Software Engineering 265
Software Development Methods
Summer 2020

Assignment 4

Due: Tuesday, August 4th, 11:55 pm via a push to your Git remote repository.
(no late submissions accepted)

Programming environment

For this assignment you must ensure your work executes correctly on the virtual machines you installed as part of Assignment #0 (which I have taken to calling `Senjhalla`). This is our "reference platform". This same environment will also be used by the course instructor and the rest of the teaching team when evaluating submitted work from students.

Any programming done outside of this reference platform might not work correctly. Such work may receive a failing grade.

Individual work

This assignment is to be completed by each individual student (i.e., no group work). Naturally you will want to discuss aspects of the problem with fellow students, and such discussion is encouraged. **However, sharing of code fragments is strictly forbidden without the express written permission of the course instructor (Zastre).** If you are still unsure regarding what is permitted or have other questions about what constitutes appropriate collaboration, please contact me as soon as possible. (Code-similarity analysis tools will be used to examine submitted work.)

Objectives of this assignment

- Use the Python 3 programming language to write a less resource-restricted implementation of `senjify`, this time using *user-defined classes* and *regular expressions*.
- Use `git` to manage changes in your source code and annotate the evolution of your solution with messages provided during commits.
- Test your code against the 20 provided test cases (i.e., those tests from assignment #2).

This assignment: `senjify4.py`

For this final assignment you will visit, for the last time, the text-formatting problem used this semester. The formatting commands as used for assignment #2 (and also assignment #3) will be used here.

The command to run the program will differ slightly from previous assignments. You are provided with a file named `tester4.py` designed to accept a command-line argument from the shell (if one exists). If the argument exists, that file is opened for input; otherwise input is assumed to be from stdin. This tester script then creates an instance of the `SENJIFY` class (which you will write) that is implemented in the `senjify4.py` file, calls the `format` method on that object, and the outputs the result returned by that `format` method.

```
% cat tests/in04.txt | ./tester4.py
```

```
% ./tester4.py tests/in04.txt
```

There are a few more structural requirements for the way your solution is to be written:

- The constructor for `SENJIFY` must accept a single parameter which is the input stream for formatting.
- The method named `format` must return a list of strings corresponding to the correct formatting result. You must complete this method. Note that this method cannot direct any output to `stdout`.
- You must write all other methods needed within the class; these methods must be all private.
- You must use regular expressions in a non-trivial manner. At the very least regular expressions must be used to extract needed information from formatting commands.

Please look at the `main` function of `tester4.py` to learn better how the script depends upon the `SENJIFY` constructor's signature and the method format. All of your Python code must appear within `senjify4.py` and all of that within the methods of class `SENJIFY`.

You are not permitted to modify `tester4.py`.

Exercises for this assignment

1. Within your `git` repo ensure there is an `a4/` subdirectory. Your `senjify4.py` must be located in your `a4/` directory.

2. Write your program. Amongst other tasks you will need to:
   - read text input from a file, line by line
   - extract substrings from lines produced when reading a file or reading an input stream
   - create and use lists in a non-trivial way
   - write a user-defined class
   - use regular expressions

3. Use the test files to guide your implementation efforts (i.e., these are located in `/home/zastre/seng265/assign2`) on the UVic lab machines. Refrain from writing the program all at once, and budget time to anticipate when "things go wrong".

4. For this assignment you can assume all test inputs will be well-formed (*i.e.*, our teaching team will not test your submission for its handling of error-formed input or for malformed command-line arguments).

What you must submit

- A single Python source file named `senjify4.py` within your `git` repository containing a solution to assignment #4.

Evaluation

Our grading scheme is relatively simple.

- "A" grade: A submission completing the requirements of the assignment which is well-structured and very clearly written. All tests pass and therefore no extraneous output is produced.
- "B" grade: A submission completing the requirements of the assignment. The code written in `tester4.py` runs without any problems; that is, all tests pass and therefore no extraneous output is produced.
- "C" grade: A submission completing most of the requirements of the assignment The code written in `tester4.py` runs with some problems.
- "D" grade: A serious attempt at completing requirements for the assignment. The code written in `tester4.py` runs with quite a few problems; some non-trivial tests pass.
- "F" grade: Either no submission given, or submission represents very little work, or no tests pass.