

# Schlussprüfung Programmiertechniken in der Computerlinguistik I HS 13

Aufgabenstellung: Simon Clematide/Martin Volk

Prüfung vom 9. Januar 2014  
Institut für Computerlinguistik, Universität Zürich

Vorname \_\_\_\_\_ Matrikelnummer \_\_\_\_\_

Nachname \_\_\_\_\_

## Für Studierende der folgenden Studiengänge:

- ☐ BA - Studiengang Computerlinguistik (Phil. Fakultät)  
☐ BA - Studiengang Computerlinguistik und Sprachtechnologie (Phil. Fakultät)  
☐ Andere:

## Nur für Lizentiatsstudierende der Computerlinguistik als ein Fach aus der Phil. Fakultät:

Strasse: \_\_\_\_\_ Hauptfach: \_\_\_\_\_

PLZ/Ort: \_\_\_\_\_ E-Mail: \_\_\_\_\_

Die Prüfungsergebnisse von ECL und PCL von den Lizentiatsstudierenden werden (zusammen) per Post verschickt. *Bitte Adresse nicht vergessen!*

Aufgabe Nr.:	1	2	3	4	5	6	7	8	9	10	11	12	Summe
Punktzahl:	12	10	4	4	14	4	14	5	2	4	2	15	90
Davon erreicht:													

Note SU: \_\_\_\_\_ Note SP: \_\_\_\_\_

Endnote: \_\_\_\_\_ Bestanden: ☐ Ja ☐ Nein

*Auf jedes separate Blatt mit Lösungen den Nachnamen schreiben!*

Viel Erfolg!

## Wichtige Hinweise

Maximale Punktzahl: 90 Pro Punkt sollte ungefähr 1 Minute gebraucht werden.

Hinweise: Bitte schreiben Sie in einem knappen, aber verbalen Stil (keine Stichwortsammlungen). Bei inhaltlichen Auswahlendungen, wo einfach alles spontan hingeschrieben und Falsches wie Korrektes munter vermischt wird, behalten wir uns Abzüge vor. Erlaubtes Hilfsmittel ist die Referenzkarte zu Python. Die Antworten können **deutsch** (bevorzugt) oder **englisch** sein.

### 1. UNIX-Kommandos für Schlagzeilenkorpus (12 Punkte)

In einem Unterverzeichnis namens `2013/` befinden sich genau 365 Textdateien mit **der** Haupt-Schlagzeile einer Boulevardzeitung von jedem Tag.

Jede Datei ist nach folgendem Muster benannt: `MM-DD.txt` (MM=Monat numerisch, DD=Tag numerisch). Z.B. enthält `2013/06-01.txt` die Schlagzeile vom 1. Juni 2013.

Jede Datei hat genau eine Zeile, welche den **tokenisierten** Text der Schlagzeile enthält. Fiktives Beispiel:

`SP-Politikerin heiratet SVP-Kantonsrat .`

- 3

(a) Mit welchem Terminalbefehl (oder Befehlen) lassen sich alle Schlagzeilen des Jahres aneinandergehängt in eine Datei `2013.txt` ablegen?

.....
- 2

(b) Mit welchem Terminalbefehl (oder Befehlen) lassen sich alle Schlagzeilen vom Monat Oktober aneinandergehängt auf dem Terminal anzeigen?

.....
- 3

(c) Wie lautet der Grep-Befehl, um aus der Datei `2013.txt` alle **Bindestrich-Komposita** auf dem Terminal anzuzeigen, welche mit einem Parteikürzel ("`SP`", "`CVP`", "`SVP`", "`FDP`") beginnen. Z.B. soll "`SP-Politikerin`" oder "`FDP-Bundesrat`" einzeln auf einer Ausgabezeile erscheinen.

.....
- 3

(d) Der Befehl `tr ' ' '\n'` kann verwendet werden, um alle Leerzeichen in Newlines zu verwandeln. Beschreiben Sie knapp und exakt, was der folgende Befehl auf dem Terminal **ausgibt** (also nicht, was die einzelnen Werkzeuge machen). Erfinden Sie ein Beispiel, wie die **erste** und **letzte** Zeile der Ausgabe aussehen könnte.

