# Peer-review of assignment 5 for *INF3331-torjuskd*

Thomas Ballo, thomagb, thomagb@student.matnat.uio.no
Atle Nærum Eriksen, atlene, atlene@ifi.uio.no
Gudrun Sigfusdottir, gudrunsi, gudrunsi@student.matnat.uio.no

November 13, 2016

## 1 Review

The program was tested using python3 on Linux and OS X.

### General feedback

The assignment is well conducted and all in all a good job. A README.md file which explains how to run the different programs would have been appreciated.

Both Python programs are well structured with good use of functions, making it easier to read the code and understand the functionality. Good use of docstrings the functions to explain what they do.

#### Notes for improvement

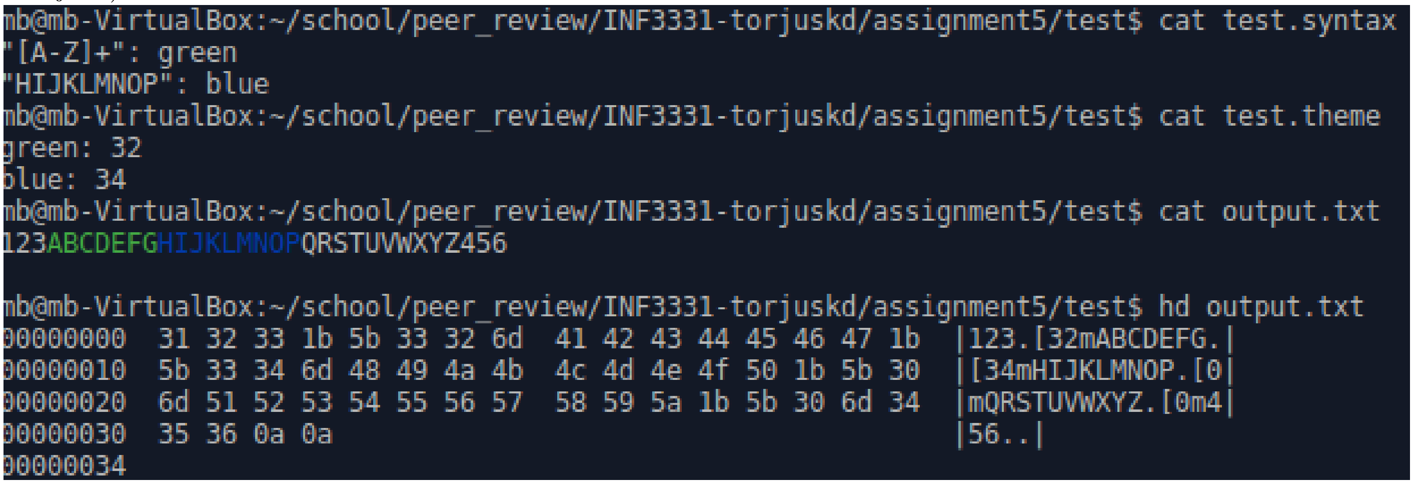
- Lines shouldn't be longer than 79 characters according to the PEP 8 style guide. Some long lines like line 51 and 52 in `highlighter.py` makes it harder to read and understand the code.

### Assignment 5.1: Syntax highlighting

The layout of the program is decent and the use of "if __file__ == "__main__"" is very welcomed. However, in general there are quite a few things that are not so great:

