IWGS1_WS1920_Zettel10

July 7, 2020

1 Informatische Werkzeuge in den Geistes- und Sozialwissenschaften I

1.1 Hausaufgabe 10 (Projekt.1, 100 Punkte)

Erschienen: 21.12.2019
Abgabe bis: 12.01.2020

Bitte laden Sie Ihre Notebooks bis 23:59 Uhr am Abgabetag in Ihrer Übungsgruppe bei StudOn hoch.

Wenn Ihnen einige der hier verwendeten Konzepte unbekannt sind oder Sie nicht wissen, wie Sie fortfahren sollen, können Sie die Vorlesungsunterlagen zu Rate ziehen oder jederzeit Fragen im Forum oder auf Slack stellen, im Tutorium nachfragen, oder Ihrem Tutor eine Mail schreiben.

1.1.1 Weihnachtsamnestie

Wie bereits letztes Jahr wollen wir auch in diesem Durchlauf von IWGS eine "Weihnachtsamnestie" durchführen. Das bedeutet, dass Sie alle dazu eingeladen sind, auch vergangene Übungszettel an Ihre Tutoren (per Mail) einzureichen. Diese werden dann genau so korrigiert wie andere Übungszettel auch und Sie erhalten gleichermaßen Punkte und Feedback.

Wir sind der Meinung, dass Abgabefristen helfen können, sich tatsächlich zu den Übungszetteln aufzuraffen, aber wir wollen nicht, dass diese im Weg stehen, sich auch später noch mit vielleicht verpassten oder nicht abgegebenen Zetteln auseinander zu setzen. Wenn Sie also bisher nicht alle Zettel abgegeben haben, können Sie sicherlich davon profitieren. Wir nehmen auch weiterhin gerne Abgaben zu alten Zetteln an.

1.1.2 Einleitung in das Projekt

Über die nächsten Wochen werden Sie die Gelegenheit haben, in den IWGS-Übungen Ihr erstes Softwareprojekt mit mehreren Komponenten entwickeln. Alle Komponenten werden von Ihnen programmiert werden, basierend auf Techniken die Sie bereits gelernt haben und solchen, die Sie noch lernen werden. Am Ende soll ein browser-basiertes, leicht erweiterbares *Choose-Your-Own-Adventure*-Abenteuer, ein so genanntes Spielbuch, entstehen.

Den Anfang bildet dieses Übungsblatt. Hier werden wir uns damit auseinander setzen, Inhalte aus Dateien auszulesen und zu unseren Zwecken weiter zu verarbeiten, ähnlich wie auf Zettel 8. Das

wird notwendig sein, da wir alle einzelnen Abschnitte unseres Abenteuers (inklusive Titel, Text und den verschiedenen Handlungsmöglichkeiten) aus Dateien auslesen werden wollen. So können auch Leute die Geschichte schreiben und erweitern, die nicht programmieren können.

1.1.3 Aufgabe P1.1 (Dateien aus Unterordnern, 25 Punkte)

Zunächst wird Ihre Aufgabe sein, alle Dateien ausfindig zu machen, die die Endung `.txt` aufweisen, da wir die einzelnen Elemente der Geschichte in solchen Dateien abspeichern wollen. Wir wollen außerdem den Überblick in unserer Ordnerstruktur behalten, also werden wir alle relevanten Dateien nicht direkt neben dieses Notebook legen, sondern in einen Unterordner namens `resources`.

Sie können weiterhin mit der Funktion `open()` arbeiten, um Dateien zu öffnen. Allerdings werden Sie zusätzlich zum eigentlichen Dateinamen auch den Ordner, in dem sich die Datei befindet, angeben müssen.

```
# Öffnet die Datei "example.txt", die im gleichen Ordner wie das Notebook liegt.
open("example.txt")

# Öffnet die Datei "example.txt", die im Unterordner "resources" liegt.
open("resources/example.txt")
```

Um an alle Dateinamen in einem Ordner zu kommen, können Sie die Methode `os.listdir()` verwenden. Übergeben Sie dieser einen Ordnernamen als String als Parameter, bekommen Sie eine Liste von Dateinamen in diesem Ordner zurück. Denken Sie daran, dass Sie zunächst die Bibliothek `os` importieren müssen, bevor Sie diese Methode nutzen können.

Für diese Aufgabe (und folgende) könnten außerdem die String-Methoden `startswith()` und/oder `endswith()` hilfreich sein. Diese geben einen Boolschen Wert zurück, wenn der String tatsächlich mit dem als Parameter übergebenen String beginnt oder endet.

```
text = "Das Pferd frisst keinen Gurkensalat"
text.endswith("Gurkensalat")     # True
text.endswith("Kartoffelsalat") # False
```

Aufgabe: Schreiben Sie ein kurzes Python-Programm, das alle Dateien im Unterordner `resources` mit der Dateiendung `.txt` auflistet und jeweils angibt, wie viele Zeichen in diesen Dateien enthalten sind.

[1]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

1.1.4 Aufgabe P1.2 (Finden des Titels, 20 Punkte)

Als nächstes wollen wir von einer gegebenen Storyelement-Datei im Ordner `resources` den Titel herausfinden. Der Titel einer Story-Datei (falls sie einen hat) ist immer in der ersten Zeile zu finden und wird dadurch markiert, dass diese erste Zeile mit dem String `"TITLE;"` beginnt. Ist also die erste einer Zeile einer Datei `"TITLE;London Bridge"` so hat die Datei den Titel "London Bridge". Ist die erste Zeile der Datei `"London Bridge is falling down!"`, so hat sie keinen Titel (weil der entsprechende Marker fehlt).

