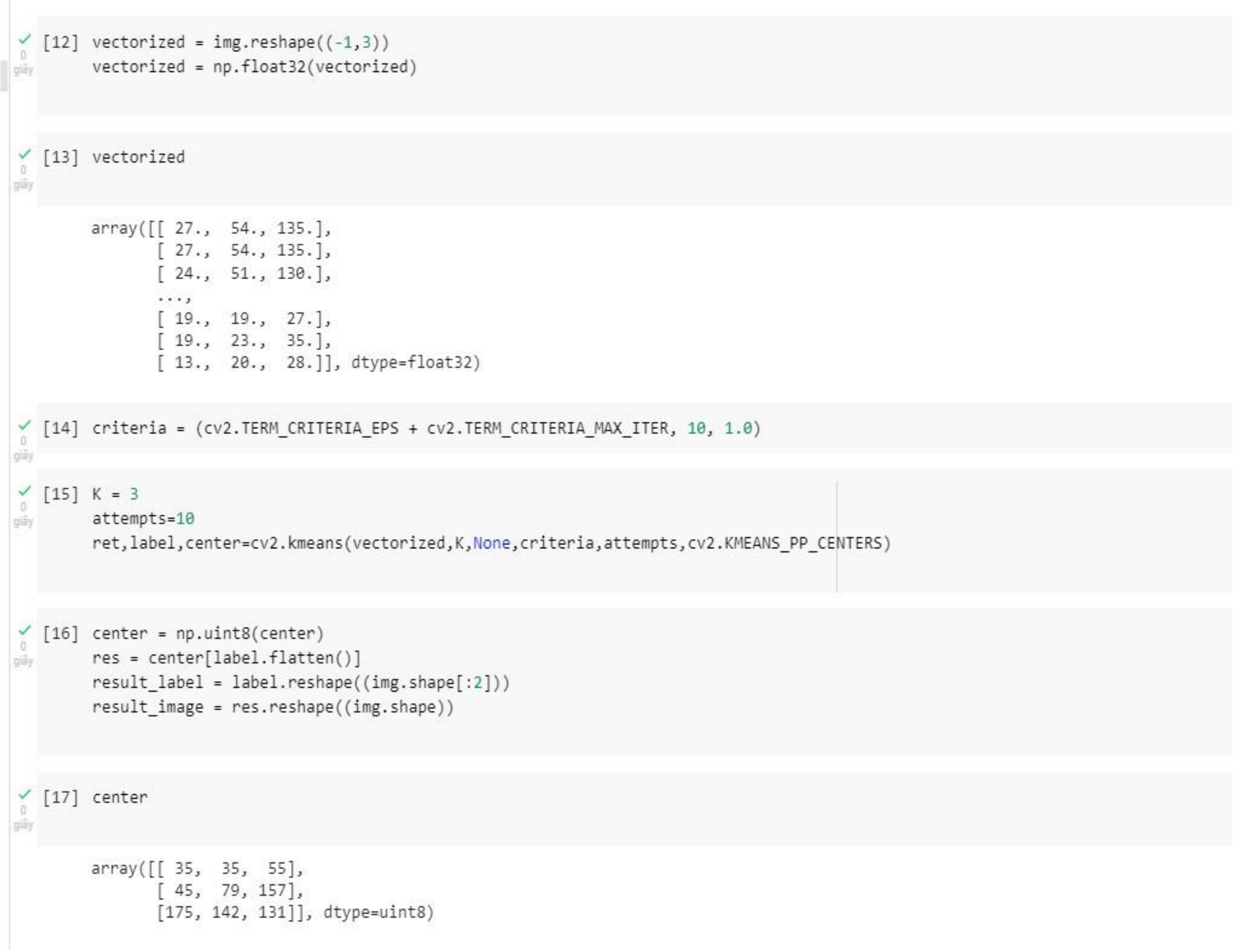


```
[10] all_names = get_subfiles(path_Data)
     print("Number of Images:", len(all_names))
     IMG = []
     for i in range(len(all_names)):
         tmp = cv2.imread(path_Data + all_names[i])
         IMG.append(tmp)

     ImageDB = IMG.copy()
     NameDB = all_names
```

```
     Number of Images: 28
```

```
[11] FileName = 'Emotion.jpg'
     idx = NameDB.index(FileName)
     print("Selected Image : ", "\nIndex ", idx, "\nName ", NameDB[idx])

     image_orig = ImageDB[idx]
     image_orig = ResizeImage(image_orig, 300, 0)
     img = cv2.cvtColor(image_orig,cv2.COLOR_BGR2RGB)
     image_gray = cv2.cvtColor(image_orig,cv2.COLOR_BGR2GRAY)
     image_hsv = cv2.cvtColor(image_orig, cv2.COLOR_BGR2HSV)
     image_ycbcr = cv2.cvtColor(image_orig, cv2.COLOR_BGR2YCR_CB)
     ShowImage([image_orig, img, image_gray, image_hsv, image_ycbcr], 1, 5)
```

```
     Selected Image :
     Index  16
     Name  Emotion.jpg
```



```
[12] vectorized = img.reshape((-1,3))
     vectorized = np.float32(vectorized)
```

```
[13] vectorized
```

```
array([[ 27.,  54., 135.],
       [ 27.,  54., 135.],
       [ 24.,  51., 130.],
       ...,
       [ 19.,  19.,  27.],
       [ 19.,  23.,  35.],
       [ 13.,  20.,  28.]], dtype=float32)
```

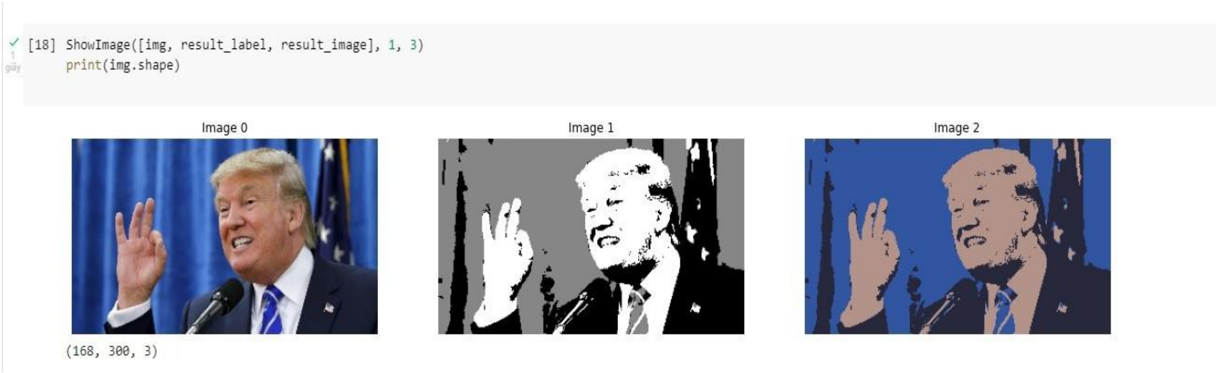
```
[14] criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
```

```
[15] K = 3
     attempts=10
     ret,label,center=cv2.kmeans(vectorized,K,None,criteria,attempts,cv2.KMEANS_PP_CENTERS)
```

```
[16] center = np.uint8(center)
     res = center[label.flatten()]
     result_label = label.reshape((img.shape[:2]))
     result_image = res.reshape((img.shape))
```

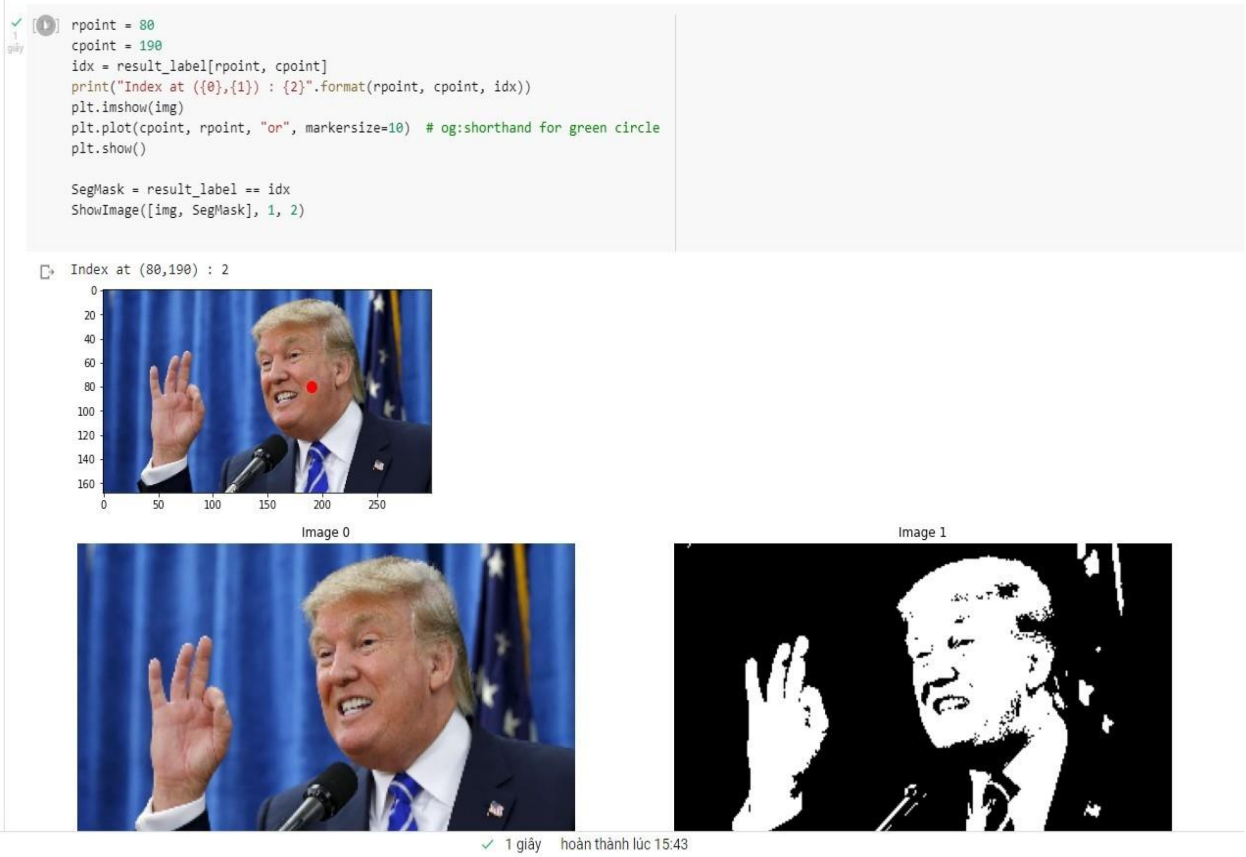
```
[17] center
```

```
array([[ 35,  35,  55],
       [ 45,  79, 157],
       [175, 142, 131]], dtype=uint8)
```



