Task documentation SYN: Syntax highlighting in Python 3 in IPP 2013/2014

Name and surname: Ladislav Šulák

Login: xsulak04

1 Assignment

The task was to create a script in language `Python 3`, which will highlight syntax of input text file based on input format file containing regular expressions with relevant list of format commands . This list contains one or more commands, which are actually `HTML` tags.

2 Implementation

There wasn't used object-oriented programming (OOP) and data type dictionary. Instead of that, functions, global variables and a few lists are being used. My solution is based on two things:

- Finite state machine,
- Iteration on list which contains position where to apply a format command, and `HTML` opening/closing tag .

The program is divided into several parts, which are described below.

2.A Argument parsing

Argument parsing is implemented in function `paramProcessing()`. The entire parsing is solved as a cycle, which iterates through list of input parameters, so I didn't use any of support libraries. If there is an error state, the script ends with exit code 1. Besides parameters for determining name of input and output files files, and parameter for printing a help message, there is also an argument `--br` which will add tag `<br />` before every end of the line. Others tags have two parts: opening and closing.

2.B Processing of input format file

This processing is also connected with regular expressions and operations with them. The first step is to separate a regular expression (which is moved and translated into Python's regular expressions in function `RegexpProcessing(regexpIN)` after that) from format commands and then separate format commands itself (separation implemented in function `formatProcessing()`).

There would be two possible solutions of translating regular expressions:

- Replacing an input string with regular expression for translating into correct form,
- Finite state machine.

I realise that first choice would be much more harder for checking every regular expression form, so the second case was chosed.

Finite state machine is implemented in function `RegexpProcessing(regexpIN)` as a cycle which read an input string given from format file, character by character. This function returns a translated regular expression, which is saved also with format commands into a global list.

In the next step regular expression is used to find a location of corresponding string, and string itself (which provides a function `commandProcessing()` and `addPositions(regexp, begin, end, count)`).

2.C Inserting tags

The problem of inserting tags into an input string was following: if positions of all tags are stored , and tag is inserted, position of every character is altered. There are also a few solutions of this problem:

- If tag is inserted, every position higher than actual is altered – add lenght of inserted tag to every higher position

This solution consumes a high amount of procesor 's time – need to alter a positions in almost every time when inserting.

- Positions and tags are stored into list, which is ordered from biggest to smallest possitions. Information about correct sequence of tags from input file is also stored. In case that more tags are inserted on the same position we use this number to achieve correct order.

This solution has been chosen because of it's effectivity.

2.D NQS extension

NQS extension is a reduction of two or more characters '+ ' or '*'. It is implemented in a finite state machine. In the case that next token is one of these characters, it checks last character in output string, and will perform a reduction.

3 Conclusion

Script was developed and tested on school servers Merlin with operating system CentOS 6.5 64bit and Eva with operating system FreeBSD 9.x 64bit. I have used provided tests and also my own test script in Bash. I focused on regular expressions of all kind. For checking correctness of output expression I used online Python regexp testers and debuggers.