`$ tr ' ' '\n' < 2013.txt | sort | uniq -c | sort -n`

.....

.....

.....

.....
- 1

(e) Wieso kann es Sinn machen, solche Aufgaben mit UNIX-Kommandos zu lösen und z.B. nicht in Python?

.....

**2. Python-Theorie (10 Punkte)** Kreuzen Sie **alle** korrekten Aussagen an. Es gibt 0.5 Punkte pro richtige Antwort, 0.5 Punkte **Abzug** pro **falsche** Antwort.

- 2** (a) Objekte
- ☐ Jedes Attribut ist ein Objekt.
  - ☐ Jedes Objekt ist ein Attribut.
  - ☐ Jede Methode ist ein Attribut.
  - ☐ Jedes Objekt ist eine Methode.
- 2** (b) Oberklassen
- ☐ Jede selbstdefinierte Klasse hat eine Oberklasse.
  - ☐ Unterklassen eines Objekts heissen Instanzen.
  - ☐ Jedes Objekt ist eine Instanz einer Klasse.
  - ☐ Die oberste Klasse heisst `object`.
- 2** (c) Zeichenketten
- ☐ Der String `"\n"` und der String `'\n'` sind die gleiche Zeichenkette.
  - ☐ Der String `"abc\n"` und `r"abc\n"` sind die gleiche Zeichenkette.
  - ☐ Die Länge der Zeichenkette `"\ndas\tART\nHaus\tNN\n"` ist grösser als 16.
  - ☐ Solange in einem Stringliteral `s` nur ASCII-Zeichen vorkommen, ist es egal, ob wir das Stringliteral als `u's'` oder `'s'` schreiben.
- 2** (d) Veränderliche Datenstrukturen
- ☐ Normale Strings (d.h. Objekte vom Typ `str`) haben im Gegensatz zu Unicode-Zeichenketten (d.h. Objekte vom Typ `unicode`) den Vorteil, dass sie veränderlich (*mutable*) sind.
  - ☐ Die Schlüssel in Dictionaries (`dict`) müssen unveränderlich (*immutable*) sein.
  - ☐ Mengen (`set`) können nur unveränderliche Datenstrukturen als Elemente haben.
  - ☐ Die Werte in Dictionaries (`dict`) müssen immer unveränderlich sein.
- 1** (e) Identität
- ☐ Zwei Objekte `o1` und `o2`, welche identisch sind (d.h. es gilt `id(o1) == id(o2)`), haben immer denselben Wert (d.h. es gilt `o1 == o2`).
  - ☐ Die beiden Variablen `a` und `b` sind identisch nach der Verarbeitung dieses Codes.  

```
a = "a"
b = a
```
- 1** (f) Generatorausdrücke und Listenkomprehensionsausdrücke
- ☐ Die Funktion `len` lässt sich auf Generatorausdrücke anwenden.
  - ☐ Die Funktion `len` lässt sich auf Listenkomprehensionsausdrücke anwenden.

**4 3. Import-Statement (4 Punkte)** Was unterscheidet die beiden Versionen des import-Statements:

```
import nltk          ### (A)
from nltk import *   ### (B)
```

.....

.....

.....

.....

.....

.....

.....

- 4. Ausdrücke (*expressions*) vs. Anweisungen (*statements*) (4 Punkte)** Was ist der Unterschied und das Verhältnis zwischen Ausdrücken und Anweisungen?

.....

.....

.....

.....

.....

.....

.....

