# IWGS2_SS20_Zettel1

July 7, 2020

# 1 Informatische Werkzeuge in den Geistes- und Sozialwissenschaften II

## 1.1 Hausaufgabe 1 (Wiederholung von IWGS 1)

**Erschienen:** 25.04.2020
**Abgabe bis:** 03.05.2020

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.

## 1.2 Einführung

In IWGS 2 werden wir weitere Tools lernen, die in der Praxis nützlich für digitale Geistes- und SozialwissenschaftlerInnen sein können. Wie schon am Ende von IWGS 1 werden wir dazu ein Projekt erstellen, was sich diesmal über das komplette Semester zieht. Dieses Projekt wird diesmal ein voll funktionsfähiger Webserver sein, der eine Bilddatenbank verwaltet. BenutzerInnen wird es möglich sein, über eine Web-Oberfläche Einträge zu der Datenbank hinzuzufügen oder zu löschen. Außerdem können Metadaten angegeben und Annotationen hinzugefügt werden.

Dazu bauen wir auf die erlernten Konzepte aus IWGS 1 auf. Dieses Übungsblatt dient dazu, diese zu wiederholen und schon kleine Schritte in Richtung unseres Servers zu machen.

Wie schon letztes Semester möchten wir Sie dazu ermutigen, jederzeit mit Fragen auf uns Tutoren zuzukommen. Wir helfen sehr gerne bei Unklarheiten! Außerdem: Ein größeres Projekt wie dieses löst sich am Besten in Teams. Gerne können Sie deshalb die Hausaufgaben in Gruppen erledigen!

## 1.3 Aufgabe 1.1 (Dateisystem)

Zur Erinnerung: Wenn Sie in Python Funktionalität aus anderen Modulen verwenden möchten, müssen Sie dazu zunächst das entsprechende Modul importieren. Um beispielsweise das os-Modul verwenden zu können, importieren Sie am Anfang Ihrer Skripte:

```
import os
```

Mit dem os-Modul können Sie das Betriebssystem (os=Operating System) nach Informationen zu Dateien fragen.

Zusammen mit diesem Aufgabenblatt finden Sie einen Ordner *data*. In diesem befinden sich ein paar wenige Bilder, die für dieses erste Übungsblatt als unsere "Datenbank" fungieren. Später ersetzen wir das mit einer echten Datenbank.

**Aufgabe:** Schreiben Sie in der nächsten Zelle ein Python-Programm, welches os.listdir verwendet, um eine Liste mit allen Dateien in dem Ordner *data* zu bekommen. Geben Sie die Liste aus.

[ ]:
```
# Shift + Enter zum Ausfuehren
```

### 1.4 Aufgabe 1.2 (Dateinamen filtern mit Regulären Ausdrücken)

In der ausgegebenen Liste sollten Sie 4 Dateien sehen, von denen 3 Bilder sind. Außerdem ist in dem Ordner eine Textdatei. Wir sind nicht an dieser Datei interessiert, immerhin schreiben wir eine Bilddatenbank!

**Aufgabe:** Erweitern Sie in der nächsten Zelle Ihr Programm von oben. Erstellen Sie eine zweite Liste *images*, welche nur die Namen von Bilddateien enthält. Schreiben Sie dazu eine Schleife, die für jede Datei entscheidet, ob es sich um ein Bild handelt und wenn ja den Namen zu *images* hinzufügt. Sie können auch in der nächsten Zelle noch Ihre Python-Variablen aus der letzten Teilaufgabe verwenden. Wichtig dafür ist nur, dass die erste Zelle vorher mal ausgeführt wurde.

Um zu entscheiden, ob eine Datei ein Bild ist oder nicht, verwenden wir in dieser Aufgabe Reguläre Ausdrücke. Der Dateiname für eine Bilddatei folgt dazu dem folgenden Muster: - Der Name der Datei selber besteht aus Buchstaben (groß und klein), Ziffern, Bindestrichen und Unterstrichen. - Auf den Namen folgt ein Punkt. - Die Dateiendung ist entweder jpg oder png (für unsere Beispiele).

Sie können die Hilfsfunktion `valid_match` verwenden, um zu prüfen ob ein String Ihrem Muster entspricht.

Geben Sie am Ende die neue Liste *images* wieder aus, und stellen Sie sicher, dass wirklich nur genau die 3 Bilddateien darin enthalten sind.

[ ]:
```
import re

def valid_match(regex, string):
    if re.fullmatch(regex, string) is None:
        return False
    else:
        return True



# Ihr Code hier. Shift + Enter zum Ausfuehren
```

## 1.5 Aufgabe 1.3 (Bottle)

In dieser Teilaufgabe machen wir ein paar kleine Schritte in Richtung unseres Webservers. Zusammen mit diesem Aufgabenblatt finden Sie eine Python-Datei *image_database.py*, die das Grundgerüst eines Bottle-Servers enthält.

