Dokumentace úlohy SYN: Zvýraznění syntaxe v Python 3 do IPP 2015/2016
Jméno a příjmení: Patrik Segedy
Login: xseged00

Úvod

Tato dokumentace pojednává o řešení druhého projektu do předmětu IPP. Jedná se o skript v jazyce Python ve verzi 3.4.3. Účelem skriptu je automatické zvýrazňování různých částí textu na základě formátovací tabulky.

Dokumentace obsahuje způsob zpracování argumentů, zpracování formátovací tabulky, způsob konverze regulárního výrazu a aplikace zadaného formátování na text.

Zpracování parametrů

K zpracování parametrů je použitá metody `parse_args()` z modulu `argparse`. Tato funkce prohledává argumenty příkazové řádky, udělá konverzi na požadovaný typ a vrací objekt typu `Namespace`.

Při volání metody parse_args() se odchytávají výjimky, které jsou nahrazeny chybovým kódem podle zadání.

Zpracování formátovací tabulky

Pro zpracování formátovací tabulky slouží funkce `readFormatFile()`, která čte formátovací soubor po řádcích a rozdělí řádek pomocí funkce `partition()` podle znaku '/'. Výstupem funkce `partition()` je trojice řetězců, řetězec před znakem '/', řetězec obsahující posloupnost znaků '/' a řetězec, který se nachází za posloupností znaků '/'. První a třetí posloupnost se uloží do seznamu.

Výstupem funkce `readFormatFile()` je seznam seznamů ve tvaru:
`[['regulární výraz','formátovací příkazy'],[...]]`.

Konverze regulárního výrazu

K vyhodnocení regulárního výrazu je potřebné ho nejprve převést na regulární výraz typu, který používá jazyk Python.

Konverze probíhá kombinací regulárních výrazů a stavového automatu. Regulární výrazy nejdřív zkontrolují některé z podmínek, aby byl výraz platný. Například, že výraz nemůže obsahovat dvě a více znaků '|' za sebou, nebo že nemůže začínat tečkou.

Stavový automat má 3 stavy:

- S - je počáteční stav, ve kterém se řeší nepovolené znaky za aktuálním znakem, negace jednoho znaku a přechází se do ostatních stavů
- negate - řeší negaci před závorkou pomocí *negative lookahead* a kontroluje, zda se neguje pouze jeden znak uvnitř závorek
- perc - překládá regulární výrazy začínající znakem '%' do formátu regulárních výrazů pro jazyk Python

Ostatní chyby jako například zlý počet závorek kontroluje kompilátor regulárních výrazů jazyka Python, voláním `re.compile()` nad konvertovaným regulárním výrazem. V případě chyby se odchytí výjimka a nahradí se chybovým kódem podle zadání.

Aplikace formátovaní

Pomocí funkce `convertTag(inTag, openTag)`, převedu formátovací příkazy z formátovací tabulky na požadované tagy, které uložím do seznamu. Argument `inTag` jsou vstupní formátovací příkazy a argument `openTag` udává, zda je tag otevírací nebo uzavírací. Když se konvertují uzavírací tagy, tak se seznam na konci obrátí pomocí funkce `reverse()`.

Pro každou dvojici regulárního výrazu a formátovacích příkazů se najde ve vstupním souboru pozice začátku a konce řetězce, který odpovídá regulárnímu výrazu. Následně se do seznamu uloží dvojice, která

obsahuje pozici a seznam tagů na dané pozici. Otevírací tagy se přidávají na konec seznamu a uzavírací na začátek seznamu. Seznam pak může vypadat následovně:

```
[('3', ['<b>', '<i>']),('4', ['</i>', '</b>']),(...),...]
```

Obsah vstupního souboru se prochází po znaku. Když na pozici nemá být žádný tag, tak se znak zkopíruje do výstupního řetězce. Když se narazí na pozici, na které má být nějaký tag, tak se do výstupního řetězce uloží všechny tagy, které mají být na dané pozici a znak který má následovat za nimi. Na konci se výstupní řetězec zapíše do výstupního souboru.

Problémem ze začátku bylo, že tagy které se nacházeli za posledním znakem nebyli zapsány do výstupu. Tenhle problém vznikl z důvodu, že tyto tagy měli pozici větší, jako byla maximální pozice ve vstupním souboru. Proto se ještě nakonec do výstupu přidají tagy, které mají index větší než maximální index ve vstupním souboru.

Závěr

Projekt byl vyvíjený na referenčním školním serveru merlin, takže byla zaručena funkčnost řešení na požadované verzi jazyka Python a nebylo potřebné instalovat žádné moduly.

K testování byli využité zveřejněné ukázkové testy s doplněnými testy a upravené na automatické porovnání souborů pomocí utility diff.