Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Dr. Stefan Wölfl, Thorsten Engesser
Wintersemester 2015/2016

Universität Freiburg
Institut für Informatik

Übungsblatt 6

Abgabe: Freitag, 4. Dezember 2015, 20:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet06` 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/ws1516/info1/wiki/hinweise.html`

Überprüfen Sie, dass Sie alle Lösungen ins Repository hochgeladen haben (z.B. mit dem Befehl `svn status`). Überprüfen Sie auch die Webseite Ihres SVN-Unterverzeichnisses:

`https://daphne.informatik.uni-freiburg.de/ws1516/InformatikI/svn/$RZLOGIN`

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

Aufgabe 6.1 (PEP8 und Dokumentation; Punkte: 3)

Die Abgaben zu diesem Übungsblatt werden daraufhin geprüft, ob Ihr Code PEP8-konform und Ihre Funktionen mittels Doc-Strings ausreichend dokumentiert sind. Bei diesem Übungsblatt können Sie hierfür Punkte sammeln!
Beachten Sie hierzu auch die Hinweise im Python Style Guide, den Sie im Wiki zur Vorlesung finden.

Hinweis:
Benutzen Sie einen Style-Checker, um alle von Ihnen eingereichten Python-Dateien daraufhin zu prüfen. Das in der Vorlesung angegebene Tool pep8 berücksichtigt nicht die in PEP8 beschriebenen Namenskonventionen für Funktionen, etc. Hierzu kann (falls nicht bereits vorhanden) das Tool flake8 und die Erweiterung pep8-naming in den meisten Python-Installationen einfach nachinstalliert werden (z.B. mit pip3 install flake8).

Aufgabe 6.2 (Mengen-Operationen; Punkte: 3+3; Datei: `sets.py`)

(a) Definieren Sie eine Funktion `powerset(s)`, die für eine beliebige Menge `s` die Potenzmenge berechnet und zurückgibt.

(b) Definieren Sie eine Funktion `subsets(s, k)`, die angewendet auf eine Menge `s` und eine natürliche Zahl $\texttt{k} \geq 0$, die Menge gerade jener Teilmengen von `s` zurückgibt, die genau `k` Elemente enthalten.

Sie dürfen die Teilaufgaben in beliebiger Reihenfolge bearbeiten und die zuerst definierte Funktion in der Definition der anderen verwenden. Schreiben Sie für die beiden Funktionen in (a) und (b) jeweils vier Testfunktionen (`test_powerset_xyz()` und `test_subsets_xyz()`, `xyz` ist dabei durch Namen Ihrer Wahl zu ersetzen).

Hinweis: In dieser Aufgabe sollen ausschließlich Mengen (keine Listen!) benutzt werden. Das Importieren von anderen Modulen ist bei der Bearbeitung beider Teilaufgaben nicht erlaubt. Achten Sie ferner darauf, dass die an diese Funktionen übergebenen Mengen nach Ausführung der Funktion noch dieselben Mengen sind!

Aufgabe 6.3 (Datenbank; Punkte: 3+3+3; Datei: `db.py`)

Wir wollen im Folgenden einfache Datenbank-Operationen implementieren. Eine Datenbank besteht hierbei aus mehreren *Tabellen*, in denen Daten abgelegt sind. Jede Tabelle besteht aus *Einträgen* wie im nachfolgenden Beispiel:

c_id	family_name	given_name
ww	White	Walter
sw	White	Skyler
sg	Goodman	Saul

Dabei stehen in der ersten Zeile der Tabelle die verschiedenen Attribute und jede weitere Zeile enthält einen Dateneintrag mit den entsprechenden Attributwerten.
Im Folgenden realisieren wir die Einträge als Dictionaries mit Attribut-Werte-Paaren, d.h. die Attribute werden als Schlüssel dieser Dictionaries verwendet. Eine Tabelle ist dann eine Liste solcher Einträge (Duplikat-Einträge sind möglich). Betrachten Sie dazu auch die folgenden Beispieltabellen:

```
>>> series = [
...     {'s_id': 'bb', 'title': 'Breaking Bad'},
...     {'s_id': 'bcs', 'title': 'Better Call Saul'}]

>>> characters = [
...     {'c_id': 'ww', 'family_name': 'White', 'given_name': 'Walter'},
...     {'c_id': 'sw', 'family_name': 'White', 'given_name': 'Skyler'},
...     {'c_id': 'sg', 'family_name': 'Goodman', 'given_name': 'Saul'}]

>>> series_characters = [
...     {'c_id': 'ww', 's_id': 'bb'},
...     {'c_id': 'sw', 's_id': 'bb'},
...     {'c_id': 'sg', 's_id': 'bb'},
...     {'c_id': 'sg', 's_id': 'bcs'}]
```

In den folgenden Teilaufgaben beschreiben wir die Datenbank-Operationen, die von Ihnen implementiert werden sollen. Beachten Sie dabei, dass die an die Funktion übergebenen Tabellen *nicht* verändert werden sollen. Testen Sie Ihre Implementierungen mit Hilfe von jeweils zwei Test-Funktionen.

(a) Implementieren Sie eine Funktion `project(tbl, keys)`, die die *Projektion* einer Tabelle `tbl` auf die Attributmenge `keys` berechnet und zurückgibt. Die Projektion ist dabei eine neue Tabelle, die aus `tbl` alle Spalten "ausblendet", deren Attribute in `keys` nicht vorkommen.

```
>>> proj = project(characters, {'c_id', 'given_name'})

>>> from pprint import pprint
>>> pprint(proj)
[{'c_id': 'ww', 'given_name': 'Walter'},
 {'c_id': 'sw', 'given_name': 'Skyler'},
 {'c_id': 'sg', 'given_name': 'Saul'}]
```

(b) Implementieren Sie eine Funktion `select(tbl, **kwargs)`, die die *Selektion* einer Tabelle `tbl` auf die in `kwargs` übergebenen Attribut-Werte-Paare berechnet und zurückgibt. Die Ergebnistabelle besteht hier aus genau den Einträgen aus `tbl`, die auf allen in `kwargs` benutzten Attributen mit den in `kwargs` übergebenen Werten übereinstimmen.

Hinweis: Zur Vereinfachung gehen wir hier davon aus, dass alle Attributnamen in den Tabellen valide Variablennamen in Python sind.

```
>>> sel = select(characters, family_name='White')

>>> from pprint import pprint
>>> pprint(sel)
[{'c_id': 'ww', 'family_name': 'White', 'given_name': 'Walter'},
 {'c_id': 'sw', 'family_name': 'White', 'given_name': 'Skyler'}]
```

(c) Implementieren Sie eine Funktion `join(tbl1, tbl2, *args)`, die den *Verbund* beliebig vieler Tabellen `tbl1`, `tbl2`, ..., `tblN` berechnet (für `N` ≥ 2) und zurückgibt. Der Verbund zweier Tabellen `tbl1` und `tbl2` ist hierbei eine Tabelle, die für je zwei Einträge `e1` der Tabelle `tbl1` und `e2` der Tabelle `tbl2` einen Eintrag enthält, sofern die Werte auf allen gemeinsamen Attributen übereinstimmen. Der Eintrag selbst besteht aus allen Attribut-Werte-Paaren, die in einem der beiden Einträge `e1` und `e2` vorhanden sind (siehe nachfolgendes Beispiel).

Implementieren Sie zunächst eine Hilfsfunktion `_join` für den Verbund zweier Tabellen. Implementieren Sie dann den Verbund von mehreren Tabellen `join(tbl1, tbl2, ..., tblN)` durch Hintereinanderschaltung der Hilfsfunktion, also z.B. durch: `_join(...(_join(tbl1, tbl2), ...), tblN)`.

```
>>> x = join(series_characters, series, characters)

>>> from pprint import pprint
>>> pprint(x)
[{'c_id': 'ww',
  'family_name': 'White',
  'given_name': 'Walter',
  's_id': 'bb',
  'title': 'Breaking Bad'},
```

```
{'c_id': 'sw',
 'family_name': 'White',
 'given_name': 'Skyler',
 's_id': 'bb',
 'title': 'Breaking Bad'},
{'c_id': 'sg',
 'family_name': 'Goodman',
 'given_name': 'Saul',
 's_id': 'bb',
 'title': 'Breaking Bad'},
{'c_id': 'sg',
 'family_name': 'Goodman',
 'given_name': 'Saul',
 's_id': 'bcs',
 'title': 'Better Call Saul'}]
```

Aufgabe 6.4 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet06` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Bezug zur Vorlesung, Interessantes, etc.).