Semesterarbeit Teil 1b

AWD FS 2018

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

Inhaltsverzeichnis

[2 Abbildungsverzeichnis 3](#_Toc507450603)

[3 Tabellenverzeichnis 4](#_Toc507450604)

[4 Vorbereitung 5](#_Toc507450605)

[5 CSV Spezifikation 6](#_Toc507450606)

[6 Das Modul CSV 7](#_Toc507450607)

# Abbildungsverzeichnis

**No table of figures entries found.**

# Tabellenverzeichnis

[Tabelle 1: Risikoanalyse anhand RFC 4180 6](#_Toc507450595)

[Tabelle 2: Bestimmung der Verantwortlichkeit zur Minimierung der Risiken 7](#_Toc507450596)

# 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. Auch wurde auf eine ausgiebige Fehlerbehandlung verzichtet.

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 4180[[1]](#footnote-1). Die Ergebnisse werden Stichwortartig festgehalten.

|  |  |  |
| --- | --- | --- |
| # | Erkenntnis | Mögliches Risiko / Konsequenzen |
| 1 | Meme 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

Das Modul CSV

Gemäss der Aufgabenstellung muss für die Implementation das Python Modul CSV[[2]](#footnote-2) verwendet werden.

Im Folgenden soll dessen Funktionsumfang geprüft werden. Einerseits müssen relevante Informationen für die Implementierung festgehalten, anderseits die in Kapitel 5 gesammelten Risiken mit der Funktionalität des Moduls verglichen werden.

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

|  |  |  |  |
| --- | --- | --- | --- |
| # | Risiko | Verantwortung | Bemerkung |
| 1 | Meme Type = text/csv | CSV Module | csv.Sniffer |
| 2 | Verschiedene Spezifikationen vorhanden | CSV Module | csv.Sniffer |
| 3 | End line feed ist optional | CSV Module | Standard-Feature |
| 4 | Header ist optional | CSV Module | csv.Sniffer.hasHeader() |
| 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

Die Dokumentation des CSV Moduls zeigt, dass das Modul genau aus dem Grund entwickelt wurde, möglichst einfach CSV Dateien zu lesen und schreiben.

1. https://tools.ietf.org/html/rfc4180.html#ref-4 [↑](#footnote-ref-1)
2. https://docs.python.org/3/library/csv.html [↑](#footnote-ref-2)