Peer-review of assignment 4 for *INF3331-nirusant*

Reviewer 1, claesjog, claesjog@ifi.uio.no
Reviewer 2, shashaj, shashaj@student.matnat.uio.no
Reviewer 3, cklevenb, cklevenb@gmail.no

October 8, 2018

1 Review

System

The following system was used for this review:

- Mac OS High Sierra 10.13.6 v
- GNU Bash 3.2.57 v
- Python 3.6.4

Assignment 4.1

The code works as expected and is fast and efficient. I noticed that you are using camelcase for you variables, witch is not wrong, but I would recommend you to read the `PEP 8 Style Guide for Python` that suggest underscores after each word.

The docstrings are informative and well explained. However I would like to see a docstring at the beginning of you class explaining what it is for and what arguments you can pass in to it. In certain editors you can hover over a class you called and immediate see what arguments it takes and what it is used for. This is good to have in mind if you are writing code that other people are going to use.

The code is very clean and the variable names are self explanatory. However I have one small problem with the unnecessary use of 7-lines of code in the method `makeMandelbrotSet` as seen below.

```
numIt = self.numIt
pixelWidth = self.pixelWidth
pixelHeight = self.pixelHeight
xMinVal = self.xMinVal
xMaxVal = self.xMaxVal
yMinVal = self.yMinVal
yMaxVal = self.yMaxVal
```

This part should be removed and replaced with the snippet-below using the `self.` instance.

Suggested code:

```
xstep = abs(self.xMinVal - self.xMaxVal) / self.pixelWidth
Re = self.xMinVal

ystep = abs(self.yMinVal - self.yMaxVal) / self.pixelHeight
Im = self.yMinVal

img = []
for j in range(0, self.pixelHeight + 1):
    line = []
    for i in range(0, self.pixelWidth + 1):
        line.insert(i, self.mandelbrot(complex(Re,Im), self.numIt))
        Re += xstep

    img.append(line)
    Im += ystep
    Re = self.xMinVal
return np.asmatrix(img, dtype=int)

# Note: also made some spaces in between calculations and commas
```

Overall a well written script.

Assignment 4.2

The code works and is faster than the code made with pure python.

The documentation is still informative and well explained, except the missing docstring in the start of the class.

The implementation with numpy is very well made and also by avoiding the `abs()-function` you save time in your script. The only exception is the unnecessary 7-lines of code in `makeMandelbrotSet` that could have been removed, as mentioned in `Assignment 4.1`.

The report is well done and includes what is requred.

Overall very well done.

Assignment 4.3

The code works and is even faster than numpy, witch is great. However the task was to reuse code the from task 4.1(`mandelbrot_1.py`) and compare the use of numba versus python. Since this implementation is a combination of `mandelbrot_1.py` and `mandelbrot_2.py` this is wrong, but uses the right implementation of the numba extension.

I would prefer that the docstring explained the code instead of referring to another code in another python file. Then the reviewer would not have to navigate to another file to read the documentation if he/she had forgot how the function worked. And since the `makeMandelbrotSet-function` takes a lot of arguments it should have been explained in it's docstring. Last thing is that the code in this function could have been slightly more commented to get a broader understanding of the code, but was not to hard to understand.

The report is well done and includes results for previous test as well with the new test, witch is neat.

By not including that the script is combined with both `mandelbrot_1.py` and `mandelbrot_3.py` it is well done implementation of numba.

Assignment 4.4

I had trouble running the setup code, but solved it by doing this in your `setup.py` file:

```
import numpy
setup(ext_modules = cythonize("mandelbrot_4_1.pyx"),
      include_dirs=[numpy.get_include()])
```

I am testing on a mac so I guess that was the problem.

The code then works as expected.

The documentation is very well and it is easy to understand what have been done. However I would recommend to comment, in the docstring, the use of the different arguments the function takes.

All though the script is small and self explanatory it can for others be hard to understand "cryptic" variables. In this case referring to variables such as `tmpRe` and `tmpIm` that should be `temp_real` and `temp_imag`. This is an easy fix to avoid confusion and it does not take a lot of effort. In this case I would like to see more use of both `PEP 8` and `PEP 20` that says

> Explicit is better than implicit.

Regarding the c-compiler is all the variables and functions declared properly with the right types. I can not find any missing declarations.

The `report4.txt` is well done and explains the test made including all previous results.

Overall a well written script with a good implementation of Cython.

