Semesterarbeit Teil 1b: Einlesen von Daten

Aufgabenstellung

Nicht für alle Folgen und Funktionen gibt es eine mathematische Definition; viele werden definiert durch Tabellen (z.B. bei empirischen Daten). Gängige Format für solche Daten sind Excel-Tabellen oder Dateien im csv (comma separated values) Format; dabei können Excel-Tabellen im csv Format exportiert werden.

Schreiben Sie eine Funktion, die eine csv Datei einliest und dabei eine zweidimensionale Python-Liste erstellt.

(Wir werden diese Funktion auch später verwenden.)

Lesen Sie damit z.B. Daten des statistischen Amtes aus http://www.bfs.admin.ch/bfs/portal/de/index/themen.html ein.

Analyse

Das File-Script, welches für diese Aufgabe zu implementieren ist, muss folgende Bedingungen erfüllen:

- Semikolonbasierte CSV-Datei auslesen und in einen zweidimensionalen Array umwandeln.
- Die Quelle zum Auslesen steht frei. Ich habe mich für die Fibonacci-Tabelle aus Teil 1a entschieden. Diese Tabelle habe ich mit Excel in eine einfache, semikolongetrennte CSV-Datei umgewandelt:

Zahl	Fibonacci	Calls
1	1	1
2	1	3
3	2	5
4	3	9
5	5	15
6	8	25
7	13	41
8	21	67
9	34	109
10	55	177
11	89	287
12	144	465
13	233	753
14	377	1219
15	610	1973
16	987	3193
17	1597	5167
18	2584	8361
19	4181	13529
20	6765	21891

Implementation

Struktur

Für diese Übung habe ich 3 Python-Scripts erstellt:

- «test.py»: Dieses Script dient zur Überprüfung der Implementationen.
- «file.py»: Dieses Script enthält die Funktionen zum Auslesen einer Datei.

Test-Script

Dieses Script ist ganz simpel gehalten und verwende ich nur um meine Funktion zu testen.

Das Script liest die Datei «fibonacci.csv» aus dem aktuellen Ordner und gibt deren Inhalt als Python-Array zurück.

File-Script

Dieses Script enthält die Funktionen zum Auslesen einer CSV-Datei:

«read_csv_to_array(filename, csv_seperator = CSV_SEPERATOR)»

Diese Funktion liest die genannte Datei aus und wandelt diese in ein 2D-Array um. Da CSV-Dateien nicht zwangsläufig Semikolonbasiert sind, kann man optional auch noch zum Beispiel ein Komma mitgeben.

Die Funktion öffnet die Datei mit Leserechten (da wir die Schreibrechte nicht benötigen) und erstellt ein neues Array «result». Dann geht sie jede Linie aus der Datei durch, entfernt die Linefeed-Zeichen und teilt die Linie mit dem Parameter «csv_seperator» in einen Array auf, dieses Array wird dann dem Array «result» hinzugefügt. Sobald alle Linien ausgelesen wurden, wird die Datei wieder geschlossen und das Array «result» wird zurückgegeben.

«convert_value_to_number_if_possible(value)» & «convert_array_to_number_array_if_possible(input)»

Diese zwei Funktionen dienen zur Umwandlung von Daten in Zahlen, soweit dies möglich ist. Diese Funktionen habe ich zusätzlich implementiert, da ich mir dachte, wenn wir dieses File-Script zukünftig für dieses Modul verwenden, werden wir bestimmt mit Zahlen aus den CSV-Dateien arbeiten, somit wäre es doch praktisch, wenn man das ausgelesene Array umwandeln könnte, damit man Zahlen hat und nicht Texte.

Die erste Funktion «convert_value_to_number_if_possible(value)» prüft, ob der Parameter «value» eine Liste / ein Array ist, falls dem so ist, wandelt sie das Array mit der zweiten Funktion um und gibt dieses zurück. Falls dem nicht so ist, wird versucht, den Parameter «value» umzuwandeln, dafür wird die «float(value)»-Funktion von Python verwendet. Da diese Funktion jedoch fehlschlagen könnte, falls die Daten nicht in eine Zahl umgewandelt werden können, habe ich die Try-Catch-Funktion von Python verwendet. Wenn also die Daten keine gültige Nummer sind, werden diese direkt zurückgegeben, ohne Umwandlung.

Die zweite Funktion «convert_array_to_number_array_if_possible(input)» wendet die «map(function, input)» an, um ein Array umzumappen, dies bedeutet, jeder Eintrag im Array wird durch die mitgegebene Funktion umgewandelt. Als Funktion geben wir die erste Funktion an, damit werden alle Einträge in der Liste in Nummern umgewandelt.

Der grosse Vorteil in dieser Implementation liegt darin, dass man jetzt die erste Funktion für jeden Typ von Daten verwenden kann:

1. Wenn der Input eine Zahl ist, wird eine Zahl zurückgegeben.
2. Wenn der Input ein Text ist, wird versucht diesen Text umzuwandeln.
3. Wenn der Input ein Array ist, wird für jeden Eintrag wieder beim ersten Punkt gestartet. Somit können auch mehrdimensionale Arrays ohne Umstände umgewandelt werden.

Anwendungsbeispiele für diese Funktionen sind in der «test.py»-Datei vorhanden.