Name:

Geburtsdatum:

Matrikelnummer:

Klausur
Informatische Werkzeuge in den Geistes- und Sozialwissenschaften 1

18. Februar 2021

	Nur zur Korrektur, bitte freilassen!										
Aufgabe	1.1	1.2	2.1	2.2	2.3	3.1	3.2	4.1	4.2	Summe	Note
Möglich	4	3	10	5	4	5	7	10	12	60	
Erreicht											

Klausurnote: Bonuspunkte: Endnote:

Dir „Lösungen“ der Aufgaben in diesem Dokument sollen den Studierenden als Anfangspunkt für die Beantwortung der Aufgaben dienen. Trotz aller Bemühungen kann es zu unvollständigen oder sogar Fehlern in den „Lösungen“ kommen. Da die Korrektur und Benotung der gestellten Aufgaben niemals lediglich auf einen „Vergleich mit der Musterlösung“ hinausläuft, ist dies auch nicht so schlimm. In jedem Fall sollten die Studierenden die Lösungen nachvollziehen und im Prinzip mittels des Lehrstoffs selbst verifizieren können.

Sollten Sie „Lösungen“ finden, die Sie nicht verstehen oder sogar für fehlerhaft halten diskutieren Sie diese am besten mit den Tutoren oder auf dem Kursforum und benachrichtigen Sie die Lehrenden; wir werden sie dann gegebenenfalls baldigst korrigieren.

1 Grundlagen und Verständnis

Aufgabe 1.1 (Listen und Dictionaries) 4 Pkt

4 min

Erklären Sie kurz Unterschiede und Gemeinsamkeiten von *Listen* und *Dictionaries* in der Programmiersprache python.

Lösung: Sowohl Listen als auch Dictionaries sind Container für Daten. Eine Liste ist jedoch *geordnet*, also mit einer bestimmten Reihenfolge versehen. Daher kann man auf Listenelemente über deren Position in der Liste zugreifen.

Dictionaries dagegen speichern immer Schlüssel/Wert-Paare (key/value-pairs), nicht nur die Werte selbst. Daher kann man auf die Werte über ihren Schlüssel zugreifen.

Aufgabe 1.2 (Dateigrößen und Dateneinheiten) 3 Pkt

Sortieren Sie diese folgenden Dateien der Größe nach, von der kleinsten zur größten. Geben Sie dazu nur die Nummer (n) in der ersten Spalte an. 3 min

```
(1) totally_legally_downloaded_movie.mkv (3.6  Gigabytes)
(2) datensatz.sql                        (400  Exbibytes)
(3) final.pdf                            (1    Mebibyte)
(4) empty.txt                            (0    Byte)
(5) exercise_routine.pdf                 (1025 Kibibyte)
(6) ling_corpus.txt                      (3.5  Gibibytes)
```

Lösung: Die korrekte Lösung lautet:

```
(4) empty.txt                            (0    Byte)
(3) final.pdf                            (1    Mebibyte = 1024 Kibibyte)
(5) exercise_routine.pdf                 (1025 Kibibyte)
(1) totally_legally_downloaded_movie.mkv (3.6  Gigabytes)
(6) ling_corpus.txt                      (3.5  Gibibytes = 3.76 Gigabytes)
(2) datensatz.sql                        (400  Exbibytes)
```

---

2 Digitale Dokumente

Aufgabe 2.1 (HTML Formular Personendaten) 10 Pkt

Schreiben Sie eine gültige HTML-Seite, die eine passende Überschrift enthält und ein HTML-Formular, das aus zwei Textfeldern mit erklärender Beschriftung, zwei beschrifteten Radio-Buttons sowie einem „Absenden“-Button besteht. 10 min

Die Textfelder sollen mit „Name:“ und „Geburtsjahr:“ beschriftet sein. Die Radio-Buttons (von denen jeweils nur einer ausgewählt sein können soll!) sollen mit „Lebendig“ und „Verstorben“ beschriftet sein.

Wird das Formular mit dem „Absenden“-Button abgeschickt, soll der Benutzer oder die Benutzerin auf eine Seite mit einer Empfangsbestätigung weiter geleitet werden.

Sie können hierfür annehmen, dass die (bereits existierende!) entsprechende Datei mit Namen `bestaetigung.html` im gleichen Verzeichnis abgelegt ist. Eine weitere Funktionalität (wie z.B. Auswertung oder Einbindung der Angaben) ist nicht notwendig.