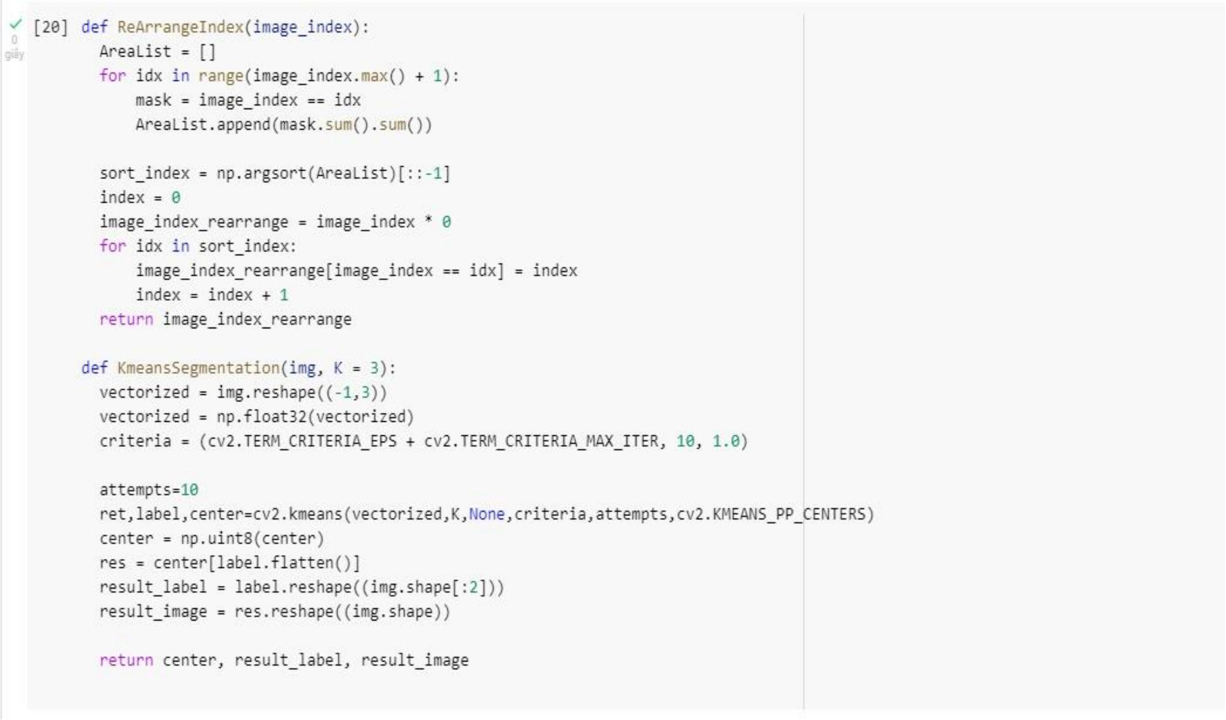
```
[18] ShowImage([img, result_label, result_image], 1, 3)
     print(img.shape)
```

```
(168, 300, 3)
```

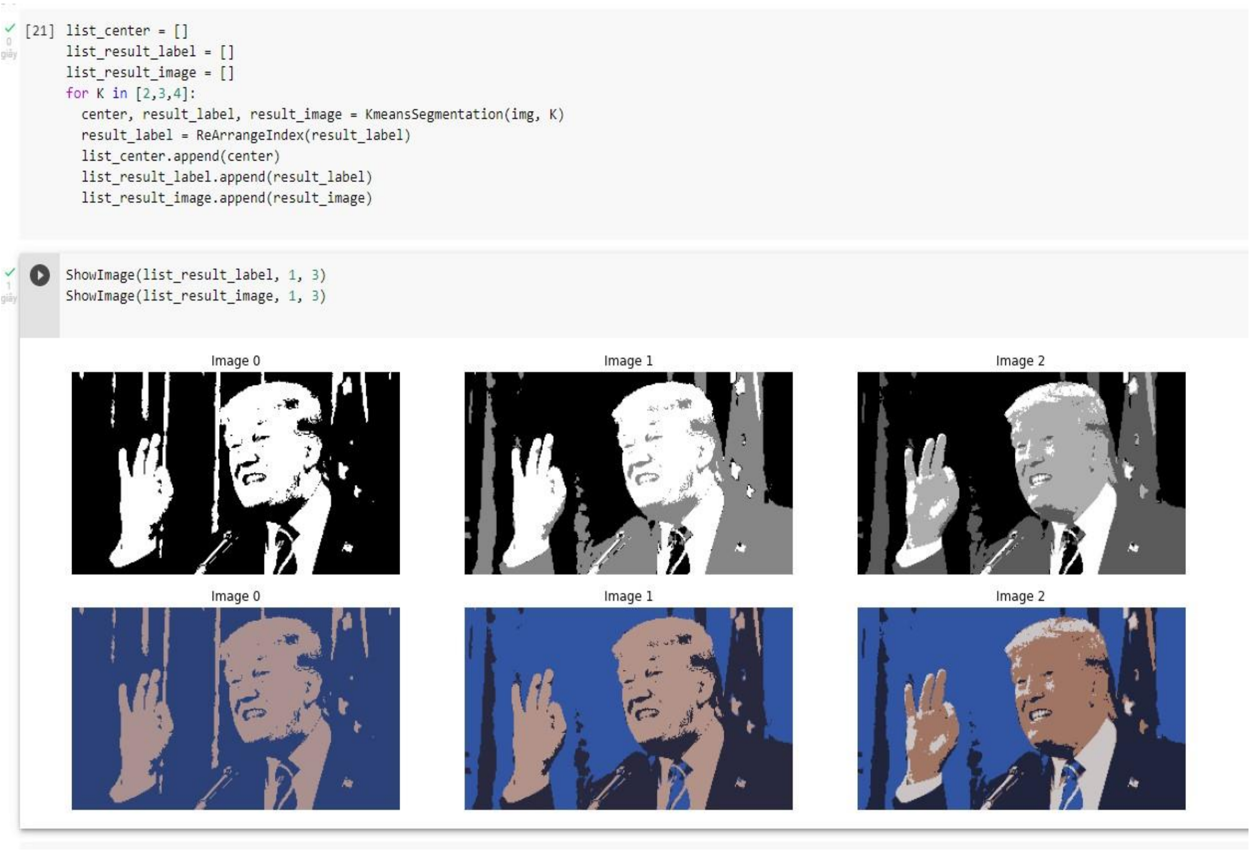


```
rpoint = 80
cpoint = 190
idx = result_label[rpoint, cpoint]
print("Index at ({0},{1}) : {2}".format(rpoint, cpoint, idx))
plt.imshow(img)
plt.plot(cpoint, rpoint, "or", markersize=10)  # og:shorthand for green circle
plt.show()

SegMask = result_label == idx
ShowImage([img, SegMask], 1, 2)
```

```
Index at (80,190) : 2
```

