Contours are the boundary of an object
Connected Component analysis: is used to find object in a image.

1) Script:Import necessary packages and image path

```
# import the necessary packages
import numpy as np
import cv2

img_path = "license_plate.png"
```

2) *Script:Load the image,convert to grayscale and apply threshold*

```
# load the input image from disk, convert it to grayscale, and
# threshold it
image = cv2.imread(img_path)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
thresh = cv2.threshold(gray, 0, 255,
    cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
```

3) Script: apply the opencv connected component analysis to the threshold of the binary image

```

# apply connected component analysis to the thresholded image
output = cv2.connectedComponentsWithStats(
    thresh, 4, cv2.CV_32S)
(numLabels, labels, stats, centroids) = output
print(numLabels)
```

Result: The `cv2.connectedComponentsWithStats return` NumLabels is the number of connected objects in the binary image which in this case will print 17 on the console,labels which is the integer ID of destination labelled image , the individual statistical property of the image,including the bounding box coordinates and area (in pixels).The `centroids` (i.e., center) *(x, y)*-coordinates of each connected component

The following code is apply on the each connected object in the numLabels

Script:

```
# loop over the number of unique connected component labels
for i in range(0, numLabels):
    # if this is the first component then we examining the
    # *background* (typically we would just ignore this
    # component in our loop)
    if i == 0:
        text = "examining component {}/{} (background)".format(
            i + 1, numLabels)

    # otherwise, we are examining an actual connected component
    else:
        text = "examining component {}/{}".format( i + 1, numLabels)

    # print a status message update for the current connected
    # component
    print("[INFO] {}".format(text))

    # extract the connected component statistics and centroid for
    # the current label
    x = stats[i, cv2.CC_STAT_LEFT]
    y = stats[i, cv2.CC_STAT_TOP]
    w = stats[i, cv2.CC_STAT_WIDTH]
    h = stats[i, cv2.CC_STAT_HEIGHT]
    area = stats[i, cv2.CC_STAT_AREA]
    (cX, cY) = centroids[i]
```

The first object in the numLabel is usually the background.The if statement in the is setting the text to be used in describing when printing to the console.The important piece here is to get the statistics of individual objects. where x,y are the top left corner of the component object,w,h are width and height,cX and cY are the center of the object.

Script: Draw a rectangle around the object with green color and circle with red color using the statistic and centroid values obtained above

```
    # clone our original image (so we can draw on it) and then draw
    # a bounding box surrounding the connected component along with
    # a circle corresponding to the centroid
    output = image.copy()
    cv2.rectangle(output, (x, y), (x + w, y + h), (0, 255, 0), 3)
    cv2.circle(output, (int(cX), int(cY)), 4, (0, 0, 255), -1)
```

Script: Constructing a mask for this considered connected object using it label ID

```
    # construct a mask for the current connected component by
    # finding a pixels in the labels array that have the current
    # connected component ID
    componentMask = (labels == i).astype("uint8") * 255
```

Script: Showing each object with rectangle boundaries and center,with the component mask

```
    # show our output image and connected component mask
    cv2.imshow("Output", output)
    cv2.imshow("Connected Component", componentMask)
```

Results:For this image we numLabels=17

Filtering connected object:The above process was able to split connected objects.We can filter connected objects to only show some of the connected objects.Repeating the same process above to the end of step 3:

Script:mask store the filter character

```
# initialize an output mask to store all characters parsed from
# the license plate
mask = np.zeros(gray.shape, dtype="uint8")
```

Script: the looping of each connected component.The x,y,w,h,area is the same concept as above,but keepWidth,keepHeight,keepArea are used to filter which component to keep based on the conditions

```

# loop over the number of unique connected component labels, skipping
# over the first label (as label zero is the background)
for i in range(1, numLabels):
    # extract the connected component statistics for the current
    # label
    x = stats[i, cv2.CC_STAT_LEFT]
    y = stats[i, cv2.CC_STAT_TOP]
    w = stats[i, cv2.CC_STAT_WIDTH]
    h = stats[i, cv2.CC_STAT_HEIGHT]
    area = stats[i, cv2.CC_STAT_AREA]

    # ensure the width, height, and area are all neither too small
    # nor too big
    keepWidth = w > 5 and w < 50
    keepHeight = h > 45 and h < 65
    keepArea = area > 500 and area < 1500
```

Script:if the component passed the above condition then we construct a mask for this component and do the bitwise_or of the original image and the current

```
    # ensure the connected component we are examining passes all
    # three tests
    if all((keepWidth, keepHeight, keepArea)):
        # construct a mask for the current connected component and
        # then take the bitwise OR with the mask
        print("[INFO] keeping connected component {}".format(i))
        componentMask = (labels == i).astype("uint8") * 255
        mask = cv2.bitwise_or(mask, componentMask)
```

Script:to show result outside the for loop

```
# show the original input image and the mask for the license plate
# characters
cv2.imshow("Image", image)
cv2.imshow("Characters", mask)
```

Result: