Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel — Universität Freiburg
Dr. Stefan Wölfl, Thorsten Engesser, Tim Schulte — Institut für Informatik
Wintersemester 2016/2017

Übungsblatt 7

Abgabe: Freitag, 09. Dezember 2016, 20:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet07` 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`

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

Aufgabe 7.1 (PEP8 und Dokumentation; Punkte: 2)

Die Abgaben zu diesem Übungsblatt werden daraufhin geprüft, ob Ihr Code PEP8-konform und Ihre Funktionen mittels Doc-Strings ausreichend dokumentiert sind. Beachten Sie hierzu die Hinweise im Python Style Guide, den Sie im Guide finden. Ab sofort wird dies für alle Abgaben vorausgesetzt, bei diesem Übungsblatt können Sie hierfür noch Punkte sammeln!

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 7.2 (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 7.3 (Kryptoanalyse; Dateien: `caesar.py`, `crack_caesar.py`; Punkte: 3+5)
Laden Sie die Datei `caesar.py` von der Webseite der Übungen zur Vorlesung herunter und speichern Sie diese im Unterverzeichnis zu diesem Übungsblatt.
`http://gki.informatik.uni-freiburg.de/teaching/ws1617/info1/python/caesar.py`
In dieser Datei wird die Funktion `caesar(text, shift, charset=ascii_letters)` definiert, die die Cäsarverschiebung[1] implementiert. Dabei wird in dem an die Funktion übergebenen String `text` jedes im Zeichensatz `charset` vorkommende Zeichen um `shift` viele Zeichen verschoben. Alle anderen Zeichen bleiben in der zurück gegebenen Zeichenfolge unverändert.

(a) Modifizieren Sie den Programmcode unterhalb der Zeile `if __name__ == ...` so, dass der Eingabetext von der Standardeingabe gelesen wird und das verschlüsselte Resultat in eine Datei geschrieben wird. Der Dateiname soll zusammen mit der Anzahl der zu verschiebenden Stellen (in dieser Reihenfolge) als Kommandozeilenargumente übergeben werden. Behandeln Sie Fehler (wie z.B. eine falsche Anzahl von Kommandozeilenparametern) auf für den Benutzer sinnvolle Weise. Wir vereinbaren ferner, dass die Eingabe durch eine Zeile mit dem einzigen Zeichen `.` beendet wird.

[1] `https://de.wikipedia.org/wiki/Caesar-Verschlüsselung`

(b) Schreiben Sie ein Programm `crack_caesar.py`, das eine Cäsar-verschlüsselte Textdatei einliest und dann versucht, den enthaltenen Text automatisch und ohne Kenntnis der tatsächlichen Verschiebung zu entschlüsseln. Als einziger Kommandozeilenparameter soll der Dateiname übergeben werden. Der (im besten Fall korrekt) entschlüsselte Text soll schließlich auf der Standardausgabe ausgegeben werden. Behandeln Sie Fehler (z.B. die Angabe eines falschen oder ungültigen Dateinamens) auch hier auf für den Benutzer sinnvolle Weise.

Ihre Entschlüsselung soll hierbei die folgende Idee implementieren: die (vermutete) Entschlüsselung ist ein Text mit einer Buchstabenverteilung, die der Buchstabenverteilung in natürlichsprachlichen Texten am ähnlichsten ist (für verschiedene Sprachen finden Sie die zu erwartende Verteilung z.B. auf `https://de.wikipedia.org/wiki/Buchstabenhäufigkeit`).

Man bestimmt also zunächst für die verschiedenen (Rück-)Verschiebungen die relativen Häufigkeiten $p(c)$ der im Text vorkommenden Buchstaben $c \in \{A, \ldots, Z\}$ (Groß- und Kleinbuchstaben werden nicht unterschieden) und vergleicht diese dann mit den zu erwartenden relativen Häufigkeiten $\bar{p}(c)$ der Sprache des zu entschlüsselnden Textes (in unserem Fall gehen wir davon aus, dass der verschlüsselte Text in "Englisch" ist). Zum Vergleich zweier Häufigkeitsverteilungen verwenden Sie den sogenannten Chi-Quadrat-Test, der ein Maß für die Größe der Abweichung ist:

$$\chi^2 = \sum_{c \in \{A,\ldots,Z\}} \frac{(p(c) - \bar{p}(c))^2}{\bar{p}(c)}.$$

Das Maß ist also kleiner, je ähnlicher sich p und $\bar{p}$ sind.

Hinweis: Aus der Datei `caesar.py` können Sie in der Datei `crack_caesar.py` die Funktion `caesar` wie folgt importieren:

```
from caesar import caesar
```

Aufgabe 7.4 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet07` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Interessantes, etc.).