Name:

Geburtsdatum:

Matrikelnummer:

Nachklausur
Informatische Werkzeuge in den Geistes- und Sozialwissenschaften 1

18. Juni 2020

	Nur zur Korrektur, bitte freilassen!											
Aufgabe	1.1	1.2	1.3	2.1	2.2	3.1	3.2	4.1	4.2	4.3	Summe	Note
Möglich	4	4	5	5	8	6	4	5	9	10	60	
Erreicht												

Klausurnote: Bonuspunkte: Endnote:

The „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.

1 Grundlagen

Aufgabe 1.1 In digitalen Dokumenten, was ist der Unterschied zwischen einem *Plaintext*-Format und einem *Markup*-Format? 4 Pkt 4 min

Lösung:Plaintext bedeutet, dass wirklich nur der Text selbst angegeben wird. Alle Zeichen in diesem Format werden auch als Teil des Textes angesehen.

Bei einem Markup-Format werden bestimmte Kontrollsequenzen als Anweisungen zur Formatierung des eigentlichen Textes interpretiert.

Aufgabe 1.2 4 Pkt

Was ist in der Programmiersprache python der Unterschied zwischen einer *Liste* und einem *Dictionary*? 4 min

Lösung:Sowohl Listen als auch Dictionaries sind Container für Daten. Eine Liste ist jedoch *geordnet*, also mit einer bestimmten Reihenfolge versehen, was auf Dictionaries nicht zutrifft. Auf der anderen Seite speichern Dictionaries immer Schlüssel/Wert-Paare (key/value-pairs), nicht nur die Werte selbst.

Aufgabe 1.3 Was ist ein *Dateipfad*? Wie unterscheidet sich dieser von einem *Dateinamen*? Was bedeuten in diesem Kontext *relativ* und *absolut*? 5 Pkt 5 min

Lösung:Ein Dateipfad ist eine Zeichenfolge die eine Datei oder ein Verzeichnis in einem Dateisystem beschreibt. Ein Dateiname ist teil eines Dateipfades, enthält aber keine Informationen darüber, in welchen Verzeichnissen (oder auf welchem Laufwerk) sich die Datei befindet.

Ein absoluter Pfad beschreibt den vollständigen Pfad vom Ursprung des Dateisystems und ist eindeutig. Viele Betriebssysteme erlauben allerdings auch, relative Dateipfade zu einem anderen Dateipfad (wie dem aktuellen Verzeichnis) zu benutzen, bei dem Teile des Pfades ausgelassen werden können und vom Betriebssystem aus dem Kontext ergänzt werden.

2 Reguläre Ausdrücke

Aufgabe 2.1 (RegEx für Nummernschilder)

Autokennzeichen (Nummernschilder) in Deutschland bestehen aus einem "Unterscheidungszeichen" (ein bis drei Großbuchstaben für den Ort, an dem das Fahrzeug registriert wurde) und einer "Kennummer" (ein bis zwei Großbuchstaben und ein bis vier Ziffern *ohne* führende Null). 5 Pkt 5 min

Unterscheidungszeichen und Kennnummer sowie Buchstaben der Kennummer und Ziffern der Kennummer werden entweder mit einem Doppelpunkt, einem Bindestrich oder einem Leerzeichen getrennt.

Geben Sie einen regulären Ausdruck an, der Autokennzeichen nach dem obigen Muster matcht. Es ist dabei keine Voraussetzung, dass alle Trennzeichen in einem Autokennzeichen gleich sind.

Beispiele: “KA-PA-777”, “BI KJ 4990”, “LIP Q 9” und “B:AN:1337”

Lösung:Hier ist eine Lösung:

```
[A-Z]{1,3}[ :-][A-Z]{1,2}[ :-][1-9][0-9]{0,3}
```

Aufgabe 2.2 (RegEx mit Konsonant Vokal und Klammerung)

8 Pkt
8 min

Geben Sie einen regulären Ausdruck an, der eine ununterbrochene Aneinanderreihung beliebiger Länge von Ausdrücken matcht, die jeweils mit einer öffnenden Klammer (also “(”,“{”,“[” oder “<”) beginnen, gefolgt von genau einem Konsonanten, genau einem Vokal und genau einer Ziffer (in dieser Reihenfolge) und mit einer schließenden Klammer (“)”,“}”,“]” oder “>”) beendet werden.

Es ist keine Voraussetzung, dass die öffnende Klammer zur schließenden Klammer passen muss. Ebenfalls gibt es keine Vorgabe dafür, ob eventuelle Gruppierungen “capturing” oder “non-capturing” sein sollen.

Beispiele: `"(no2)","[na0]","[zi5}","[ne5>[re0]"`

Lösung:Ein solcher Ausdruck wäre zum Beispiel:

```
(?:[\[({<][bcdfghjklmnpqrstvwxyz][aeiou]\d[\])}>])*
```

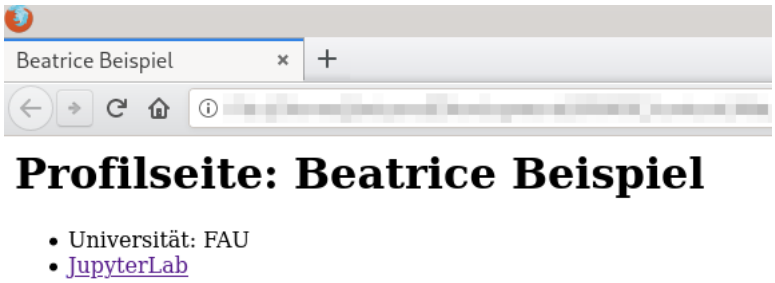
Wichtig ist, dass ein Paar eckiger Klammern escapt werden muss, damit die tatsächlichen eckigen Klammern gematcht werden können.

