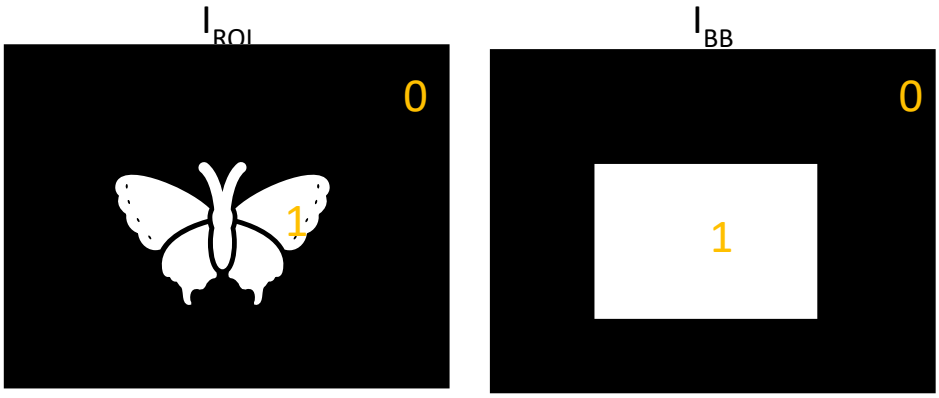
# Lab 01: elements of corrections

Q1: 100% of butterfly pixels are in its bounding box



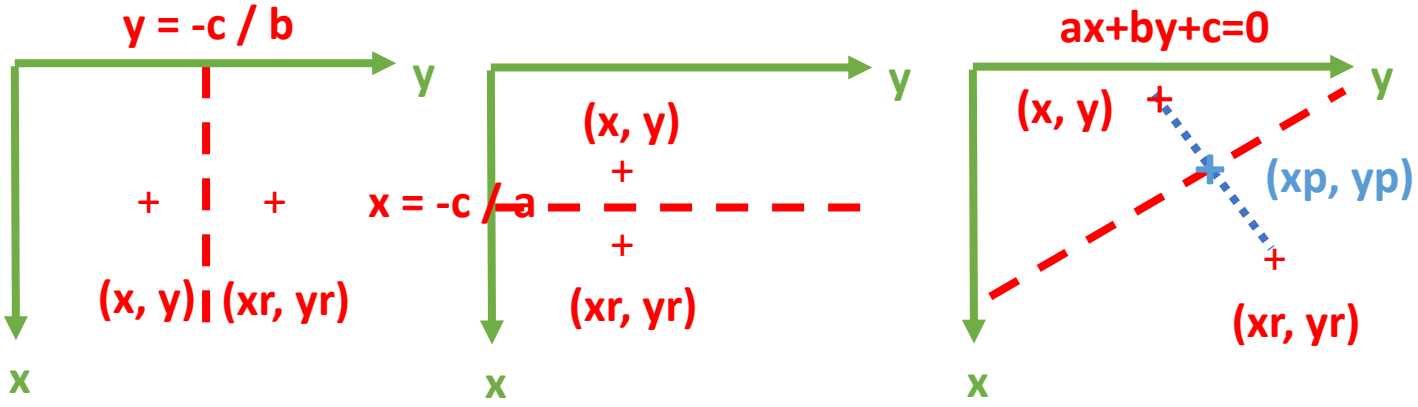

$$|I_{ROI} \cap I_{BB}| \,/\, |I_{ROI}| = 1$$

with $I_{ROI}$ the binary image of the region of interest (the butterfly), $I_{BB}$ the binary image of its bounding box, and $|I|$ the number of pixels in the binary image I with intensity value equal to 1 ($|I|$ is also called the area of I).
The area can be computed using the sum function from Numpy, and the intersection the logical_and function from Numpy.

