Name:

Geburtsdatum:

Matrikelnummer:

Klausur
Informatische Werkzeuge in den Geistes- und Sozialwissenschaften 1

07. Februar 2018

	Nur zur Korrektur, bitte freilassen!												
Aufgabe	1.1	1.2	1.3	1.4	2.1	2.2	2.3	3.1	4.1	4.2	4.3	Summe	Note
Möglich	2	2	2	2	5	5	7	10	3	8	9	55	
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 2 Pkt

Was ist in der Programmiersprache python der Unterschied zwischen einer *Liste* und einem *Dictionary*? 2 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 1.2 Welche Vorteile hat das Kapseln von Programmcode in einzelne Funktionen? 2 Pkt

Lösung:Das Kapseln von Programmcode in separate Funktionen erhöht die Übersichtlichkeit des Programms und erlaubt, einmal geschriebenen Code wiederzuverwenden und mit anderen Startwerten aufzurufen. Dies reduziert auch die Fehleranfälligkeit des Programms. 2 min

Aufgabe 1.3 Was ist der Unterschied zwischen einer URI und einer URL? 2 Pkt

Lösung:Die Abkürzung URI steht für *Uniform Resource Identifier*, hier geht es also um die Identifizierung von Ressourcen, anders als bei URLs (*Uniform Resource Locator*), wo die Lokalisierung / Ansteuerung im Vordergrund steht. 2 min

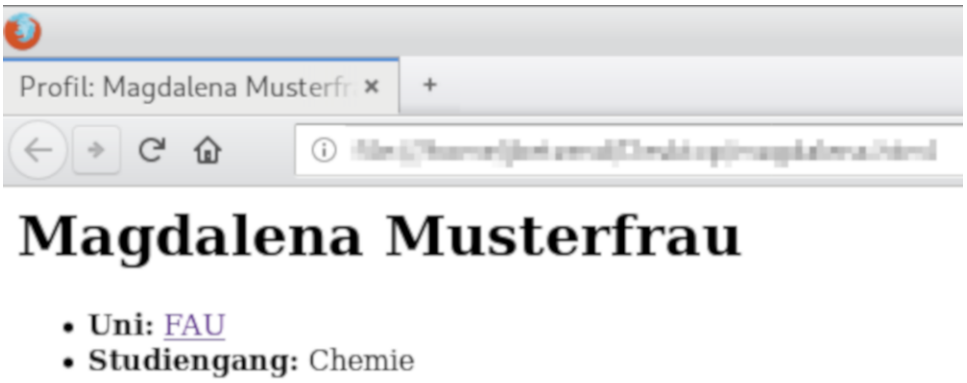
URIs sind eine Obermenge von URLs. Jede URL ist auch eine URI aber nicht jede URI ist eine URL. Alle Adressen von Websites sind URLs (und URIs), ISBN-Nummern sind jedoch nur URIs, keine URLs.

Aufgabe 1.4 Wie spielen HTML und CSS zusammen? 2 Pkt

Lösung:Wir verwenden HTML um die Struktur und den Inhalt einer Website anzugeben. CSS wird benutzt, um das Design zu verändern. 2 min

2 HTML und CSS

Aufgabe 2.1 Schreiben Sie eine gültige HTML-Datei, die in einem Browser etwa so dargestellt wird wie in folgendem Bild zu sehen: 5 Pkt 5 min



Die genaue Art der Überschrift können Sie frei wählen. Das Wort „FAU“ soll ein Link sein, der auf folgende Seite führt: `https://www.fau.de`

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

```
<html>
  <head>
    <title>Profil: Magdalena Musterfrau</title>
  </head>
  <body>
    <h1>Magdalena Musterfrau</h1>
    <ul>
      <li><b>Uni:</b> <a href="https://www.fau.de">FAU</a></li>
      <li><b>Studiengang:</b> Chemie</li>
    </ul>
  </body>
</html>
```

Aufgabe 2.2 Geben Sie für jede der folgenden Anforderungen eine valide CSS-Regel an: 5 Pkt 5 min

- Zentrieren Sie Überschriften (<h1>) horizontal.
- Geben Sie dem Element mit der ID "blue" eine blaue Schriftfarbe (z.B. #0000FF).
- Geben Sie Paragraphen (<p>) der Klasse hasBorder einen Rand der Breite 1px.

Lösung:Korrekte CSS-Regeln sind:

- h1 {text−align: center;}
- #blue {color:#0000FF;}
- p.hasBorder {border−style: solid; border−width: 1px;} oder p.hasBorder {border: solid 1px}

Aufgabe 2.3 Mit dem CSS-Selektor ol p sprechen Sie alle Paragraphen (<p>) innerhalb von numerierten Listen (<ol>) an (geschachteltes CSS). In Abbildung 1 sehen Sie ein Beispiel. 7 Pkt 7 min

Nehmen wir nun an, wir wollen in einer Tabelle wie in Abbildung 2 arbeiten. Es kommen sowohl Zeilen mit der Klasse even als auch mit der Klasse odd vor. Nur Zeilen der Klasse even sollen selektiert werden.

Wie müsste der Selektor für alle Zeilen der Klasse even innerhalb einer Tabelle mit der ID importantTable aussehen?