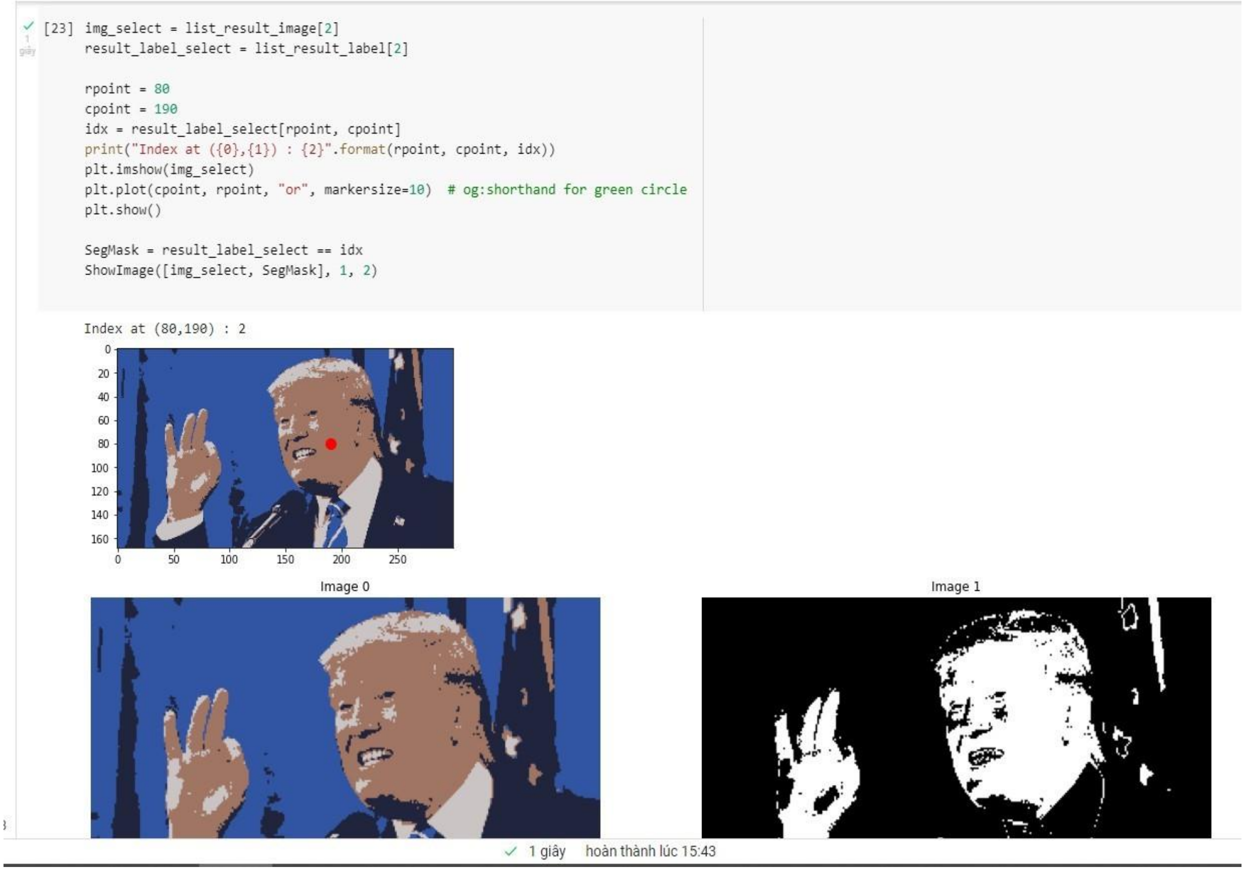


```
[20] def ReArrangeIndex(image_index):
       AreaList = []
       for idx in range(image_index.max() + 1):
           mask = image_index == idx
           AreaList.append(mask.sum().sum())

       sort_index = np.argsort(AreaList)[::-1]
       index = 0
       image_index_rearrange = image_index * 0
       for idx in sort_index:
           image_index_rearrange[image_index == idx] = index
           index = index + 1
       return image_index_rearrange

     def KmeansSegmentation(img, K = 3):
       vectorized = img.reshape((-1,3))
       vectorized = np.float32(vectorized)
       criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

       attempts=10
       ret,label,center=cv2.kmeans(vectorized,K,None,criteria,attempts,cv2.KMEANS_PP_CENTERS)
       center = np.uint8(center)
       res = center[label.flatten()]
       result_label = label.reshape((img.shape[:2]))
       result_image = res.reshape((img.shape))

       return center, result_label, result_image
```



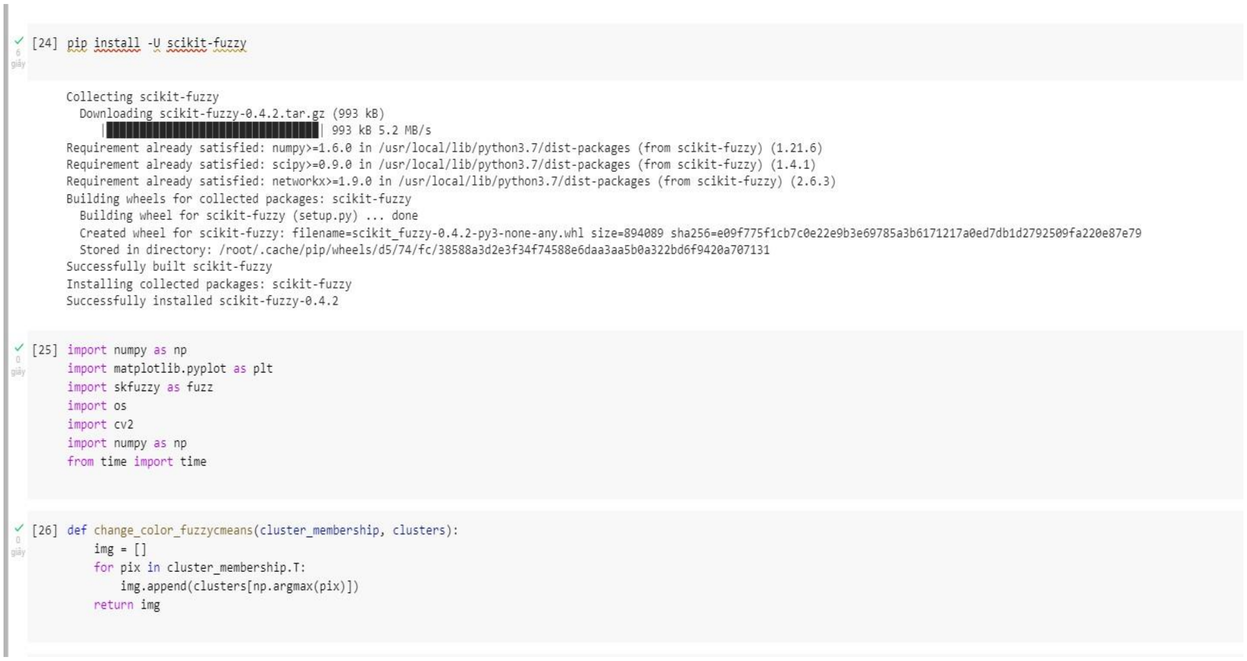
```
[21] list_center = []
     list_result_label = []
     list_result_image = []
     for K in [2,3,4]:
       center, result_label, result_image = KmeansSegmentation(img, K)
       result_label = ReArrangeIndex(result_label)
       list_center.append(center)
       list_result_label.append(result_label)
       list_result_image.append(result_image)
```

```
ShowImage(list_result_label, 1, 3)
ShowImage(list_result_image, 1, 3)
```



```
[23] img_select = list_result_image[2]
     result_label_select = list_result_label[2]

     rpoint = 80
     cpoint = 190
     idx = result_label_select[rpoint, cpoint]
     print("Index at ({0},{1}) : {2}".format(rpoint, cpoint, idx))
     plt.imshow(img_select)
     plt.plot(cpoint, rpoint, "or", markersize=10)  # og:shorthand for green circle
     plt.show()

     SegMask = result_label_select == idx
     ShowImage([img_select, SegMask], 1, 2)
```

```
Index at (80,190) : 2
```

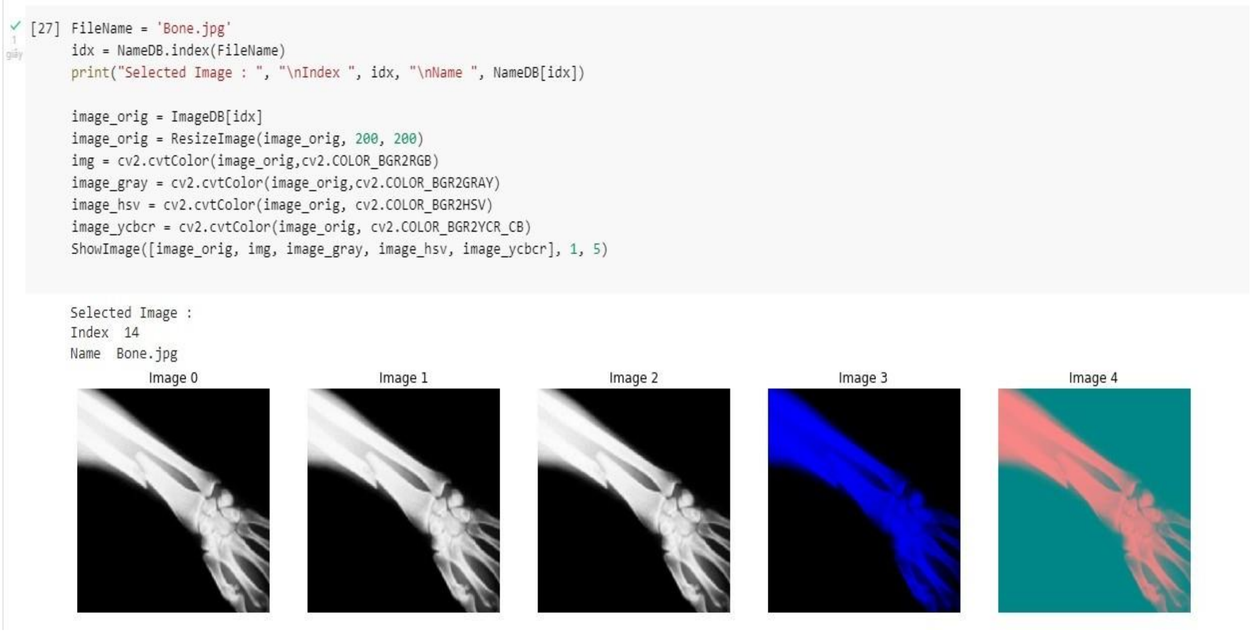


```
[24] pip install -U scikit-fuzzy
```

```
Collecting scikit-fuzzy
  Downloading scikit-fuzzy-0.4.2.tar.gz (993 kB)
     |████████████████████████████████| 993 kB 5.2 MB/s
Requirement already satisfied: numpy>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from scikit-fuzzy) (1.21.6)
Requirement already satisfied: scipy>=0.9.0 in /usr/local/lib/python3.7/dist-packages (from scikit-fuzzy) (1.4.1)
Requirement already satisfied: networkx>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from scikit-fuzzy) (2.6.3)
Building wheels for collected packages: scikit-fuzzy
  Building wheel for scikit-fuzzy (setup.py) ... done
  Created wheel for scikit-fuzzy: filename=scikit_fuzzy-0.4.2-py3-none-any.whl size=894089 sha256=e09f775f1cb7c0e22e9b3e69785a3b6171217a0ed7db1d2792509fa220e87e79
  Stored in directory: /root/.cache/pip/wheels/d5/74/fc/38588a3d2e3f34f74588e6daa3aa5b0a322bd6f9420a707131
Successfully built scikit-fuzzy
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.4.2
```