3 HTML und CSS

Aufgabe 3.1 (HTML-Seite mit Textfeld)

6 Pkt
6 min

Schreiben Sie eine gültige HTML-Datei, die in einem Browser etwa so dargestellt wird wie in folgendem Bild zu sehen:



Die genaue Art der Überschrift können Sie frei wählen. Der Hyperlink im zweiten Listenelement soll auf `http://jupyter.kwarc.info/` weiterleiten.

Lösung:Hier ist die offensichtlichste Lösung. Man beachte das <title> Element. Dies ist im Bild zu sehen und darf nicht vergessen werden. Hier ist eine von vielen Möglichkeiten.

```
<html>
  <head>
    <title>Beatrice Beispiel</title>
  </head>
  <body>
        <h1>Profilseite: Beatrice Beispiel</h1>

        <ul>
        <li>Universit&auml;t: FAU</li>
        <li><a href="http://jupyter.kwarc.info">JupyterLab</a></li>
        </ul>

  </body>
</html>
```

Aufgabe 3.2 Geben Sie für jede der folgenden Anforderungen eine valide CSS-Regel an: 4 Pkt

4 min

- Geben Sie allen `<h1>`-Elementen eine blaue Schriftfarbe.
- Zentrieren Sie alle `<p>`-Elemente mittig auf der Seite.
- Geben Sie allen `<img>`-Elementen einen Margin von 20px in alle Richtungen.
- Setzen Sie die Schriftgröße aller Hyperlinks auf 25pt.

Lösung:Korrekte CSS-Regeln sind:

- h1 { color : blue; }
- p { text−align: center; }
- img { margin: 20px 20px 20px 20px; }
- a { font−size: 25pt; }

---

4 Programmieren in Python

Aufgabe 4.1 (Python-Verständnisfrage)

Was gibt das Programm in Abbildung 1 aus, wenn es ausgeführt wird? 5 Pkt

---

Lösung:Es wird "ACCESS DENIED" ausgegeben. Das richtige Passwort ist zwar in der List vorhanden, es werden allerdings nur die ersten 3 Einträge überprüft. Das korrekte Passwort ist erst der vierte Eintrag. 5 min

---

Aufgabe 4.2 (Python-Programmieraufgabe File-IO)

Schreiben Sie eine python-Funktion reverseLines, die einen Dateipfad als Argument entgegen nimmt, diese Datei mit open(filename, mode) öffnet, zeilenweise ausliest und in einer zweiten Datei namens `reversed.txt` alle Zeilen aus der ersten Datei in umgekehrter Reihenfolge abspeichert. 9 Pkt

9 min

Lösung:Eine Lösung ist diese hier:

```python
def isCorrectPassword(pw):
    return pw == "friend"

inputs = ["admin", "12345", "password", "friend", "swordfish"]
counter = 0
passed = False

while counter < 3:
    passed = isCorrectPassword(inputs[counter])
    if passed:
        break
    else:
        counter = counter + 1

if passed:
    print("ACCESS GRANTED!")
else:
    print("ACCESS DENIED!")
```

Abbildung 1: Was macht dieses Programm?

```python
def reverseLines(filename):
    # Open file for reading.
    infile = open(filename, "r")

    # Start with empty string.
    rev = ""

    # Read line-by-line and accumulate reversed.
    for line in infile:
        rev = line + rev

    # Write reversed string to target file.
    outfile = open("reversed.txt", "w+")
    outfile.write(rev)

    # Close files.
    infile.close()
    outfile.close()
```

Aufgabe 4.3 (Python-Programmieraufgabe RegEx)

10 Pkt
10 min

Schreiben Sie eine python-Funktion deleteByRegex, die einen regulären Ausdruck (also einen String) als Argument entgegen nimmt und alle Dateien im aktuellen Verzeichnis (`"."`) löscht, deren Dateiname von dem regulären Ausdruck gematcht wird.

So sollte zum Beispiel nach dem Aufruf von deleteByRegex("\d+\.txt") eine Datei namens `31415.txt` gelöscht werden, eine Datei namens `survives.txt` allerdings nicht.

Hinweis: Ihnen könnten hierzu die Bibliotheksfunktionen os.listdir(directory), os.remove(filename) und re.fullmatch(regex,string) nützlich erscheinen.

Denken Sie auch an alle benötigten **import**-Statements.

Lösung:Eine Lösung ist diese hier:

```
import os
import re

# This has extra helpful print statments that were not required.
def deleteByRegex(regex):
    # For all files in current directory...
    for candidate in os.listdir("."):
        # ... check if filename matches regex.
        if re.fullmatch(regex,candidate):
            # If so delete.
            os.remove(candidate)
            print(f"File {candidate} matches. Deleted.")
        # Otherwise ignore.
        else:
            print(f"File {candidate} does not match. Ignoring.")
```
