On the structure of this practise, and on how to submit the solutions

This practise consists on exercises identified by number. You need to upload in canvas **one single `.zip` file** with the filename

`{your-full-name}-graded-practise-1.zip`.

This zip file should contain **only** directories named `exercise01`, `exercise02`, ..., `exercise10`, etc. The names of the directories **have to be** the lower case word "exercise" followed (without spaces) by a 2-digits number indicating the number of exercise.

There should be one file named `solution.py` and a directory named `output` (if applicable). The file `solution.py` should contain the solution of the corresponding exercise. The directory `output` is a directory that you need to create, if applicable, and write there the outputs.

Each exercise has a *number of points*. While the perfect score is 100 points, you can try to score *more* than 100 by implementing the *bonus* problems. The BONUS problems are optional, you can *exceed* the perfect score by implementing them.

Exercise 1 - 20 points - directory: `exercise01`

Implement, in the file `exercise01/solution.py`, a function that receives a path to an existing image, a `NEW_HEIGHT` in pixels, a `NEW_WIDTH` in pixels, and outputs a new image with the specified number of pixels of width and height. Use regular sampling and billinear interpolation.

The output image should be written to the directory `exercise1/output`, and it should be named the same as the input image, with the suffix `_{new_height}_{new_width}` in the name.

For example, it should be possible to execute

```
python exercise01/solution.py lenna.png 100 100
```

and the command should resize the image `lenna.png` to 100×100 pixels and should write the result in `exercise01/output/lenna_100_100.png`

Expected results

The outputs that are expected in the `exercise01/output` directory are the outputs generated by the following commands:

```
python exercise01/solution.py lenna.png 100 100
python exercise01/solution.py lenna.png 100 200
python exercise01/solution.py lenna.png 200 100
python exercise01/solution.py lenna.png 150 250
```

Exercise 2 - 20 points - directory: `exercise02`

Implement, in the file `exercise02/solution.py`, a function that receives `H_PIXELS`, `W_PIXELS`, `H_CELLS`, `W_CELLS` and produces an image of `H_PIXELS` × `W_PIXELS` pixels, which consists of a board of `H_CELLS` of height and `W_CELLS` of width.

Consider the list of colors `colors = [black, white, red, green, blue, yellow]`, and paint the cell i, j of the board with the color `colors[ (i + j) % len(colors)]`

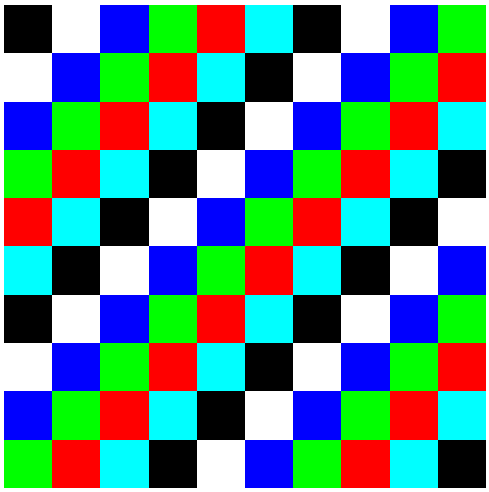
The output image should be written to
`exercise02/output/board_{H_PIXELS}_{H_CELLS}_{W_PIXELS}_{W_CELLS}.png`

Expected results

The outputs that are expected in the `exercise01/output` directory are the outputs generated by the following commands:

```
python exercise02/solution.py 100 100 10 10
python exercise02/solution.py 100 200 2 2
python exercise02/solution.py 200 100 1 1
python exercise02/solution.py 150 250 10 10
```

For example, the output of the first of 4 commands above should look like this:



Exercise 3 - 20 points - directory: `exercise03`

Based on the exercise 2 of the class number 5, implement a python script that displays the input image `lowcontrast.png` and, in the same window, one slider that variates the contrast when moving it. Optimize the implementation in order for the change of contrast to happen quick enough such that the moving of the slider is *natural*. There should not be any perceptible delay in updating the contrast when moving the slider.

Expected results

The following command:

```
python exercise03/solution.py
```

should create a window that displays the image mentioned, and a slider. Moving the slider should change the contrast of it.

Exercise 4 - 20 points - directory: `exercise04`

Following the explanation available in the material of the class 6, implement a change in the color scales of the pixels of `lenna.png`, using a mask that consists of white background and a circle centered in the center of the image, tangent to all the four sides of the image (the image is square), filled with color blue. Save the resulting image in `exercise04/output/lenna-colorscale.png`.

Exercise 5 - 20 points - directory: `exercise05`

Based on the exercises 1 and 2 of the class 8, filter the image `lenna.png` with filters box, bartlet and Gaussian of orders 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25.

Do it both with the original image in RGB and in grayscale.

Create a simple HTML page containing a table that shows all the images at the same time. The columns of the table should be: `original`, 3×3, 5×5, 7×7, ..., 25×25. The rows of the table should be *box-grayscale*, *box-rgb*, *bartlett-rgb*, *bartlett-grayscale*, *Gaussian-rgb*, *Gaussian-grayscale*, *laplacian-rgb*, *laplacian-grayscale*.

For the laplacians, fill only the cells 3×3 and 5×5.

All the files (included the HTML page) should be, as explained before, in the directory `excercise08/output`.

Exercise 6 - BONUS EXERCISE - 40 points - directory: `exercise06`

We created during the class 3, in the exercise 2.1, a simple animation in which one circle was moving and bouncing against the borders of the image.

Create an animation with multiple circles that move and can bounce not only against the borders of the image, but also against other circles. Make tham have different speeds and different colors.

Expected results

The command `python exercise06/solution.py` should create a window that displays the animation described.