Harvester Bot
Task 1A – Basic Image Processing Operations

Goal:
1. To learn Python and use it to:
- Identify shapes
- Identify colours of shapes
- Output a .csv file with an image of objects with contours and labels

Please find the *task1A_main.py* file in the folder named *Test Images*. Modify the sections of *task1A_main.py* to accomplish the following:

Given:
1. We have given a set of five images named: **test1.png, test2.png, test3.png, test4.png and test5.png** in a folder named **Test Images**. This folder is a sub-folder inside the **Task 1A** folder as shown in Figure 1.

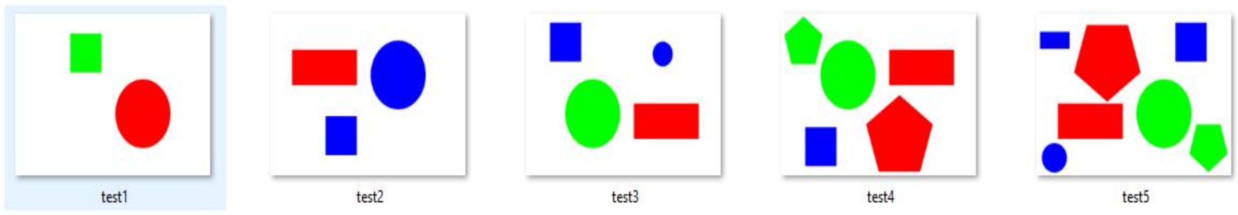


Figure 1: Test Images

Every image has a certain number of objects. Each object is defined by two features, viz. **Colour and Shape**. Objects in a given image vary in colour and shape and may appear multiple times in an image. An example image shown in Figure 2.

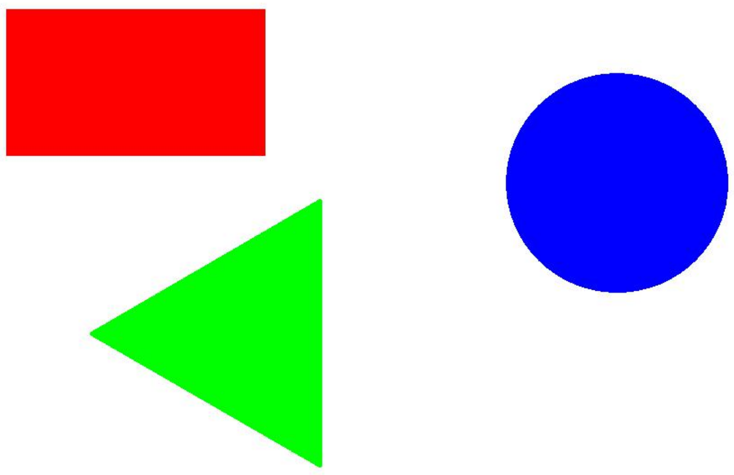


Figure 2: Example Image

Problem Description:

For each image in the **Test Images** folder, add your methods or functions to the Python script file in order to find the Colour and Shape of each object in that image.

Your code should return a List of Lists; where each individual List contains - image name followed by List of **Colour-Shape** of each object contained in that image, in a comma separated values List.

You have to find out the **Colour and Shape** of each object in the image and concatenate them into one string and then store it in the List.

Considering the image **"example.png"** shown in Figure 2, output would be as follows:

[“example.png”, [“Red-Rectangle”], [“Green-Triangle”], [“Blue-Circle”]]

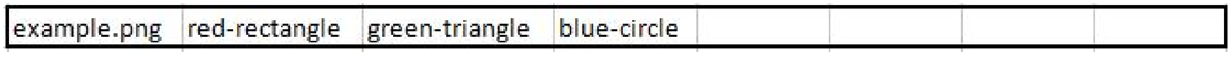
You should also return an image of objects with **Contours** and **Labels**. A **Contour** in Image processing is simply an outline along the boundary of the Object. Contours are used for shape analysis, object detection and recognition. **Label** is text with color and shape of the object. An example is shown in Figure 4.

Note:

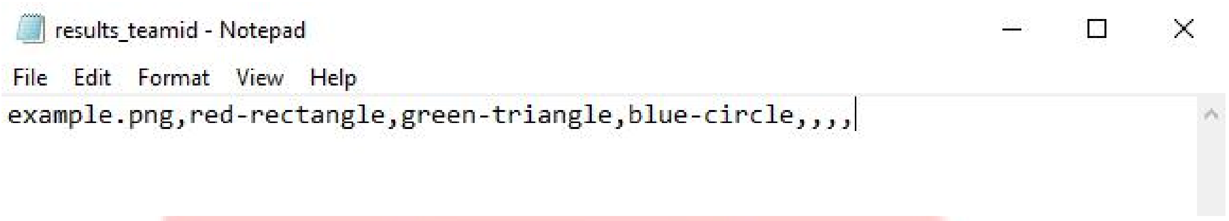
1. An object will be one of the three colours, viz. **Red, Green and Blue.**
2. An object will be one of the five shapes, viz. **Triangle, Rectangle, Square, Pentagon and Circle**.

