Dokumentace úlohy SYN: Zvýraznění syntaxe v Pythonu 3 do IPP 2011/2012
Jméno a příjmení: Pavel Frýz
Login: xfryzp00

Dokumentace k projektu IPP: Zvýraznění syntaxe v Pythonu 3

1. Zadání

Úkolem projektu bylo implementovat skript, pro zvýraznění vstupního souboru, dle zadaného formátovacího souboru. Skript na vstupu přijímá libovolný textový soubor a na výstup pak generuje tento soubor doplněný o formátovací tagy specifikované ve formátovacím souboru.

2. Popis řešení

Zpracování parametrů

Pro zpracování parametrů je využit modul `argparse`, který kromě zpracování parametrů generuje způsob použití a nápovědu. Prvně se vytvoří instance třídy `ArgumentParser`, poté se přidají požadované parametry a nakonec se zavolá metoda `parse_args` pro zpracování parametrů. Pokud zpracování proběhne úspěšně, zkontroluje se, zda nebyl některý parametr zadán vícekrát nebo parametr `--help` nebyl kombinován s jiným parametrem. Pokud nastane chyba při zpracování parametrů, skript končí s chybovým kódem 1. Zadané parametry lze zkrátit, pokud zkrácením nedojde k nejednoznačnosti parametrů.

Načtení vstupního souboru

Pokud je zadán parametr `--input=SOUBOR`, skript načítá vstup ze zadaného souboru, v opačném případě načítá vstup ze standardního vstupu. Pokud se nepodaří vstupní soubor otevřít, skript končí s chybovým kódem 2.

Generování výstupního souboru

Pokud je zadán parametr –`output=SOUBOR`, program vypíše výsledný dokument do zadaného souboru, jinak ho vypíše na standardní výstup. Pokud není zadán parametr `--format=SOUBOR`, tak skript pouze zkopíruje vstup na výstup, v opačném případě se vstup doplní o formátovací tagy, dle zadaného souboru. Zpracování formátovacího souboru probíha po řádcích. Každý řádek se rozdělí znakem tabulátoru na část s regulárním výrazem a na část s formátovacími řetězci. Zadaný regulární výraz je pak pomocí posloupnosti regulárních výrazů Pythonu, upraven na regulární výraz Pythonu. Z části s formátovacími řetězci jsou pak vytvořeny dva řetězce, jeden s počátečními tagy a druhý s koncovými tagy. Pořadí tagů v řetězci s počátečními tagy je stejné jako pořadí formátovacích řetězců na řádku, koncové tagy jsou v opačném pořadí. Řetězec s počátečními tagy je poté uložen do asociativního pole, kde klíče tvoří pozice řetězců, které vyhovují danému regulárnímu výrazu, ve vstupním souboru. Řetězec s koncovými tagy je uložen do téhož pole, kde klíče tvoří pozice konců těchto řetězců. Pokud se na dané pozici již nachází nějaký řetězec s tagy, tak je řetězec s počátečními tagy připojen k tomuto řetězci zleva a řetězec s koncovými tagy zprava. Poté co jsou zpracovány všechny řádky formátovacího souboru, projde se pole s tagy seřazené sestupně podle klíče, a do vstupního souboru se na dané pozice vloží tagy z tohoto pole. Pokud byl zadán přepínač `--br`, tak se ještě přidá na konce řádků tag `<br />`. Pokud nastane při zpracování formátovacího souboru chyba, skript končí s návratovým kódem 2, pokud se nepodaří otevřít výstupní soubor skript končí s návratovým kódem 3.

Zpracování regulárního výrazu

Prvně se odstraní všechny výskyty znaku '.' značící konkatenaci. Poté se nahradí posloupnosti znaků '+' nebo '*' znakem '+', pokud v posloupnosti nebyl znak '*', jinak se nahradí znakem '*'. Poté se nahradí tyto znaky i když mezi nimi byla závorka. Pak jsou escapovány všechny znaky, které mají speciální význam pouze v regulárních výrazech Pythonu. Následně jsou nahrazeny výrazy začínající '!' a nakonec výrazy začínající '%'.

3. Rozšíření

V programu je implementováno rozšíření NQS, které přidává podporu použití '+' nebo '*' za sebou. Toho bylo docíleno nahrazením této posloupnosti pouze jedním z těchto znaků. Viz zpracování regulárního výrazu.

4.Závěr

Program byl testován na dodaných a mnou vytvořených testech. Během testování bylo odhaleno několik chyb, které byly následně opraveny. Testování probíhalo na počítači s operačním systémemu Ubuntu a školním serveru Merlin s operačním systémem CentOS. Výsledný program dodržuje formát vstupních a výstupních dat, díky tomu je možné využití ve skriptech nebo spolupráce s jinými programi.