Name:

Geburtsdatum:

Matrikelnummer:

Nachklausur
Informatische Werkzeuge in den Geistes- und Sozialwissenschaften 1

17. April 2019

	Nur zur Korrektur, bitte freilassen!												
Aufgabe	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2	4.1	4.2	4.3	Summe	Note
Möglich	2	2	4	6	8	6	4	4	3	8	8	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 In digitalen Dokumenten, was ist der Unterschied zwischen einem *Plaintext*-Format und einem *Markup*-Format? 2 Pkt 2 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 Beschreiben Sie kurz die Rollen von „Servern“ und „Clients“ im World Wide Web und wie diese typischerweise miteinander interagieren. 2 Pkt 2 min

Lösung:Das World Wide Web basiert auf einer Server-Client Architektur in der Server Dateien und Dokumente (z.B. Videos oder Webseiten) zur Verfügung stellen und Clients (z.B. Browser) diese dem Nutzer oder der Nutzerin anzeigen. Normalerweise kommunizieren Server und Clients dabei über das HTTP (HyperText Transport Protocol).

Aufgabe 1.3 Nennen Sie zwei Arten von Kontrollflusselementen und geben Sie jeweils ein Beispiel in der Programmiersprache python. 4 Pkt 4 min

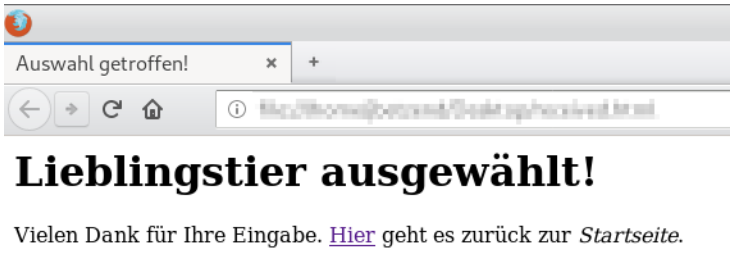
Lösung:Kontrollflusselemente:

- Bedingte Verzweigung (**if**-statements)
- Schleifen (**for**-loops)

---

2 HTML und CSS

Aufgabe 2.1 Schreiben Sie eine gültige HTML-Datei, die in einem Browser wie folgt dargestellt wird: 6 Pkt 6 min



Die genaue Art der Überschrift können Sie frei wählen. Das Wort „Hier“ soll ein Link sein, der auf folgende Seite führt: `https://tiergarten.nuernberg.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>Auswahl getroffen!</title>
```

```
  </head>
  <body>
    <h1>Lieblingstier ausgew&auml;hlt!</h1>
    Vielen Dank f&uuml;r Ihre Eingabe.
    <a href="https://tiergarten.nuernberg.de">Hier</a> geht es zur&uuml;ck zur
    <i>Startseite</i>.
  </body>
</html>
```

Aufgabe 2.2 Schreiben Sie eine gültige HTML-Seite, die eine passende Überschrift enthält, einen kurzen Text der nach dem Lieblingstier fragt und letztlich ein HTML-Formular, dass aus einem Dropdown-Menü und einem „Absenden“-Button besteht. 8 Pkt 8 min

Im Dropdown-Menü sollen zwei verschiedene Tierarten oder mehr auswählbar sein. Wird das Formular mit dem „Absenden“-Button abgeschickt, soll der Benutzer oder die Benutzerin auf eine Seite mit einer Empfangsbestätigung weiter geleitet werden (siehe auch Aufgabe 2.1).

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

Lösung:Hier ist eine von vielen Möglichkeiten.

```
<html>
  <head>
    <title>Hier Ihr Lieblingstier!</title>
  </head>
  <body>
        <h1>Lieblingstier ausw&auml;hlen!</h1>
        Unten finden Sie eine Selektion von hervorragenden Tieren.<br>
        Bitte w&auml;hlen Sie ihr liebstes aus und dr&uuml;cken Sie auf
        "Absenden!".<br><br>

        <form action="bestaetigung.html">
                <select name="animals">
                        <option value="sloth">Faultier</option>
                        <option value="shark">Hai</option>
                        <option value="pigeon">Taube</option>
                        <option value="mantisshrimp">Fangschreckenkrebs</option>
                </select>
                <input type="submit" value="Absenden"/>
        </form>
  </body>
</html>
```

Aufgabe 2.3 Geben Sie für jede der folgenden Anforderungen eine valide CSS-Regel an: 6 Pkt 6 min

- Geben Sie allen `<p>` und `<h1>`-Elementen mit einer(!) Regel eine rote Schriftfarbe.

- Geben Sie dem Element mit der ID "imblue" eine blaue Hintergrundfarbe.
- Geben Sie allen `<div>`-Elementen einen Margin von 15px in alle Richtungen.
- Zentrieren Sie allen Text in Elementem der Klasse `centered` horizontal.

Lösung:Korrekte CSS-Regeln sind:

- p, h1 { color : red; }
- #imblue { background−color : blue; }
- div { margin: 15px 15px 15px 15px;}
- .centered { text−align : center; }

---

3 Reguläre Ausdrücke

Aufgabe 3.1 In dieser Aufgabe sollen Sie einen regulären Ausdruck (Regex) für korrekte International Bank Account Numbers (IBANs) angeben. Die genaue Form von IBANs ist abhängig vom Land, wir suchen nach einem regulären Ausdruck für korrekte IBANS aus allen DACH-Ländern (d.h. Deutschland, Österreich und Schweiz). 4 Pkt 4 min

Deutsche IBANs sind die Buchstaben “DE” gefolgt von 20 Ziffern, Österreichische IBANs sind die Buchstaben “AT” gefolgt von 18 Ziffern und Schweizer IBANs sind die Buchstaben “CH” gefolgt von 19 Ziffern.

Beispiele:

- DE89370400440532013000 (Deutschland)
- AT611904300234573201 (Österreich)
- CH9300762011623852957 (Schweiz)

Lösung:Eine korrekte Lösung ist der reguläre Ausdruck:

```
"DE\d{20}|AT\d{18}|CH\d{19}"
```

Aufgabe 3.2 Alle IBANs haben ein „elektronisches Format“ (ohne Leerstellen, z.B. DE89370400440532013000) und ein „Druckformat“ (mit Leerstelle nach jeweils vier Zeichen, z.B. DE89 3704 0044 0532 0130 00). 4 Pkt 4 min

Geben Sie einen regulären Ausdruck an, der für deutsche IBANs (!) sowohl gegen das elektronische Format als auch gegen das Druckformat korrekt matcht.

Lösung:Eine korrekte Lösung ist der reguläre Ausdruck:

```
"DE\d{20}|DE\d\d \d{4} \d{4} \d{4} \d{4} \d\d"
```

---

4 Programmieren in Python

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

```
foo = "IWGS"
bar = foo + foo + foo
baz = bar + bar + bar

print(len(baz))
```

Lösung:Es wird "36" ausgegeben.

Aufgabe 4.2 Schreiben Sie eine python-Funktion dict2list, die zwei Argumente (ein Dictionary und einen Boolschen Wert) übergeben bekommt und eine Liste zurück gibt. Wird der Funktion True als Boolscher Wert übergeben, so soll die Liste alle *Keys* des Dictionarys enthalten. Wird stattdessen False übergeben, soll die Liste alle *Values* des Dictionarys enthalten. 8 Pkt
8 min

Beispiel: Betrachten Sie den folgenden Fall:

```
# Source: https://en.wikipedia.org/wiki/Mae_Jemison
exampleDictionary = {
  "name" : "Mae Carol Jemison",
  "birthday" : "1956-10-17",
  "gender" : "female",
  "occupation" : "astronaut",
  "mission" : "STS-47"
}
```

Wenn Sie die Funktion mit dict2list(exampleDictionary,True) aufrufen, soll die Liste ['name', 'birthday', 'gender', 'occupation', 'mission'] zurück gegeben werden.

Bei dem Aufruf dict2list(exampleDictionary,False) stattdessen hingegen die Liste ['Mae Carol Jemison', '1956-10-17', 'female', 'astronaut', 'STS-47']

Lösung:One solution is the following:

```
def dict2list(inp, keyflag):
  # All cases start with an empty list.
  ret = []

  # If the second argument is True.
  if keyflag:
    # Append all keys to ret.
    for key in inp:
      ret.append(key)
  # If the second argument is False.
  else:
    # Apeend all values to ret.
    for key in inp:
      ret.append(inp[key])

  # Return accumulated list
```

```
  return ret

print(dict2list(exampleDictionary,True))
print(dict2list(exampleDictionary,False))
```

Aufgabe 4.3 8 Pkt

8 min

Schreiben Sie ein Programm in python das den Nutzer oder die Nutzerin nach einem Passwort fragt. Ihnen steht dabei eine validate-Funktion zur Verfügung, die ein Argument (String) nimmt und entweder True oder False zurück gibt, je nachdem ob es sich um das korrekte Passwort handelt oder nicht. Diese validate-Funktion müssen Sie *nicht* implementieren.

Gibt der Nutzer oder die Nutzerin das korrekte Passwort ein, so soll „Zugang gewährt!“ ausgegeben werden. Ist das eingegebene Passwort falsch, soll Ihr Programm erneut fragen, jedoch maximal zehn Mal. Ist das Passwort auch bei der zehnten Eingabe nicht korrekt, soll das Programm „Zugang verweigert!“ ausgeben.

Lösung:Eine mögliche Lösung ist die folgende:

```
# Zu Beginn sind 10 Versuche uebrig und kein Zugang.
count = 10
access = False

# Solange nicht entweder alle Versuche aufgebraucht
# sind oder schon das richtige PW eingegeben wurde:
while ((count > 0) and (not access)):
  eingabe = input("Bitte Passwort eingeben: ")
  if validate(eingabe):
    access = True
  else:
    count -= 1

# Satus ausgeben
if access:
  print("Zugang gewaehrt!")
else:
  print("Zugang verweigert!")
```