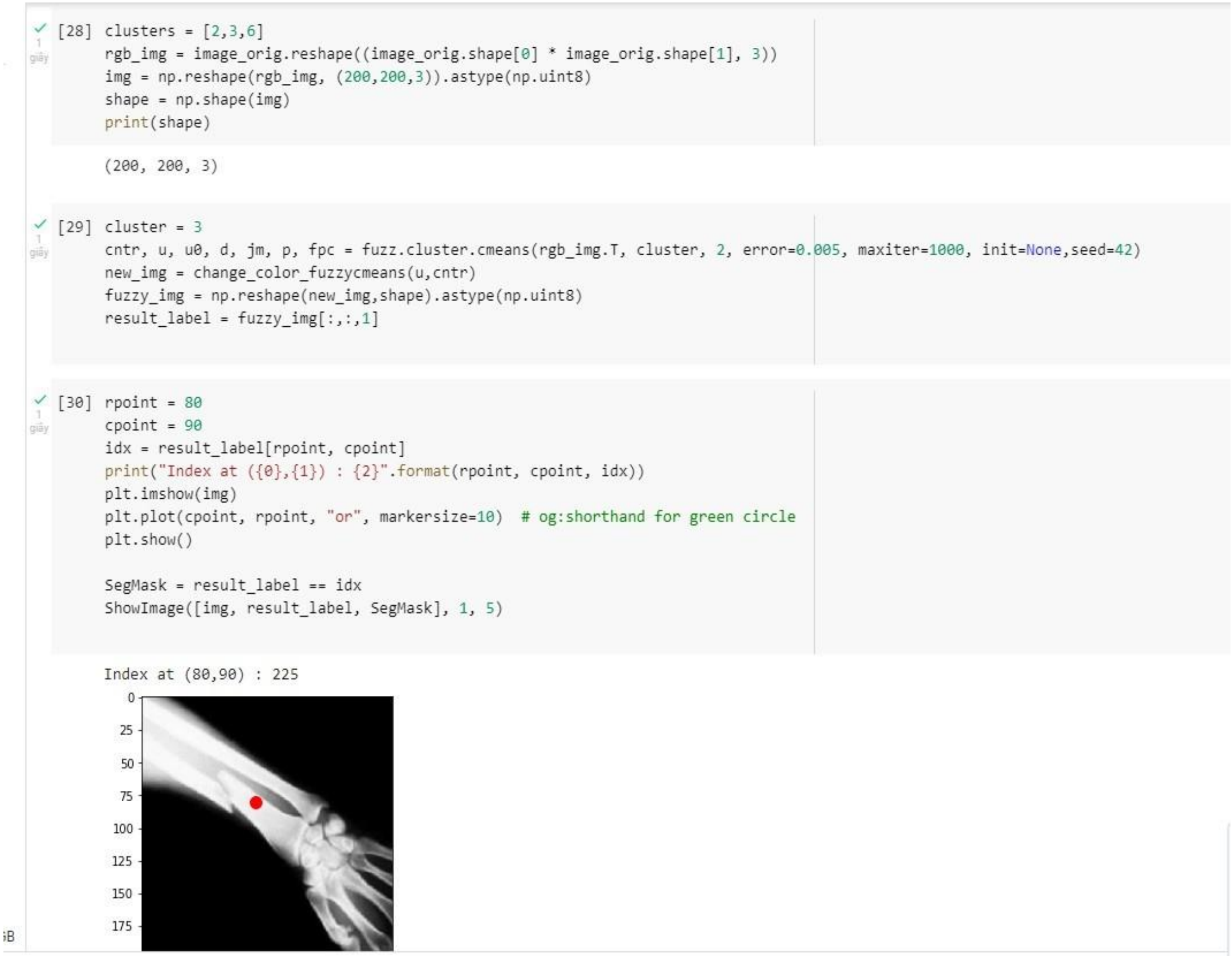
```
[25] import numpy as np
     import matplotlib.pyplot as plt
     import skfuzzy as fuzz
     import os
     import cv2
     import numpy as np
     from time import time
```

```
[26] def change_color_fuzzycmeans(cluster_membership, clusters):
         img = []
         for pix in cluster_membership.T:
             img.append(clusters[np.argmax(pix)])
         return img
```



```
[27] FileName = 'Bone.jpg'
     idx = NameDB.index(FileName)
     print("Selected Image : ", "\nIndex ", idx, "\nName ", NameDB[idx])

     image_orig = ImageDB[idx]
     image_orig = ResizeImage(image_orig, 200, 200)
     img = cv2.cvtColor(image_orig,cv2.COLOR_BGR2RGB)
     image_gray = cv2.cvtColor(image_orig,cv2.COLOR_BGR2GRAY)
     image_hsv = cv2.cvtColor(image_orig, cv2.COLOR_BGR2HSV)
     image_ycbcr = cv2.cvtColor(image_orig, cv2.COLOR_BGR2YCR_CB)
     ShowImage([image_orig, img, image_gray, image_hsv, image_ycbcr], 1, 5)
```

```
Selected Image :
Index  14
Name   Bone.jpg
```



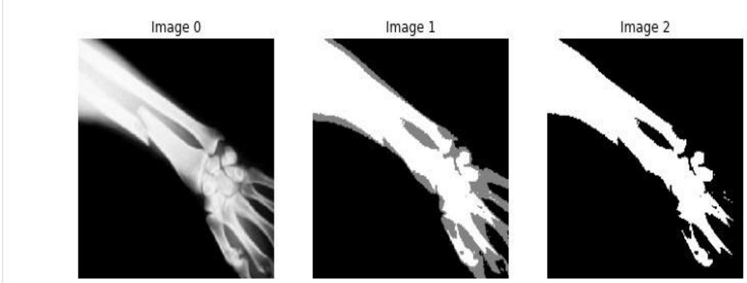
```
[28] clusters = [2,3,6]
     rgb_img = image_orig.reshape((image_orig.shape[0] * image_orig.shape[1], 3))
     img = np.reshape(rgb_img, (200,200,3)).astype(np.uint8)
     shape = np.shape(img)
     print(shape)
```

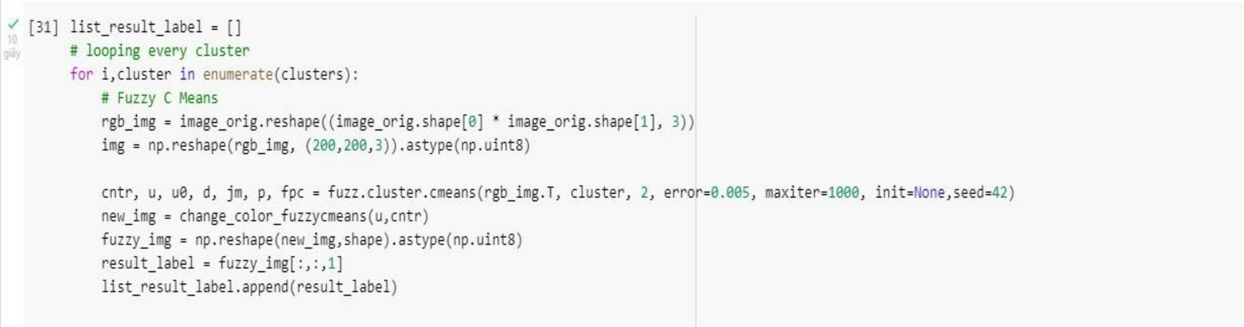
```
(200, 200, 3)
```

```
[29] cluster = 3
     cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(rgb_img.T, cluster, 2, error=0.005, maxiter=1000, init=None,seed=42)
     new_img = change_color_fuzzycmeans(u,cntr)
     fuzzy_img = np.reshape(new_img,shape).astype(np.uint8)
     result_label = fuzzy_img[:,:,1]
```

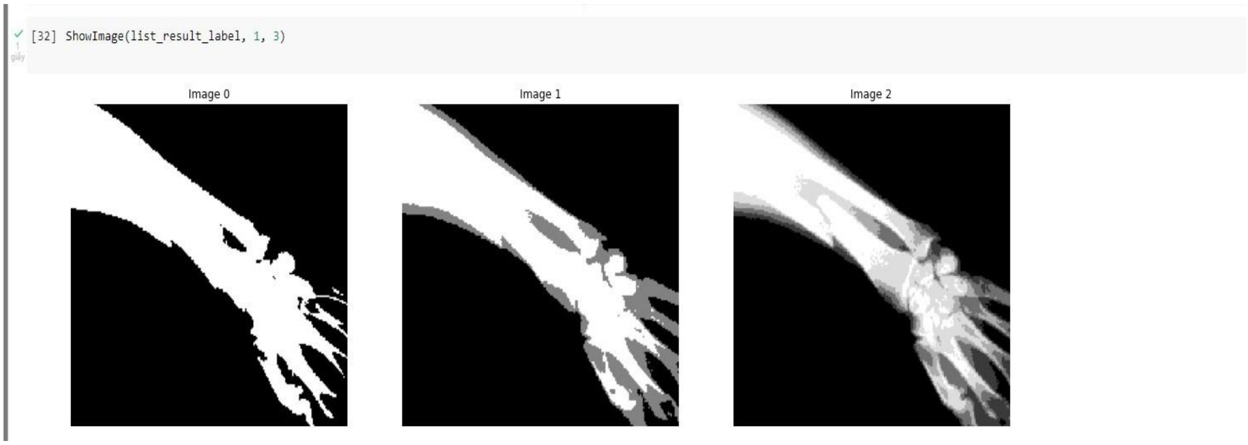
```
[30] rpoint = 80
     cpoint = 90
     idx = result_label[rpoint, cpoint]
     print("Index at ({0},{1}) : {2}".format(rpoint, cpoint, idx))
     plt.imshow(img)
     plt.plot(cpoint, rpoint, "or", markersize=10)  # og:shorthand for green circle
     plt.show()

     SegMask = result_label == idx
     ShowImage([img, result_label, SegMask], 1, 5)
```

```
Index at (80,90) : 225
```

