This week I could familiarize myself again with the python and Object Oriented logics. The alternation between the teaching and the practical allowed me to create all the chunks of code without falling behind.

The last part of the practical was a good exposure to a more branched Object Oriented structure. The exercise of walking through the 5 files, understanding what the different methods in the classes are doing and in which other section they are called was a nice example of how an Object Oriented system should look like.

Regarding the exercise of this week I could sort either by X and/or Y. I generate 3 plots: random, sorted by X and sorted by Y. In my sort function, I made use of the function “deepcopy” to create a backup of the original object “PointField”, applying the sorting to an image of it. The time measures reveal that the in-built function is clearly faster than the one we created. Here the results:

Sorting by X 20 took 0.00115 (115e-05)

Sorting by Y 20 took 0.00105 (105e-05)

Sorting by X **with in-built** function 20 took 2.294e-05

Sorting by Y **with in-built** function 20 took 1.751e-05

Curiously the second time the functions (both mine and the in-built) are called they take slightly less time than the first time (e.g. 0.00115 vs 0.00105, there is a difference of 0.01 milli-seconds), but I am not sure what is causing this.

Regarding possible applications of the sorting function, I do not have any specific one coming to my mind. However, the sorting function is always an evergreen procedure used in many processes. In the R environment for example, it is often one of the first thing to do when importing a dataset before start manipulating it.