- line.strip() on line 28 is redundant (but no biggie).
- `data = read_data` on line 47 is redundant
- The regex on line 43 could have been compiled since it doesn't change inside the loop (but no biggie).
- Naming. Even though you document your code, the symbol names are not sufficient to be descriptive, which partly renders the documentation moot. Examples:
  - highlight(): What does it highlight? It's not an instance method, so there's no instance context to bind meaning to.
  - syntax=False: A better name could perhaps be something along the lines of is_syntax_file=False? Because that's what you are actually asking.
  - readrulefile() returns a list as per the documentation. A list of what? To use it you would need to know what it contains.
  - Similarly, highlight() takes some arguments that also are lists, but that doesn't tell us anything about the lists.
  - The filename argument of highlight(), does it hold a filename or a file *path*? It's the latter, and the symbol name should reflect that.
  - highlight() is said to return <none>, however it returns a string if a source file was given.
  - readrulefile(), rulefilename, and more: Can easily distract your focus by being non-standard. We recommend using underscores between word boundaries. This is obviously even more important the longer the name becomes or it won't be readable.
  - This item is debatable, but the semantic meaning of what you're doing should play an important role in how you name things. For example, the function readrulefile() is also used to load theme files in addition to syntax files. However, the theme information is not particularly rule-based. The regexes define rules for what should be searched for in a string, but color codes aren't exactly rules. You can look at it that way, though, so food for thought.

The highlighting part of the program appeared cryptic at first sight, and from a readability perspective it could be improved. We were suspicious about whether the approach of ”fixing up the colors” would work in cases where there was more than two colors to work with (since that’s what the regex expects), such as a region within a region within a region. We did some tests and found out this didn’t appear to be working as expected, but surprisingly the simpler test of a region within a region (standard overlapping) also failed. Even though the code seems correct in theory, there is clearly something wrong in practice, and it could be worth looking into (or rather, for a future project, use more varied test subjects).

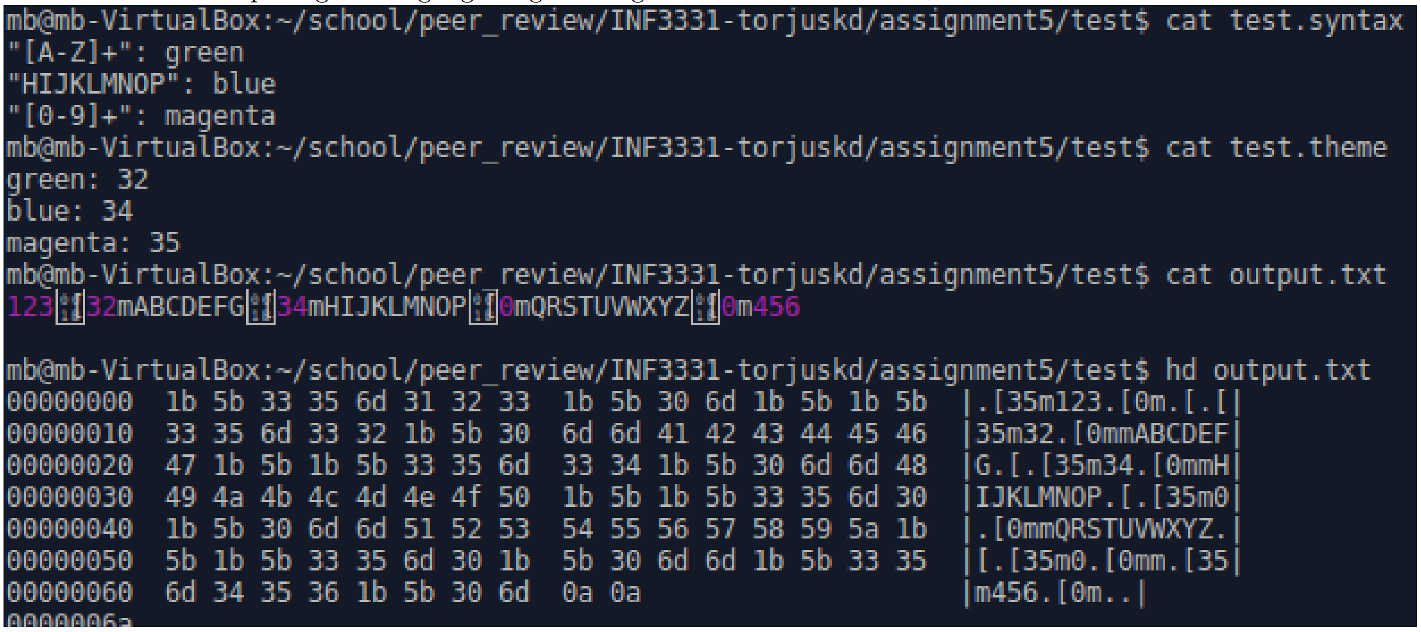


```
mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ cat test.syntax
"[A-Z]+": green
"HIJKLMNOP": blue
mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ cat test.theme
green: 32
blue: 34
mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ cat output.txt
123ABCDEFGHIJKLMNOPQRSTUVWXYZ456

mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ hd output.txt
00000000  31 32 33 1b 5b 33 32 6d  41 42 43 44 45 46 47 1b  |123.[32mABCDEFG.|
00000010  5b 33 34 6d 48 49 4a 4b  4c 4d 4e 4f 50 1b 5b 30  |[34mHIJKLMNOP.[0|
00000020  6d 51 52 53 54 55 56 57  58 59 5a 1b 5b 30 6d 34  |mQRSTUVWXYZ.[0m4|
00000030  35 36 0a 0a                                       |56..|
00000034
```

