

Semesterarbeit Teil 1b

AWD FS 2018

Simon Egli [simon.egli@students.ffhs.ch]

1 Inhaltsverzeichnis

2	Tabellenverzeichnis	3
3	Vorbereitung	4
4	CSV Spezifikation	5
5	Das Modul CSV	6
6	Implementation	8
6.1	Der Parameter path	9
6.2	Der Parameter seperateHeader	9
6.3	Der Parameter removeEmptyLines	9


2 Tabellenverzeichnis

Tabelle 1: Risikoanalyse anhand RFC 4180	5
Tabelle 2: Bestimmung der Verantwortlichkeit zur Minimierung der Risiken	6

3 Vorbereitung

Dieses Dokument beinhaltet die Lösung der Semesterarbeit Teil 1b. Das gestellte Problem wird nach dem Ansatz «Think first, then act» gelöst.

Erst nach der theoretischen Konzeption sollen Python Funktionen implementiert werden. Die so geschaffenen Quellcode-Referenzen werden wie folgt dargestellt:



```
from PythonDatei import *  
print( funktion( parameter ) )
```

Unter Berücksichtigung des Dateipfads, können diese kopiert und eigenständig ausgeführt werden.

Als Arbeitsumgebung (IDE) wird Visual Studio Code verwendet:

<https://code.visualstudio.com/>

Um Python entwickeln zu können, wird die von Microsoft zur Verfügung gestellte Extension verwendet:

<https://marketplace.visualstudio.com/items?itemName=ms-python.python>

Zudem wird der Funktionsumfang der IDE mit der Extension Code Runner erweitert:

<https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner>

Wichtige Erkenntnisse und Hinweise werden dargestellt als:



Dies ist ein wichtiger Hinweis.

Der mit dieser Arbeit abgegebene Quellcode entstand aus dem Anspruch heraus erste Erfahrungen in der Python-Programmierung zu sammeln und mathematische Fertigkeiten zu schulen.

Der Quellcode entspricht nicht den in der Softwareentwicklung üblichen Qualitätsansprüchen.

Das heisst, es werden keine automatischen Tests abgegeben, die die Korrektheit der Implementation verifizieren. Auf eine korrekte Fehlerbehandlung wurde geachtet.

4 CSV Spezifikation

Die Erfahrung hat gezeigt, dass einlesen von Daten meist keine grosse Herausforderung darstellt. Wichtigster Punkt dabei ist die Wahl des Werkzeugs (Programmiersprache, Bibliotheken).

Die korrekte Verarbeitung (parsen) hingegen ist oftmals nicht trivial. Dabei gilt, je generischer die Spezifikation, desto höher die Anforderungen an die Implementation.

Aus diesem Grund soll in einem ersten Schritt die Spezifikation des Dateiformats CSV auf mögliche Risiken hin untersucht werden. Als Quelle dient der RFC Standard 4180¹. Die Ergebnisse werden Stichwortartig festgehalten.

#	Erkenntnis	Mögliches Risiko / Konsequenzen
1	Mime Type = text/csv	Wie werden CSV Dateien als solche erkannt?
2	Verschiedene Spezifikationen vorhanden	Viele Spezialfälle die beachtet werden müssen
3	End line feed ist optional	Gefahr einer leeren Zeile nach dem parsen
4	Header ist optional	Kopfzeile muss zuverlässig erkannt werden
5	Felder können optional in Doppelte Anführungszeichen gesetzt werden	Doppelte Anführungszeichen und deren „escape“ Möglichkeiten müssen speziell beachtet werden
6	Common usage of CSV is US-ASCII	Interpretation von Zeichen ausserhalb des ASCII Character-Sets muss geprüft werden.
7	Möglichkeit zur Ausführung von Schadsoftware	Muss analysiert werden.
8	Keine Formate spezifiziert.	Müssen z.B. Daten erkannt werden, wenn ja welche Formate?
9	Keine Eindeutige Spezifikation für Leer-Felder	Muss analysiert werden.
10	Excel bietet die Möglichkeiten Formel zu hinterlegen und Felder miteinander zu verbinden.	Wie werden Verknüpfungen geschaffen und wie werden diese verarbeitet?

Tabelle 1: Risikoanalyse anhand RFC 4180

¹ <https://tools.ietf.org/html/rfc4180.html#ref-4>

5 Das Modul CSV

Gemäss der Aufgabenstellung muss für die Implementation das Python Modul CSV² verwendet werden.

Im Folgenden soll dessen Funktionsumfang geprüft werden. Einerseits müssen relevante Informationen für die Implementierung festgehalten, andererseits die in Kapitel 4 gesammelten Risiken mit der Funktionalität des Moduls verglichen werden.