Q2: The butterfly fills 59% of its bounding box

$|I_{ROI}| \,/\, |I_{BB}| = 0.59$

Q3: Compute the coordinated (xr, yr) of the symmetric point of (x,r) with respect to a line in **the image space**




with (xp, yp) the **orthogonal projection of (x, y) onto the line ax + by + c = 0** (to compute it, read Projection_orthogonale, wikipedia)

Q4: Invert x with xr, and y with yr

Q5: You have to use the mean function form Numpy, mask_arr, arr, sym_arr and **2

Q6: using the min and argmin function from Numpy, you should **automatically** find

```
Best initial parameters: [0, 1, -324]
Best initial loss: 0.016286986732576305
```

Q6bis: There are 3 variables to optimize, namely the parameters of the line, a, b and c. Thus, the Nelder-Mead method creates at each iteration a 3-simplex, namely a tetrahedron.

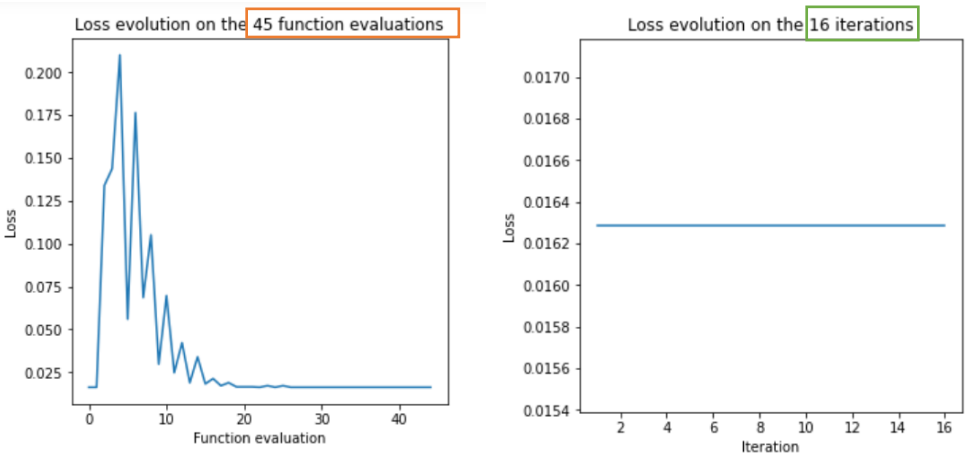
Q7: You have to use the minimize function of Scipy with input

- sym_loss (fun)
- best parameters automatically obtained from Q6 (x0)
- 'Nealder-Mead' (method)
- 'return_all' = True (options, cf documentation)
- (input_arr, mask_arr, back_val) (args)
  With input_arr=mask1_arr, mask_arr = mask1_arr and back_val=np.min(mask1_arr)

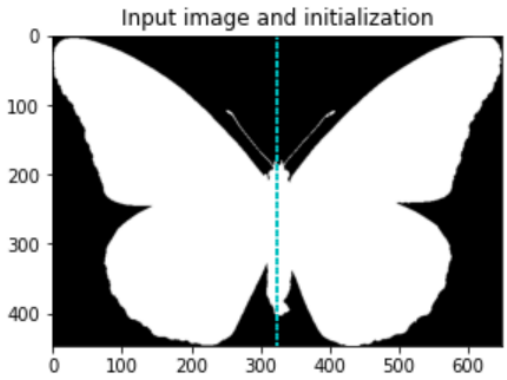
Q8:

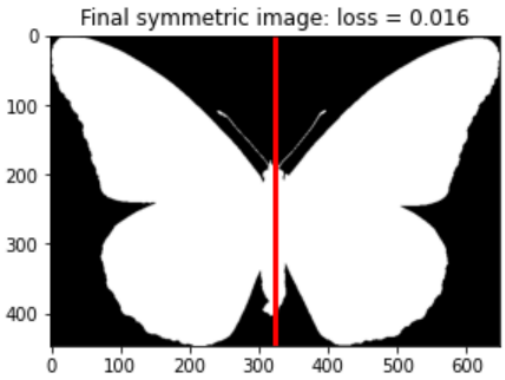
```
       allvecs: [array([   0.,    1., -324.]), array([   0.,    1., -324.]), array([   0.,    1., -324.]), arra
y([   0.,    1., -324.]), array([   0.,    1., -324.]), array([   0.,    1., -324.]), array([   0.,    1., -32
4.]), array([   0.,    1., -324.]), array([   0.,    1., -324.]), array([   0.,    1., -324.]), array([   0.,
  1., -324.]), array([   0.,    1., -324.]), array([   0.,    1., -324.]), array([   0.,    1., -324.]), array
([   0.,    1., -324.]), array([   0.,    1., -324.])]
 final_simplex: (array([[ 0.00000000e+00,  1.00000000e+00, -3.24000000e+02],
       [ 1.56250000e-05,  1.00000000e+00, -3.24000000e+02],
       [ 6.91364778e-06,  1.00018868e+00, -3.24055358e+02],
       [ 7.06318206e-06,  1.00013077e+00, -3.24054742e+02]]), array([0.01628699, 0.01628699, 0.01628699, 0.0162
8699]))
           fun: 0.016286986732576305
       message: 'Optimization terminated successfully.'
          nfev: 45
           nit: 16
        status: 0
       success: True
             x: array([   0.,    1., -324.])
```

Q9 vs Q10:

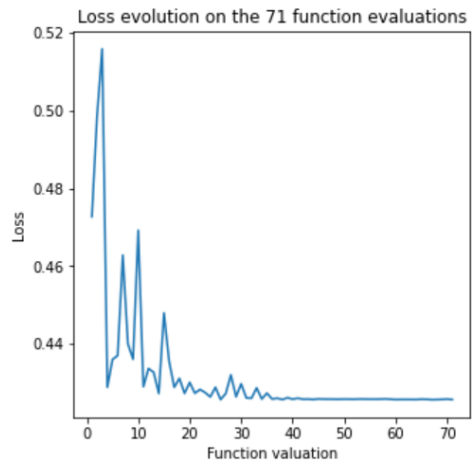


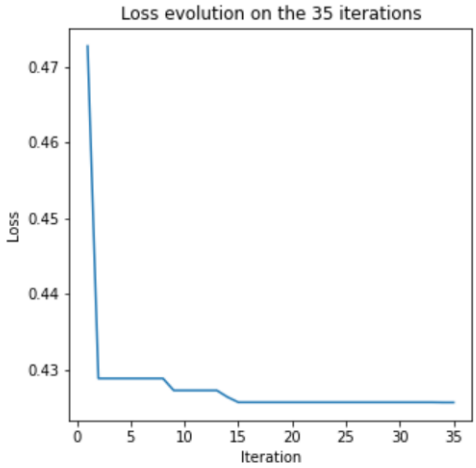
Q11:

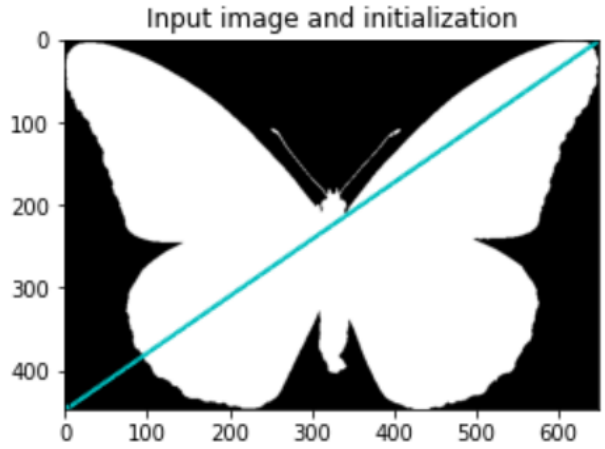


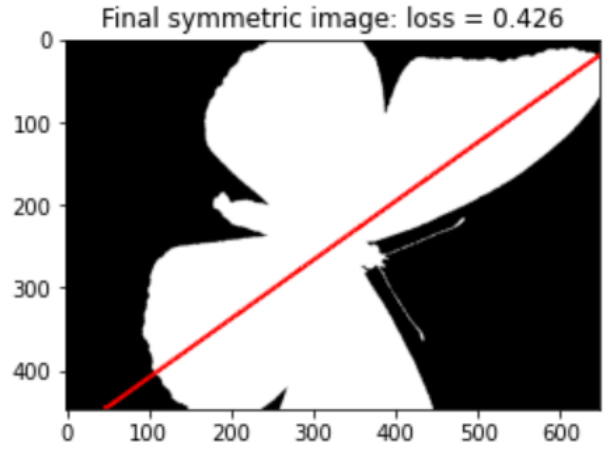




Q12: Change x0 with init_ps[2]