As you can see the colors are not properly closed in this example, nor is the green region reopened after leaving the blue region.

We then added a simple regex to highlight digits and got this result:

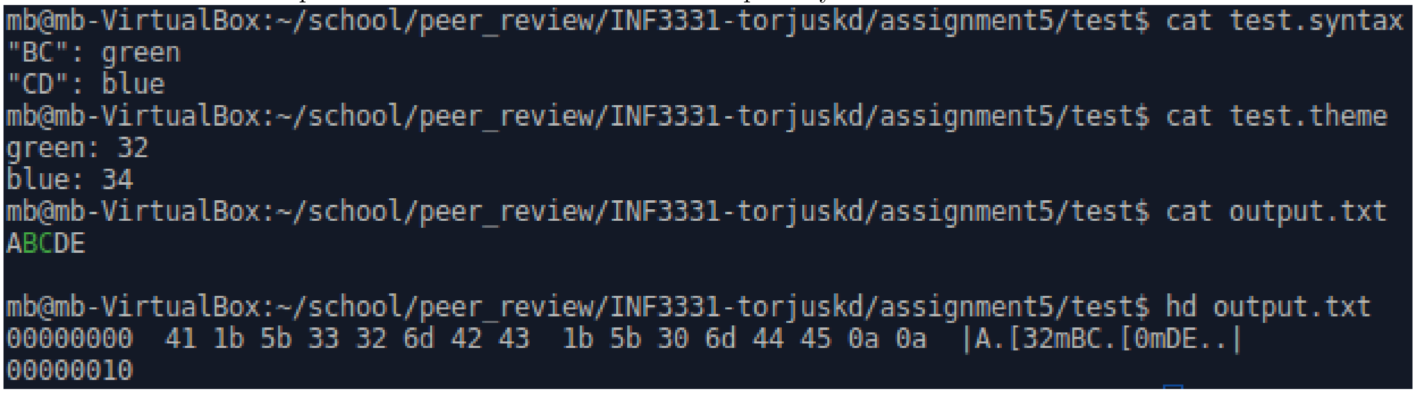


```
mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ cat test.syntax
"[A-Z]+": green
"HIJKLMNOP": blue
"[0-9]+": magenta
mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ cat test.theme
green: 32
blue: 34
magenta: 35
mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ cat output.txt
123␛[32mABCDEFG␛[34mHIJKLMNOP␛[0mQRSTUVWXYZ␛[0m456

mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ hd output.txt
00000000  1b 5b 33 35 6d 31 32 33  1b 5b 30 6d 1b 5b 1b 5b  |.[35m123.[0m.[.[|
00000010  33 35 6d 33 32 1b 5b 30  6d 6d 41 42 43 44 45 46  |35m32.[0mmABCDEF|
00000020  47 1b 5b 1b 5b 33 35 6d  33 34 1b 5b 30 6d 6d 48  |G.[.[35m34.[0mmH|
00000030  49 4a 4b 4c 4d 4e 4f 50  1b 5b 1b 5b 33 35 6d 30  |IJKLMNOP.[.[35m0|
00000040  1b 5b 30 6d 6d 51 52 53  54 55 56 57 58 59 5a 1b  |.[0mmQRSTUVWXYZ.|
00000050  5b 1b 5b 33 35 6d 30 1b  5b 30 6d 6d 1b 5b 33 35  |[.[35m0.[0mm.[35|
00000060  6d 34 35 36 1b 5b 30 6d  0a 0a                    |m456.[0m..|
0000006a
```

