Dokumentace úlohy CSV: CSV2XML v Python 3 do IPP 2016/2017

Jméno a příjmení: Adam Šulc

Login: xsulca00

Řešení projektu CSV bylo implementováno v jazyce Python 3 s využitím některých modulů, které jsou jeho součástí.

Prvně se řeší získávání zadaných voleb programu a jednoduché, intuitivní rozhraní, které bude tyto volby reprezentovat. K tomuto účelu jsem použil modul „argparse“ naprosto vyhovující mým požadavkům.

Nejprve bylo potřeba načíst vstupní CSV soubor, který dostal program na vstupu. K tomuto účelu jsem využil modul CSV, který disponuje potřebnými funkcemi a umožnil mi dle nastavených parametrů načíst jednotlivé řádky a sloupce, přes které jde iterovat skrze iterátory. Tato podoba nebyla úplně vhodná, neboť jsem potřeboval jednotlivé buňky nejen modifikovat, ale i přistupovat k nim skrze indexy.

Za tímto účelem jsem překopíroval již zpracovaný CSV soubor do struktury listu, obsahující listy, které obsahují jednotlivé řetězce (buňky).

Poté, co překopíruji jednotlivé buňky a řádky, provedu záměnu nevalidních znaků XML (entity reference) jako např. ampersand, uvozovky za jejich ekvivalenty odpovídající definici XML. Následně se generuje XML hlavička a kořenový element, pokud nebylo pomocí voleb programu zadáno jinak. V případě, že první řádek odpovídá hlavičce, podle které se budou následně generovat elementy, udělám hlubokou kopii první řádku, kterou samostatně uložím do proměnné a smažu první řádek v proměnné, obsahující všechny řádky a sloupce CSV souboru. Během této operace se kontroluje, zda-li je název pro výsledný element, který bude uzavírat jednu buňku, validní a v případě záporné odpovědi se pokusí nahradit nevalidní znaky v názvu validním znakem a poté opět zkontroluje, zda-li název je správný dle syntaxe XML.

Pokud byla zadána volba pro zotavení s chyby, vezme se délka prvního řádku (případně řádku, která odpovídá hlavičce) a zkrátí se délka řádku na odpovídající počet buněk případně doplní se a vyplní do potřebného počtu sloupců. Toto neplatí v případě volby „all columns“. Následuje hlavní cyklus, který generuje výsledný XML řetězec. Zvolil jsem prosté řešení odpovídající tisknutí jednotlivých konstrukcí namísto použití funkcí z modulů, neboť nebylo zapotřebí řešit komplexní struktury jazyka a některé volby programu by bylo nemožné triviálně implementovat. V hlavním cyklu se postupně prochází řádky a jejich sloupce obalené XML elementy, které se podle zadaných voleb programu tisknou do výsledného XML řetězce.

Funkce pro ověření, zda-li název elementu je validní, předepisuje vzor do „regex“ modulu jazyka Python odpovídající přesné definici dle XML standardu. Zde jsem narazil na problémy kódování jednotlivých povolených znaků, které odpovídaly Unicode formátu. Problém jsem vyřešil zakódováním řetězce do posloupnosti bytů a následným dekódováním do UTF-8.