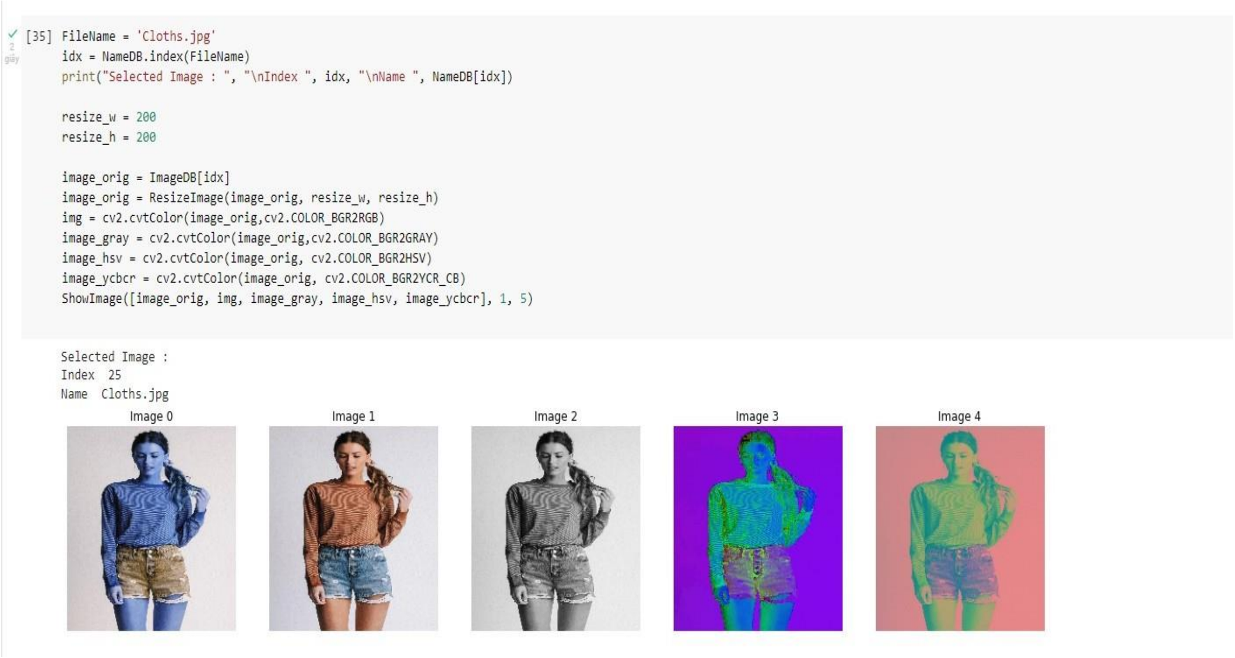




```
[31] list_result_label = []
     # looping every cluster
     for i,cluster in enumerate(clusters):
         # Fuzzy C Means
         rgb_img = image_orig.reshape((image_orig.shape[0] * image_orig.shape[1], 3))
         img = np.reshape(rgb_img, (200,200,3)).astype(np.uint8)

         cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(rgb_img.T, cluster, 2, error=0.005, maxiter=1000, init=None,seed=42)
         new_img = change_color_fuzzycmeans(u,cntr)
         fuzzy_img = np.reshape(new_img,shape).astype(np.uint8)
         result_label = fuzzy_img[:,:,1]
         list_result_label.append(result_label)
```

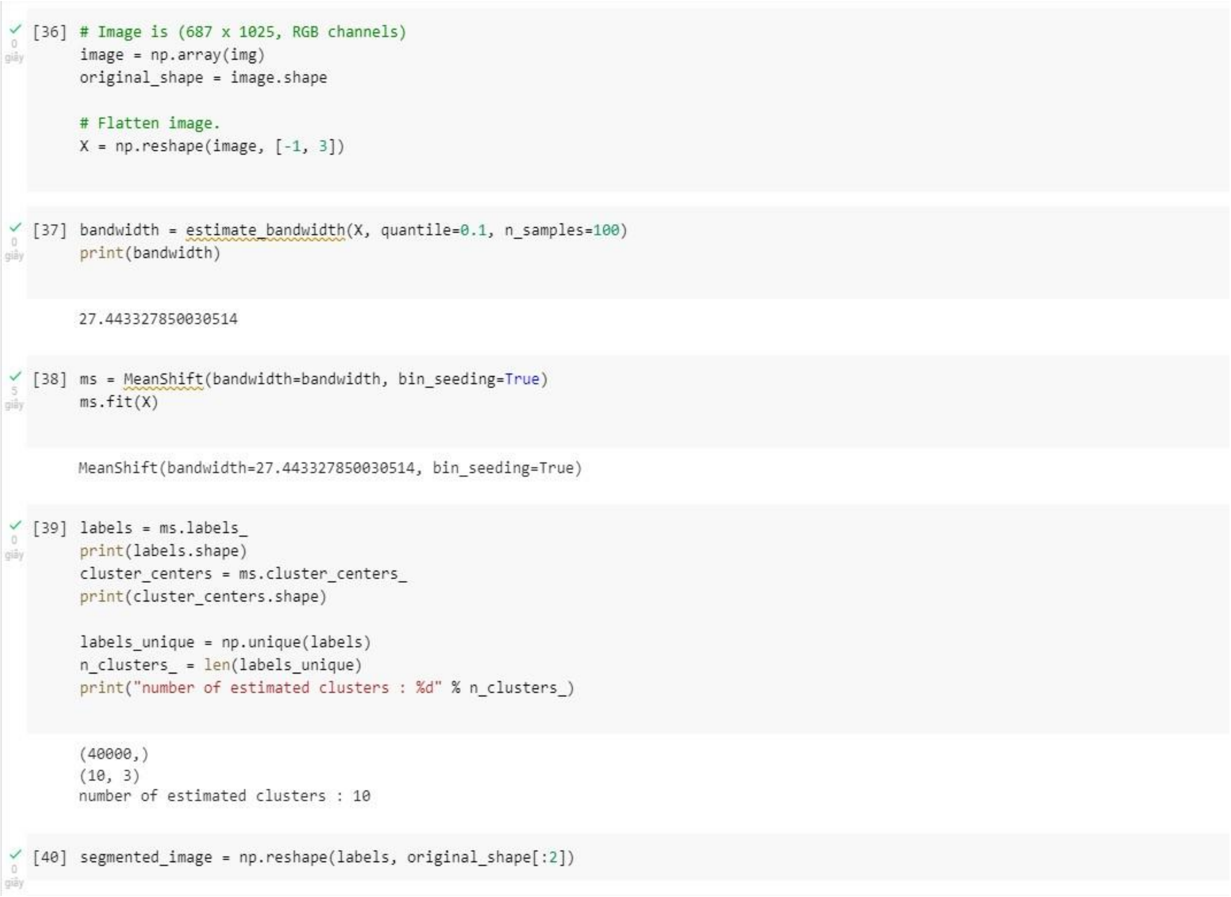


```
[32] ShowImage(list_result_label, 1, 3)
```



```
[35] FileName = 'Cloths.jpg'
     idx = NameDB.index(FileName)
     print("Selected Image : ", "\nIndex ", idx, "\nName ", NameDB[idx])

     resize_w = 200
     resize_h = 200

     image_orig = ImageDB[idx]
     image_orig = ResizeImage(image_orig, resize_w, resize_h)
     img = cv2.cvtColor(image_orig,cv2.COLOR_BGR2RGB)
     image_gray = cv2.cvtColor(image_orig,cv2.COLOR_BGR2GRAY)
     image_hsv = cv2.cvtColor(image_orig, cv2.COLOR_BGR2HSV)
     image_ycbcr = cv2.cvtColor(image_orig, cv2.COLOR_BGR2YCR_CB)
     ShowImage([image_orig, img, image_gray, image_hsv, image_ycbcr], 1, 5)
```

```
Selected Image :
Index  25
Name  Cloths.jpg
```



```
[36] # Image is (687 x 1025, RGB channels)
     image = np.array(img)
     original_shape = image.shape

     # Flatten image.
     X = np.reshape(image, [-1, 3])
```

```
[37] bandwidth = estimate_bandwidth(X, quantile=0.1, n_samples=100)
     print(bandwidth)
```

```
27.443327850030514
```

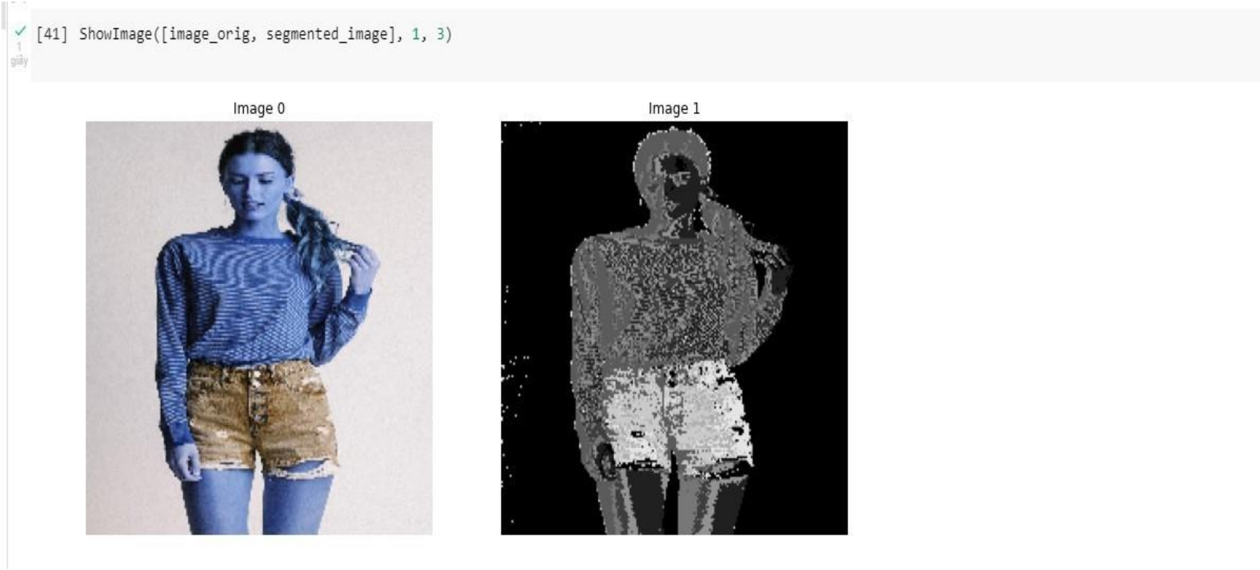
```
[38] ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
     ms.fit(X)
```

```
MeanShift(bandwidth=27.443327850030514, bin_seeding=True)
```

