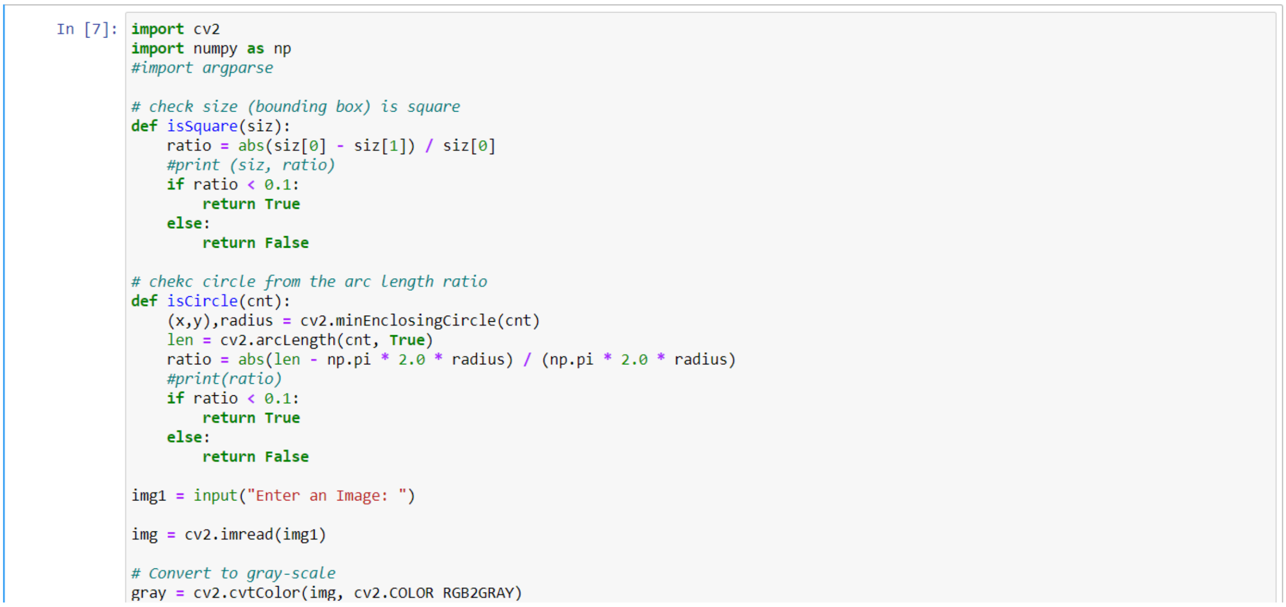
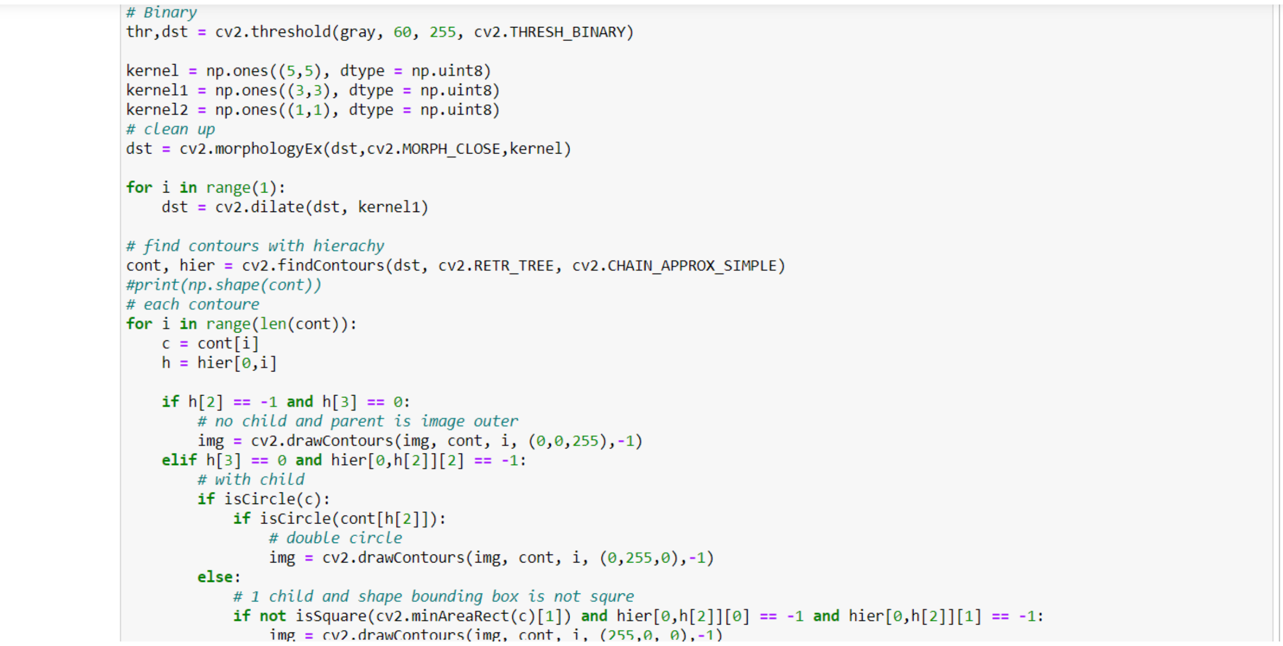
PROBLEM STATEMENT 6