Wichtig: Die letzte Zeile in dem Skript startet den Server auf einem gegebenen Port. Ändern Sie den Port auf eine andere Zahl (kleiner als 65535), um Konflikte mit Ihren Kommilitonen zu vermeiden.

Wie auch letztes Semester starten wir unseren Bottle-Server aus einem Terminal. Dazu müssen Sie zunächst im Terminal in den Übungsordner navigieren. Zur Erinnerung: Der Befehl `cd` (change directory) wechselt das Verzeichnis. Um in den richtigen Ordner zu gelangen, tippen Sie also `cd Uebungen/IWGS2` und drücken Enter. Von dort aus können Sie das Skript mit dem Befehl `python3 image_database.py` ausführen. Ihre Webseite ist dann verfügbar unter http://jupyter.kwarc.info:32500/. (Sie müssen nur dort auch die Zahl am Ende auf Ihren Port umstellen.) Zum Beenden drücken Sie im Terminal, aus dem Sie den Server gestartet haben, die Tastenkombination `Strg + C`.

Stellen Sie sicher, dass Ihr Webserver funktioniert, indem Sie ihn ausführen und die entsprechende URL im Browser besuchen. Die Seite sollte leer sein (weil wir noch keinen Inhalt eingefügt haben), aber sie sollte erreichbar sein.

**Aufgabe:** Schreiben Sie in der Datei *home.tpl* gültiges HTML, welches eine Webseite erzeugt, die eine Überschrift "Bilddatenbank IWGS" und einen Paragraph mit etwas einleitendem Text enthält. Die Größe der Überschrift und den Inhalt des Paragraphen können Sie frei wählen.

Testen Sie Ihren Server wieder (Sie müssen ihn dazu neu starten). Sie sollten ihre Webseite nun sehen!

## 1.6 Aufgabe 1.4 (Bilder anzeigen)

In dieser letzten Teilaufgabe werden wir die Bilder im *data*-Verzeichnis auf der Webseite anzeigen. Fügen Sie dazu am Anfang des *image_database.py*-Skripts Ihren Code aus Teilaufgabe 1.1 und 1.2 ein, um die Liste mit allen Bilddateien in die Liste *images* zu laden.

Der nächste Schritt ist, diese Liste mit Dateinamen auf Ihrer Webseite anzuzeigen. Die Route `home` ist dafür verantwortlich, die Webseite anzuzeigen. Sie ruft dazu die von Ihnen erstelle HTML-Datei *home.tpl* auf. Damit *home.tpl* die Liste mit Dateinamen anzeigen kann, muss sie Zugriff auf diese Daten haben. Übergeben Sie deshalb in der `home`-Route die Liste mit Dateinamen an das Template.

Zur Erinnerung: Sie können in *tpl*-Dateien Python-Code schreiben und Python-Variablen verwenden.

```
<!-- Random HTML-Stuff -->

    % for i in range(10) :                Python-Code startet mit einem %.
        <p>Das ist Element {{i}}<p>       Mit {{VARIABLE_NAME}} können wir auf Python-Variablen z
    % end                                 Eigenheit von TPL: Schleifen und ifs enden mit end. Das

<!-- More HTML-Stuff -->
```

**Aufgabe:** Fügen Sie in *home.tpl* eine HTML-Liste ein, deren Inhalt mit den Dateinamen aus der Python-Liste gefüllt wird. Ihre Liste sollte somit am Ende drei Elemente haben (für *bild0.jpg*, *bild1.jpg*, *bild2.png*). Verwenden Sie dazu eine Schleife und die Liste!

Testen Sie den Server regelmäßig. Das macht die Fehlersuche deutlich einfacher, als wenn Sie erst allen Code schreiben und danach testen! Denken Sie daran, dass Sie den Server nach jeder Änderung neu starten müssen (Strg + C zum Beenden, dann nochmal `python3 image_database.py` ausführen).

Als letzten Schritt wollen wir die Bilder anzeigen. Ändern Sie dazu die HTML-Liste so, dass jedes Listenelement ein Hyperlink ist. Der Text des Links soll der Dateiname sein. Der Link selber soll auf eine Webseite mit der URL `/static/DATEI_NAME` führen, also zum Beispiel für *bild0.jpg* auf `/static/bild0.jpg`.

Ihre Webseite sollte nun eine Liste mit Links anzeigen. Beim Klicken auf einen Link sollte das entsprechende Bild angezeigt werden.