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

Torjus Dahle, torjuskd, torjuskd@uio.no
Daniel Egelandsdal Osen, danieleo, danieleo@student.matnat.uio.no
Oleksandr Korpusov, oleksako, alpost@zoho.com

October 16, 2016

1 Introduction

1.1 Goal

The review should provide feedback on the solution to the student. The main goal is to *give constructive feedback and advice* on how to improve the solution. You, the peer-review team, can decide how you organise the peer-review work between you.

1.2 Guidelines

For each (coding) exercise, one should review the following points:

- Is the code **working as expected**? For non-internal functions (in particular for scripts that are run from the command-line), does the program handle invalid inputs sensibly?
- Is the code **well documented**? Are there docstrings and are the useful?
- Is the code written in **Pythonic way** [1]? Is the code easy to read? Are the variable/class/function names sensible? Do you find overuse of classes or not sufficient use of functions or classes? Are there parts of the program that are hard to understand?
- Can you find **unnecessarily complicated parts** of the program? If so, suggest an improved implementation.
- List the programming parts that are not answered.

Use (shortened) code snippets where appropriate to show how to improve the solution.

1.3 Points

The review is completed by pushing the review Latex and PDF files to each of the reviewed repositories. The name of the files should be *feedback.tex* and *feedback.pdf*.

You will get up to 10 points for delivering the peer-reviews. Each of you should contribute to the review roughly equivalently - your team will get the same number of points[2].

1.4 Useful Latex snippets

Here is some sample usage of Latex.

1.4.1 Sample code

```
import sys
print "This is a sample code"
sys.exit(0)
```

1.4.2 Mathematical equation

$$2\pi > 6 \tag{1}$$

2 Review

Tested on Ubuntu 14.04.1 using Python 3.4.3.

[1]https://www.python.org/dev/peps/pep-0020/
[2]In case a team-member does not contribute, please email simon@simula.no

General feedback

One aspect you can work on is testing your code, and eliminating unnecessary bugs. The documentation of your code is good - just write more of it! :) Your code is efficient, readable, you have kept it simple and understandable throughout the assignment - keep it this way. It is clear that you know how to program well, just remember the little things. ;)

Assignment 4.1: Python implementation

The code is ok and clean looking, but there are a few issues. The first one is you shebang:

```
#Looks like this:
#!/user/bin/env python3

#You probably want something like this instead:
#!/usr/bin/env python3 # "usr" is the default "user"-dir on unix systems

#(and on OS X it would be /Users)
```

Another really unfortunate thing is that the program won't actually run because of another tiny mistake:

```
#Instead of this:
if "__name__" == "__main__": # here you compare the string "__name__" to the string "__main__"

#This is what you probably wanted to do:
if __name__ == "__main__":
```

The imported ”time” is being overshadowed with the defined time float variable.

The script also doesn't specify a filename, so no image file is saved to disk or shown to the user. After after fixing these issues I still get some runtime warnings:

```
./mandelbrot_1_fixed.py:22: RuntimeWarning: overflow encountered in cdouble_scalars
  if (f_c * numpy.conj(f_c) > 2**2):
./mandelbrot_1_fixed.py:22: RuntimeWarning: invalid value encountered in cdouble_scalars
  if (f_c * numpy.conj(f_c) > 2**2):
./mandelbrot_1_fixed.py:21: RuntimeWarning: overflow encountered in cdouble_scalars
  f_c = f_c*f_c + c
./mandelbrot_1_fixed.py:21: RuntimeWarning: invalid value encountered in cdouble_scalars
  f_c = f_c*f_c + c
```

I would have tried to look into the cause of these warnings, but your script completes successfully (after the aforementioned changes) so I guess it doesn't matter much. Your code is readable, pythonic, well documented and concise. You have done a good job, I would just advise you to test your code more thoroughly.

Assignment 4.2: numpy implementation

Much of the same (as on 4.1) applies here, except that here you added a function call outside the main test, so the code actually runs (but will lead to problems later). This time the code is working as expected, well documented, pythonic and has efficient use of numpy.

Assignment 4.3: Integrated C implementation

Providing a comment on how to install could have been a good idea. Other than that, the code is good. You had very similar runtimes using cython compared to numpy. One reason for that might be that you rely heavily on numpy arrays, instead of using more primitive c-type arrays, and your solution is fundamentally the same. Concise readable code, simple but perhaps a little bit lacking in documentation/comments/instructions.

Assignment 4.4: An alternative integrated C implementation

Again lacking install/setup instructions. Code seems good. Maybe it ran on your system, but I couldn't quite install it/get it to run:

```
mandelbrot_4.i:7: Error: Unable to find 'numpy.i'
error: command 'swig' failed with exit status 1
```

Assignment 4.5: User interface

The user interface is really good, even interactive. Well done!

Assignment 4.6: Packaging and unit tests

No test framework used. There is also no apparent way to run the tests. Maybe instructions are lacking again? Same problem as earlier here with the imported time being overshadowed.

And unfortunately, mandelbrot_2.py has its mandelbrot-method called when imported here:

```
#from "test_mandelbrot.py"
from mandelbrot_2 import mandelbrot #this calls the  mandelbrot_2 method directly. Again probably not what you want to do.
```

For a outsider there is no way to tell what the expected input value for the variable iterations is in the test-function.

```
def test(iterations, max_escape_time):
```

Assignment 4.7: More color scales + art contest

The user is able to freely choose color scale, great! Not much to say here, other than job well done.

Assignment 4.8: Self replication

No file called replicator.py found. There are several (although tricky) ways to do this. One example is:

```python
rep = r"print('rep = r\"'+ rep +'\"\neval(rep)')"
eval(rep)
```