In Bezug auf die Risiken, können folgende Aussagen gemacht werden:

#	Risiko	Verantwortung	Bemerkung
1	Mime Type = text/csv	CSV Module	<code>csv.Sniffer</code>
2	Verschiedene Spezifikationen vorhanden	CSV Module	<code>csv.Sniffer</code>
3	End line feed ist optional	CSV Module	Standard-Feature
4	Header ist optional	CSV Module	<code>csv.Sniffer.hasHeader()</code>
5	Felder können optional in Doppelte Anführungszeichen gesetzt werden	CSV Module	Standard-Feature
6	Common usage of CSV is US-ASCII	Python / CSV Module	Default encoding ist unicode.
7	Möglichkeit zur Ausführung von Schadsoftware	CSV Module / Entwickler	Ist immer möglich -> Die Verantwortung muss geteilt werden.
8	Keine Formate spezifiziert.	CSV Module	Standard-Feature
9	Keine Eindeutige Spezifikation für Leer-Felder	CSV Module	Standard-Feature
10	Excel bietet die Möglichkeiten Formel zu hinterlegen und Felder miteinander zu verbinden.	Entwickler	Muss eigenhändig implementiert werden

Tabelle 2: Bestimmung der Verantwortlichkeit zur Minimierung der Risiken

² <https://docs.python.org/3/library/csv.html>

Ein Zitat aus der CSV Modul Dokumentation bestätigt, dass die Intension des Moduls nahezu deckungsgleich mit den festgehaltenen Risiken ist:

«The lack of a well-defined standard means that subtle differences often exist in the data produced and consumed by different applications. These differences can make it annoying to process CSV files from multiple sources. Still, while the delimiters and quoting characters vary, the overall format is similar enough that it is possible to write a single module which can efficiently manipulate such data, hiding the details of reading and writing the data from the programmer. »

Beachtet werden müssen die beiden Punkte:

7) Möglichkeit zur Ausführung von Schadsoftware

Software kann nie als 100% sicher gelten. Um möglichst hohe Sicherheit bieten zu können, muss eine Applikation regelmässig gewartet und verbessert werden.

Da es sich hier um eine Übung handelt, wird die Applikation mit Appel an die Eigenverantwortung und der Mahnung, nur CSV-Dateien aus vertrauenswürdigen Quellen einzulesen, freigegeben.



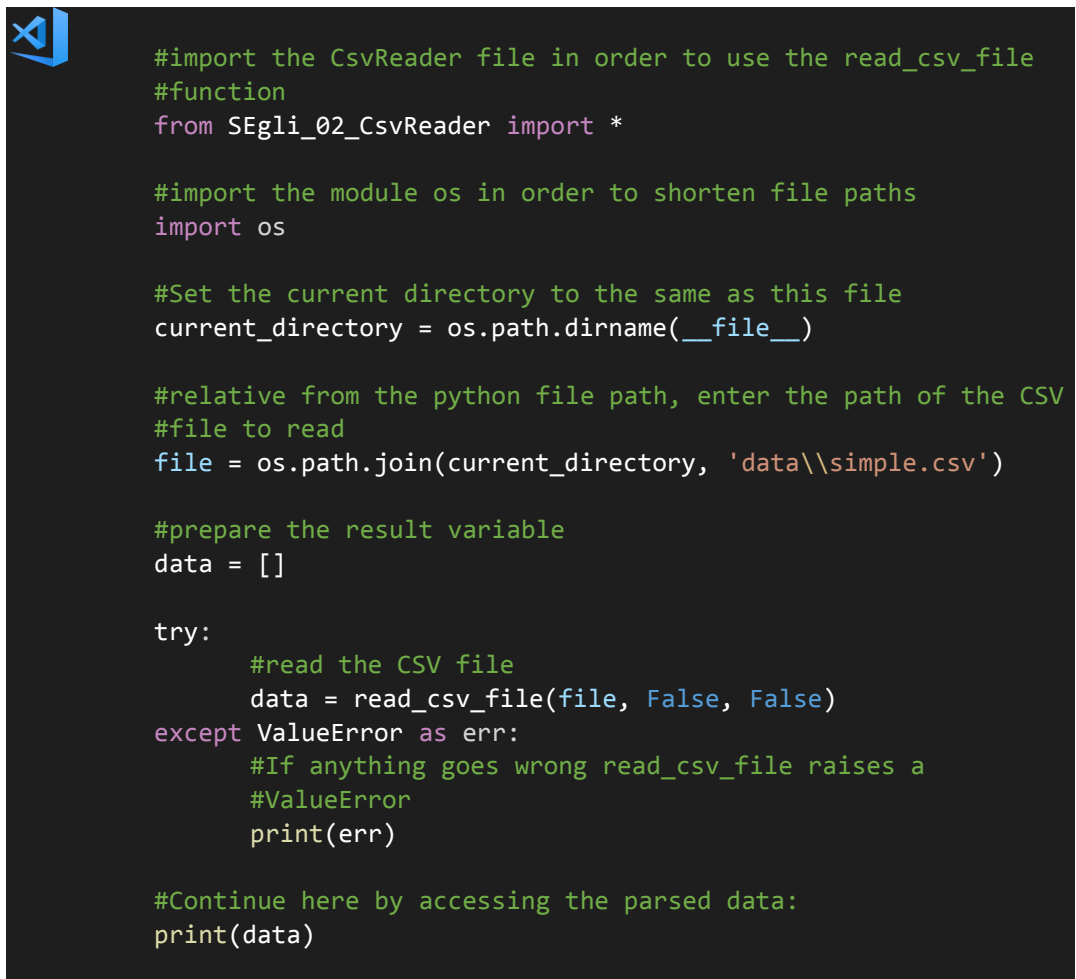
Die meisten gängigen Antivirus Programme bieten die Möglichkeit einzelne Dateien auf Schadsoftware zu prüfen.