PS6-1: Part Identification and Classification:

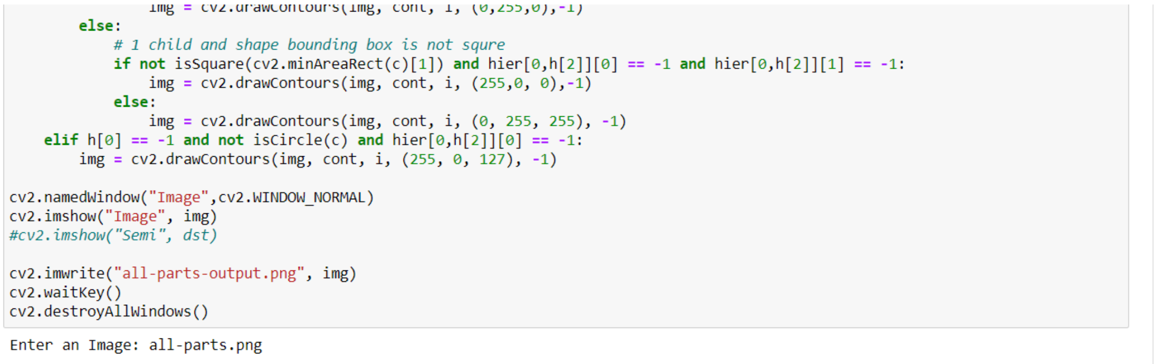
CODE:



```
In [7]: import cv2
        import numpy as np
        #import argparse

        # check size (bounding box) is square
        def isSquare(siz):
            ratio = abs(siz[0] - siz[1]) / siz[0]
            #print (siz, ratio)
            if ratio < 0.1:
                return True
            else:
                return False

        # chekc circle from the arc length ratio
        def isCircle(cnt):
            (x,y),radius = cv2.minEnclosingCircle(cnt)
            len = cv2.arcLength(cnt, True)
            ratio = abs(len - np.pi * 2.0 * radius) / (np.pi * 2.0 * radius)
            #print(ratio)
            if ratio < 0.1:
                return True
            else:
                return False

        img1 = input("Enter an Image: ")

        img = cv2.imread(img1)

        # Convert to gray-scale
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
```



```
# Binary
thr,dst = cv2.threshold(gray, 60, 255, cv2.THRESH_BINARY)

kernel = np.ones((5,5), dtype = np.uint8)
kernel1 = np.ones((3,3), dtype = np.uint8)
kernel2 = np.ones((1,1), dtype = np.uint8)
# clean up
dst = cv2.morphologyEx(dst,cv2.MORPH_CLOSE,kernel)

for i in range(1):
    dst = cv2.dilate(dst, kernel1)

# find contours with hierachy
cont, hier = cv2.findContours(dst, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
#print(np.shape(cont))
# each contoure
for i in range(len(cont)):
    c = cont[i]
    h = hier[0,i]

    if h[2] == -1 and h[3] == 0:
        # no child and parent is image outer
        img = cv2.drawContours(img, cont, i, (0,0,255),-1)
    elif h[3] == 0 and hier[0,h[2]][2] == -1:
        # with child
        if isCircle(c):
            if isCircle(cont[h[2]]):
                # double circle
                img = cv2.drawContours(img, cont, i, (0,255,0),-1)
        else:
            # 1 child and shape bounding box is not squre
            if not isSquare(cv2.minAreaRect(c)[1]) and hier[0,h[2]][0] == -1 and hier[0,h[2]][1] == -1:
                img = cv2.drawContours(img, cont, i, (255,0, 0),-1)
```

