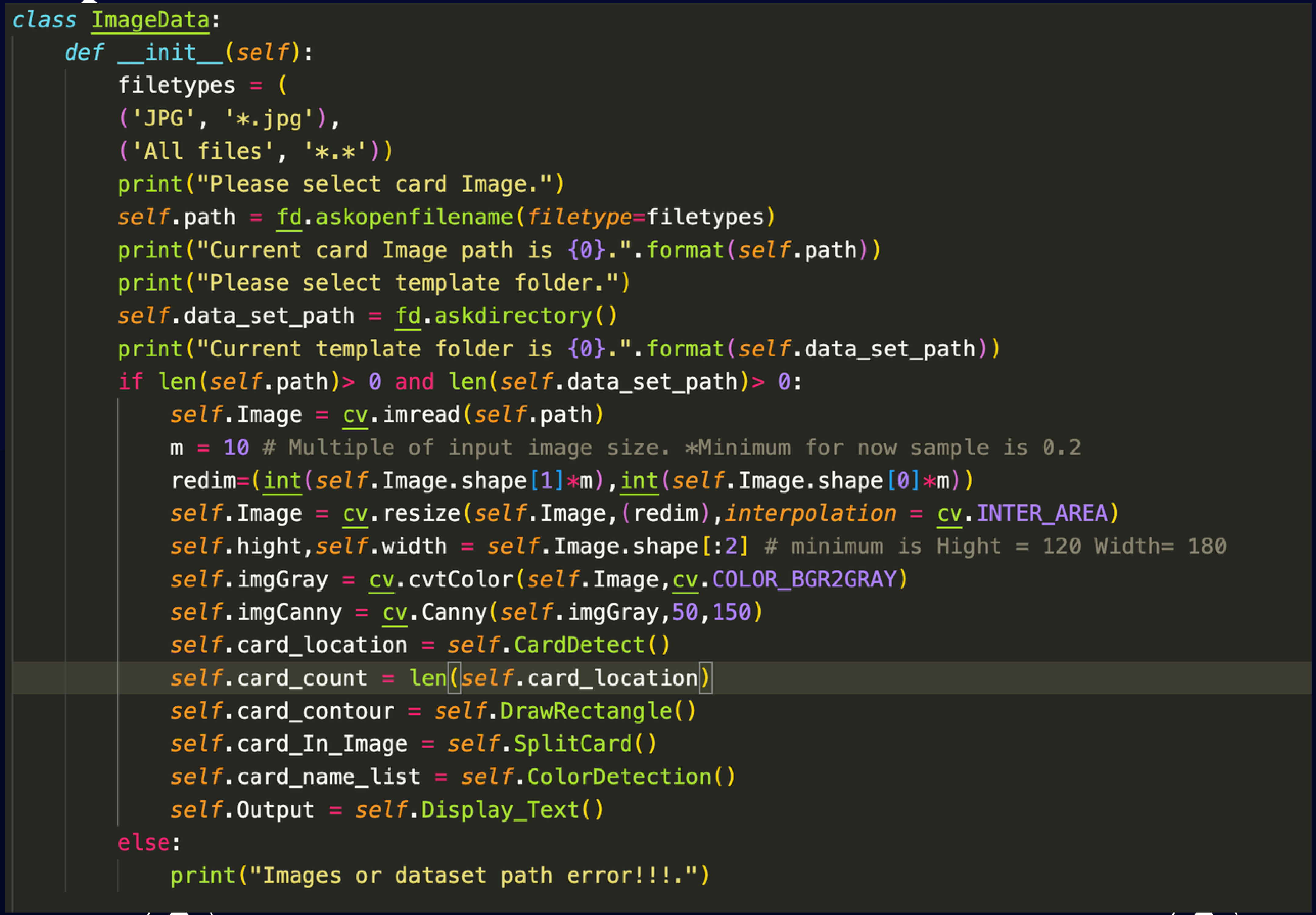
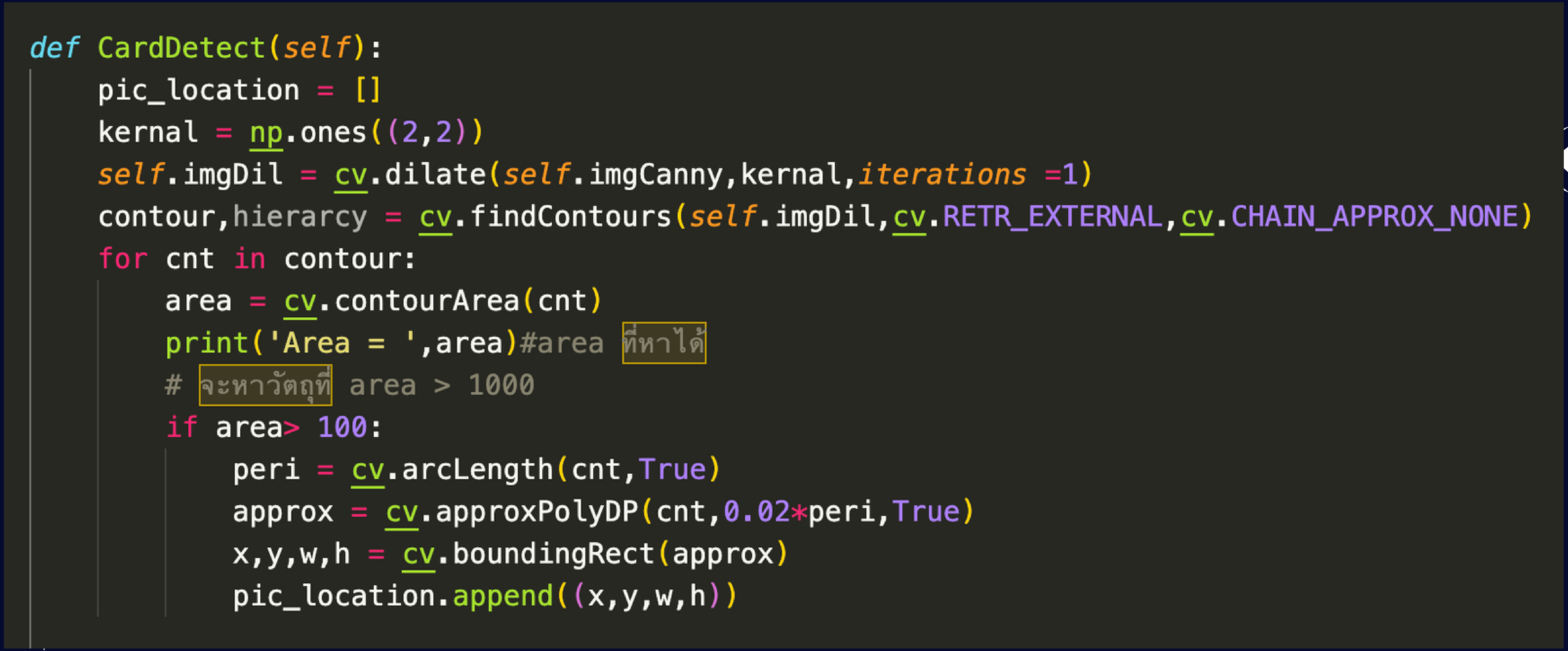
ASSIGNMENT 02

Image object



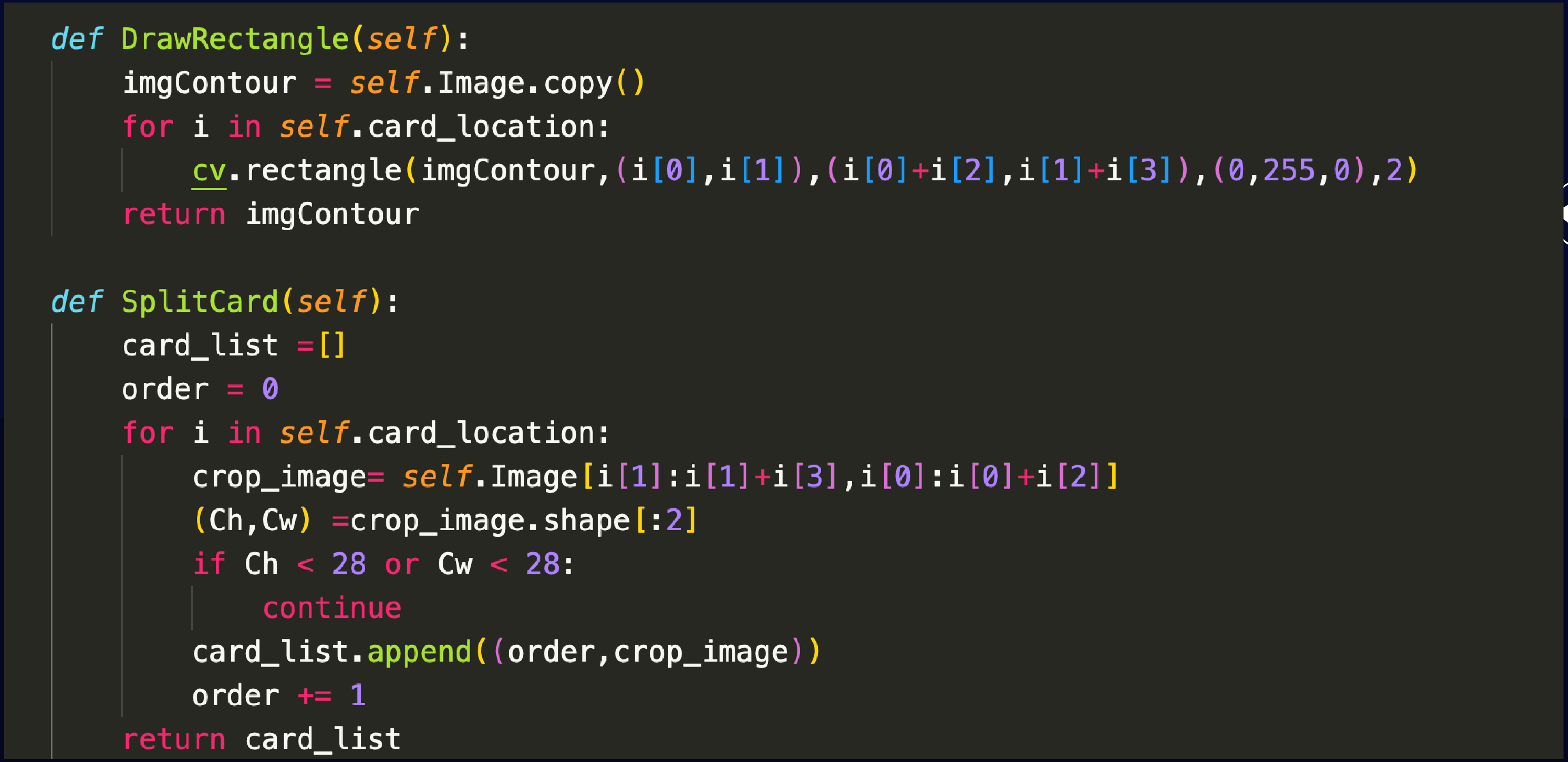
```python
class ImageData:
    def __init__(self):
        filetypes = (
        ('JPG', '*.jpg'),
        ('All files', '*.*'))
        print("Please select card Image.")
        self.path = fd.askopenfilename(filetype=filetypes)
        print("Current card Image path is {0}.".format(self.path))
        print("Please select template folder.")
        self.data_set_path = fd.askdirectory()
        print("Current template folder is {0}.".format(self.data_set_path))
        if len(self.path)> 0 and len(self.data_set_path)> 0:
            self.Image = cv.imread(self.path)
            m = 10 # Multiple of input image size. *Minimum for now sample is 0.2
            redim=(int(self.Image.shape[1]*m),int(self.Image.shape[0]*m))
            self.Image = cv.resize(self.Image,(redim),interpolation = cv.INTER_AREA)
            self.hight,self.width = self.Image.shape[:2] # minimum is Hight = 120 Width= 180
            self.imgGray = cv.cvtColor(self.Image,cv.COLOR_BGR2GRAY)
            self.imgCanny = cv.Canny(self.imgGray,50,150)
            self.card_location = self.CardDetect()
            self.card_count = len(self.card_location)
            self.card_contour = self.DrawRectangle()
            self.card_In_Image = self.SplitCard()
            self.card_name_list = self.ColorDetection()
            self.Output = self.Display_Text()
        else:
            print("Images or dataset path error!!!.")