Zu diesem Zwecke können sich die Methoden `readline()` (`readlines()`!) als hilfreich erweisen, welche, aufgerufen auf einer frisch geöffneten Datei, genau die erste Zeile der Datei zurück gibt.

Aufgabe: Schreiben Sie ein kurzes Pythonprogramm, dass zunächst einen Dateinamen abfragt. Danach soll überprüft werden, ob diese Datei existiert. Falls sie das nicht tut, soll eine Fehlermeldung ausgegeben werden. Falls doch soll die erste Zeile der Datei eingelesen werden. Startet diese mit dem String `"TITLE;"`, so soll eine Erfolgsmeldung ausgegeben werden, die den Dateinamen und den Titel (= den Rest der Zeile, *ohne* das `"TITLE;"`) enthält. Startet Sie mit einem anderen String, so soll abermals ein Fehler gemeldet werden, diese Datei hätte keinen Titel.

[2]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

1.1.5 Aufgabe P1.3 (Finden der Fortsetzungen, 25 Punkte)

Die nächste wichtige Aufgabe ist es, die unterschiedlichen Handlungsmöglichkeiten aus den Storydateien zu extrahieren. Diese bestehen für unsere Zwecke immer aus einem Marker `OPTION;`, einem Ziel (welche Datei aufgerufen werden soll, wenn diese Option ausgewählt wird) und einer Beschreibung der Handlungsoption für die Spielenden. In unseren Dateien werden diese durch `;` getrennt. Eine mögliche Handlungsoption würde also so aussehen:

`OPTION;elephant;Ich mache meinen Frieden damit, heute keine Erdnüsse zu essen und gebe sie alle dem Elefanten.`

Diese Handlungsoptionen müssen jetzt aus den Dateien ausgelesen werden. Wichtig ist dabei, dass immer klar bleibt, welches *Ziel* mit welcher *Beschreibung* verknüpft sein soll. Speichern Sie diese also am besten jeweils zusammen in einer Liste oder einem Tupel. Der Marker interessiert uns allerdings nicht, sobald wir mit dem Auslesen fertig sind (er ist ja auch immer gleich). Also muss dieser nicht weiter gespeichert werden.

Ein weiterer Hinweis: Mit der String-Methode `split()` können Sie einen String an einem beliebigen Trennzeichen teilen. Sie erhalten dann eine Liste mit den getrennten Teilen des Strings als Elemente, ohne irgendwelche Trennzeichen.

```
text = "comma,separated,values"
text.split(",") # Evaluiert zur Liste ['comma','separated','values']
```

Aufgabe: Schreiben Sie ein weiteres kurzes Pythonprogramm, dass zunächst einen Dateinamen abfragt. Danach soll überprüft werden, ob diese Datei existiert. Falls sie das nicht tut, soll eine Fehlermeldung ausgegeben werden. Falls doch sollen alle Zeilen der Datei, die mit dem String `"OPTION;"`beginnen, in Ziel und Beschreibung geteilt werden. Bilden Sie jeweils aus einem Ziel und der dazugehörigen Beschreibung eine Liste oder ein Tupel. Ihr Programm soll am Ende eine Liste von allen diesen zusammengehörigen Paaren, die in der Datei vorkommen, ausgeben.

[3]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

1.1.6 Aufgabe P1.4 (Finden des Textes, 15 Punkte)

Nachdem Titel und Handlungsoptionen aus dem Text extrahiert wurden, bleibt nun lediglich der eigentliche Erzähltext der Datei übrig, also alles was kein Titel und keine Handlungsoption ist. Es wird notwendig werden, auch diesen leicht zugänglich zu machen.

Aufgabe: Schreiben Sie nun eine Python-*Funktion* (ein Programm, siehe Skript) namens `contents` die einen Dateinamen (also einen String) als Parameter entgegen nimmt. Die Funktion soll zuerst überprüfen, ob die Datei existiert. Falls nein, soll `None` zurück gegeben werden. Falls doch, soll die Funktion alle Zeilen der Datei sammeln, die *nicht* mit `"TITLE;"` oder `"OPTION;"` anfangen, und zusammen als einen großen String zurück geben.

[4]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```

1.1.7 Aufgabe P1.5 (Konkretisierung des Textes, 15 Punkte)

Dies ist eine Folgeaufgabe zu P1.4. In der finalen Version des Abenteuers soll der Name der Hauptfigur anpasspar sein. Spielende sollen selbst entscheiden können, unter welchem Namen sie am Abenteuer teilnehmen. Um diesen Effekt zu erreichen werden alle Storydateien nur den Platzhalter `"NAME"` enthalten, wann immer auf den tatsächlichen Namen der Figur referenziert wird. Dieser Platzhalter wird dann von Ihnen durch den gegebenen Namen ersetzt werden.

Aufgabe: Schreiben Sie ein kurzes Python-Programm, dass zunächst einen Dateinamen und dann einen Charakternamen abfragt. Lesen Sie den Erzähltext der entsprechenden Datei mit `contents()` aus P1.4 ein und ersetzen Sie mit der `re`-Bibliothek und regulärem Ausdruck den Platzhalter `"NAME"` durch den angegebenen Charakternamen. Geben Sie den resultierenden String aus.

[5]:
```
# Ihr Code hier! (Doppelklick zum Editieren, Shift + Enter um die Zelle␣
↪auszuführen)
```