Übungen 1

Arbeiten Sie in im NLTK-Tutorial Part1, Kapitel 1: *Programming Fundamentals and Python* die Abschnitte 2.1 (*Getting Started*) bis inklusive 2.5 (*Making Decisions*) durch.

Aufgabe 1 (entspricht NLTK 2.3.3.1)

Weisen Sie der Variable **kette** das Objekt **'colorless'** zu. Ändern Sie diese Kette durch *slicing* und Verkettung in **'colourless'** und weisen Sie der neuen Kette den Namen **kette1** zu, so dass folgende Auswertungen möglich sind:

```
>>> kette
'colorless'
>>> kette1
'colourless'
```

Aufgabe 2 (entspricht NLTK 2.3.3.3)

Verwenden Sie *slicing*, um von den folgenden Wörtern die Affixe abzuschneiden: *dishes, falling, nationality, undo, preheat*. Weisen Sie diese Wörter der Reihe nach den Variablen **wort1 - wort5** zu. Weisen Sie die Ergebnisse dann den Variablen **wort1a - wort5a** zu, so dass u.a. folgende Auswertungen möglich sind:

```
>>> word3
'nationality'
>>> word3a
'nation'
```

Aufgabe 3

Weisen Sie der Variable **kette** das Objekt **'preheated'** zu. Erzeugen anschließend 4 weitere Objekte (denen Sie die Namen **kette1 - kette4** zuweisen), und zwar indem Sie **kette** wie folgt 'slicen':

- mit 2 positiven Indizes
- mit zwei negativen Indizes
- mit einem positiven und einem negativen Index
- mit einem negativen und einem positiven Index

Es sollten u.a. folgende Auswertungen möglich sein:

```
>>> kette
'preheated'
>>> kette2 #bzw. kette1, kette3, kette4
'heat'
```

Aufgabe 4

Weisen Sie der Variable **name** eine Zeichenkette zu, die Ihren Vornamen, Ihren zweiten Namen und Ihren Nachnamen enthält. (Wenn Sie keinen zweiten Namen haben, erfinden Sie einen). Erzeugen Sie auf der Grundlage von **name** anschließend eine Zeichenkette, die nur noch Ihren zweiten Namen enthält, und weisen dieser den Namen **mittelname** zu. Es sollte folgende Auswertung möglich sein (natürlich mit Ihren Namen):

```
>>> name
'Helge Maria Schneider'
>>> mittelname
'Maria'
```

Aufgabe 5

Erzeugen Sie auf Grundlage von **name** eine Zeichenkette **name1**, die **name** in umgekehrter Reihenfolge wiedergibt. Es sollte folgende Auswertung möglich sein:

```
>>> name1
'redienhcS airaM egleH'
```

Aufgabe 6

Weisen Sie der Liste **['homer', 'marge', 'bart', 'lisa', 'maggie']** den Namen **liste** zu. Wandeln Sie die Liste in eine Kette namens **kette** um, so dass folgende Auswertungen möglich sind:

```
>>> liste
['homer', 'marge', 'bart', 'lisa', 'maggie']
>>> kette
'homer marge bart lisa maggie'
```

Aufgabe 7

Erzeugen Sie auf Grundlage von **liste** aus Aufgabe 6 eine Zeichenkette namens **kette**, in der die einzelnen Listenelemente durch einen Bindestrich separiert sind, so dass folgende Auswertung möglich ist:

```
>>> kette
'homer-marge-bart-lisa-maggie'
```

Aufgabe 8

Erzeugen Sie auf Grundlage von **liste** aus Aufgabe 6 eine Liste namens **liste1**, die die Elemente von **liste** in umgekehrter Reihenfolge wiedergibt, so dass die folgende Auswertung möglich ist:

```
>>> liste1
['maggie', 'lisa', 'bart', 'marge', 'homer']
```

Aufgabe 9

Weisen Sie der Variablen **kette** die Zeichenkette 'abacadaeafag' zu. Erzeugen Sie auf Basis von **kette** eine neue Zeichenkette namens **kette1**, die mit Ausnahme des 'a' alle Elemente aus **kette** enthält, Wenn aufg1.py kompiliert ist, so dass folgende Auswertungen möglich sind::

```
>>> kette
'abacadaeafag'
>>> kette1
' b c d e f g'
```

In der nächsten Aufgabe sollen Sie eine **Funktion** definieren. Im NLTK-Tutorial finden Sie Information zu diesem Thema erst an späterer Stelle. Für unsere Zwecke reichen ein paar knappe Anmerkungen.