```

Card Detection



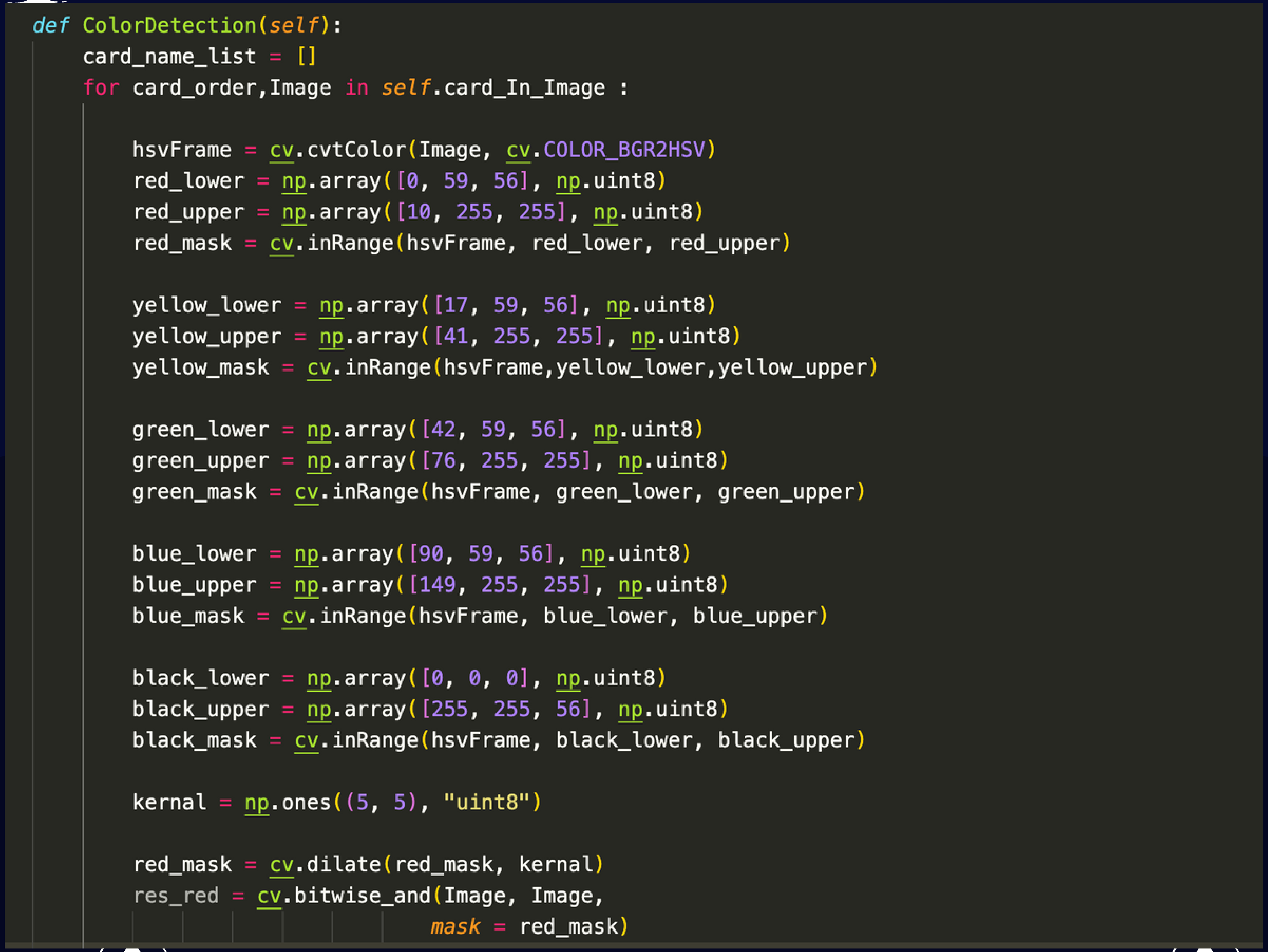
```
def CardDetect(self):
    pic_location = []
    kernal = np.ones((2,2))
    self.imgDil = cv.dilate(self.imgCanny,kernal,iterations =1)
    contour,hierarcy = cv.findContours(self.imgDil,cv.RETR_EXTERNAL,cv.CHAIN_APPROX_NONE)
    for cnt in contour:
        area = cv.contourArea(cnt)
        print('Area = ',area)#area ที่หาได้
        # จะหาวัตถุที่ area > 1000
        if area> 100:
            peri = cv.arcLength(cnt,True)
            approx = cv.approxPolyDP(cnt,0.02*peri,True)
            x,y,w,h = cv.boundingRect(approx)
            pic_location.append((x,y,w,h))
```

Draw & SplitCard



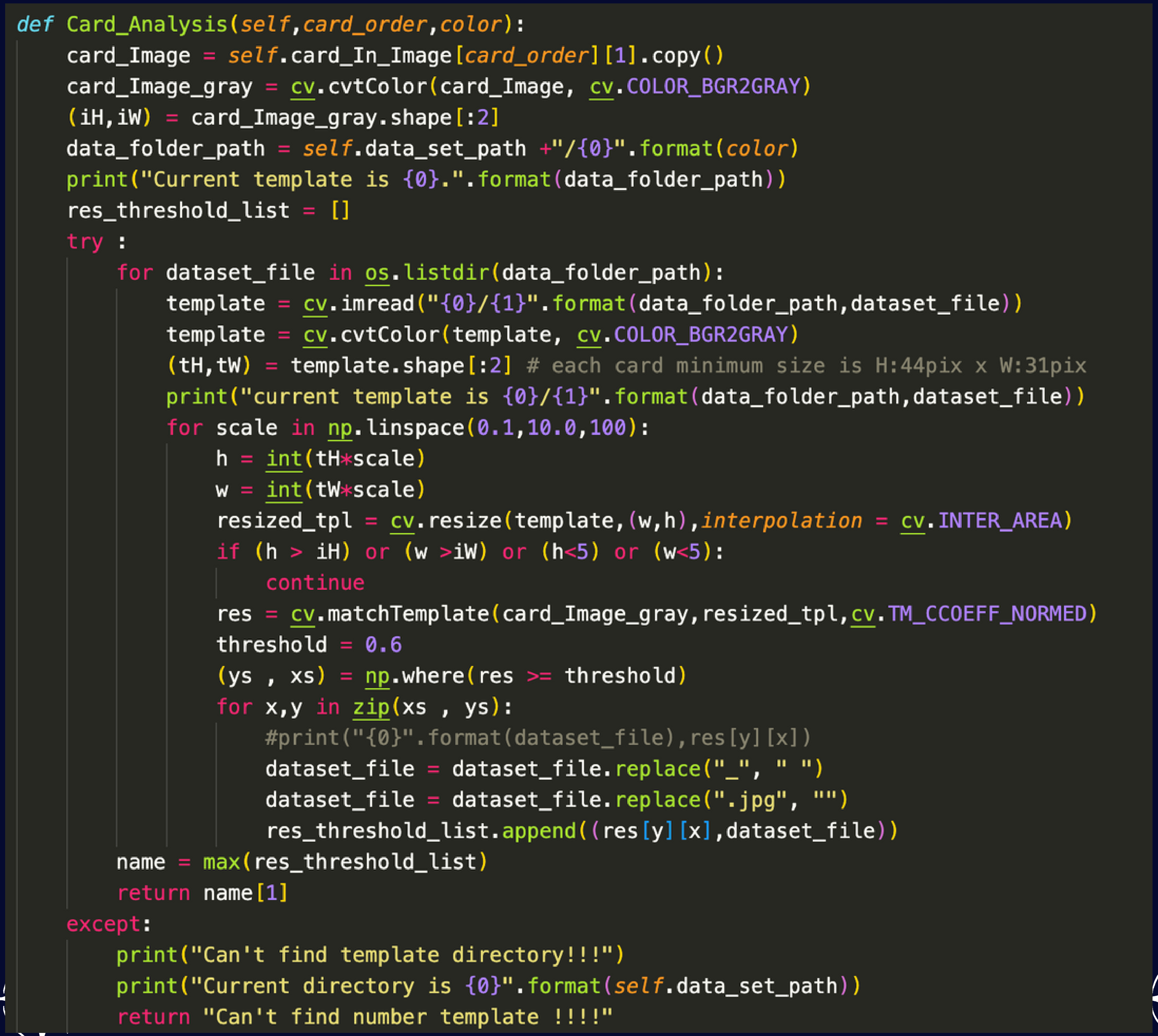
```python
def DrawRectangle(self):
    imgContour = self.Image.copy()
    for i in self.card_location:
        cv.rectangle(imgContour,(i[0],i[1]),(i[0]+i[2],i[1]+i[3]),(0,255,0),2)
    return imgContour

def SplitCard(self):
    card_list =[]
    order = 0
    for i in self.card_location:
        crop_image= self.Image[i[1]:i[1]+i[3],i[0]:i[0]+i[2]]
        (Ch,Cw) =crop_image.shape[:2]
        if Ch < 28 or Cw < 28:
            continue
        card_list.append((order,crop_image))
        order += 1
    return card_list
```

Color Detection



