Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Dr. Stefan Wölfl, Thorsten Engesser, Tim Schulte
Wintersemester 2016/2017

Universität Freiburg
Institut für Informatik

Übungsblatt 5

Abgabe: Freitag, 25. November 2016, 20:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet05` im Wurzelverzeichnis Ihrer Arbeitskopie des SVN-Repositories an. Ihre Lösungen werden dann in Dateien in diesem Unterverzeichnis erwartet.

Beachten Sie bitte bei allen Aufgaben die *Hinweise zur Bearbeitung der Übungsaufgaben* unter der folgenden URL:

`http://gki.informatik.uni-freiburg.de/teaching/ws1617/info1/guide/hinweise.html`

Insbesondere soll jede von Ihnen implementierte Funktion einen minimalen Doc-String enthalten. Dazu gehört eine Kurzbeschreibung der Funktion sowie die Dokumentation der Parameter und des Rückgabewerts. Siehe:

`http://gki.informatik.uni-freiburg.de/teaching/ws1617/info1/guide/styleguide.html#doc-strings`

Bewertet wird bei allen Aufgaben die letzte Version, die zur Deadline des Übungsblattes auf dem SVN-Server eingereicht ist.

Aufgabe 5.1 (Liste glätten; Datei: `flatten.py`; Punkte: 4+4)

Wir nennen eine Liste *verschachtelt*, wenn sie mindestens eine weitere Liste als Element enthält. Die geglättete Version einer (verschachtelten oder nicht-verschachtelten) Liste ergibt sich wie folgt: jedes Element L, das eine Liste ist, wird (unter Beibehaltung der Reihenfolge) durch die Elemente der geglätteten Version von L ersetzt; jedes andere Element wird (unter Beibehaltung der Reihenfolge) übernommen.

Hinweis: Bei der Bearbeitung der folgenden Aufgaben sollen nur Hilfsmittel verwendet werden, die bisher in der Vorlesung eingeführt wurden. Lösungen, die zusätzliche Module importieren, werden nicht bewertet.

(a) Definieren Sie eine rekursive Funktion `flatten(lst)`, die die geglättete Version der übergebenen Liste `lst` berechnet und zurückgibt. Ihre Funktion soll dabei eine komplett neue Liste zurückgeben und darf die übergebene Liste bzw. alle betrachteten Teillisten nicht verändern. Betrachten Sie dazu das folgende Beispiel:

```
>>> egg = [3,4,[[5]]]
>>> spam = [[[1, 2, egg], (6, [7]), 8], 9, False]
>>> flatten(spam)
[1, 2, 3, 4, 5, (6, [7]), 8, 9, False]
>>> spam
[[[1, 2, [3, 4, [[5]]]], (6, [7]), 8], 9, False]
>>> egg
[3, 4, [[5]]]
```

(b) Definieren Sie eine rekursive Funktion `flatten_in_place(lst)`, die eine beliebig Liste `lst` übernimmt und die Liste selbst glättet (in Gegensatz zu (a)). Bei einem Aufruf von `flatten_in_place` sollen auch alle Teillisten geglättet werden. Betrachten Sie dazu folgendes Beispiel:

```
>>> egg = [3,4,[[5]]]
>>> spam = [[[1, 2, egg], (6, [7]), 8], 9, False]
>>> flatten_in_place(spam)
>>> spam
[1, 2, 3, 4, 5, (6, [7]), 8, 9, False]
>>> egg
[3, 4, 5]
```

Aufgabe 5.2 (Wort-Baum; Datei: `words.py`; Punkte: 3+2+2+3)

In dieser Aufgabe geht es darum, eine Zeichenfolge (einen String) einzulesen und dabei eine Datenstruktur anzulegen, die es später erlaubt für ein gegebenes Wort zu entscheiden, ob und wie oft dieses Wort in der Zeichenkette vorkommt. Unter einem *Wort* verstehen wir im Folgenden jede endliche Folge von Buchstaben des deutschen Alphabets (also den Zeichen a, b, c, ..., z, A, B,..., Z, ä, Ä, ö, Ö, ü, Ü, ß) der Länge ≥ 1. Je zwei Wörter in der Zeichenfolge werden durch eine nicht-leere, endliche Folge von Zeichen, die nicht zu diesen Buchstaben gehören (z.B., Leerzeichen, Satzzeichen, Zeilenumbrüche), getrennt.

Laden Sie das Template `words.py` von der Vorlesungswebsite herunter. Dieses enthält bereits eine Funktion `next_word(s)`, die angewendet auf einen String `s` ein Tupel `(word, rest)` zurückgibt, wobei `word` das erste Wort (im Sinne der Spezifikation) in `s` ist und `rest` die Zeichenfolge ist, die in `s` auf `word` folgt.

Es soll nun ein Suchbaum der in einem String `s` vorkommenden Wörter erzeugt werden: Jeder Knoten des Suchbaumes wird durch eine Liste

```
[ltree, rtree, word, n]
```

repräsentiert. Dabei ist `word` ein Wort, `n` die Anzahl der Vorkommnisse von `word` in `s`, `ltree` der linke und `rtree` der rechte Teilbaum. Als Ordnungsrelation verwenden wir Python's lexikographische Ordnung von Strings. Das heißt, ein Wort `w` wird im linken Teilbaum eines Knotens `[ltree, rtree, word, n]` eingefügt bzw. gesucht, falls der Vergleich `w < word` den Wert `True` zurückgibt, etc. In Blattknoten sind `ltree` und `rtree` jeweils der Wert `None`.

(a) Definieren Sie eine Funktion `word_tree(s)`, die aus dem übergebenen String `s` diesen Suchbaum erzeugt und zurückgibt. Natürlich kann Ihre Funktion eine selbst-definierte Hilfsfunktion verwenden.

(b) Definieren Sie eine Funktion `word_freq(tree, word)`, die für einen solchen Suchbaum `tree` und ein Wort `word`, die in `tree` hinterlegte Anzahl der Wortvorkommnisse von `word` zurückgibt. Falls das Wort in dem Baum nicht vorkommt, soll die Funktion den Wert `0` zurückgeben.

(c) Definieren Sie eine Funktion `print_tree(tree)`, die alle in `tree` abgelegten Wörter und die in `tree` jeweils hinterlegte Anzahl der jeweiligen Wortvorkommnisse zeilenweise (pro Zeile ein Wort und dessen Anzahl) ausgibt. Dabei soll der Baum in symmetrischer Reihenfolge (*In-Order*) traversiert werden.

Die Ausgabe könnte in etwa wie folgt aussehen:

```
>>> s = "spam eggs spam eggs ham spam hamham Spam eggs hamham"
>>> tr = word_tree(s)
>>> print_tree(tr)
Spam    : 1
```

```
eggs    : 3
ham     : 1
hamham  : 2
spam    : 3
```

(d) Definieren Sie eine Funktion `freq_words(tree)`, die eine verschachtelte Liste mit den Wörtern aus `tree` erzeugt und zurückgibt. In der zurückgegebenen Liste soll das Element mit Index n eine Liste sein, die genau die Wörter mit der Worthäufigkeit $n + 1$ enthält. Das letzte Element der Rückgabeliste darf keine leere Liste sein. Betrachten Sie dazu das folgende Beispiel:

```
>>> freq_words(word_tree('spam eggs spam ham eggs spam spam spam'))
[['ham'], ['eggs'], [], [], ['spam']]
```

Aufgabe 5.3 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet05` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Interessantes, etc.).