Probeklausur
Informatische Werkzeuge in den Geistes- und Sozialwissenschaften I

11. Januar 2020

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 & Begrifflichkeiten

Aufgabe 1.1 Erklären Sie die Begriffe *Bit*, *Byte*, *Kilobyte* und *Mebibyte*. Erläuteren Sie außerdem kurz die Beziehungen, die diese Begriffe untereinander haben. — 5 Pkt, 5 min

Lösung:Ein Bit ist die fundamentale Einheit für Information. Bits können entweder den Wert 1 oder 0 annehmen.

Ein Byte sind 8 Bit. Ein Kilobyte sind (Nach SI-Präfix-Nomenklatur) 1000 Byte. Manchmal werden auch 1024 Byte als ein “Kilobyte” bezeichnet. Um hier Präzision in der Sprache zu erlauben wurde der Begriff "Kibibyte"für 1024 Bytes eingeführt. Ein Mebibyte sind 1024 Kibibyte.

Aufgabe 1.2 In digitalen Dokumenten, was ist der Unterschied zwischen einem *Plaintext*-Format und einem *Markup*-Format? — 5 Pkt, 5 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 1.3 Erläutern Sie kurz, was es mit der python-Funktion input() auf sich hat. Ist es möglich, Argumente an diese Funktion zu übergeben? Wenn ja: was passiert mit — 5 Pkt, 5 min

diesen Argumenten? Wenn nein: was passiert, wenn jemand es trotzdem tut? Worauf ist zu achten, wenn Zahlen eingelesen werden sollen?

Lösung:Mit Hilfe der input()-Funktion ist es möglich, zur Laufzeit des Programms eine Eingabe von den Benutzer'Innen abzufragen (zum Beispiel ob sie ein Haustier haben). Die Eingabe geschieht über das Kommandozeileninterface und wird als String zurück gegeben.

Es ist möglich, Strings als Argument an input() zu übergeben, diese werden dann vor dem Eingabeprompt geprintet.

2 Digitale Dokumente

Aufgabe 2.1 (HTML Ping Pong)

Geben Sie den Quellcode für zwei HTML-Seiten (`ping.html` und `pong.html`) an. 15 Pkt

Beide Seiten sollten eine Überschrift, einen kurzen Text und einen Knopf (nicht einfach nur einen Link) enthalten. Beim Klicken des Knopfes soll auf die jeweils andere Seite weiter geleitet werden, so dass durch wiederholtes Klicken zwischen den Seiten hin und her gesprungen werden kann. 15 min

Lösung:Der Trick ist, HTML Formulare zu verwenden, und die jeweils andere Seite im action Attribut als Ziel anzugeben.

```
<html>
  <head>
    <title>Ping!</title>
  </head>
  <body>
    <h1>PING!</h1>
    <p>The next logical step would be to click on the Ping! button...</p>
    <form action="pong.html">
      <input type="submit" value="Ping!"/>
    </form>
  </body>
</html>
```

```
<html>
  <head>
    <title>Pong!</title>
  </head>
  <body>
    <h1>PONG!</h1>
    <p>The next logical step would be to click on the Pong! button...</p>
    <form action="ping.html">
      <input type="submit" value="Pong!"/>
    </form>
  </body>
</html>
```

3 Reguläre Ausdrücke

Aufgabe 3.1 (Reguläre Ausdrücke I)

Geben Sie einen regulären Ausdruck an, der *genau* folgende Strings komplett matcht (also alle diese und keine anderen): 5 Pkt 5 min

cat4, hat4, mat4, bat4, apollo

Lösung:Eine mögliche Lösung wäre der reguläre Ausdruck `^[chmb]at4|apollo$`.

Da wir die RegEx-Abkürzungen für "Anfang des Strings" und "Ende des Strings" (`^` und `$`)verwenden, gehen wir sicher, dass keine anderen Strings außer der explizit aufgeführten (wie z.B. My hat4sunshine is great!) matchen. Der |-Operator erlaubt uns, auch auf den String apollo zu matchen, der das sonst gemeinsame Muster bricht.

