Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Dr. Christian Becker-Asano, Dr. Stefan Wölfl
Wintersemester 2014/2015

Universität Freiburg
Institut für Informatik

Übungsblatt 10

Abgabe: Freitag, 16. Januar 2015, 18:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet10` im Wurzelverzeichnis Ihrer Arbeitskopie des SVN-Repositories an. Ihre Lösungen werden dann entsprechend dem ersten Übungsblatt 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/ws1415/info1/hinweise_uebungen.txt`

Insbesondere müssen alle Python-Dateien und die darin definierten Funktionen entsprechend diesen Hinweisen dokumentiert und dabei PEP8 beachtet werden.

Unterbinden Sie alle nicht erforderlichen `print`-Anweisungen in Ihren Python-Dateien. Falls Sie zum Debuggen `print`-Anweisungen verwenden, benutzen Sie eine globale Variable oder Konstante, mit der Sie Ausgaben auf die Konsole ausschalten können.

Überprüfen Sie, dass Sie alle Lösungen ins Repository hochgeladen haben. Bewertet wird bei allen Aufgaben die letzte Version, die zur Deadline des Übungsblattes auf dem SVN-Server eingereicht ist.

Aufgabe 10.1 (Reguläre Ausdrücke; Punkte: 3+3; Datei: `regexp.txt`)

Beschreiben Sie umgangssprachlich, welche Mengen von Strings von den folgenden regulären Ausdrücken spezifiziert werden. Geben Sie dann für jeden regulären Ausdruck jeweils zwei Strings an. Die Anwendung des regulären Ausdrucks mittels `findall` soll dabei für den einen String eine leere Liste und für den anderen eine nicht-leere Liste zurückgeben.

(a) `r'[^\']+\d[a-d|*]'`

(b) `r'(([A-Za-z]+?)(\d)\2-\3)'`

Aufgabe 10.2 (Reguläre Ausdrücke II; Punkte: 4+1; Datei: `month.py`)

(a) Schreiben Sie eine Funktion `find_dates`, die aus einem Text alle Datumsangaben der Gestalt wie im Beispiel

```
10. März 2014, 9:05 Uhr
```

findet und als eine Liste von Dictionaries zurückgibt. Dabei soll das Leerzeichen vor der Monatsangabe optional sein. Jedes Dictionary habe dabei die Schlüssel `year`, `month`, `day`, `hour`, und `minute`, denen numerische Werte zugewiesen sind. Das Dictionary im Beispiel würde also wie folgt aussehen:

```
{'year': 2014, 'month': 3, 'day': 10, 'hour': 9,
'minute': 5}.
```

(b) Schreiben Sie für die neu implementierte Funktion eine Test-Funktion mit 4 Assertions.

Aufgabe 10.3 (Reguläre Ausdrücke und Gen-Sequenzierung; Punkte: 1+1+4+2; Datei: `augc.py`)

In der 20. Vorlesung wurde beispielhaft ein regulärer Ausdruck zum Extrahieren von URLs und Linktexten vorgestellt. In Anlehnung an dieses Beispiel, wollen wir uns mit einem Problem beschäftigen, das für die Bioinformatik relevant ist.
Bei der Transkription wird in der Genetik eine DNA-Sequenz aus Nukleinbasen, die durch die vier Zeichen `ATGC` repräsentiert werden können, in RNA übersetzt, die durch die vier Zeichen `AUGC` repräsentiert werden können[1]. Auf spezifische Details wollen wir nicht weiter eingehen, sondern uns mit der Analyse der resultierenden mRNA-Struktur beschäftigen.
Beliebig lange Strings aus den Zeichen `AUGC` sind prinzipiell möglich. Jede beliebige Dreiergruppe innerhalb dieses Strings wird als *Codon* bezeichnet, womit $4^3 = 64$ mögliche Basentripletts/Codons möglich sind[2]. Folgende (stark vereinfachte) Regeln bei der Translation einer Sequenz sind zu beachten:

- Die Übersetzung einer Nukleobasen-Sequenz kann an der ersten, zweiten oder dritten Stelle begonnen werden, wodurch sich verschiedene `Codon`-Sequenzen ergeben.
- Bei einmal festgelegtem Startpunkt werden alle weiteren Nukleobasen lückenlos zu Codons zusammengefasst und übersetzt.
- Das Codon `AUG` wird als *Startcodon* bezeichnet.
- Das Codon `UAG` wird als *Stopcodon* bezeichnet.
- Alle Codons zwischen einem Start- und einem Stopcodon sind (für uns) von Bedeutung, alles andere sollte ignoriert werden.

Nun zu Ihren Programmieraufgaben, bei denen Sie davon ausgehen dürfen, dass alle Eingabestrings keine anderen als die Zeichen `AUGC` (beliebig oft) enthalten:

(a) Implementieren Sie eine Funktion `check_for_startcodon(AUGCstring)`, die prüft, ob der Eingabestring `AUGCstring` mindestens ein Startcodon enthält und nur in diesem Fall `True` zurückliefert, sonst `False`. Ist es hierfür sinnvoll, reguläre Ausdrücke zu verwenden? Begründen Sie.

(b) Entwerfen Sie einen regulären Ausdruck, mit dem geprüft wird, ob irgendwo innerhalb eines Eingabestrings mindestens einmal auf ein Startcodon `AUG` ein Stopcodon `UAG` folgt. Hierbei darf alles zwischen diesen beiden Codons ignoriert werden, mit der Ausnahme, dass die Anzahl der dazwischen befindlichen Zeichen ein echtes Vielfaches von drei sein muss.

(c) Schreiben Sie unter Benutzung des Moduls `re` für reguläre Ausdrücke eine Funktion `extract_codon_string(AUGCstring)`, die folgende Anforderungen erfüllt:

- Alle Codons, die durch ein Start- und ein Stopcodon eingeschlossen werden, sollen aus dem Eingabestring `AUCGstring` herausgefiltert und als Liste zurückgegeben werden.

[1] Siehe auch: `http://de.wikipedia.org/wiki/Transkription_(Biologie)`
[2] Siehe auch: `http://de.wikipedia.org/wiki/Genetischer_Code#Codon`

- Der Rückgabewert der Funktion ist also eine Liste von Strings, deren jeweilge Länge ein ganzes Vielfaches von drei sein muss. Diese Bedingung sollte als Teil des verwendeten regulären Ausdrucks implementiert werden.
- Ihr regulärer Ausdruck sollte `non-greedy` sein. Zum Beispiel sollte der Eingabestring `'AUGCCCUAGAUGCCCUAG'` die Liste `['CCC', 'CCC']` als Rückgabewert liefern.
- Eingebettete `AUG`'s sind erlaubt, d.h. die Eingabe `'AUGAUGCCCUAG'` sollte `['AUGCCC']` als Rückgabewert liefern.

(d) Dokumentieren und testen Sie Ihre Implementationen wie bei den vorherigen Übungsblättern mit Hilfe entsprechender Testfunktionen.

Aufgabe 10.4 (Erfahrungen; 2 Punkte)

Legen Sie im Unterverzeichnis `sheet10` eine Textdatei `erfahrung.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Bezug zur Vorlesung, Interessantes, etc.).