Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel — Universität Freiburg
Dr. Christian Becker-Asano, Dr. Stefan Wölfl — Institut für Informatik
Wintersemester 2013/2014

Übungsblatt 5

Abgabe: Freitag, 29. November 2013, 18:00 Uhr

Bei diesem Übungsblatt erwarten wir alle Lösungen zu den Aufgaben in der jeweils angegebenen Python-Datei im Unterverzeichnis `sheet05`. Beachten Sie dabei die bekannten Formatierungshinweise für Python-Dateien und vergessen Sie nicht, alle von Ihnen verwendeten Dateien ins SVN-Repository hochzuladen!

Versehen Sie alle Ihre Funktionen in den Aufgaben 5.2 und 5.4 im Doc-String mit entsprechenden Tests, die mittels `doctest` getestet werden können. Für die Aufgaben 5.1 und 5.3 integrieren Sie die jeweils beschriebenen Tests z.B. in die oberste Skript-Umgebung, d.h. in den Anweisungblock von:

```
if __name__ == "__main__":
    <Ihre Tests ...>
```

Aufgabe 5.1 (Adressbücher als Dictionaries; 1+2+2+1 Punkte; Datei: `addressbook.py`)

In dieser Aufgabe beschäftigen wir uns mit dem Datentyp `dictionary`. Zur Übung sollen Sie Adressbuchdaten, die in Form einer Liste mit Tupeln vorliegen, schrittweise in ein Dictionary umwandeln.

Folgende Daten sind gegeben:

```
address_list = [("salutation", "givenName", "familyName",
                 "homeAddress", "homeStreetNumber", "homeCity",
                 "homeTelephoneNumber", "homeEmailAddress",
                 "workAddress", "workStreetNumber", "workCity",
                 "workTelephoneNumber", "workEmailAddress"),
                ("Mr.", "Frodo", "Baggins",
                 "Bag End", "1", "Westfarthing",
                 "+74-473-2244467", "frodo@shiremail.mea",
                 "Mount Doom", "19", "Mordor",
                 "+66-736-73666", "frodo@doomed.mea"),
                ("Mr.", "Gandalf", "the Grey",
                 "Wizard street", "42", "Valmar",
                 "+82-562-74263253", "gandalf@the-grey.mea",
                 "Bag End", "1", "Westfarthing",
                 "+74-473-4263253", "gandalf@work.mea"),
                ("Mr.", "Gandalf", "the White",
                 "Wizard street", "42", "Valmar",
                 "+82-562-74263253", "gandalf@the-white.mea",
                 "Bag End", "1", "Westfarthing",
                 "+74-473-4263253", "gandalf@work.mea")]
```

Diese Beispielliste besteht aus insgesamt vier Tupeln mit jeweils 13 Einträgen vom Typ `string`. Wenn Sie sich die Tupel als einzelne Zeilen einer Tabelle vorstellen, dann wird klar, dass das erste Tupel die Daten der Überschrift der 13 Spalten der Tabelle enthält. Jedes weitere Tupel (d.h. jede weitere Tabellenzeile) enthält dann einen Personen-bezogenen Eintrag.
Hinweise: Zur Lösung der Programmieraufgaben bietet es sich an, `for`-Schleifen mit `zip` und `enumerate` zu kombinieren.

(a) Machen Sie sich mit der Datenstruktur und den Daten in `address_list` vertraut, indem Sie eine Funktion `print_addr_list(addrlist)` schreiben. Diese Funktion bekommt eine Adressliste in der angegebenen Form als Argument übergeben und soll die darin enthaltenen Daten jeder Person zeilenweise in der Form `<typ>: <wert>` ausgeben, also in obigem Beispiel beginnend mit `salutation: Mr.` gefolgt von einer neuen Zeile mit `givenName: Frodo` usw.

(b) Schreiben Sie eine Funktion `addr_list_to_dict(addrlist)`, die eine Adressliste in obiger Form als Argument übergeben in ein geschachteltes Dictionary umwandelt und dieses mittels `return` zurückgibt. Legen Sie dazu zuerst ein leeres (äußeres) Dictionary an. Füllen Sie dieses dann schrittweise mit Einträgen, deren *Keys* fortlaufende Integer sind und deren *Values* wiederum Dictionaries sind. Diese werden im Folgenden als „innere“ Dictionaries bezeichnet. Jedes innere Dictionary enthält am Ende alle Daten genau einer Person als *Key-Value*-Paare.