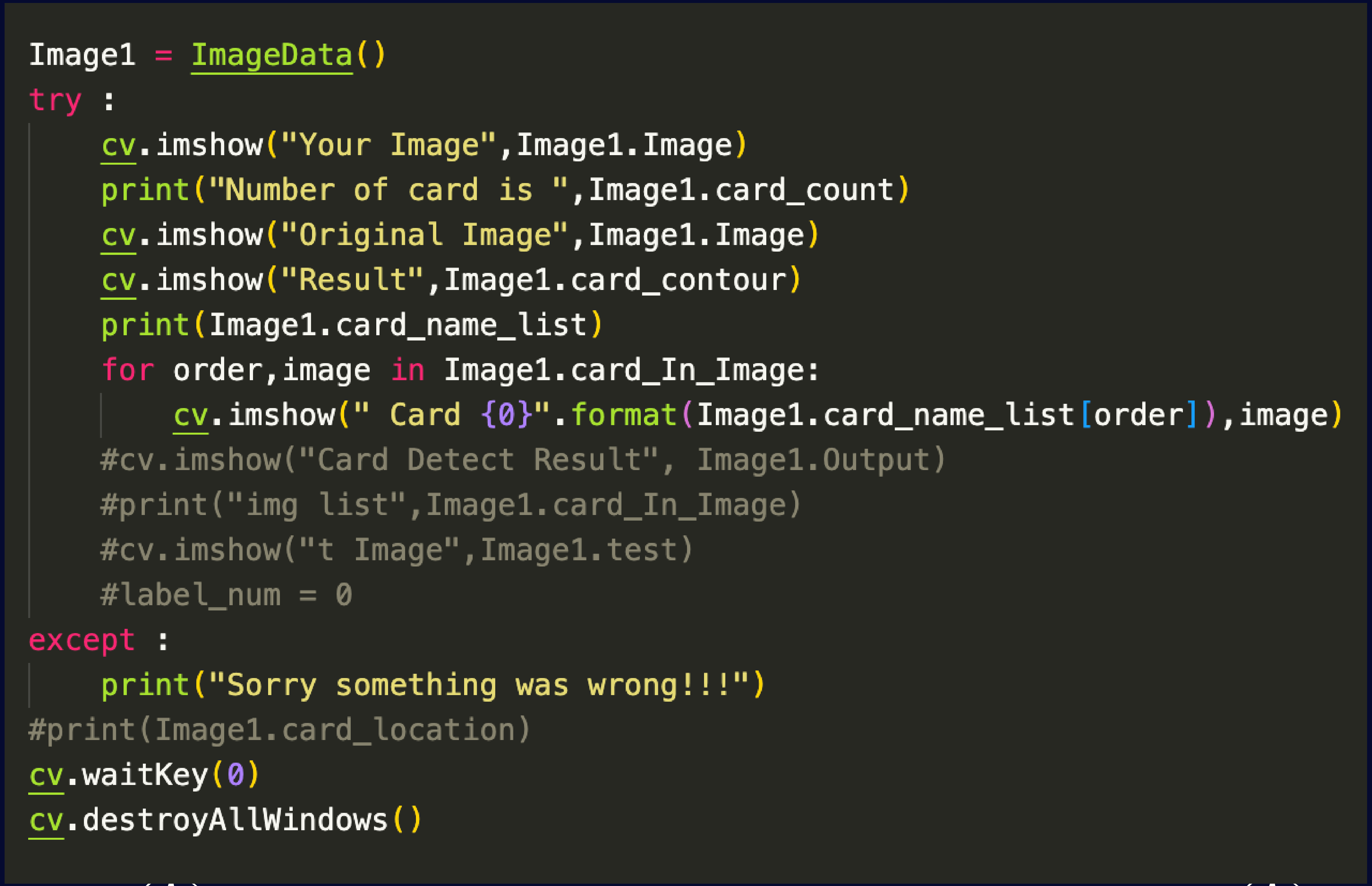
```python
def ColorDetection(self):
    card_name_list = []
    for card_order,Image in self.card_In_Image :

        hsvFrame = cv.cvtColor(Image, cv.COLOR_BGR2HSV)
        red_lower = np.array([0, 59, 56], np.uint8)
        red_upper = np.array([10, 255, 255], np.uint8)
        red_mask = cv.inRange(hsvFrame, red_lower, red_upper)

        yellow_lower = np.array([17, 59, 56], np.uint8)
        yellow_upper = np.array([41, 255, 255], np.uint8)
        yellow_mask = cv.inRange(hsvFrame,yellow_lower,yellow_upper)

        green_lower = np.array([42, 59, 56], np.uint8)
        green_upper = np.array([76, 255, 255], np.uint8)
        green_mask = cv.inRange(hsvFrame, green_lower, green_upper)

        blue_lower = np.array([90, 59, 56], np.uint8)
        blue_upper = np.array([149, 255, 255], np.uint8)
        blue_mask = cv.inRange(hsvFrame, blue_lower, blue_upper)

        black_lower = np.array([0, 0, 0], np.uint8)
        black_upper = np.array([255, 255, 56], np.uint8)
        black_mask = cv.inRange(hsvFrame, black_lower, black_upper)

        kernal = np.ones((5, 5), "uint8")

        red_mask = cv.dilate(red_mask, kernal)
        res_red = cv.bitwise_and(Image, Image,