Assignment 4.5

The implementation works as expected, but if I inserted a wrong argument such as a `string` where it should be an `int/float` the program crashes. I can also see that you have a lot of print-states made manually. This can be avoided by using `argparse` witch is a very helpful tool for these kind of argument implementations. The package have the flags, `-h` and `--help`, built in so you do not have to reinvent the wheel. Following code-snippet is an example of how you could have implemented it:

```python
import argsparse

# Calling the argparser class
parser = argparse.ArgumentParser()

# Making must-have-arguments
# Taking a value type float and prints the help instructions if the -h or --help flag is set
parser.add_argument("xmin", help="Takes a float value for the first point in the complex plane", type=float)
parser.add_argument("ymin", help="Takes a float value for the first point in the complex plane", type=float)
parser.add_argument("xmax", help="Takes a float value for the second point in the complex plane", type=float)
parser.add_argument("ymax", help="Takes a float value for the second point in the complex plane", type=float)
parser.add_argument("nx", help="Takes a int value for the width of the image", type=int)
parser.add_argument("ny", help="Takes a int value for the heigth of the image", type=int)
parser.add_argument("max_esc_time", help="Decide the maximum escape time", type=int)

# Making optional arguments that are false by default
parser.add_argument("--python", help="Will calculate the mandelbrot set with pure python",
                    type=bool, nargs='?', const=True, default=False)
parser.add_argument("--numpy", help="Will calculate the mandelbrot set with numpy",
                    type=bool, nargs='?', const=True, default=False)
parser.add_argument("--numba", help="Will calculate the mandelbrot set with numba extension",
                    type=bool, nargs='?', const=True, default=False)

# Simply put all your arguments in the variable "args" and you can access them
args = parser.parse_args()

if args.python:
    mandelbrot_1(args.max_esc_time, args.nx, args.ny, args.xmin, args.ymin, args.xmax, args.ymax)
elif args.numba:
    #Do something
...
...
...

# Note that all arguments starting with -- is optional and are set to FALSE by default, and
# the arguments passed in the command line will be checked with the type of the argument if
# it is the correct type or not.
```

By using typing `python mandelbrot.py --help` you will be provided with the information screen shown in the snippet below.

```
usage: mandelbrot.py [-h] [--python [PYTHON]] [--numpy [NUMPY]] [--numba [NUMBA]]
                     xmin ymin xmax ymax nx ny max_esc_time

positional arguments:
  xmin                 Takes a float value for the first point in the complex
                     plane
  ymin                 Takes a float value for the first point in the complex
                     plane
  xmax                 Takes a float value for the second point in the complex
                     plane
  ymin                 Takes a float value for the second point in the complex
                     plane
  nx                 Takes a int value for the width of the image
  ny                 Takes a int value for the heigth of the image
  max_esc_time       Decide the maximum sescape time

optional arguments:
  -h, --help         show this help message and exit
  --python [PYTHON]  Will calculate the mandelbrot set with pure python
  --numpy [NUMPY]    Will calculate the mandelbrot set with numpy
  --numba [NUMBA]    Will calculate the mandelbrot set with numba extension
```

Furthermore I think the docstrings is well written and the script is easy to read. But I am still missing some argument documentation for the functions.

Overall a good implementation of the user interface.

Assignment 4.6

The `setup.py` script successfully installed the `compute_mandelbrot` function, and it works as expected.

The test are well made and runs successfully with `unittest`. The test is easy to understand and self explanatory. A suggestion would be to make a test for all your other scripts as well. This it not a part of the assignment, but would be a good practice.

The docstrings are good and the script is more or less self explanatory.

Overall a well done test and a good implementation of the scripts as a package.

Assignment 4.7

Nice image of the mandelbrot set.

Assignment 4.8

The script works as expected.

The docstring for the `repeat_good-function` is very well written - Good job!
The docstring for the `repeat_bad-function` is not even there - Also good job!

Overall a very good implementation of both good and bad python code.

General feedback

I can see you are a skilled programmer, but there is always room for some improvement. I have made a small list of things that you should have in mind while programming.

- Write docstrings for all classes and functions. Describe what it is, witch arguments/types it takes and what it return. A helpful tool in Visual Studio code is `autoDocstring`. This auto-generates a good core of a docstring.
- The README.md should contain more details and could have a section for each task explaining what to run in the command-line.
- Consistent use of spaces in calculations and arguments in functions or classes

Overall very good job!