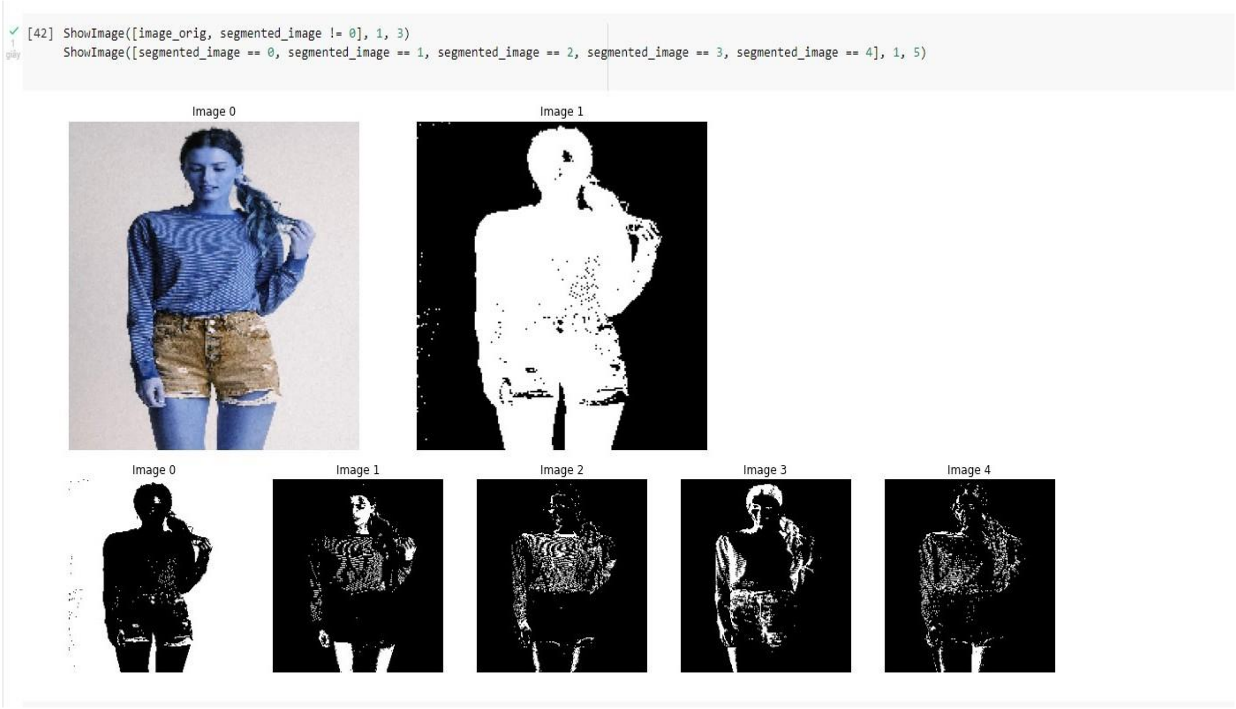
```
[39] labels = ms.labels_
     print(labels.shape)
     cluster_centers = ms.cluster_centers_
     print(cluster_centers.shape)

     labels_unique = np.unique(labels)
     n_clusters_ = len(labels_unique)
     print("number of estimated clusters : %d" % n_clusters_)
```

```
(40000,)
(10, 3)
number of estimated clusters : 10
```

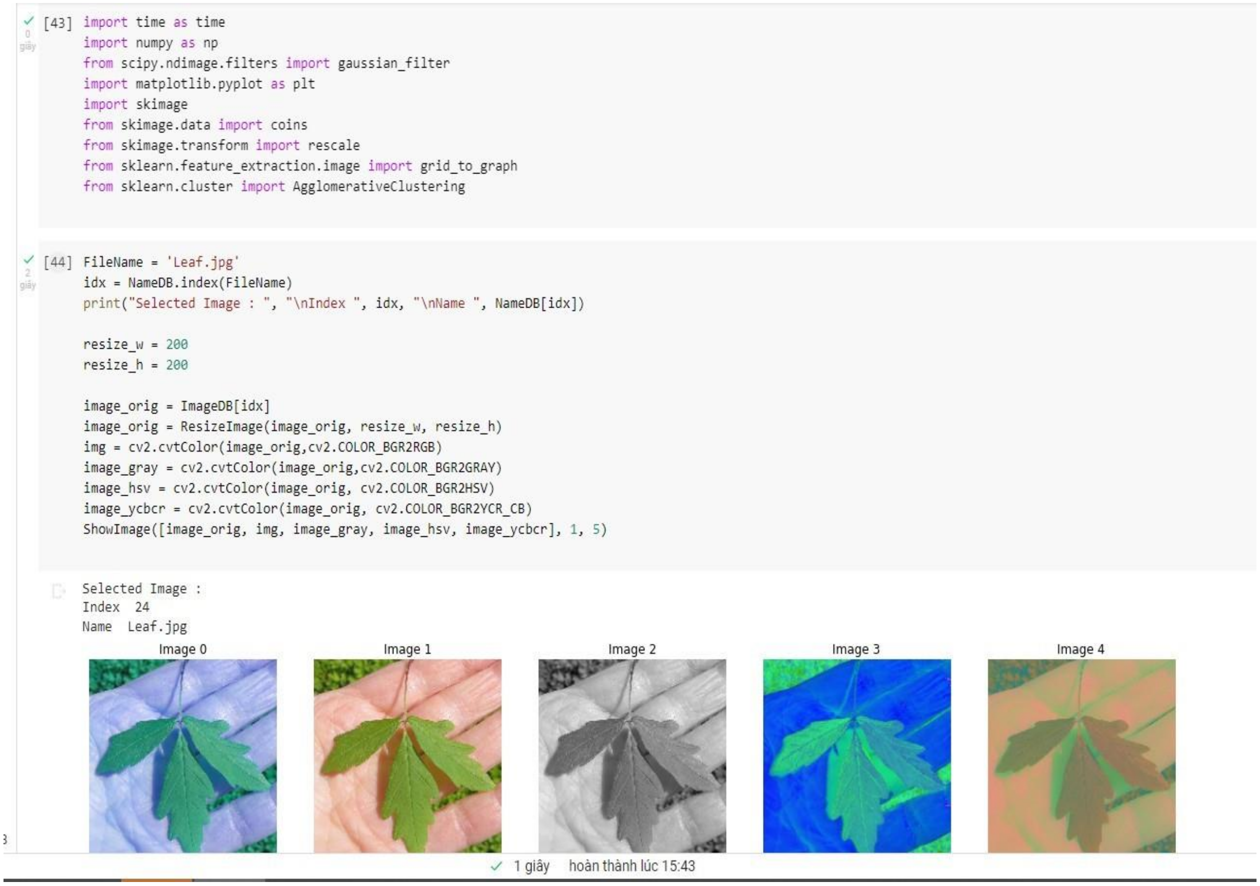
```
[40] segmented_image = np.reshape(labels, original_shape[:2])
```



```
[41]  ShowImage([image_orig, segmented_image], 1, 3)
```