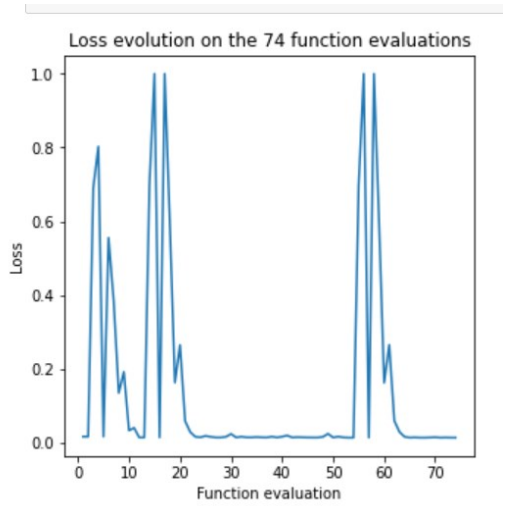


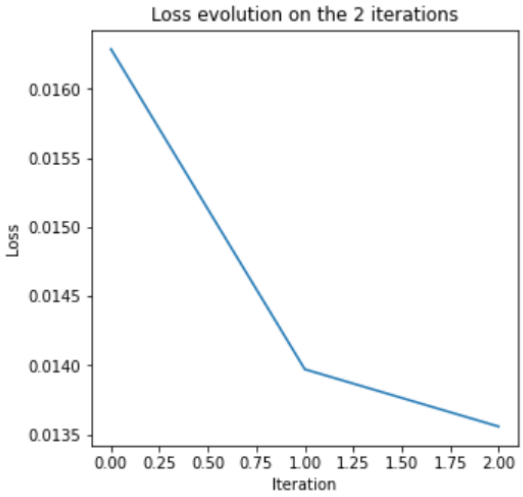



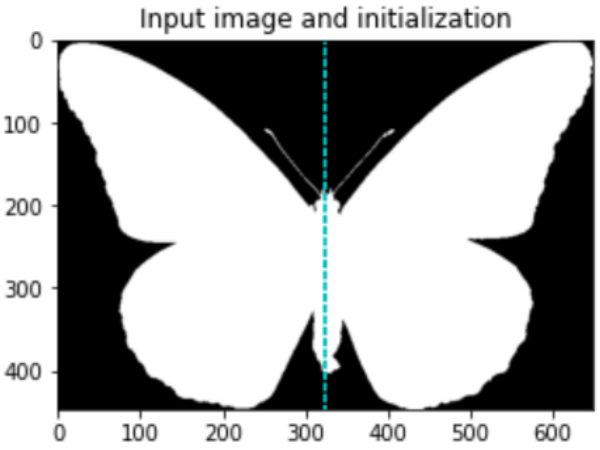



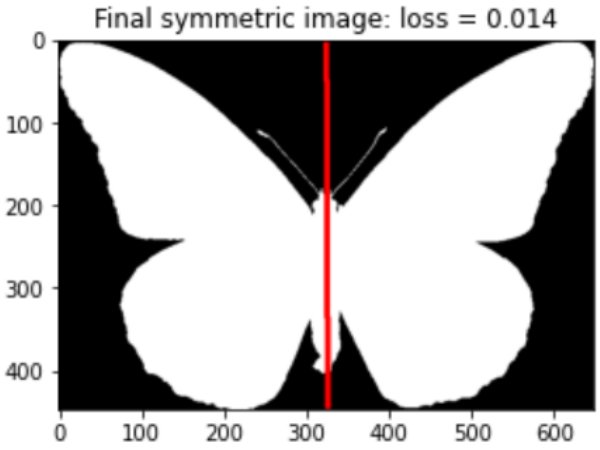



- Q13: Change x0 with the best parameters automatically obtained from Q6 (x0), and method with 'Powell'



