Programmierkurs Python II – SS 2013

Übung 1

---

1 Bäume implementieren (4 Punkte)

Implementiere eine Klasse für Bäume (siehe Folien). Die Klasse soll mindestens die folgenden Methoden implementieren:

- `__str__(self)` konvertiert einen Baum in einen String, und zwar so, dass er auch wieder als Eingabe geparst werden kann: Alle (Teil-)Bäume sind mit runden Klammern begrenzt, außer Blätter; Die Wurzel des Teilbaums steht direkt nach der öffnenden Klammer, die Kinderbäume sind mit Leerzeichen getrennt.
  Beispiel: `(S (NP Er) (VP schnarcht))`
- `__repr__(self)` konvertiert einen Baum in einen String, der wieder als Python-Code interpretiert werden kann. Beispiel:
  `Tree('S',[Tree('NP',[Tree('Er',[])]),Tree('VP',[Tree('schnarcht',[])])])`
- `__iter__(self)` liefert einen Iterator über die Knoten im Baum.
- `leaves(self)` liefert einen Iterator über die Blätter des Baums.
- `depth(self)` berechnet die Tiefe des Baums

2 Baumausdrücke parsen (4 Punkte)

Implementiere einen Parser für Baumausdrücke. Die Funktion `trees` stellen wir zur Verfügung.

Die Funktion `tokenize(string)` soll für eine Zeichenkette als Eingabe einen Iterator über die Tokens liefern:

```
>>> list(tokenize("(S (NP Peter) (VP schläft))"))
['(', 'S', '(', 'NP', 'Peter', ')', '(', 'VP', 'schläft', ')', ')']
```

Der Parser soll als Iterator implementiert werden (siehe Beispiel-Code).

```
class Parser:
    __slots___ = ['tokens']
    def __init__(self, tokens):
        self.tokens = tokens
        self._trees = trees(tokens)
    def __iter__(self):
        return self
    def __next__(self):
        return next(self._trees)
```

```
>>> t = tokenize("(S (NP Peter) (VP schläft)) (S (NP Er) (VP schnarcht))")
>>> p = Parser(t)
>>> next(p)
Tree('S', [Tree('NP', [Tree('Peter', [])]), Tree('VP', [Tree('schläft', [])])])
>>> next(p)
Tree('S', [Tree('NP', [Tree('Er', [])]), Tree('VP', [Tree('schnarcht', [])])])
>>> next(p)
StopIteration
```

Erweitere die Implementierung der Funktionen `tree` bzw. `trees` so, dass Fehler in der Eingabe korrekt behandelt werden (beispielsweise durch Werfen einer Ausnahme).

3 Nichtterminalsymbole zählen (3 Punkte)

Auf der Homepage findet sich eine kleine Beispieldatei mit einer Liste von (zwei) Bäumen. Implementiere eine funktion `countNonterminals(file)`, die Bäume aus der Datei einliest und zählt, wie häufig welche Nichtterminalsymbole vorkommen. Als Nichtterminalsymbol sollen alle Etiketten von Nicht-Blättern zählen.

Versuche, die Implementierung so zu gestalten, dass die Eingabedatei möglichst nur Zeichen- oder Zeilenweise eingelesen wird. (Es ist hier ausdrücklich erwünscht, Hilfsmethoden zu implementieren.)

Abgabe bis Donnerstag, 2. Mai 2013, 11:00 Uhr