Task Description

Task 1

In the *"Task1"* folder you will find a python file called task.py. The task has four steps:

1. Please describe concisely in your own words what the program is doing.

2. The current implementation of the TaskHandler is not really maintainable and extensible. We would like you to refactor it. After you are done it should still be able to run the scenario1.txt file without any errors.

3. Please add the new task name "count" which will calculate the total number of values of a task. After this step also file sceneraio2.txt should run without any issues.

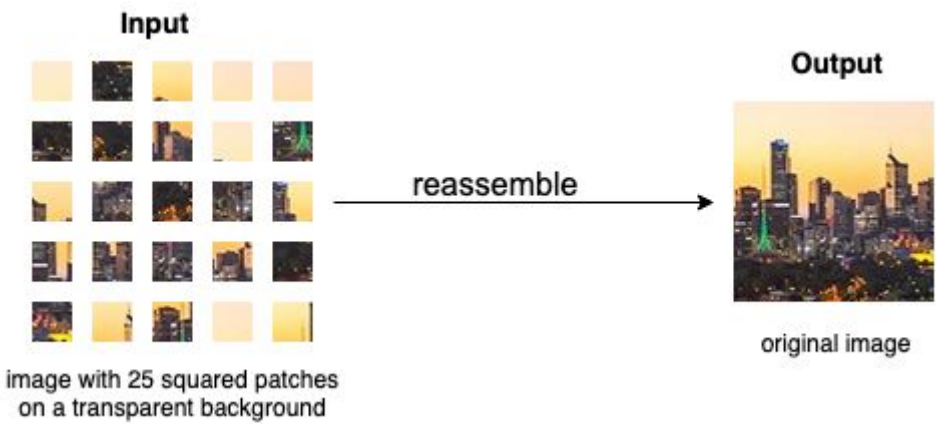
4. In the final step we would like you to make the handler more robust. In the case a task name is received which is not handled by the handler it shouldn't crash. Instead it should just give out a message notifying about the task name and continue with other tasks. After this step all scenario files should run through without any errors.

Please only submit the final solution which is able to run all files.

Task 2

The starting point is an image with a size of nxn with m squared patches inside. The small patches are randomly mixed, but put together in the right place they form a picture. Your task will be to collect these patches and reassemble them back to the original image. The parts are not rotated and the top left patch already has the right position.

We want you to implement a solution for cutting out the patches and then reassemble them back into the original image. This requires a method that checks whether the patches match. We are aware that there is no perfect solution for this problem, so it is okay if some very similar patches are swapped. But the majority of the patches should be placed at the right position inside the image.



Specification:

- the picture and the small patches are always square
- top left patch (the corner) is already at the right position
- patches are not rotated
- the patches are provided as a single image with an alpha channel

Guidelines:

- the solution should be written in Python 3.x.
- the solution may use any open-source library for image processing
- the solution must not contain any open-source library which has an implementation of a puzzle solver inside
- the solution can only take the image with the mixed patches and the output path of the reassembled image as an input
- the solution must work with any number and sizes of image patches
- the solution should be time efficient and not produce any memory leaks, regardless of the input size

Questions:

1. Which method did you use to determine the neighbouring patches and why?
2. What complexity in terms of time and memory does your solution have?

Submission

The solution to task 1 and 2 should be kept separately. Therefore two directories should be created: task_1 and task_2, which contain the respective solutions. The answer to the questions in task 1 can be added in a normal text file and put inside the corresponding directory.
Please combine both directories to one single zip archive.