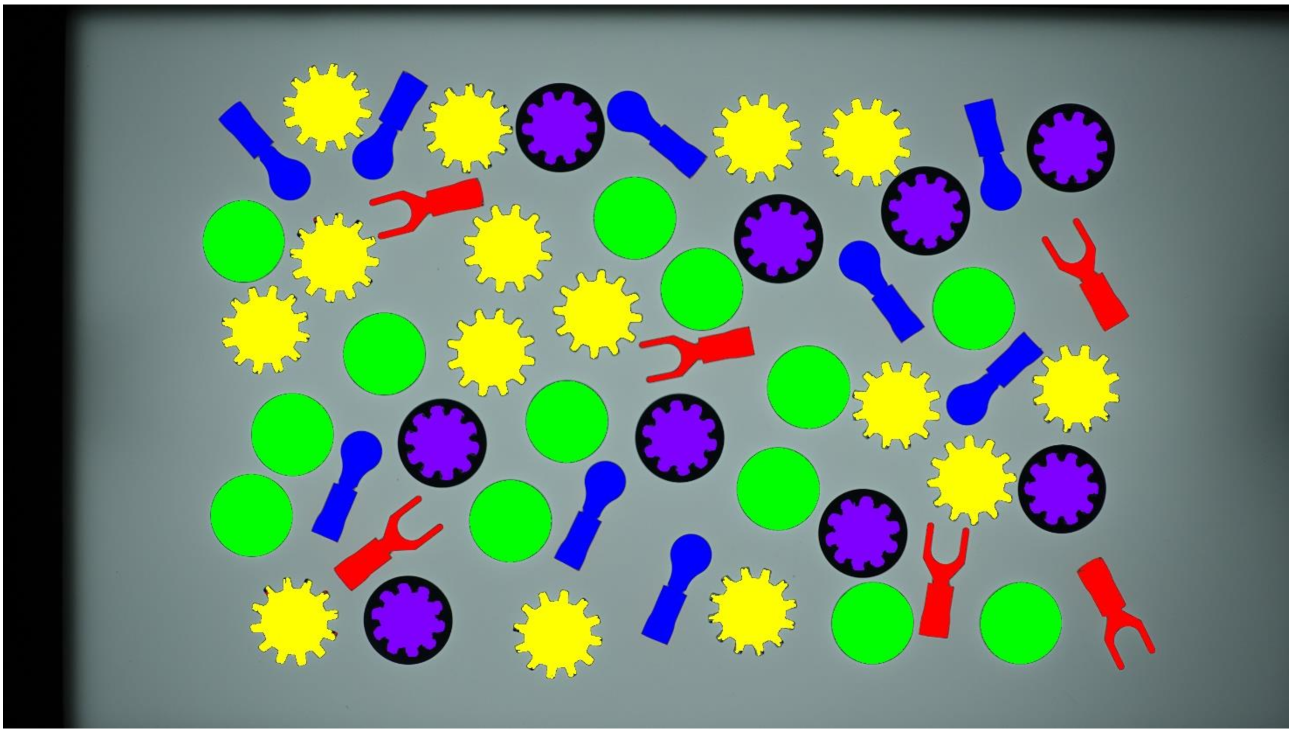


```
                img = cv2.drawContours(img, cont, i, (0,255,0),-1)
        else:
            # 1 child and shape bounding box is not squre
            if not isSquare(cv2.minAreaRect(c)[1]) and hier[0,h[2]][0] == -1 and hier[0,h[2]][1] == -1:
```

```
                img = cv2.drawContours(img, cont, i, (255,0, 0),-1)
            else:
                img = cv2.drawContours(img, cont, i, (0, 255, 255), -1)
    elif h[0] == -1 and not isCircle(c) and hier[0,h[2]][0] == -1:
        img = cv2.drawContours(img, cont, i, (255, 0, 127), -1)

cv2.namedWindow("Image",cv2.WINDOW_NORMAL)
cv2.imshow("Image", img)
#cv2.imshow("Semi", dst)

cv2.imwrite("all-parts-output.png", img)
cv2.waitKey()
cv2.destroyAllWindows()

Enter an Image: all-parts.png
```