Required Output:

1. The Python script file named *task1A_main.py* with the function ***"main"*** populated with your logic of solving the task.
2. A **.csv** file with the Lists of each image on each row i.e. five rows of data for a total of five test images in the Test Images folder. For our example, .csv file would look like Figure 3.



example.png	red-rectangle	green-triangle	blue-circle				



results_teamid - Notepad

File Edit Format View Help

```
example.png,red-rectangle,green-triangle,blue-circle,,,,
```

Figure 3: CSV output

3. For each image in the *Test Images* folder create a corresponding output image with Contours and Labels; each output image should be named as **output1.png** (for test1.png), **output2.png** (for test2.png) and so on, in the *Test Images* folder itself.

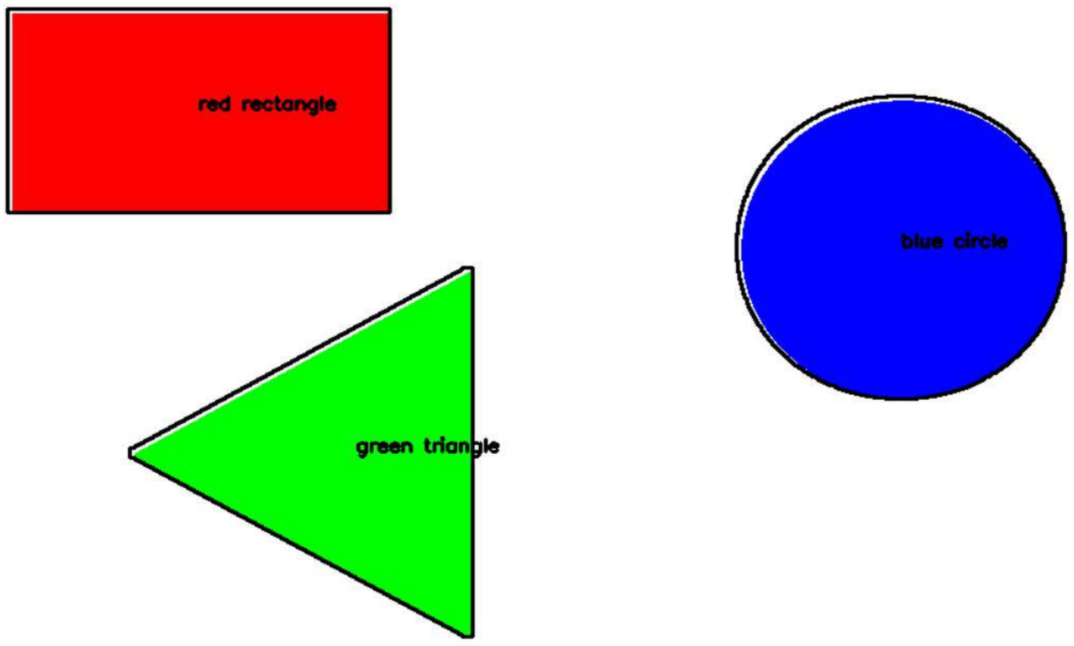


Figure 4: Output Image with Contours and other properties enlisted, overlaid on the object

To do:

1. Open **task1A_main.py** located in the folder named **Test Images**. It has two functions:
   - ◆ **main():** It returns the required Lists for the **writecsv()** function.
     It should also write the Output Images and save it in the *Test Images* folder as explained in point 3 under *"Required Output"* section above.
   - ◆ **writecsv():** This function expects two parameters as arguments and writes the elements of List of Lists one at a time into a **"results1A_teamid.csv"** file. **Do not edit this function. However you can edit the global variable - "filename" with your teamid in the *task1A_main.py* Python code file.** Please use your eYRC Team ID while actually naming the file.
     IMPORTANT: Do NOT change names of any of these functions.

Rules:

1. You need to write a **generic program**. Your code should be capable enough to detect **any number** of objects in an image and extract their properties. In addition your code will be tested on several **undisclosed images when you submit your code.**

2. Use basic knowledge of geometry to differentiate between the shapes of objects.

3. **Matching of images should be independent of orientation of the objects.** Objects in *image* might be rotated w.r.t. each other.

Result of rotation:

- For Circles , there is no difference in the images. Rotation does not affect these images.

- Triangles, Rectangles, Squares and Pentagons can be in any orientation. An example rotation for a Triangle is shown in Figure 6 below:

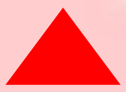


Figure 6a. Original orientation

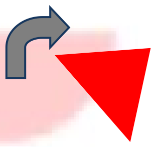


Figure 6b. Rotated Triangle

4. In case an object is not present in the image, **return nothing or an empty List.**

Happy Learning!

All The Best!!!