Task 1 – Match the Objects

Please find the *task1_main.py* file in the folder named *Task_Description*. Modify the *task1_main.py* to accomplish the following:

Given:

1. A ***board*** - having 9 **Objects** in 9 **Positions**. Figure 1 shows a sample *board*. Positions on "*board*" are numbered as shown in Figure 3a.
2. 5 images of ***containers*** - each having 16 **Objects** in 16 **Positions**. Figure 2 shows a sample "*container*". Positions on *container* are numbered as shown in Figure 3b.

Note: Each Object is defined by three features, viz. **Shape, Size and Color**.

"board" and *"container"* are given as image files in the folder /test_images

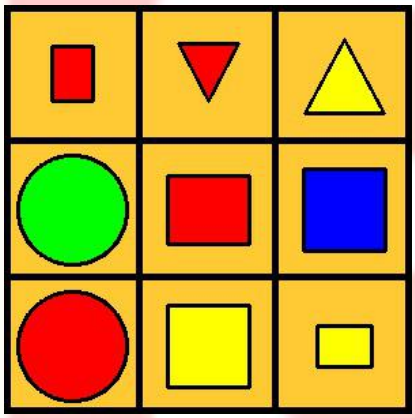

Figure 1: board

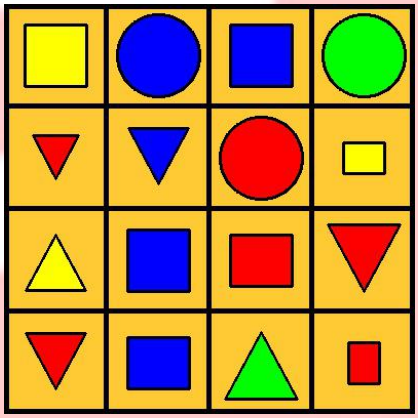

Figure 2: container

1	2	3
4	5	6
7	8	9

Figure 3a

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Figure 3b

Figure 3: Position numbering convention

Problem Description:

For a given image of a *"board"* and a *"container"*, write a python program to solve the following:

1. Find the color and shape of each object in the *"board"*.

Your code should return a *python list* having 9 *python tuples*. Each tuple has three elements, first element is the position number on the "*board",* second element is color of object viz, "***red***", "***blue***", "***green***" and "***yellow***", third element is shape of object viz, "***Triangle***", "***4-sided***" and "***Circle***".

Example: Considering Figure 1, the output should be:

[(1, "red", "4-sided"), (2, "red", "Triangle"), (3, "yellow", "Triangle"), (4, "green", "Circle"), (5, "red", "4-sided"), (6, "blue", "4-sided"), (7, "red", "Circle"), (8, "yellow", "4-sided"), (9, "yellow", "4-sided")]

2. For each object in the "*board"*, find the matching object in the "*container".* Objects are considered a **match** only if all the three features of an object viz, **Shape**, **Size** and **Color** in the "*board"* matches exactly with an object in the "*container".*

Your code should return a *python list* having 9 *python tuples*. Each tuple has two elements, first element is the position number on the *"board"* and second element is the position number of the **match** on the *"container"*. Figure 3 shows number convention to be followed for creating the tuple.

Example: Expected output for sample *"board"* and "*container"* shown in Figure 1 and Figure 2 :

[(1, 16), (2, 5), (3, 9), (4, 4), (5, 11), (6, 3), (7, 7), (8, 1), (9, 8)]

Note:

1. Both the output lists must be ordered such that tuples are arranged in **ascending order of position numbers in the "*board"*.**
2. **object color** and **shape** must be written as shown in the example.

To do/Instructions:

1. Open *task1_main.py* in editor of your choice.

2. Code file has a function called *main()*. Do not change name of this function. It expects two arguments, *board_filename* and *container_filename*. This function returns two python lists, *board_objects* and *output_list*. *board_objects* should store information about "*board"* (Problem Description 1) and *output_list* should store *information about the* match (Problem Description 2). Do not change names of any of the return variables.