10) Excel bietet die Möglichkeiten Formel zu hinterlegen und Felder miteinander zu verbinden

Die Aufgabenstellung fordert klar, dass CSV Dateien eingelesen und deren Inhalt in einer zweidimensionalen Liste abgelegt wird. Eine Unterstützung bzw. logische Verbindung von Zellen wird aus diesem Grund nicht implementiert.

6 Implementation

Nach ausführlicher Auseinandersetzung mit der Materie, wird eine Funktion `read_csv_file` implementiert, welche wie folgt genutzt werden kann:

A screenshot of a code editor showing a Python script. The script imports the `CsvReader` module from `SEgli_02_CsvReader` and the `os` module. It sets the current directory to the directory of the script file using `os.path.dirname(__file__)`. Then, it constructs the path to a CSV file named `simple.csv` in a subdirectory named `data` using `os.path.join`. A list `data` is initialized to store the results. A `try` block attempts to read the CSV file using the `read_csv_file` function. If a `ValueError` occurs, it is caught and the error is printed. Finally, the parsed data is printed.

```
#import the CsvReader file in order to use the read_csv_file
#function
from SEgli_02_CsvReader import *

#import the module os in order to shorten file paths
import os

#Set the current directory to the same as this file
current_directory = os.path.dirname(__file__)

#relative from the python file path, enter the path of the CSV
#file to read
file = os.path.join(current_directory, 'data\\simple.csv')

#prepare the result variable
data = []

try:
    #read the CSV file
    data = read_csv_file(file, False, False)
except ValueError as err:
    #If anything goes wrong read_csv_file raises a
    #ValueError
    print(err)

#Continue here by accessing the parsed data:
print(data)
```

Obiges Beispiel verwendet das Modul `OS` um den Dateipfad der aktuellen Python-Datei zu bestimmen. Der Pfad zur gewünschten CSV Datei wird relativ bestimmt. Wie genau der CSV-Pfad ermittelt wird, bleibt dem Anwender überlassen.

6.1 Der Parameter **path**

Type : **String** oder **Bytes** Object

Die Funktion **read_csv_file** erwartet einen Parameter **path**, welcher die zu lesende CSV-Datei identifiziert.

Je nach Betriebssystem werden unterschiedliche Formatierungen verlangt. Eine genaue Spezifikation kann der Python Dokumentation entnommen werden:

<https://docs.python.org/3.2/library/functions.html#open>

6.2 Der Parameter **seperateHeader**

Type : **Boolean**

Dient zur Festlegung, ob die (eventuell vorhandene) Kopfzeile des Datensatzes vom Körper separiert werden soll. Wird **True** übergeben, wird das Resultat als **Dictionary** in folgender Form zurückgegeben:



```
return {  
    "Header": header,  
    "Body": data  
}
```

Zur Bestimmung der Kopfzeile wird die Funktion **csv.Sniffer().has_header** verwendet. Kann keine Kopfzeile ermittelt werden oder ist diese leer, wird im Feld **Header** eine leere Liste zurückgegeben.

Das Feld **Body** enthält alle Zeilen des Datensatzes als Zweidimensionale Liste, ohne Kopfzeile.

Diese Funktion ist hilfreich, wenn die Kopfzeile identifiziert werden muss.

Wird **False** übergeben, wird eine zweidimensionale Liste mit allen gefundenen Datensätzen zurückgegeben.

6.3 Der Parameter **removeEmptyLines**

Type : **Boolean**

Wird **True** übergeben, so werden alle leeren Zeilen **[]** aus dem Datenkörper entfernt. Lediglich komplett leere Zeilen werden entfernt. Leere Felder werden ignoriert.

Wird **False** übergeben, bleibt der Datenkörper unverändert.