# Module 1 – Report

The script has to be launch with the following structure ;

python watermark.py *path_to_img_src path_to_watermark*

At launch, a window with the given picture opens and different functionalities are available to the user.

**Features :**

1. The user can **click** on a specific place of the picture to **dispose the watermark** on this position
2. The user can use the keys **'0' to '9'** to adjust the **transparency** of the watermark
3. The user can use the keys **'(' and ')'** to respectively **downscale and upscale** the watermark
4. The user can use the key **'r'** to **reset** the picture
5. The user can use the key **'s'** to **save** the picture
6. The user can use the key **'q'** to **close** the window.

The method **avg_value** takes as argument the position (x,y) of the user's click. It measures the average value (gray scale) of the pixels in the area of the watermark of the picture and return it. If the value is below 127 (darker) it will display the watermark in white, if the value is above 127 (brighter) it will be displayed in black.

The method **watermarking** takes as arguments the position of the user's click, the level of transparency and a bool indicating if the watermark has to be white or black. On every pixel where the watermark has been placed and for all the channels (RGB) we calculate a new pixel value based on the transparency level.

If it's black :

```
image[imy, imx, d] = image[imy, imx, d] * (9 - transparent) / 9
```

for transparency level 0 the level of the pixel is still the same, for level 9 it's set to 0 (black)

if it's white :

```
image[imy, imx, d] = image[imy, imx, d] + (255 - image[imy, imx, d]) *
(transparent / 9)
```

for transparency level 0 the level of the pixel is still the same , for level 9 it's set to 255 (white)

In both cases the in-between levels are linear.

The other methods are either trivial or GUI related.