Lösung:Der Selektor allein (ohne dazugehörige key/value pairs) ist:

table#importantTable tr.even oder einfacher (den Wert eines id Attributs gibt es immer nur einmal) #importantTable tr.even

```
...
  <ol>
    <li><p>Dies ist ein Paragraph in der Liste (wird selektiert)</p></li>
    <li>Dies ist zwar in der Liste, aber kein Paragraph (wird nicht selektiert).</li>
    ...
  </ol>

<p>Paragraph ausserhalb der Liste (wird nicht selektiert)</p>
...
```

Abbildung 1:

```
...
<table id="importantTable">
  ...
  <tr class="odd">
    <td>Philipp</td>
    <td>Kurth</td>
    <td>philipp.kurth@fau.de</td>
  </tr>

  <tr class="even">
    <td>Jonas</td>
    <td>Betzendahl</td>
    <td>jonas.betzendahl@fau.de</td>
  </tr>
  ...
</table>
...
```

Abbildung 2:

3 Reguläre Ausdrücke

Aufgabe 3.1 Eine Telefonnummer besteht (im Kontext dieser Aufgabe) aus den folgenden Bestandteilen: 10 Pkt 10 min

- Einer Vorwahl. Diese beginnt entweder mit +, gefolgt von einem zweistelligen länderspezifischen Code (LC), oder mit 0, und endet mit weiteren drei Ziffern.
  Beispiele: 0174, +49172, +32151, ...
- Einem optionalen Trennzeichen zwischen Vorwahl und Basisnummer.
  Erlaubte Trennzeichen sind Schrägstrich (/), Leerzeichen oder kein Trennzeichen.
- Einer genau siebenstelligen Basisnummer.
  Beispiele: 1234567, 5678910, ...

Es darf maximal ein Trennzeichen vorkommen und auch nur zwischen Vorwahl und Basisnummer. Erstellen Sie einen regulären Ausdruck (Regular Expression), der alle nach diesem Muster aufgebauten Telefonnummern matcht.

Beispiele für vollständige Nummern sind:

	Beginnt mit 0	Beginnt mit +LC
Trennzeichen /	0174/1234567	+49174/1234567
Trennzeichen Leerzeichen	0172 5678910	+49174 5678910
Kein Trennzeichen	01741996767	+491744345456

Lösung:Eine korrekte Lösung ist der reguläre Ausdruck:

```
"(?:0|\+\d\d)\d{3}[ /]?\d{7}"
```

4 Programmieren in Python

Aufgabe 4.1 Was gibt das Programm in Abbildung 3 aus, wenn es ausgeführt wird? 3 Pkt 3 min

```
a = 3 + 2 * 3
if a > 10:
    print("a > 10")
else:
    print("a <= 10")
```

Abbildung 3:

Lösung:Es wird "a <= 10" ausgegeben. a ist 9, da python Punkt-vor-Strichrechnung beachtet.

Aufgabe 4.2 Schreiben Sie eine python-Funktion favoriteColors, die eine Liste von Dictionaries übergeben bekommt. Jedes Dictionary enthält Informationen über eine Person. Jede Person hat einen Namen (Key: "name"), aber nur manche Personen haben eine Lieblingsfarbe (Key: "favColor"). 8 Pkt 8 min

Anmerkung: Sowohl die Keys, als auch die Values sind Strings.

Ihre Funktion soll für jede Person, für die eine Lieblingsfarbe angegeben ist, den Namen der Person und die jeweilige Farbe printen. Am Ende soll sie die Anzahl an Personen zurückgeben, für die eine Lieblingsfarbe angegeben ist.

Lösung:One solution is the following:

```
def favoriteColors(dicts):
    # Counting variable
    counter = 0

    # Iterate over all given dictionaries.
    for d in dicts:
        if "favColor" in d:
```

```
            counter = counter + 1

            # Example. Doesn't need to be exactly this.
            print("The favorite color of", d["name"], "is", d["favColor"])

    return counter
```

Aufgabe 4.3 Schreiben Sie eine python-Funktion generateList, die ein Dictionary übergeben bekommt. Ihre Funktion soll eine HTML-Seite mit einer ungeordneten Liste erstellen, in der jeder Listeneintrag einem Eintrag im Dictionary entspricht. 9 Pkt 9 min

Anmerkung: Sowohl die Keys, als auch die Values sind Strings.

Das geforderte Format für die ungeordnete Liste ist:

- Key1: Value1
- Key2: Value2
- Key3: Value3
- ...

Ihnen steht eine Hilfsmethode skeleton(body) zur Verfügung, die einen String erwartet und diesen in ein HTML-Grundgerüst einbettet. Sie müssen dieser Funktion nur den Body der Webseite übergeben, also alles *innerhalb* der <body>-Tags. Die Funktion skeleton gibt dann den zusammengesetzten String zurück.

Ihre Funktion soll am Ende den fertigen HTML-Quelltext zurückgeben.

Lösung:One solution is the following:

```
def generateList(dictionary):
    # Opening unordered list
    mystring = "<ul>"

    # Iterating over all keys of a dictionary
    for item in dictionary:
        # This is possible because both keys and values are strings.
        elem = item + ": " + dictionary[item]
        mystring += " <li>" + elem + "</li>"

    # closing list
    mystring += "</ul>

    return skeletong(mystring)
```