                                 mask = red_mask)
```

Card Analysis



```python
def Card_Analysis(self,card_order,color):
    card_Image = self.card_In_Image[card_order][1].copy()
    card_Image_gray = cv.cvtColor(card_Image, cv.COLOR_BGR2GRAY)
    (iH,iW) = card_Image_gray.shape[:2]
    data_folder_path = self.data_set_path +"/{0}".format(color)
    print("Current template is {0}.".format(data_folder_path))
    res_threshold_list = []
    try :
        for dataset_file in os.listdir(data_folder_path):
            template = cv.imread("{0}/{1}".format(data_folder_path,dataset_file))
            template = cv.cvtColor(template, cv.COLOR_BGR2GRAY)
            (tH,tW) = template.shape[:2] # each card minimum size is H:44pix x W:31pix
            print("current template is {0}/{1}".format(data_folder_path,dataset_file))
            for scale in np.linspace(0.1,10.0,100):
                h = int(tH*scale)
                w = int(tW*scale)
                resized_tpl = cv.resize(template,(w,h),interpolation = cv.INTER_AREA)
                if (h > iH) or (w >iW) or (h<5) or (w<5):
                    continue
                res = cv.matchTemplate(card_Image_gray,resized_tpl,cv.TM_CCOEFF_NORMED)
                threshold = 0.6
