**Library used:**

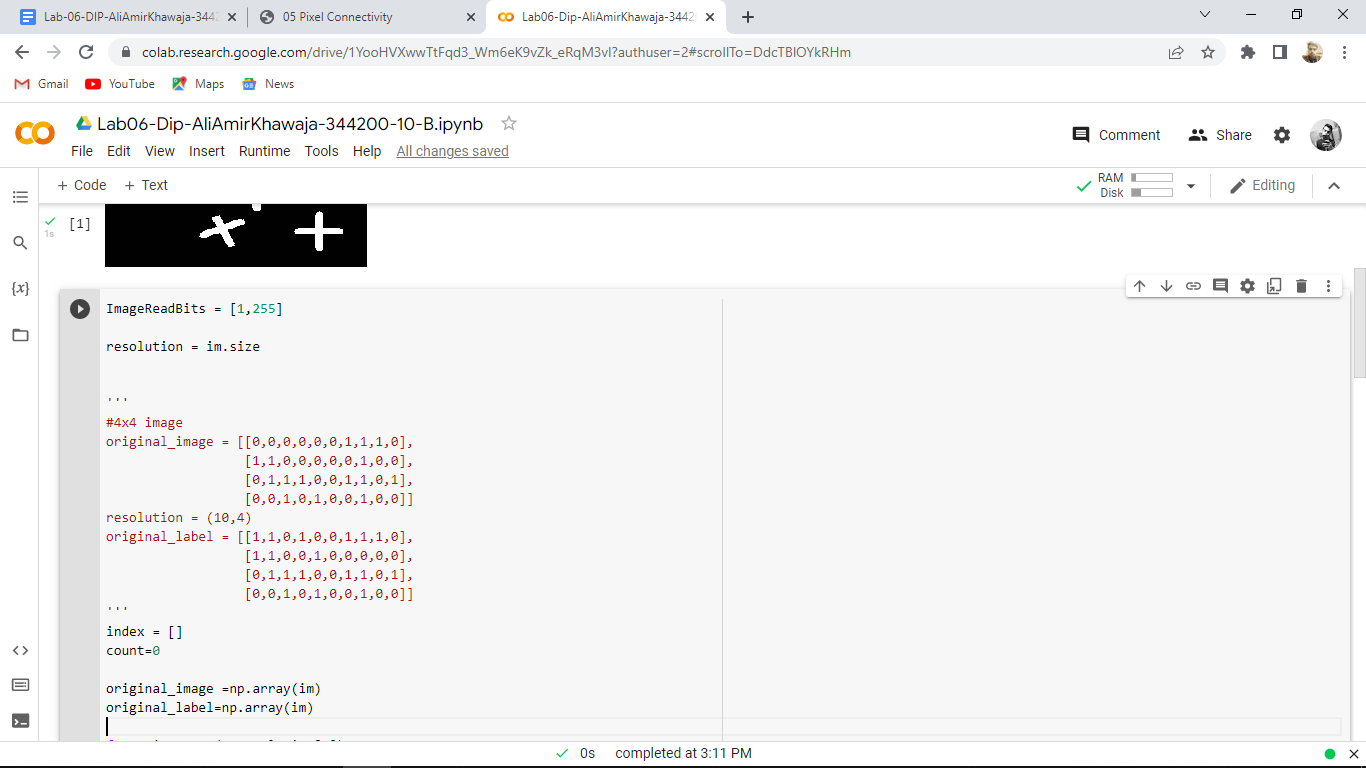
**from PIL import Image**

**import numpy as np**

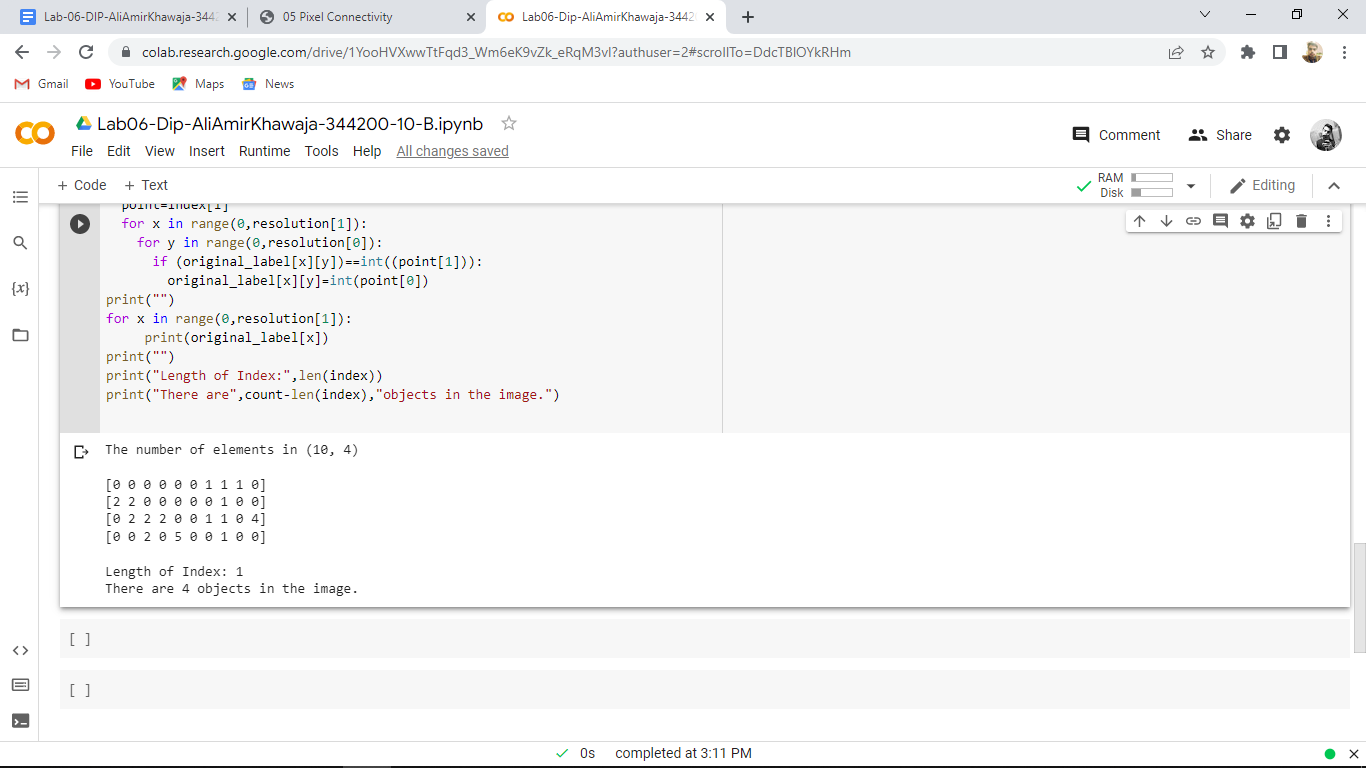
**Task 1**

* **Testing code with a demo image array.**

**Original demo image of 10x4 size.**



**ScreenShot of Demo Image of size 10x4.**



**Task 2**

**Code**

**from PIL import Image**

**import numpy as np**

**im = Image.open("/content/cc.png")**

**im**

**ImageReadBits = [1,255]**

**resolution = im.size**

**'''**

**#4x4 image**

**original\_image = [[0,0,0,0,0,0,1,1,1,0],**

**[1,1,0,0,0,0,0,1,0,0],**

**[0,1,1,1,0,0,1,1,0,1],**

**[0,0,1,0,1,0,0,1,0,0]]**

**resolution = (10,4)**

**original\_label = [[1,1,0,1,0,0,1,1,1,0],**

**[1,1,0,0,1,0,0,0,0,0],**

**[0,1,1,1,0,0,1,1,0,1],**

**[0,0,1,0,1,0,0,1,0,0]]**

**'''**

**index = []**

**count=0**

**#Using Numpy to covert image values in arrays**

**original\_image =np.array(im)**