Überprüfen Sie Ihre Funktion anhand selbst gewählter Testcases und mittels geeigneter `print`-Anweisungen. Was liefern Ihnen zum Beispiel folgende zwei Codezeilen als Ausgabe?

```
address_dict = addr_list_to_dict(address_list)
print(address_dict[1].get('givenName'))
```

(c) Nachdem Sie nun ein Dictionary erzeugt haben, fällt Ihnen auf, dass es vielleicht besser wäre, wenn das äußere Dictionary nicht durch Intergerwerte indiziert wäre, sondern mittels der jeweiligen Vornamen der Personen. Dann könnte man so auf die Daten einer Person zugreifen:

```
address_dict_new = convert_addr_dict_keys(address_dict)
print(address_dict_new['Frodo'].get('givenName'))
```

Implementieren Sie die Funktion `convert_addr_dict_keys` mit einem Parameter, die die inneren Einträge eines bereits in dictionary-Form vorliegenden Adressbuchs systematisch durchgeht. Ein neu zu generierendes (äußeres) Adressbuch soll dann als *Keys* die vorhandenen Vornamen (d.h., „givenName“-Werte) auf die jeweils richtigen Dictionaries zeigen lassen. Nutzen Sie dazu die Methode `update`, die auf Folie 11-15 eingeführt wurde.

(d) Wieso ist die Anzahl der Einträge in den beiden Dictionaries `address_dict` und `address_dict_new` unterschiedlich? Skizzieren Sie kurz eine Idee, wie Sie dieses Problem beheben könnten.

Aufgabe 5.2 (Invertieren eines Dictionary; 4 Punkte; `dicts.py`)

Definieren Sie eine Funktion `invert_dict(d)`, die angewendet auf ein Dictionary `d`, in dem jedem Schlüssel (= Key) ein unveränderliches, Hash-fähiges Objekt als Wert zugewiesen ist, ein neues Dictionary zurückgibt: in diesem werden die Werte in `d` als Schlüssel verwendet und jedem solchen Schlüssel `k` wird als Wert eine Liste jener Schlüssel von `d` zugewiesen, deren Wert (in `d`) `k` ist.

Testen Sie Ihre Funktion an mindestens vier geeigneten und selbst gewählten Beispielen.

Aufgabe 5.3 (`set` vs. `frozenset`; 4 Punkte; Datei: `notice.py`)

Wir nehmen an, dass wir einen Strom von Eingabeelementen analysieren wollen. Dabei wollen wir feststellen, wie viele verschiedene Elemente dieser Eingabestrom enthält und wie viele Elemente mindestens zweimal auftreten. Programmieren Sie dazu eine Funktion `notice` mit drei Parametern `el`, `newelements`, `oldelements`, wobei `el` ein beliebiges (Hash-fähiges) Element, `newelements` eine Menge von bereits vorher gesehenen Elementen und `oldelements` die Menge von bereits *mehr* als einmal gesehenen Elementen ist. Modifizieren Sie dazu die beiden Mengen entsprechend und geben dann als Rückgabewert der Funktion das Paar `(newelements, oldelements)` zurück. Benutzen Sie als Datentyp `set`.

Schreiben Sie eine weitere Funktion `notice_frozen`, die die gleiche Funktionalität wie `notice` besitzt, aber den Datentyp `frozenset` einsetzt.

Benutzen Sie folgende Funktion zum Testen der beiden Funktionen:

```
def test_notice(testfunc, testnew, testold, testlist):
    for el in testlist:
        testnew, testold = testfunc(el, testnew, testold)
    return len(testnew), len(testold)
```

Ein typischer Testaufruf sähe dann folgendermaßen aus:
`test_notice(notice, set(), set(), (1, 5, 3, 1))` bzw.
`test_notice(notice_frozen, frozenset(), frozenset(), (1, 5, 3, 1))`.

Testen Sie beide Funktionen auf folgenden Sequenzen:

(a) `(1, 5, 3, 1, 9, 3, 4, 1, 5)`

(b) `range(1000)`

(c) `range(100000)`

Welche Beobachtung machen Sie?

Aufgabe 5.4 (Mengen-Operationen; 2 + 1 + 1; Datei: `sets.py`)

(a) Definieren Sie eine Funktion `powerset(x)`, die angewendet auf eine Menge `x` die Menge aller Teilmengen von `x` zurückgibt.

(b) Definieren Sie eine Funktion `subsets_intersection(s, x)`, die angewendet auf zwei Mengen `s` und `x` die Menge jener Teilmengen von `s` zurückgibt, die mindestens ein gemeinsames Element mit `x` haben.

(c) Definieren Sie eine Funktion `subsets_card(s, k)`, die angewendet auf eine Menge `s` und eine natürliche Zahl `k`, die Menge jener Teilmengen von `s` zurückgibt, die genau `k` Elemente enthalten.

Verwenden Sie zur Lösung dieser Aufgaben keine Listen, sondern Mengen. Auch das Modul `itertools` sollen Sie zur Lösung nicht verwenden. Testen Sie Ihre Funktionen an jeweils mindestens vier geeigneten und selbst gewählten Beispielen.

Hinweis: Achten Sie darauf, dass die an diese Funktionen übergebenen Mengen nach Ausführung der Funktion noch dieselben Mengen sind!

Aufgabe 5.5 (Erfahrungen; 2 Punkte)

Legen Sie im Unterverzeichnis `sheet05` 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.).