                (ys , xs) = np.where(res >= threshold)
                for x,y in zip(xs , ys):
                    #print("{0}".format(dataset_file),res[y][x])
                    dataset_file = dataset_file.replace("_", " ")
                    dataset_file = dataset_file.replace(".jpg", "")
                    res_threshold_list.append((res[y][x],dataset_file))
        name = max(res_threshold_list)
        return name[1]
    except:
        print("Can't find template directory!!!")
        print("Current directory is {0}".format(self.data_set_path))
        return "Can't find number template !!!!"
```

Show Output



```python
Image1 = ImageData()
try :
    cv.imshow("Your Image",Image1.Image)
    print("Number of card is ",Image1.card_count)
    cv.imshow("Original Image",Image1.Image)
    cv.imshow("Result",Image1.card_contour)
    print(Image1.card_name_list)
    for order,image in Image1.card_In_Image:
        cv.imshow(" Card {0}".format(Image1.card_name_list[order]),image)
    #cv.imshow("Card Detect Result", Image1.Output)
    #print("img list",Image1.card_In_Image)
    #cv.imshow("t Image",Image1.test)
    #label_num = 0
except :
    print("Sorry something was wrong!!!")
#print(Image1.card_location)
cv.waitKey(0)
cv.destroyAllWindows()
```

Thank You