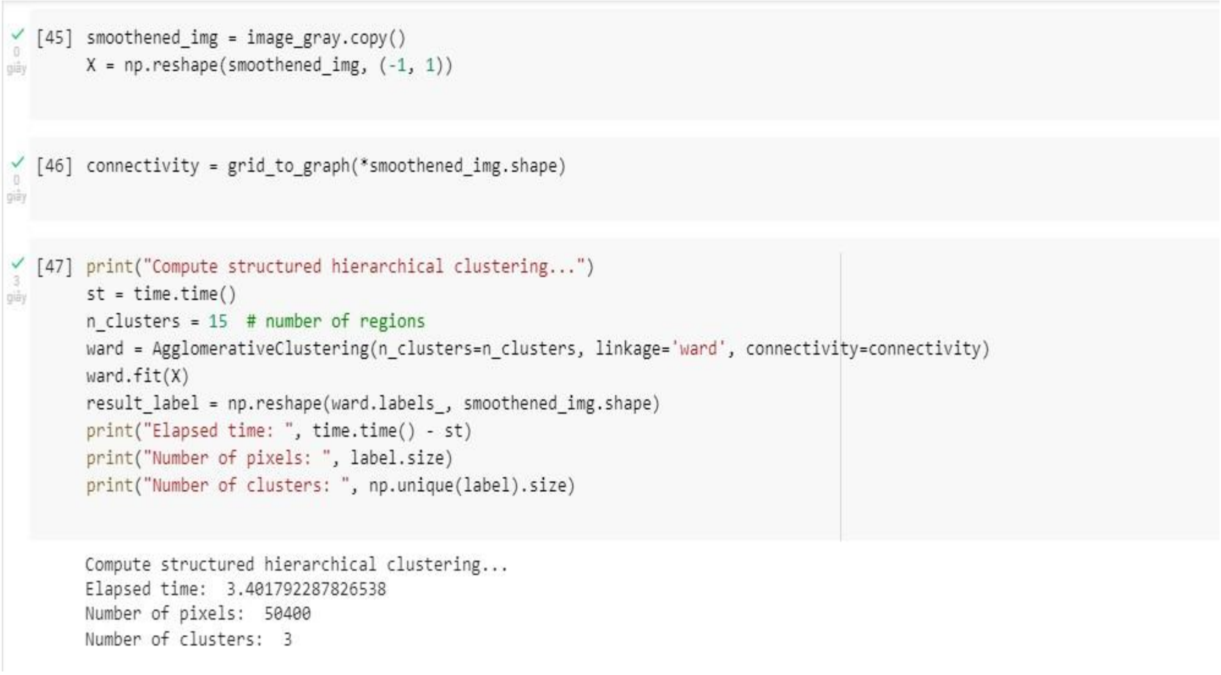
```
[42]  ShowImage([image_orig, segmented_image != 0], 1, 3)
      ShowImage([segmented_image == 0, segmented_image == 1, segmented_image == 2, segmented_image == 3, segmented_image == 4], 1, 5)
```



```
[43] import time as time
     import numpy as np
     from scipy.ndimage.filters import gaussian_filter
     import matplotlib.pyplot as plt
     import skimage
     from skimage.data import coins
     from skimage.transform import rescale
     from sklearn.feature_extraction.image import grid_to_graph
     from sklearn.cluster import AgglomerativeClustering
```

```
[44] FileName = 'Leaf.jpg'
     idx = NameDB.index(FileName)
     print("Selected Image : ", "\nIndex ", idx, "\nName ", NameDB[idx])

     resize_w = 200
     resize_h = 200

     image_orig = ImageDB[idx]
     image_orig = ResizeImage(image_orig, resize_w, resize_h)
     img = cv2.cvtColor(image_orig,cv2.COLOR_BGR2RGB)
     image_gray = cv2.cvtColor(image_orig,cv2.COLOR_BGR2GRAY)
     image_hsv = cv2.cvtColor(image_orig, cv2.COLOR_BGR2HSV)
     image_ycbcr = cv2.cvtColor(image_orig, cv2.COLOR_BGR2YCR_CB)
     ShowImage([image_orig, img, image_gray, image_hsv, image_ycbcr], 1, 5)
```

```
Selected Image :
Index  24
Name  Leaf.jpg
```

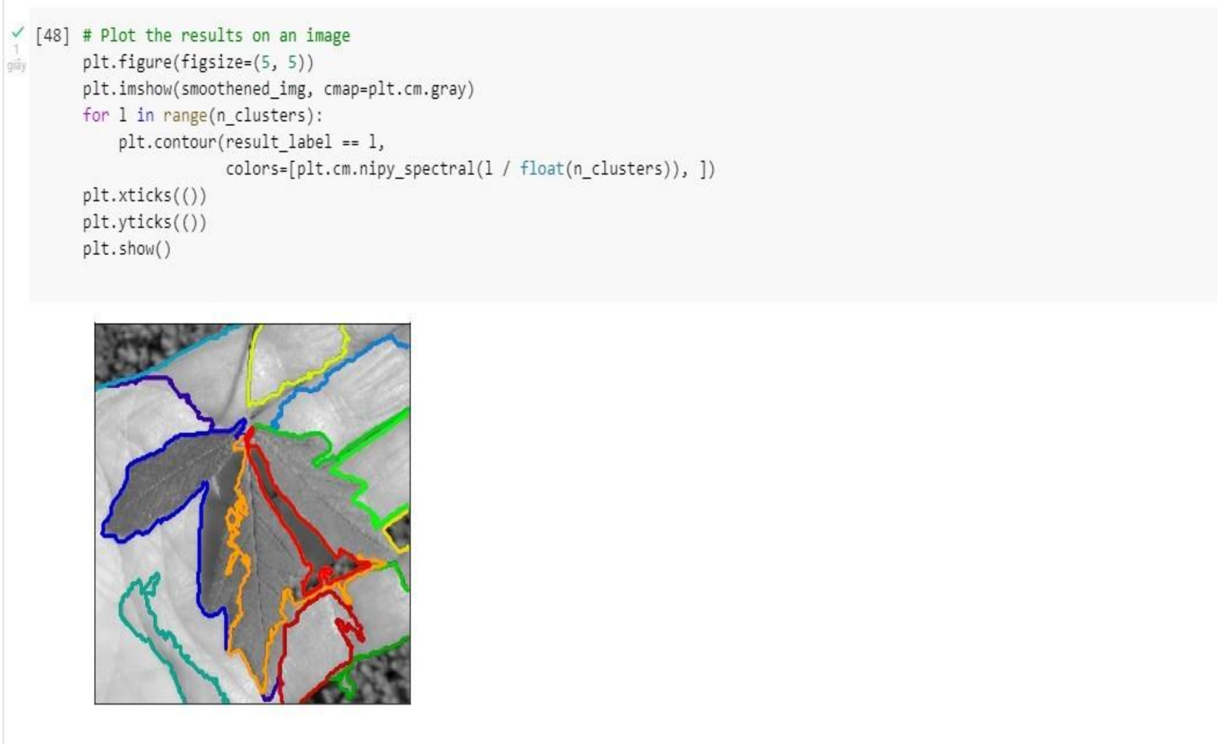


```
[45] smoothened_img = image_gray.copy()
     X = np.reshape(smoothened_img, (-1, 1))
```

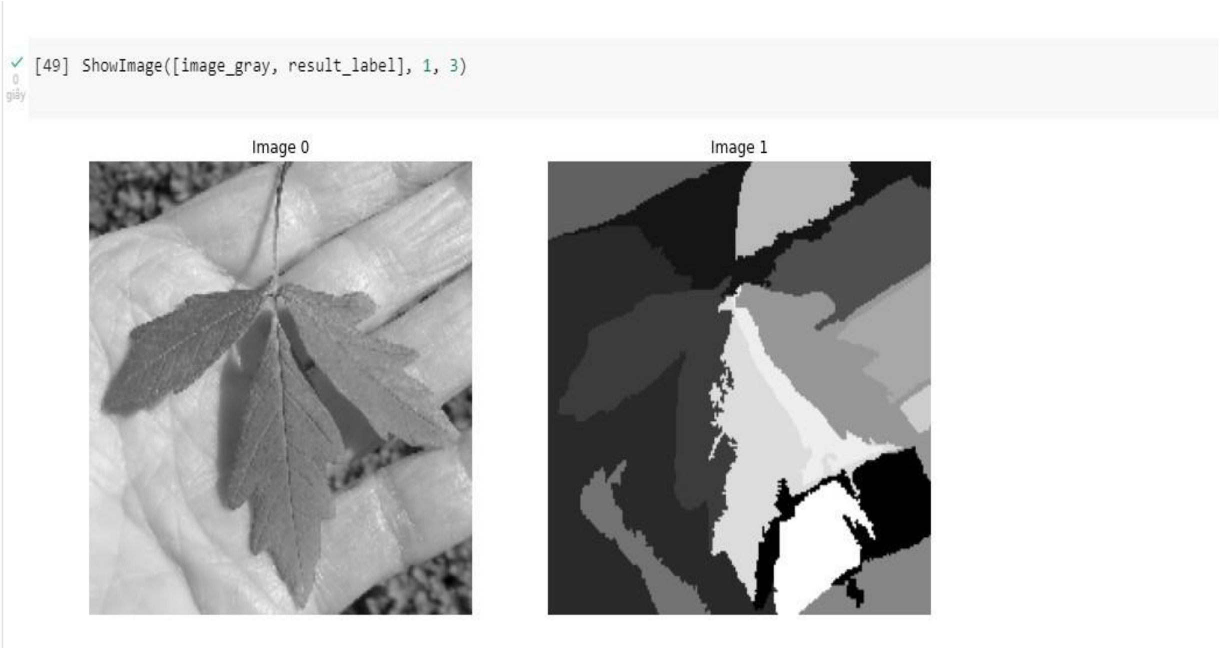
```
[46] connectivity = grid_to_graph(*smoothened_img.shape)
```

```
[47] print("Compute structured hierarchical clustering...")
     st = time.time()
     n_clusters = 15  # number of regions
     ward = AgglomerativeClustering(n_clusters=n_clusters, linkage='ward', connectivity=connectivity)
     ward.fit(X)
     result_label = np.reshape(ward.labels_, smoothened_img.shape)
     print("Elapsed time: ", time.time() - st)
     print("Number of pixels: ", label.size)
     print("Number of clusters: ", np.unique(label).size)
```

