Einführung in Python

Übung zur Vorlesung NLP
Beate Krause, Dominik Benz
22.04.2008

Ablauf der Übung

- Klärung von aktuellen Fragen zur Vorlesung
- Kurze Einführung in Python
- Einrichten einer Python-Umgebung
- Bearbeiten der Programmieraufgabe

Python ist eine...

- objektorientierte
- dynamisch getypte
- interpretierte
- interaktive
- High-Level-Programmiersprache

Zum Programmieren braucht man..

- Einen Texteditor. Komfortabel:
 - **Eclipse** (<http://www.eclipse.org/downloads/moreinfo/classic.php>)
 - **Pydev Python Plugin** (<http://pydev.sourceforge.net/>)
- den **Python-Interpreter**
 - <http://www.python.org/download/>
- für uns zusätzlich: NLTK
 - <http://nltk.sourceforge.net/>

Getting started

- Gut zum „Herumspielen“: Das Interactive Development Enviroment (IDLE)

```
1 Python 2.5.2 (r252:60911, Feb 21 2008, 13:11:45) [MSC v.1310 32 bit
2 Type "copyright", "credits" or "license()" for more information.
3 >>> print 3 + 10 * 10
4 103
5 >>> "hello,_" + "world"
6 'hello,_world'
7 >>> "spam_" * 4
8 'spam_spam_spam_spam_'
```

Variablen

- Variablennamen: **ASCII-Zeichen + Ziffern** (erstes Zeichen darf keine Ziffer sein)
- müssen **nicht deklariert** werden, werden ins Leben gerufen, sobald Wert zugewiesen wird (aber auch erst dann!)

```
1 >>> msg = 'Hi'
2 >>> num = 3
3 >>> msg * num
4 'HiHiHi'
5 >>> msg * newNum
6
7 Traceback (most recent call last):
8   File "<pyshell\#3>", line 1, in <module>
9     msg * newNum
10 NameError: name 'newNum' is not defined
```

Sequenztyp Strings

- Strings werden als Sequenz von Zeichen aufgefasst, aus denen „slices“ extrahiert werden können

```
1 >>> msg = 'Hello World'
2 >>> msg[0]
3 'H'
4 >>> msg[3]
5 'l'
6 >>> msg[1:4]
7 'ell'
8 >>> len(msg)
9 11
10 >>> msg[-1]
11 'd'
```

H	e	l	l	o		W	o	r	l	d	
0	1	2	3	4	5	6	7	8	9	10	11

Sequenztyp Liste

- kann verschiedene Elemente enthalten (Strings, Integer, andere Listen, ...)
- sind im Gegensatz zu Strings **mutable**, d.h. Inhalt kann nach Definition modifiziert werden

```
1 >>> numbers = [1, 4, 9, 16]
2 >>> colors = ['green', 'blue', 'red']
3 >>> colors[0] * numbers[1]
4 'greengreengreengreen'
5 >>> colors[0] = 'black'
6 >>> numbers[1] = 2
7 >>> colors[0] * numbers[1]
8 'blackblack'
9 >>> name = 'Saulus'
10 >>> name[0] = 'P'
11 Traceback (most recent call last):
12   File "<pyshell#6>", line 1, in <module>
13     name[0] = 'S'
14 TypeError: 'str' object does not support item assignment
```

Sequenztypen allgemein

- alle Sequenztypen unterstützen die Operationen
 - ★ Verkettung `"Gambol" + "putty" == "Gambolputty"`
 - ★ Wiederholung: `2 * "spam" == "spamsam"`
 - ★ Indizierung: `"Python"[1] = "y"`
 - ★ Mitgliedschaftstest: `"yth" in "Python"`
 - ★ Slicing: `"Monty_Python's_Flying_Circus"[6:12] = "Python"`
 - ★ Iteration: `for x in "egg"`

Sequenzen / Strings: nützliche Funktionen

```
1 >>> chomsky = ['colorless', 'green', 'ideas']
2 >>> chomsky.sort()      # Sortieren
3 >>> chomsky.reverse()   # Reihenfolge umkehren
4 >>> chomsky.append('sleep') # Elemente hinzufuegen
5
6 # Elemente aus Liste in String umwandeln
7 >>> sent = ';' .join(chomsky)
8 >>> sent
9 'colorless;green;ideas;sleep'
10
11 # String mittels Delimiter aufteilen
12 >>> sent.split(;)
13 ['colorless', 'green', 'ideas', 'sleep']
```

Kontrollstrukturen: if / else

```
1 >>> word = 'Hello World'
2 >>> if len(word) < 3:
3     print 'word_length_is_less_than_three'
4     elif len(word) == 3:
5         print 'word_length_is_three'
6     else:
7         print 'word_length_is_greater_than_three'
```

- Gültigkeitsbereich der Schleife durch whitespace festgelegt (!!)
- kein Semikolon am Ende der Zeilen

Dictionaries

- Listen sind immer mit Integers indiziert
- Verallgemeinerung („alles mit allem indizieren“): **Dictionaries**

```
1 >>> pos = {}
2 >>> pos['colorless'] = 'adv'
3 >>> pos['furiously'] = 'adv'
4 >>> pos['ideas'] = 'n'
5 >>> pos['ideas']
6 'n'
7
8 >>> pos.keys() # all keys (as list)
9 >>> pos.values() # all values (as list)
10 >>> pos.items() # all items (as key / value pairs)
11
12 >>> if 'adv' in pos:
13 # ACHTUNG: prueft nur Schluesselmenge
```

Reguläre Ausdrücke

- „Zeichenketten, die der Beschreibung von Mengen beziehungsweise Untermengen von Zeichenketten mit Hilfe bestimmter syntaktischer Regeln dienen“ (Wikipedia)
- sehr mächtig & praktisch – aber: learning by doing!!

```
1 >>> import nltk
2 >>> s = 'This is a greeeat sentence!'
3 >>> nltk.re_show('is', s)
4 Th{is} {is} a greeeat sentence!
5 >>> nltk.re_show('e+', s)
6 This is a gr{eee}at s{e}nt{e}nc{e}!
7
8 >>> import re
9 # alle nicht-Vokal / Vokal-Paare
10 >>> re.findall('([\^aeiou][aeiou])', s)
11 ['hi', ' i', ' a', 're', 'se', 'te', 'ce']
```

Reguläre Ausdrücke ctd.

- Online testen:
<http://www.fileformat.info/tool/regex.htm>

Hinweise zur Übung / NLTK

- NLTK bietet recht viele „fertige“ Funktionalität / Hilfestellung
 - z.B. praktisch zum Plotten der Häufigkeiten: `nltk.draw.plot`
 - Stopwortlisten für verschiedene Sprachen verfügbar
 - ...
- es geistert auch viel fertiger Python-NLP-Code im Netz herum
 - inspirieren OK, kopieren nicht
 - Rad nicht 2x erfinden – aber VERSTEHEN!