Aufgabe 3.2 (Reguläre Ausdrücke für Adressen)

Geben Sie einen regulären Ausdruck an, der gegen Adresszeilen der Form „91058 Erlangen" oder „33649 Bielefeld" matcht. 5 Pkt 5 min

Diese bestehen immer aus einer fünfstelligen PLZ, einem Leerzeichen und einem Ortsnamen. Ortsnamen sind für den Rahmen dieser Aufgabe zwischen einem und 20 Zeichen lang, beginnen mit einem Großbuchstaben (inkl. Umlauten) gefolgt von lediglich Kleinbuchstaben (inkl. Umlauten + ß), ohne Leerstellen oder Interpunktion.

Lösung:Ein solcher Ausdruck wäre zum Beispiel:

```
\d{5}\s[A-ZÄÖÜ][a-zßäöü]{1,19}
```

Aufgabe 3.3 (Reguläre Ausdrücke für IP-Adressen)

Geben Sie einen regulären Ausdruck an, der gegen IPv4-Adressen matcht. Diese Adressen bestehen aus genau vier Blöcken, die durch Punkte voneinander getrennt sind. Jeder Block enthält zwischen einer und drei Ziffern von 0 bis 9. 5 Pkt 5 min

Beispiele:

- `127.0.0.1`
- `131.188.6.20`
- `203.000.113.000`

Hinweis: Blöcke in tatsächlichen IPv4-Adressen nehmen nur Werte zwischen 0 und 255 an, nicht jede beliebige Kombination aus drei Ziffern wie z.B. 999. Diesen Fakt ignorieren wir in dieser Aufgabe.

Lösung:Ein solcher Ausdruck wäre zum Beispiel:

```
\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}
```

Wichtig zu beachten ist, dass der Punkt zwischen den Blöcken mit einem Backslash escapt werden muss. Sonst würden auch Ausdrücke wie der folgende zugelassen, weil der . auf alle Zeichen matcht.

```
12780+0q1
```

---

4 Programmieren in Python

Aufgabe 4.1 (Listen von Listen)

Schreiben Sie eine python-Funktion longestString, die aus einer *Liste von Listen von Strings* den längsten String findet und zurück gibt. 10 Pkt 10 min

Beispiel: Angewendet auf die Liste [["sloth","elephant","cat"],["singing"],["banana","kiwi"]] sollte Ihre Funktion "elephant" zurück geben.

Diese Liste von Listen wird Ihrer Funktion als einziges Argument übergeben werden.

Sie können für diese Aufgabe annehmen, dass alle Längen der Strings in den Listen nur einmal vorkommen (d.h. es gibt keine zwei Strings, die gleich lang sind).

Lösung:Hier ist eine mögliche Lösung mit zwei verschachtelten Schleifen, die über alle Elemente aller Listen iteriert.

```
def longestString(listoflists):
    champion = ""

    # Zwei geschachtelte For-Schleifen
    for liste in listoflists:
        for element in liste:
            if len(element) > len(champion):
                champion = element

    return champion

# Code zum Testen der Funktion (nicht gefragt in der Aufgabe).
testList = [["sloth","elephant","cat"],["singing"],["banana","kiwi"]]
print(longestString(testList)) # Prints "elephant"
```

Aufgabe 4.2 Schreiben Sie ein python-Programm, das das Würfeln von 10.000 sechsseitigen Würfeln simuliert, die Augenzahlen aufaddiert und den Durschnittswert (die Summe geteilt durch die Anzahl Durchläufe) ausgibt. 5 Pkt 5 min

Ihnen steht dafür eine Funktion wuerfel() zur Verfügung, die eine zufällige Zahl zwischen 1 und 6 (inklusive) zurück gibt.

Lösung:Eine Lösung ist diese hier:

```
# Import und wurfel()-Funktion sind nicht gefragt,
# hier aber der Vollstaendigkeit halber implementiert.
import random

def wuerfel():
    return random.randint(1,6)

# Dieses Programm war gefragt:
counter = 0
for i in range(0,10000):
    counter = counter + wuerfel()

average = counter / 10000
print("Durchschnitt:",average)
```