Eine Python-Funktion besteht aus dem Funktionsnamen und einer Reihe von Argumenten (die ggf. auch leer sein kann): Funktionsname(Argument$_1$, Argument$_2$,…Argument$_n$). Mit einer Funktion sind bestimmte Aktionen verbunden, die auf das bzw. die Argumente der Funktion angewendet werden. Im folgenden Beispiel sehen wir die Funktion **len()**, eine Funktion mit einem Argument, hier eine Zeichenkette, dessen Länge ermittelt wird:

```
>>> len('hallo miteinander')
17
```

Hier wird die Aktion 'ermittele Länge' auf das Argument angewendet und ein Ergebnis (17) zurückgeliefert. Bei der Funktion **type()** wird als Ergebnis der Datentyp zurückgeliefert, dem das Argument angehört:

```
>>> type([3, 4, 6, 7])
<type 'list'>
```

'Aktion' kann aber auch bedeuten, dass nicht ein Wert zurückgeliefert wird sondern etwas anderes passiert, z.B. dass das Argument auf dem Monitor ausgegeben wird:

```
>>> print('hallo')
hallo
```

Um nun in Python neue Funktionen zu definieren, verwenden wir den Befehl **def**. Dieser wird der zu definierenden Funktion vorangestellt, die entsprechende Zeile schließt mit einem Doppelpunkt ab und die Aktion(en), die dann vollzogen werden soll, werden eingerückt in einen Anweisungsblock in eine oder mehrere neue Zeilen geschrieben (diese Einrückung wird i.d.R. automatisch vom Editor vorgenommen):

def Funktionsname (Argument$_1$, Argument$_2$,…Argument$_n$):
    Anweisungsblock

Beispiel: wir definieren mit Bezug auf **print** die Funktion **drucke(Sequenz)**:

```
>>> def drucke(x):
        print(x)
```

Anschließend können wir die Funktion mit einer beliebigen Liste oder Zeichenkette als Argument aufrufen und das gewünschte Ergebnis wird auf dem Monitor ausgegeben:

```
>>> drucke('mein name ist hase')
mein name ist hase
```

Aufgabe 10

Definieren Sie mithilfe des Befehles **def** eine Funktion namens **drucke_alphabetisch()**, die die Elemente einer Sequenz (Liste oder Zeichenkette) in alphabetischer Reihenfolge und ohne Umbruch auf dem Monitor ausgibt. Dazu müssen Sie eine **for**-Schleife verwenden. Es sollte die folgende Auswertung möglich sein:

```
>>> drucke_alphabetisch('bxyafwce')
a b c e f w x y
```

Aufgabe 11

Weisen Sie der Liste `['der','junge','ist','sehr','unglluecklich']` den Namen `liste` zu. Arbeiten Sie diese Liste mit einer `for`-Schleife so ab, dass Sie eine neue Liste namens `liste1` erzeugen, in der nur noch die Elemente aus `liste` enthalten sind, deren Länge >= 4 ist. Es sollten die folgenden Auswertungen möglich sein:

```
>>> liste16
['der', 'junge', 'ist', 'sehr', 'unglluecklich']
>>> liste17
['junge', 'sehr', 'unglluecklich']
```

Aufgabe 12 (entspricht NLTK 2.4.6.9)

Arbeiten Sie `liste` aus Aufgabe 11 mit einer `for`-Schleife und der Funktion `len()` so ab, dass Sie eine neue Liste namens `liste1` erzeugen, in der die Längen der Elemente aus `liste` aufgeführt sind. Es sollte die folgende Auswertungen möglich sein::

```
>>> liste11
[3, 5, 3, 4, 12]
```

Aufgabe 13 (entspricht NLTK 2.4.6.11)

Weisen Sie der Zeichenkette `'the dog barks in the garden'` die Variable `kette` zu. Nutzen Sie dann die Funktion `index()` in Kombination mit *slicing*, um eine Liste namens `liste` zu erzeugen, in der alle Wörter aus `kette` bis exklusive 'in' enthalten sind. Es sollten die folgenden Auswertungen möglich sein:

```
>>> kette
'the dog barks in the garden'
>>> liste
['the', 'dog', 'barks']
```

Aufgabe 14

Nutzen Sie *Slicing*, die Index-Funktion und Verkettung, um auf Grundlage von `kette` aus Aufgabe 13 eine Zeichenkette namens `kette1` zu erzeugen, bei der die letzten drei Wörter ('in the garden') am Anfang stehen. Es sollte die folgende Auswertungen möglich sein::

```
>>> kette1
'in the garden the dog barks'
```

Aufgabe 15 (entspricht NLTK 2.5.5.1.1)

Weisen Sie der Zeichenkette `'she sells sea shells by the sea shore'` die Variable `kette` zu. Definieren Sie anschließend eine Funktion `drucke_sh(kette)`, bei der die Wörter, die mit 'sh' beginnen, zeilenweise auf dem Monitor ausgegeben werden. Es sollten die folgenden Auswertungen möglich sein:

```
>>> kette
'she sells sea shells by the sea shore'
>>> drucke_sh(kette14)
she
shells
shore
```

Aufgabe 16 (entspricht NLTK 2.5.5.1.3)

Erzeugen Sie auf Grundlage von `kette` aus Aufgabe 14 eine neue Zeichenkette namens `kette1`, in der vor jedem Wort, das mit 'se' beginnt, das Wort 'like' eingebaut ist. Es sollte die folgende Auswertungen möglich sein:

```
>>> kette1
' she like sells like sea shells by the like sea shore'
```