**original\_label=np.array(im)**

**#Initializing the label's array**

**for x in range(0,resolution[1]):**

**for y in range(0,resolution[0]):**

**original\_label[x][y]=0**

**for x in range(0,resolution[1]):**

**for y in range(0,resolution[0]):**

**if original\_image[x][y]==1:**

**original\_image[x][y]=0**

**#Main code for finding the number of objects in the image**

**print("The number of elements in",resolution)**

**#All the conditions to find the objects by traversing thought each data point**

**for x in range(0,resolution[1]):**

**for y in range(0,resolution[0]):**

**if original\_image[x][y]!=0:**

**if x==0 and y==0:**

**count=count+1**

**original\_label[x][y]=count**

**elif x==0 and y!=0:**

**if original\_image[x][y-1]!=0:**

**original\_label[x][y]=original\_label[x][y-1]**

**else:**

**count=count+1**

**original\_label[x][y]=count**

**elif y==0 and x!=0:**

**if original\_image[x-1][y]!=0:**

**original\_label[x][y]=original\_label[x-1][y]**

**else:**

**count=count+1**

**original\_label[x][y]=count**

**elif x!=0 and y!=0:**

**if original\_image[x][y-1]==0 and original\_image[x-1][y]==0:**

**count=count+1**

**original\_label[x][y]=count**

**elif (original\_image[x][y-1]!=0 and original\_image[x-1][y]==0):**

**original\_label[x][y]=original\_label[x][y-1]**

**elif (original\_image[x][y-1]==0 and original\_image[x-1][y]!=0):**

**original\_label[x][y]=original\_label[x-1][y]**

**elif original\_image[x][y-1]!=0 and original\_image[x-1][y]!=0:**

**if original\_label[x][y-1]==original\_label[x-1][y]:**

**original\_label[x][y]=original\_label[x][y-1]**

**elif original\_label[x][y-1]!=original\_label[x-1][y]:**

**original\_label[x][y]=original\_label[x][y-1]**

**if (original\_label[x][y-1])>(original\_label[x-1][y]):**