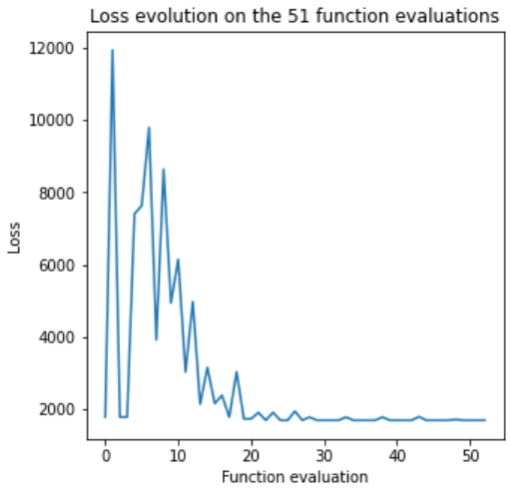



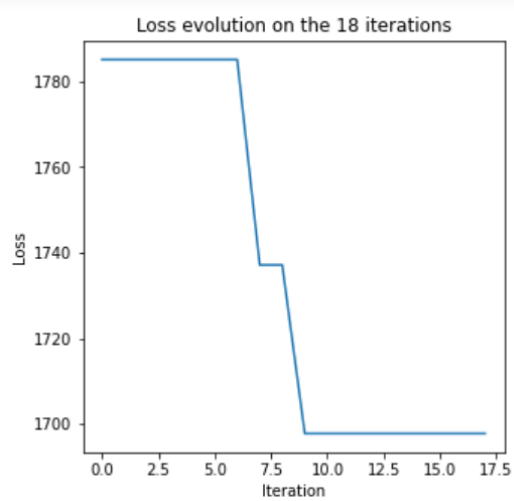



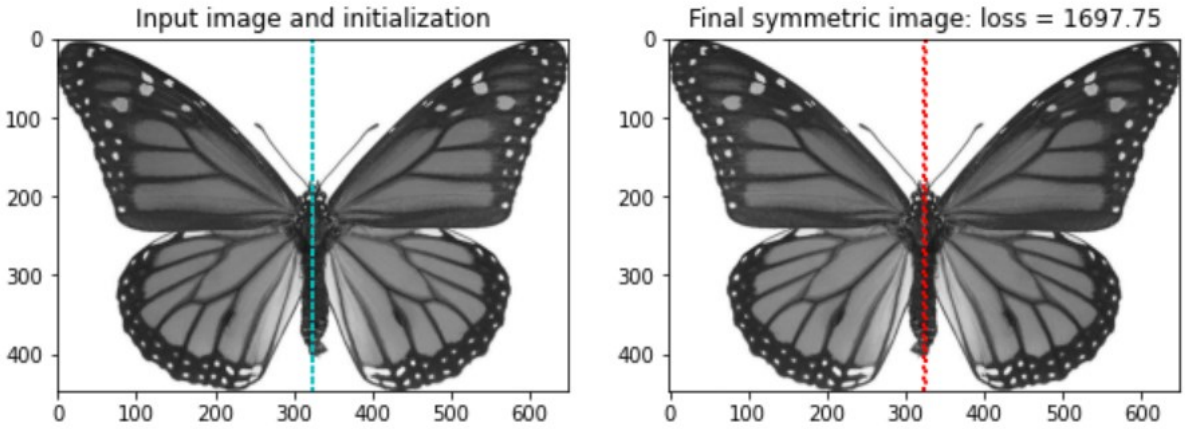




Q13: change input_arr with glim1_arr and back_val with np.max(glim1_arr)

```
Best initial parameters: [0, 1, -324]
Best initial loss: 1785.135350520619
```






That is quite satisfying, because the intensity values are in [0, 255], and not in [0,1] anymore!!