# Peer-review of assignment 4 for *INF3331-yosephog*

Ine Marie Larsson, inemla, inemla@student.matnat.uio.no
Kristian Geli Nilsen, kristgn, kristgn@uio.no
Ngoc Teresa Nguyen, ngoctn, ngoctn@student.matnat.uio.no

October 9, 2018

## 1 Review - *to be filled out*

I used Python version 3.7.0 on the Darwin operating system for this review. The review is written by Ine Marie Larsson.

### Assignment 4.1

The code works, and it outputs an image of the mandelbrot set. The area drawn was easily altered by making adjustments to the parameters 'ymin', 'ymax', 'xmin' and 'xmax' in the main method of the script. Furthermore, all computations are done with pure python, and numpy is only used for storing numbers. Good job! This delivery answers all but one of the programming parts in the probem statement. The colouring was not saved as an image. This could be done by adding the following line of code:

```
    #save the image as e.g. image.png
    plt.savefig('image.png')
```

There were no docstrings accompanying the functions, and only one comment was provided. In terms of function names, the name 'compute' does not provide much insight as to what the function computes. For someone less familiar with the problem specification, it is not clear from the name what this function does. I found the following section of the code a bit difficult to follow:

```
    for ii,i in enumerate(x):
    for jj,j in enumerate(y):
        list[ii][jj]=compute(complex(i,j),maxiter)
    return list
```

As a suggestion on how to improve the code, I would have considered labeling the variables slightly differently and include more comments in order to improve readability, e.g.:

```
        lst=[[0] * len(x) for lx in range(len(y))]

        #fill the lst array with the values from the compute function
        for i, x_real in enumerate(x):
        for j, y_imag in enumerate(y):
            lst[i][j] = compute(complex(x_real, y_imag), maxiter)
        return lst
```

You have done a good job, but take a look at the variable/function names.

### Assignment 4.2

The code works. It outputs an image of the mandelbrot set. The report contains all the relevant information about the parameters, as well as the runtimes for the python and numpy versions of the code. It seems to me that you have used vectorization where this is possible. There are no docstrings, but more comments than in part 4.1., which is good. I was new to the np.ravel() function, so I appreciated the comment associated with it. However the # symbol should be used for comments that are not docstrings. As a suggestion on how to improve your code, I suggest splitting the mandel function in to two functions. If, for instance, you let one function generate the initial grid, and another generate the values, I think it would be easier to understand what happens where. This delivery answers all the programming parts. Good job!

### Assignment 4.3

The code works. It outputs the image and calculates the time spent. I have the same comments about variable, and function, names as I did in assignment 4.1. There are no docstrings, or comments. The report contains all the relevant parameters, as well as the runtime from the three different implementations of the code. There is also a discussion on the advatages/dis- advantages of Numba. All programming parts of this assignment has been answered.

### Assignment 4.4

Not applicable.

## Assignment 4.5

I tried running the program with the following three command line arguments:

```
python3 mandelbrot_1.py -1 1 -1 1 100 100 100 python bilde.png
python3 mandelbrot_1.py -1 1 -1 1 100 100 100 numpy bilde.png
python3 mandelbrot_1.py -1 1 -1 1 100 100 100 numba bilde.png
```

The program runs. It outputs an image, which is also saved with the name I specified on the command line. A possible improvement to the user interface would be to have some default arguments, such that not all arguments must be specified by the user. When I called it with the –help flag, it only provided information about the "choice" and "file_name" arguments. Here, you have used the variable name 'liste' instead of 'list', as was used in the three previous sections. This is preferable, as list is a built-in function in Python. All programming parts of the assignment has been answered.

## Assignment 4.6

The installation works. The "compute_mandelbrot.py" script runs, and the method "compute_mandelbrot" does what it is supposed to, except for the default argument for the "max_escape_time" variable, which is set to 100 instead of 1000. It returns an array and saves the plot if a file name is provided.

With regards to the unit tests, your README file did not contain any information on how to run the tests. I attempted to run them with the pytest framework, but this attempt halted. I would suggest giving the tests names that more readily communicates what they are testing, e.g. "test_region_entirely_outside" and "test_region_not_outside".

## Assignment 4.7

There is a contribution to the art contest, and the image looks really cool. There was not provided an option to choose between different colour scales.

## Assignment 4.8

In the good function, you first allocate a variable temp as an empty string. Then you make additions to this string in the for- loop. When you concatenate two strings with a '+', a new string is made. As a suggestion on how to improve your code, try making the temp variable a list instead. This way you can append the list without generating new string objects:

```
1        def good_repeat(text):
2
3            temp=[]
4            for i in range(len(text)):
5                temp.append(str(text[i] * (i+1)).capitalize())
6                temp.append('-')
7
8            #join the elements of the list to form a single string
9            return ''.join(temp)
```

In the bad function, you have complicated the code by including a lot of if-statements - good!

## General feedback

I would suggest including docstrings. This makes it easier to quickly get a grasp of what the function does, which parameters it takes and what it returns. Furthermore, I would suggest adding a blank space between an operator and a value/variable name. Also, consider variable/function names. All in all, your code does what it is supposed to, and you make good use of numpy where this is permitted. Furthermore, the axis in the plots are given the correct values, which makes the plot easy to interpret.