OUTPUT:

All-parts-output.png



Modifications Made:

I studied the code posted by the professor and I observed that there he has taken all possible combinations into consideration. So, I tried all the possible combinations for the internal lock washer to get the desired output. Later I have used the else condition to make the external lock washers to yellow.

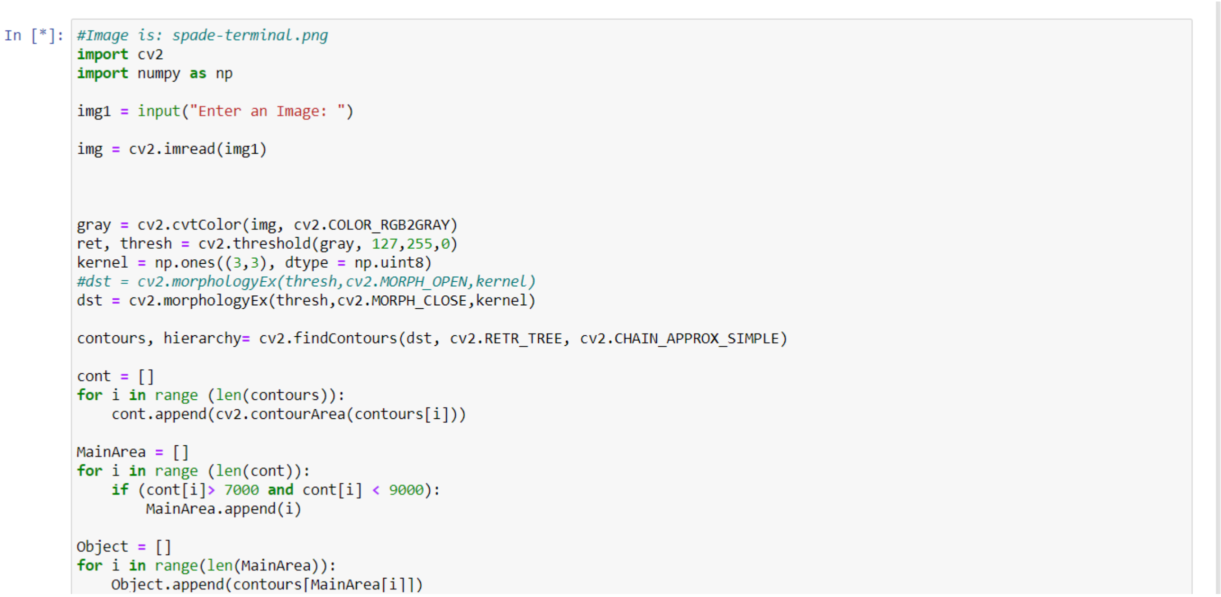
Operating System: Windows 10

IDE Used: Jupyter Notebook

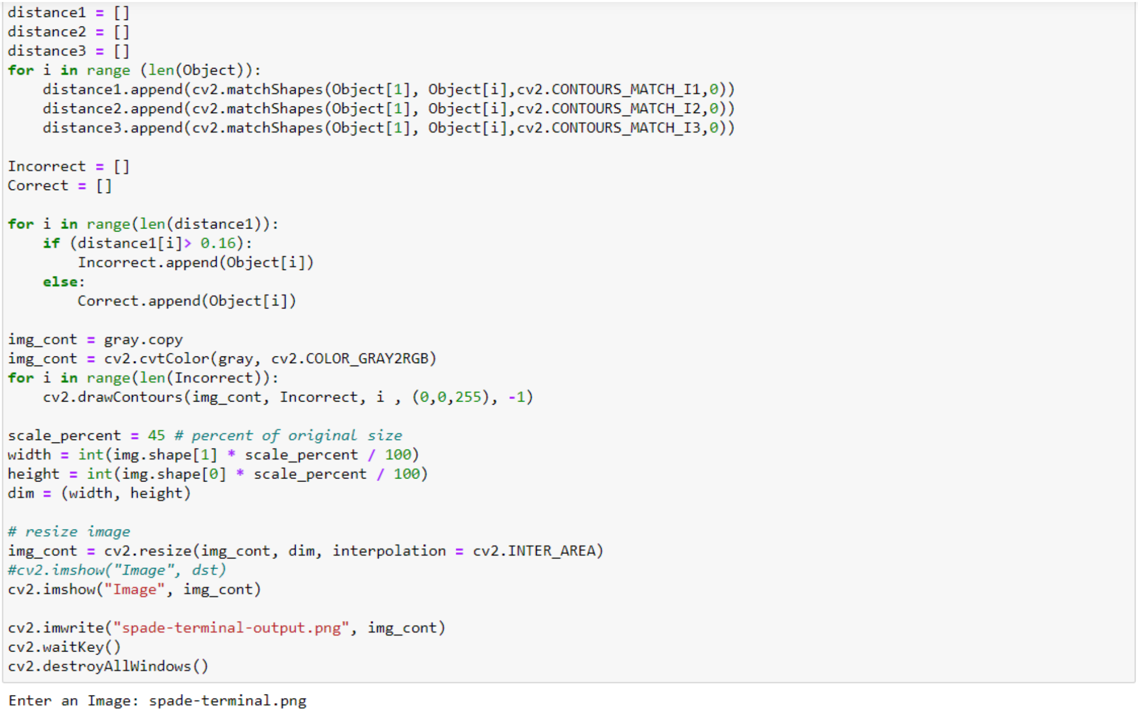
Number of Hours Spent: 4 hours

Ps6-2: Detecting Defective Parts

CODE:



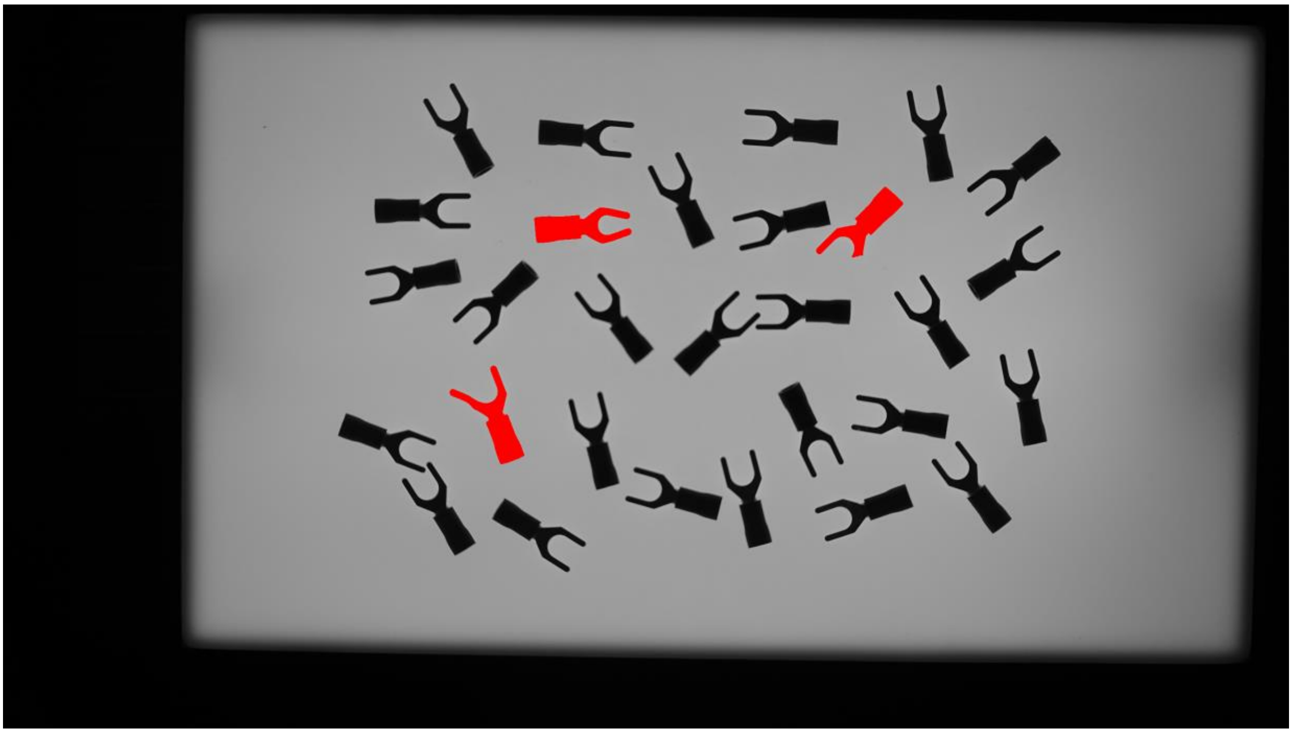
```
In [*]: #Image is: spade-terminal.png
        import cv2
        import numpy as np

        img1 = input("Enter an Image: ")

        img = cv2.imread(img1)


        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        ret, thresh = cv2.threshold(gray, 127,255,0)
        kernel = np.ones((3,3), dtype = np.uint8)
        #dst = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel)
        dst = cv2.morphologyEx(thresh,cv2.MORPH_CLOSE,kernel)

        contours, hierarchy= cv2.findContours(dst, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        cont = []
        for i in range (len(contours)):
            cont.append(cv2.contourArea(contours[i]))

        MainArea = []
        for i in range (len(cont)):
            if (cont[i]> 7000 and cont[i] < 9000):
                MainArea.append(i)

        Object = []
        for i in range(len(MainArea)):
            Object.append(contours[MainArea[i]])
```



```
distance1 = []
distance2 = []
distance3 = []
for i in range (len(Object)):
    distance1.append(cv2.matchShapes(Object[1], Object[i],cv2.CONTOURS_MATCH_I1,0))
    distance2.append(cv2.matchShapes(Object[1], Object[i],cv2.CONTOURS_MATCH_I2,0))