- 5. Schleifen (14 Punkte)** Ein Hacker hat Teile des Programms zur Formatierung von textuellen Häufigkeitstabellen böswillig mit Kommentaren der Form `#X#` überschrieben. Mit Hilfe eines Beispieloutputs sollte es Ihnen möglich sein, die Kommentare mit Variablen oder Literalen zu ersetzen, damit das Programm wieder lauffähig wird.

```
import nltk

cfd = nltk.ConditionalFreqDist(
    (genre, word)
    for genre in nltk.corpus.brown.categories()
    for word in nltk.corpus.brown.words(categories=genre))

genres = ['news', 'religion', 'hobbies']
modals = ['can', 'could', 'may', 'might', 'must', 'will']

def tabulate(cfdist, words, categories):
    print '%-16s' % #A#,
    for #B# in #C#:
        print '%6s' % #B#,
    print
    for #D# in #E#:
        print '%-16s' % #D#,
        for #F# in #G#:
            print '%6d' % cfdist[#H#][#I#],
        print
```

```
tabulate(cfd, modals, genres)
```

Output des Programms:

Category	can	could	may	might	must	will
news	93	86	66	38	50	389
religion	82	59	78	12	54	71
hobbies	268	58	131	22	83	264

#A#	_____	#F#	_____
#B#	_____	#G#	_____
#C#	_____	#H#	_____
#D#	_____	#I#	_____
#E#	_____		

**4 6. Ausgabe von Ausdrücken mit Listen und anderen Sequenzen (4 Punkte)**

```
>>> names = ['Carla', 'Bruno', 'Adam']
>>> print names[-1][0]      Ausgabe: _____
>>> names.sort()
>>> print names[0]          Ausgabe: _____
>>> del names[0]
>>> print names[0]          Ausgabe: _____
>>> names.append('Daniel')
>>> print names             Ausgabe: _____
```

**7. Trunkierung von Wörtern (14 Punkte)** Trunkierung ist eine alte Technik im *Information Retrieval* (IR), um die Anzahl der Indexterme eines Dokuments klein zu halten. Bei der Trunkierung auf eine feste Anzahl Zeichen werden alle Tokens von ihrem Ende her auf eine maximale Anzahl Buchstaben gekürzt, z.B. auf 6 Buchstaben. Weiter sollen **einbuchstabige** Tokens wie im IR üblich ignoriert werden. Folgender Beispielcode illustriert dies:

```
>>> doc = ['Alle', 'Dokumente', 'dieser', 'Dokumentensammlung', ':']
>>> trunk(doc, 6)
['Alle', 'Dokume', 'dieser', 'Dokume']
```

Die Funktion `trunk(l, n)` evaluiert zur selben Liste wie `l`, wobei alle Strings auf eine maximale Länge `n` gekürzt sind. Hinweis: `"b"[:4]` evaluiert zu `"b"` und `"abc"[:2]` zu `"ab"`.

Definieren Sie nun die **Funktion** `trunk`, zuerst ohne Listenkomprehension, dann mit.

**7 (a) Ohne Listenkomprehension**

```
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

**7 (b) Mit Listenkomprehension**

```
.....
.....
.....
.....
.....
```

- 5 8. Reguläre Ersetzung (5 Punkte)** Gegeben sei eine Sammlung von HTML-Dateien mit der folgenden Struktur:

```
...
<div class="content">Bla ... bla</div>
<div class="footer">2014. All rights reserved. </div>
...
```

Definieren Sie einen regulären Such- und Ersetzungsausdruck, der einen String (`htmlstring`) mit dem Inhalt einer ganzen HTML-Datei reduziert auf den textuellen Inhalt des HTML-Tags `<div class="content">`, d.h. auf `Bla ... bla`.

`pattern = _____`

`replacement = _____`

`text = re.sub(pattern, replacement, htmlstring)`

- 2 9. Wozu ist das Regex-Flag (`?u`) nützlich in der Sprachverarbeitung? (Antwort mit einem Beispiel) (2 Punkte)**

.....

.....

.....

.....

- 4 10. Pipeline für Webseiten-Verarbeitung (4 Punkte)** Welche Schritte der Verarbeitung sind typischerweise notwendig, um eine einfache korpuslinguistische Verarbeitung einer HTML-Datei hinter einer Web-Adresse durchführen zu können (z.B. Vokabulargröße bestimmen)?

.....

.....

.....

.....

.....

.....

- 2 11. Was ist der Unterschied zwischen lokalen und globalen Namen? (2 Punkte)**

.....

.....

.....

.....

