Task documentation SYN: Syntax Highlighting in Python 3 for IPP 2015/2016
Name and surname: Andrej Barna
Login: xbarna01

1 Overview

This documentation describes my implementation of a Python 3 script, the purpose of which is to insert markup tags into the input text, based on rules defined in the formatting file. Tags inserted into the text are identical to HTML tags used for text formatting.

2 Format file parsing

Formatting records must be parsed in order to be used for text formatting. The format I have selected for the records' storage, is a list of n-tuples, one n-tuple for each formatting record. At the first position in the n-tuple of the record is the regular expression of the record, after which formatting tags are stored. Each formatting tag is represented by a pair, the identifier of the tag is at the first position and the parameter of the tag is stored at the second position, which is used only for `color` and `size` tags.

2.1 Regular expression translation

The format of regular expressions used in formatting records is impractical. In order to make them usable by standard Python functions these regular expressions must be translated into the standard PCRE format. The translation itself is performed by the `translateRegex()` function, which replaces lexems from the original regular expression with their PCRE counterparts.

The most problematic part of the translation process is the negation of a group of expressions, which is matching only single characters. For this purpose the `resolveNegatedGroup()` function is called, which translates this group into a valid PCRE character class. After the translation process is finished, the translated PCRE expression is compiled to verify its correctness.

3 Match search

The search for matches is performed separately for each formatting record. This process is done by the `findMatches()` function. First, the `re.finditer()` function is called, using the regular expression from the formatting record. Then each match, represented by a pair of beginning and ending indexes, is stored in a list, which is then stored in a master list. After the search was performed for all formatting records, the master list containing all matches in separate sublists for each formatting record is returned from the function.

4 Tag insertion

Tags are inserted into the text in the order, in which formatting records were defined. Since matches are stored as pairs of beginning and ending indexes of the match, if one or more tags were inserted into the text before one of these indexes, these indexes must be corrected to preserve the consistency of highlighting. Because of that, tags are inserted from the end of the text to the beginning, in order to decrease the number of match corrections.

5 Extensions

The NQS extension is included in this implementation of the script. It's performed by the `nqsResolve()` function, which replaces consecutive quantifiers with a corresponding single quantifier.