    distance3.append(cv2.matchShapes(Object[1], Object[i],cv2.CONTOURS_MATCH_I3,0))

Incorrect = []
Correct = []

for i in range(len(distance1)):
    if (distance1[i]> 0.16):
        Incorrect.append(Object[i])
    else:
        Correct.append(Object[i])

img_cont = gray.copy
img_cont = cv2.cvtColor(gray, cv2.COLOR_GRAY2RGB)
for i in range(len(Incorrect)):
    cv2.drawContours(img_cont, Incorrect, i , (0,0,255), -1)

scale_percent = 45 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)

# resize image
img_cont = cv2.resize(img_cont, dim, interpolation = cv2.INTER_AREA)
#cv2.imshow("Image", dst)
cv2.imshow("Image", img_cont)

cv2.imwrite("spade-terminal-output.png", img_cont)
cv2.waitKey()
cv2.destroyAllWindows()

Enter an Image: spade-terminal.png
```

OUTPUT:

Spade-terminal-output.png



Process:

First, I identified the contours by gauging the areas of the contours in the image given. Then I isolated the contours by knowing the threshold and the matchShapes() was used to know the distances. I recognized the distances and identified the high valued distances. Later I colored imperfect shapes to red by setting a threshold.

Operating System: Windows 10

IDE Used: Jupyter Notebook

Number of Hours Spent: 4 hours