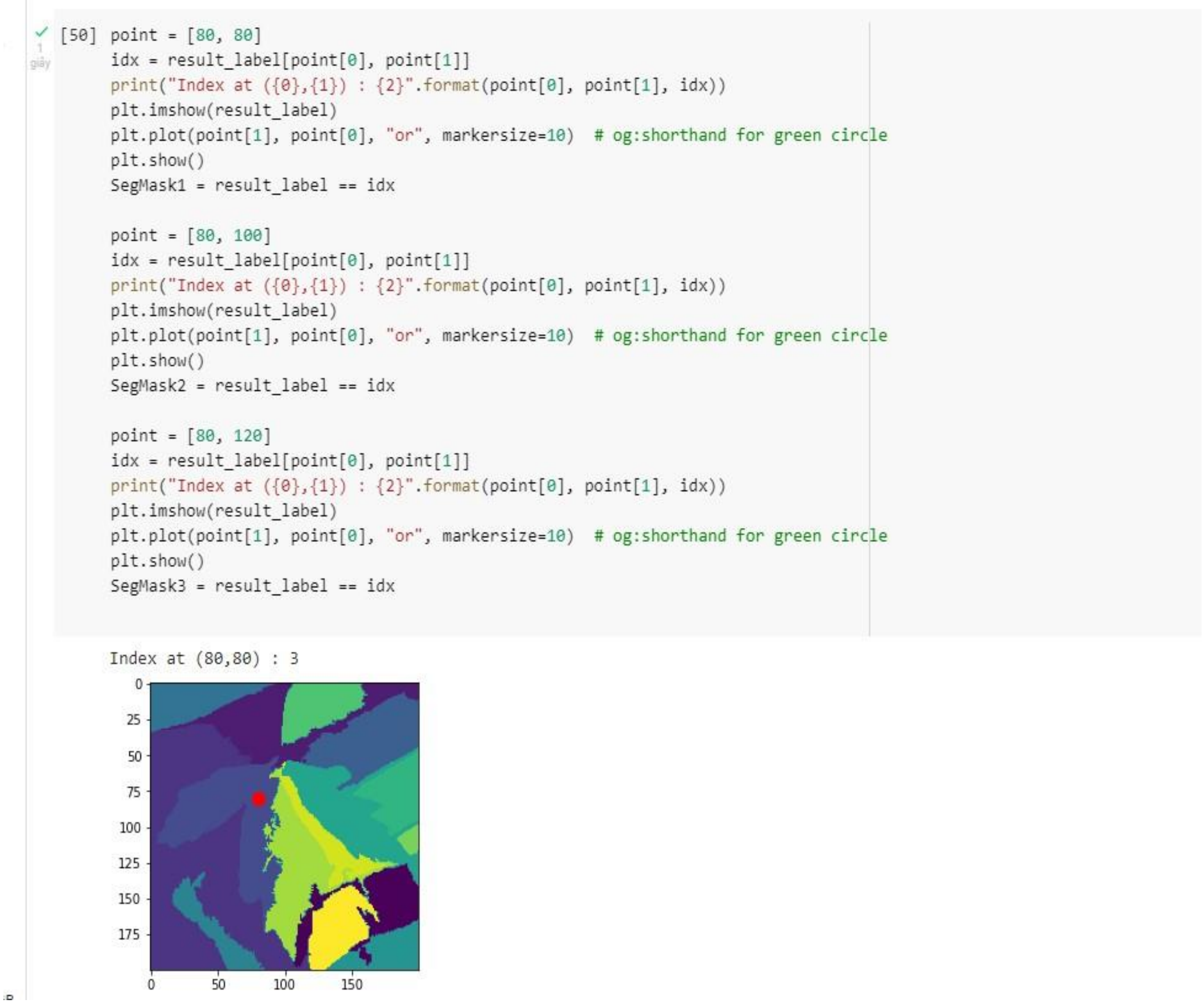
```
Compute structured hierarchical clustering...
Elapsed time:  3.401792287826538
Number of pixels:  50400
Number of clusters:  3
```



```
[48] # Plot the results on an image
     plt.figure(figsize=(5, 5))
     plt.imshow(smoothened_img, cmap=plt.cm.gray)
     for l in range(n_clusters):
         plt.contour(result_label == l,
                     colors=[plt.cm.nipy_spectral(l / float(n_clusters)), ])
     plt.xticks(())
     plt.yticks(())
     plt.show()
```

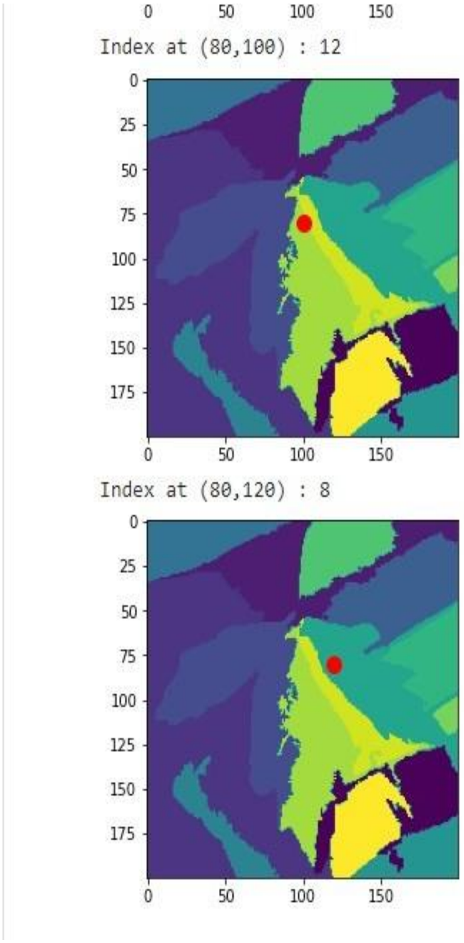


```
[49] ShowImage([image_gray, result_label], 1, 3)
```



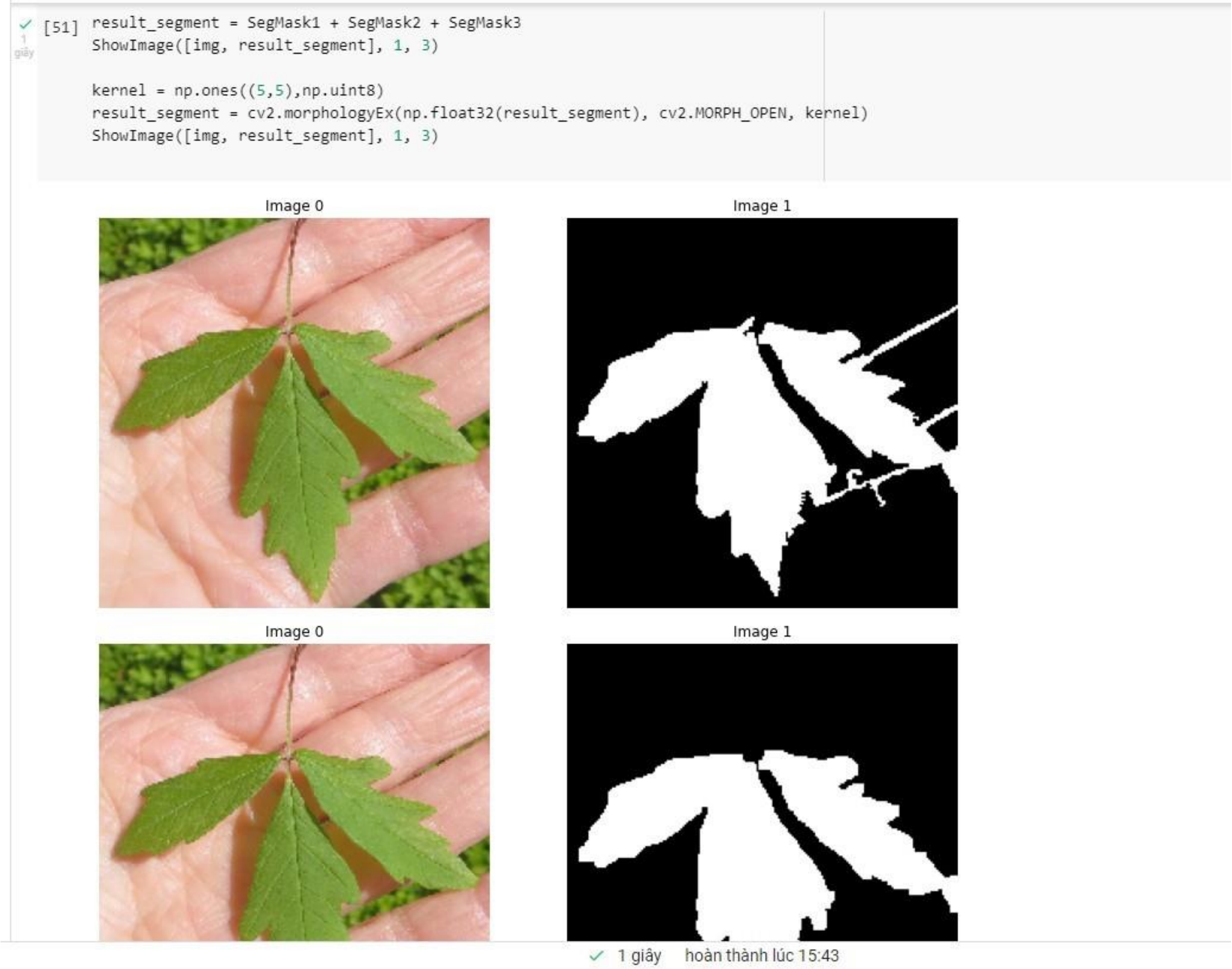
```
[50] point = [80, 80]
     idx = result_label[point[0], point[1]]
     print("Index at ({0},{1}) : {2}".format(point[0], point[1], idx))
     plt.imshow(result_label)
     plt.plot(point[1], point[0], "or", markersize=10)  # og:shorthand for green circle
     plt.show()
     SegMask1 = result_label == idx

     point = [80, 100]
     idx = result_label[point[0], point[1]]
     print("Index at ({0},{1}) : {2}".format(point[0], point[1], idx))
     plt.imshow(result_label)
     plt.plot(point[1], point[0], "or", markersize=10)  # og:shorthand for green circle
     plt.show()
     SegMask2 = result_label == idx

     point = [80, 120]
     idx = result_label[point[0], point[1]]
     print("Index at ({0},{1}) : {2}".format(point[0], point[1], idx))
     plt.imshow(result_label)
     plt.plot(point[1], point[0], "or", markersize=10)  # og:shorthand for green circle
     plt.show()
     SegMask3 = result_label == idx
```

```
Index at (80,80) : 3
```



```
Index at (80,100) : 12
```

```
Index at (80,120) : 8
```



[51]

```
result_segment = SegMask1 + SegMask2 + SegMask3
ShowImage([img, result_segment], 1, 3)

kernel = np.ones((5,5),np.uint8)
result_segment = cv2.morphologyEx(np.float32(result_segment), cv2.MORPH_OPEN, kernel)
ShowImage([img, result_segment], 1, 3)
```

✓ 1 giây hoàn thành lúc 15:43