Lösung: Hier ist eine von vielen Möglichkeiten.

```
<html>
    <body>
      <h1>Lebenseckdaten:</h1>
        <form action="./bestaetigung.html">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name"/>
            <br/>

            <label for="birth">Geburtsjahr:</label>
            <input type="text" id="birth" name="birth"/>
            <br/>

            <input type="radio" id="living" name="aliveornot"/>
            <label for="living">Lebendig</label>

            <input type="radio" id="dead" name="aliveornot"/>
            <label for="dead">Verstorben</label>

            <br/><input type="submit" value="Submit"/>
        </form>
    </body>
</html>
```

Aufgabe 2.2 (Mentales CSS) 5 Pkt

Gegeben ist der Quelltext der folgenden HTML-Seite: 5 min

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            body {background-color: green;}
```

```
            h1 {color: blue;}
            div {
                    background—color : red;
                    font—size: 20px;
                    padding: 20px;
                  }
        </style>
    </head>
    <body>

        <h1>Dies ist eine Ueberschrift!</h1>
        <div><p>Und dies ein Paragraph.</p></div>
    </body>
</html>
```

Beschreiben Sie, welche Elemente auf der fertig dargestellten HTML-Seite zu sehen sind, wie sie angeordnet sind und welche Farbe sie haben.

Lösung: Die Website hat einen grünen Hintergrund und beginnt mit einer blauen Überschrift (`<h1></h1>`). Die Überschrift lautet „Dies ist eine Überschrift!“.

Darunter befindet sich eine rote Box (`<div></div>`). In dieser Box steht vertikal zentriert (alternativ: ein in alle Richtungen um 20px gepaddeter) schwarzer Text der Schriftgröße 20px: „Und dies ein Paragraph.“.

Aufgabe 2.3 (XPath) 4 Pkt

Im Kontext von `XML`-Dokumenten, was versteht man unter einem *XPath*? Geben Sie ein Beispiel für einen XPath mit mindestens drei Komponenten an und erklären Sie, was damit erreicht wird. 4 min

---

Lösung: Die *XML Path Language* ist eine Abfragesprache für XML-Dokumente, ähnlich wie reguläre Ausdrücke für unstrukturierten Text. Ein XPath-Ausdruck adressiert Teile eines XML-Dokuments, das dabei als Baum betrachtet wird.

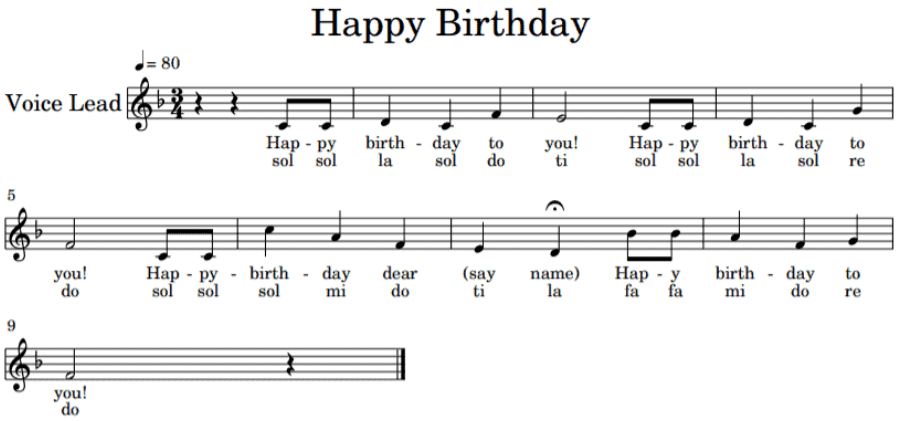
Beispiel: Der XPath `/bookstore/book[price>35]/title` gibt die Titel aller Bücher im Buchladen `bookstore` an, deren Preis über 35 liegt.

---

3 Reguläre Ausdrücke

Aufgabe 3.1 (RegEx für Solfège) 5 Pkt

In der Musiktheorie und -didaktik wird oft auf die so genannte Solfège-Tonlehre zurück gegriffen. Hierbei werden den einzelnen Noten bestimmte Silben zugeordnet (traditionellerweise *do, re, mi, fa, so(l), la* und *ti*). Oft wird Musik auch (teilweise) mit diesen Silben notiert. 5 min



Beispiele für Musikstücke in Solfège:

1. `sol sol la sol do ti sol sol la sol re do sol sol sol mi do ti la fa fa mi do re do`
2. `mi re do re mi mi mi re re re mi so so mi re do re mi mi mi mi re re mi re do`
3. `so la so fa mi fa so re mi fa mi fa so so la so fa mi fa so re so mi do`

Geben Sie einen regulären Ausdruck an, der gültige Musikstücke, also Aneinanderreihungen beliebig vieler dieser Silben, jeweils getrennt durch Whitespace, akzeptiert. Sie können davon ausgehen, dass auch die letzte Silbe von Whitespace gefolgt wird. Ihr Ausdruck soll sowohl mit der Silbe *so* als auch mit der Silbe *sol* funktionieren. Musikalische Aspekte sollen in dieser Aufgabe keine Rolle spielen.

Lösung: Hier ist eine Lösung:

```
((do|re|mi|fa|so|sol|la|ti)\s)*
```

Aufgabe 3.2 (RegEx für Polynome) 7 Pkt

7 min

Ein Polynom ist (im Rahmen dieser Aufgabe) eine Reihe von Summanden, voneinander getrennt durch Rechenzeichen, also entweder `+` oder `-`. Alle Summanden haben dabei die Form `ax^b`, wobei `a` und `b` jeweils beliebige natürliche Zahlen (also z.B. 7, 0 oder 314...) sein können. Der Faktor `a` kann allerdings auch entfallen.

Beispiele für Polynome:

1. `5x^2+3x^1+0x^0`
2. `x^4-x^3+2x^4`
3. `x^2`

Geben Sie einen regulären Ausdruck für Polynome wie oben beschrieben an. Beachten Sie dabei, dass in einem gültigen Polynom mindestens ein Summand existieren muss und auf den letzten Summand kein Rechenzeichen folgt.

Lösung: Hier ist eine Lösung:

```
(\d*x\^\d+[+-])*(\d*x\^\d+)
```

4 Programmieren in Python

Aufgabe 4.1 (Ermittlung der Lebensspanne) 10 Pkt
10 min

Gegeben sind drei Dictionaries. Das erste, names, enthält Strings als Werte, die Namen von historischen Personen entsprechen. Die anderen beiden, births und deaths, enthalten Integers als Werte, die Geburts- bzw. Todesjahren entsprechen.

Die Schlüssel in allen drei Dictionaries sind Integers, sodass Name einer Person und deren Geburts- bzw. Todesjahre alle den gleichen Schlüssel im jeweiligen Dictionary haben:

```
names = {
    1 : "Sophie Germain",
    17 : "Emmy Noether",
    8 : "Evariste Galois",
    10 : "Leonhard Euler"
    # ...
}
```

```
births = {
    8 : 1811,
    1 : 1776,
    17 : 1882,
    10 : 1707
    # ...
}
```

```
deaths = {
    17 : 1935,
    1 : 1831,
    8 : 1832,
    10 : 1783
    # ...
```

Schreiben Sie ein python-Programm, das ermittelt, welche Person in diesem Datenset am jüngsten gestorben ist. Es soll diese Information und das Alter, in dem die Person verstarb ausgeben.

Lösung: Hier ist eine mögliche Lösung:

```
# Set variables to amounts outside of actual range
record_id = 0
record_age = 9999

# For every known person, do the following:
for idnr in names:

    # Calculate (approximate) age on death:
    age = deaths[idnr] - births[idnr]

    # Find out if this is a new record, and if so...
    if age < record_age:

        # Set the new records
        record_id = idnr
        record_age = age

# Print the records
print(names[record_id], "died the youngest at roughly", record_age, "years old.")
```

Aufgabe 4.2 (Ping-Pong Webserver) 12 Pkt
12 min

Geben Sie den python-Quellcode für folgende drei Routen eines `bottle`-Webservers an. Dieser Webserver soll auf einen HTML GET-Request mit dem Pfad ...

1. . . . /ping ein HTML-Fragment zurück geben, das nur einen Link auf den Pfad /pong enthält.

2. . . . /pong ein HTML-Fragment zurück geben, das einen HTML-Link auf den Pfad /ping enthält, aber darüber hinaus an Log-Datei `zugriffe.log` eine Zeile mit der aktuellen Uhrzeit anhängt. Außerdem soll auf der Seite angezeigt werden, wie oft dieser Prozess schon durchgeführt wurde (diese Information ist aus der Log-Datei zu ermitteln, benutzen Sie dafür keine Variable außerhalb der Route).

3. /reset abermals ein HTML-Fragment mit einem Link zurück auf den Pfad /ping zurück geben, aber außerdem die Log-Datei löschen und den Zähler damit auf 0 zurück setzen.

Dabei können Sie auf eine bereits implementierte Funktion uhrzeit() zurück greifen, die die aktuelle Uhrzeit als String zurück gibt. Vermeiden Sie dabei auch eventuelle Fehler die bei wiederholtem Aufruf der /reset-Route (wie zum Beispiel beim Löschen einer Datei, die nicht existiert) vorkommen könnten.

Hinweis: Zur Erinnerung: die os Bibliothek bietet die Methoden os.path.isfile(path) zum Überprüfen der Existenz einer Datei und os.remove(path) zum Löschen einer Datei an.

Lösung: Hier ist eine mögliche Lösung:

```
from bottle import route, run
from datetime import datetime
import os

# Muss nicht in der Klausur angegeben werden
def uhrzeit():
    return datetime.now().strftime("%Y-%m-%d %H:%M:%S")

# This route is always just a link to /pong
@route('/ping')
def ping():
    return "<a href=\"./pong\">Ping!</a>"

# This route links to /ping, but also appends
# a line to the file named 'zugriffe.log' and displays
# the number of lines after the link.
@route('/pong')
def pong():
    # Open file, append one line, close file.
    file = open("zugriffe.log","a")
    file.write(uhrzeit() + "\n")
    file.close()

    # Open file, read number of lines, close file.
    again = open("zugriffe.log")
    count = len(again.read().splitlines())
```

```
    again.close()

    return "<a href=\"./ping\">Pong!</a> " + str(count)

# This route deletes the file named 'zugriffe.log' of it exists.
@route('/reset')
def reset():
    if os.path.isfile('zugriffe.log'):
        os.remove('zugriffe.log')

    return "Reset successfull! <a href=\"./ping\">Ping!</a>"

# Bitte die Portnummer auf einen anderen Wert
# setzen, um sich beim Testen nicht gegenseitig
# in die Quere zu kommen.
run(host='0.0.0.0', port=32444)
```