It now becomes apparent why it’s a bad idea to have compact, cryptic code instead of longer, more mundane step-by-step code, because this would probably be very hard to debug to fix.

The highlighting approach taken is to substitute the output on every regex match, which was pointed out by an instructor on Piazza to be a problematic solution. Here is an example why:



```
mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ cat test.syntax
"BC": green
"CD": blue
mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ cat test.theme
green: 32
blue: 34
mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ cat output.txt
ABCDE

mb@mb-VirtualBox:~/school/peer_review/INF3331-torjuskd/assignment5/test$ hd output.txt
00000000  41 1b 5b 33 32 6d 42 43  1b 5b 30 6d 44 45 0a 0a  |A.[32mBC.[0mDE..|
00000010
```

The regex first matches BC and colors that part correctly. Now, despite CD actually existing in the input the regex is unable to match it because the input has been modified in-place, and there is now ”[0m” in between C and D.

Interestingly, despite these shortcomings task 5.2, 5.3 and 5.4 seem to be working just fine. It may have something to do with the way things are overlapping in those files compared to our tests, but we can't say for sure.

## Assignment 5.2: Python syntax

The program works as expected, but the second theme file is missing. There is a file called python.theme2 in the directory, but it appears to be empty. python.syntax is well written and takes into consideration all possible mismatches.

The regex accounts for proper spacing between things, such that e.g. both "a=1" and "a = 1" are both valid, that's nice to see. Some of the regexes are more open than they could have been, e.g. the string "def-part" (from our test files) gets matched even though it's not a def-statement. Similarly, "class ¡anything here¿: ¡some code¿" matches the whole line instead of capturing only the keywords required. Perhaps this was your intention, perhaps not. It's still worth restricting the regex as much as possible so that f.ex. "I'm going to class today" isn't classified as a class definition (just an example, your program doesn't do this).

Strings match both multi-line, double-quoted, single-quoted, and even strings that have matching quotes inside the string itself, which should from that point on cancel the string (and this is colored correctly). Kudos.

The non-capturing group at the start of most of the regexes is probably not necessary.

## Assignment 5.3: Syntax for your favorite language

Java: Again, well written and takes into consideration a lot of different aspects of the language.

## Assignment 5.4: Syntax for your second favorite language

SiMPLE: Yet again, the syntax file is extensive and takes a lot into consideration. This seems to work as expected. Good job.

## Assignment 5.5: Superdiff

The program works as expected and is well documented. It is quite easy to read and the number of functions is acceptable. It does seem overly complicated, but by the way it is implemented, there is no obvious way to simplify it. By treating the file-lines as arrays and then poping the stack would eliminate the need for the line1N and line2N variables and shorten the code drastically.

When no output filename is provided it would have been nice if the program printed the output to the terminal instead. Now it doesn't really do anything when no output file is provided.

The algorithm is not very smart. For instance when comparing two files where the first line has been moved to the end of the file, the output is:

```
  +b
  +c
  +d
  +e
  0a
  +
  -b
  -c
  -d
  -e
  -
```

This is technically not wrong, although adding and removing the last empty line is unnecessary. But the output would have been a lot easier to read if it was:

```
- a
0 b
0 c
0 d
0 e
+ a
```

## Assignment 5.6: Coloring diff

The coloring works as expected.