**index.append(str(original\_label[x-1][y])+","+str(original\_label[x][y-1]))**

**else:**

**index.append(str(original\_label[x][y-1])+","+str(original\_label[x-1][y]))**

**#End of conditions**

**count=count-1**

**list=[]**

**for x in range(0,len(index)):**

**if list.\_\_contains\_\_(index[x])==False:**

**list.append(index[x])**

**#Second pass replacing labels if there 4-Connectivity**

**temp\_str=""**

**for i in range(0,len(index)):**

**point=list[i]**

**for x in range(0,resolution[1]):**

**for y in range(0,resolution[0]):**

**temp\_str=point.split(",")**

**if (original\_label[x][y])==int(temp\_str[1]):**

**original\_label[x][y]=int(temp\_str[0])**

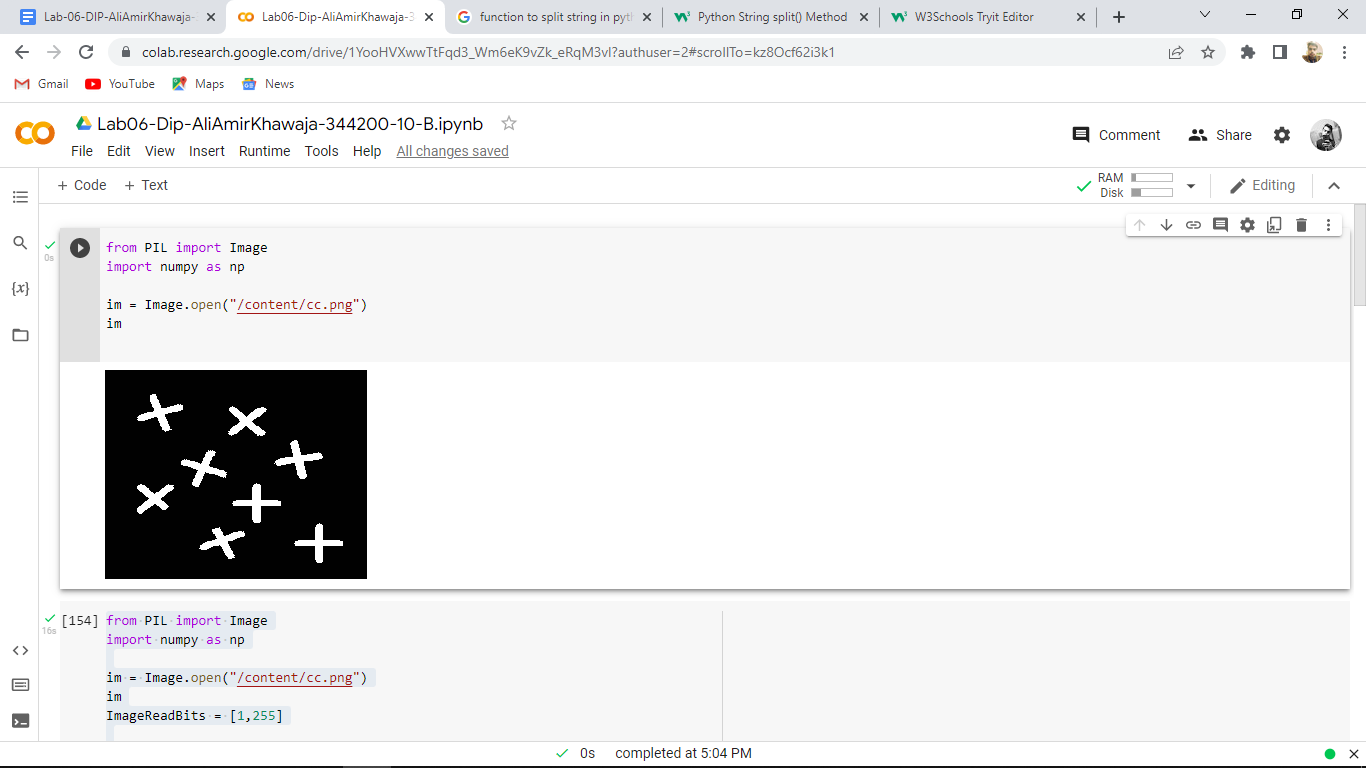
**#Printing out the results**

**print("")**

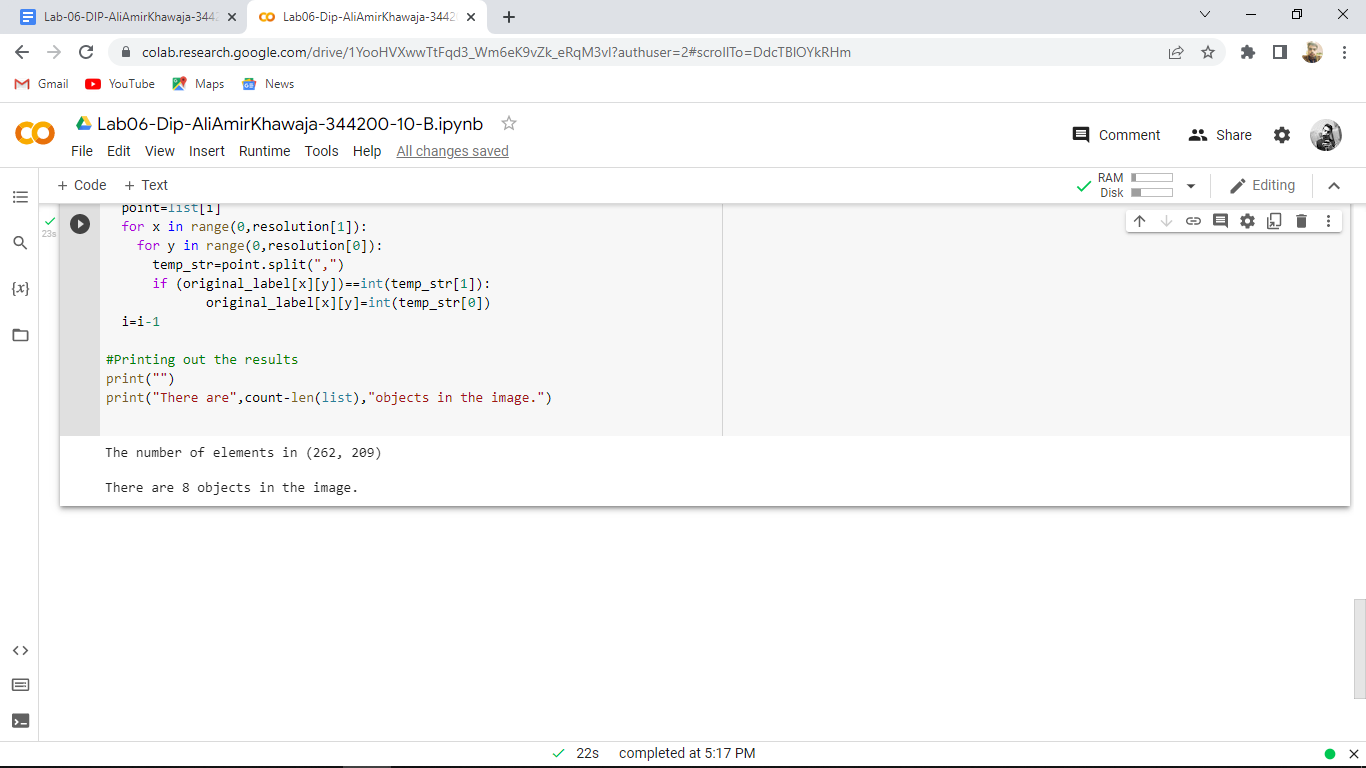
**print("There are",count-len(list),"objects in the image.")**

**Screenshot**

Original image:



Results After Execution:



**Task 3**

**Code:**

**# Getting the input image**

**img = cv2.imread("/content/cc.png", 0)**

**# Converting those pixels with values 1-127 to 0 and others to 1**

**img = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)[1]**

**# Applying cv2.connectedComponents()**

**num\_labels, labels = cv2.connectedComponents(img)**

**# Map component labels to hue val, 0-179 is the hue range in OpenCV**

**label\_hue = np.uint8(179\*labels/np.max(labels))**

**blank\_ch = 255\*np.ones\_like(label\_hue)**

**labeled\_img = cv2.merge([label\_hue, blank\_ch, blank\_ch])**

**# Converting cvt to BGR**

**labeled\_img = cv2.cvtColor(labeled\_img, cv2.COLOR\_HSV2BGR)**

**# set bg label to black**

**labeled\_img[label\_hue==0] = 0**

**# Showing Original Image**

**plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))**

**plt.axis("off")**

**plt.title("Orginal Image")**

**plt.show()**

**#Showing Image after Component Labeling**

**plt.imshow(cv2.cvtColor(labeled\_img, cv2.COLOR\_BGR2RGB))**

**plt.axis('off')**

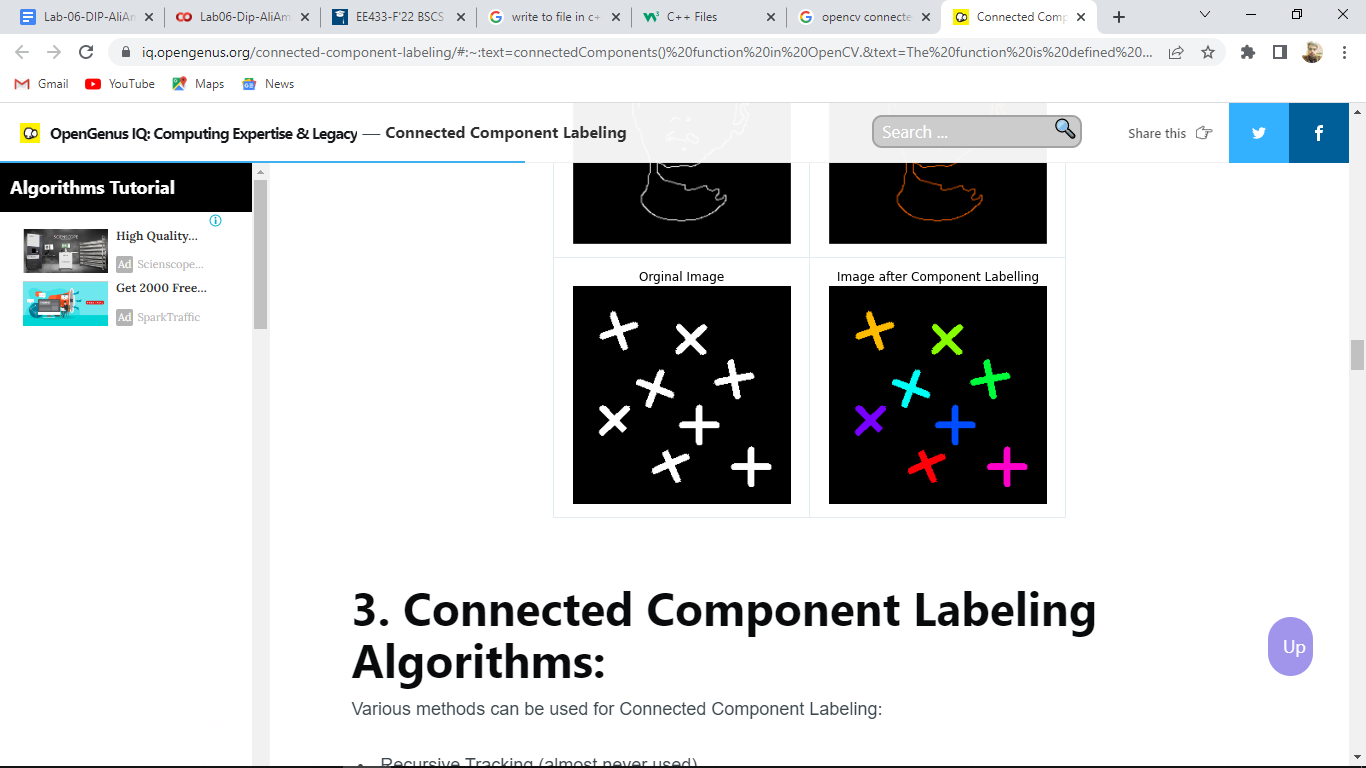
**plt.title("Image after Component Labeling")**

**plt.show()**

**Note:**

Code is taken from the website: https://iq.opengenus.org/connected-component-labeling/#:~:text=connectedComponents()%20function%20in%20OpenCV.&text=The%20function%20is%20defined%20so,path%20to%20the%20original%20image.

**Output:**

****

**Findings:**

* The results from both tasks show that in the image there are 8 objects.
* The other is considered as background.