3. Add required variables, functions according to your requirement to get the desired output.

4. Follow the guidelines mentioned in *task1_main.py* to complete the above task.

5. After completing your program, run the file *test_task1_main.py.* This is a python program that will check your program to check correctness of code in finding desired output for the given "*board"* and "*container" combinations*. You are required to run this program without making any changes in it.

6. In order to run *test_task1_main.py*, you have to install a python code testing framework called *nose*. This can be done by going to terminal/command prompt and type ***pip install nose***

7. On executing *test_task1_main.py, you should get output as shown in Figure 4.*

8. You should ideally get "OK" for all eleven test cases. Information about Test cases are mentioned at the end of this document.

9. Submission:
   a) You must save screen-shot of terminal window showing output for all test cases as "*output.png"*
   b) Save **program files** and ***"output.png"*** in folder called Task-1
   c) Compress the folder into zip format. Upload the **Task_1.zip** on the Portal.

```
% python test_LM_main_simple.py
nose.config: INFO: Ignoring files matching ['^\\.', '^_', '^setup\\.py$']
test_LM_main_simple.test_main_board_shapes ... ok
test_LM_main_simple.test_final_output_list('test_images/board_8.jpg', 'test_images/container_1.jpg', 0) ... ok
test_LM_main_simple.test_final_output_list('test_images/board_8.jpg', 'test_images/container_2.jpg', 1) ... ok
test_LM_main_simple.test_final_output_list('test_images/board_8.jpg', 'test_images/container_3.jpg', 2) ... ok
test_LM_main_simple.test_final_output_list('test_images/board_8.jpg', 'test_images/container_4.jpg', 3) ... ok
test_LM_main_simple.test_final_output_list('test_images/board_8.jpg', 'test_images/container_5.jpg', 4) ... ok
test_LM_main_simple.test_final_output_list_if_sorted('test_images/board_8.jpg', 'test_images/container_1.jpg', 0) ... ok
test_LM_main_simple.test_final_output_list_if_sorted('test_images/board_8.jpg', 'test_images/container_2.jpg', 1) ... ok
test_LM_main_simple.test_final_output_list_if_sorted('test_images/board_8.jpg', 'test_images/container_3.jpg', 2) ... ok
test_LM_main_simple.test_final_output_list_if_sorted('test_images/board_8.jpg', 'test_images/container_4.jpg', 3) ... ok
test_LM_main_simple.test_final_output_list_if_sorted('test_images/board_8.jpg', 'test_images/container_5.jpg', 4) ... ok

----------------------------------------------------------------------
Ran 11 tests in 0.224s

OK
```

Figure 4: Output - After running *test_task1_main.py*

Rules:

1. **Matching of images is independent of orientation of the objects.** However objects in "*board" and "container"* will be rotated only in multiples of 90 degrees w.r.t. each other.
   Result of rotation:
   - For Circles and Squares, there are no differences in the images. Rotation does not affect these images.
   - For Rectangles, the following two orientations are possible, which are **equivalent**.

   - Triangles can be in any of the following four possible orientations. All these four shapes are **equivalent**.

2. In case object is not present in the *"container"*, **return 0** as the **second element of tuple.**

3. In case multiple matches are found in the *"container"*, **return lowest numbered position** of match in the "*container*" as **second element of tuple**.

4. You need to write a **generic program**. Note that we have provided five (05) "*container*" images for testing purposes. In addition *your code will be tested on several* ***private*** *"container"* ***images when you submit your code.***

5. Team is encouraged to design their own test *"container"* images.

Information About Test Cases:

There are a total of 11 test cases used to check your program. Following are details of each test case:

Testcase 1: Check Expected Output 1 i.e. Position, Color and Shape of 9 objects. Your code will pass this case if you have

a) saved output as **list of tuples**,
b) used **correct names** for color and shapes
c) Sorted the list in **ascending order of position numbers in the "*board*".**

Testcases 2-6: Check Expected Output 2 by running your code for all "*container*" images provided to you and check if *output_list* returned by main() function has all objects correctly matched.

Testcases 7-11: It will check if *output_list* is saved in **ascending order of position numbers in the "*board*".**

Happy Learning

All The Best!!!