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 6

Abgabe: Freitag, 6. Dezember 2013, 18:00 Uhr

Bei diesem Übungsblatt erwarten wir alle Lösungen zu den Aufgaben in der jeweils angegebenen Python-Datei im Unterverzeichnis `sheet06`. Beachten Sie dabei die bekannten Formatierungshinweise für Python-Dateien und vergessen Sie nicht, alle von Ihnen verwendeten Dateien ins SVN-Repository hochzuladen!

Versehen Sie die Funktionen in der Aufgabe 6.2 im Doc-String mit entsprechenden Tests, die mittels `doctest` geprüft werden können.

Aufgabe 6.1 (BF and PEP8, Punkte: 3+3, Datei: `bf_interpreter.py`)

In der Datei `bf_interpreter.py` zu diesem Übungsblatt finden Sie die Implementierung eines Interpreters für Brainf∗ck-Programme, der ähnlich in der Vorlesung vorgestellt wurde.

(a) Ergänzen Sie den Doc-String der in diesem Modul definierten Funktion `bfinterpret` um eigene, selbst gewählte Tests.

(b) Leider ist die in diesem Modul angegebene Implementierung von verschiedenen Leuten editiert worden, die ganz unterschiedlichen Programmier-Konventionen folgen. Modifizieren sie also den Programm-Code so, dass er in sich konsistent und insbesondere PEP8-konform wird (siehe `http://www.python.org/dev/peps/pep-0008/`). Auch die bereits bisher in den Übungen angeforderten Formatierungshinweise für Python-Dateien sollen eingehalten werden. Benutzen Sie dazu auch geeignete Style-Checker.

Hinweis: 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 nachinstalliert werden (z.B. mit `pip3 install flake8`).

Aufgabe 6.2 (Operieren mit Dateien und Ausnahmen; Punkte: 4+4+4)

Wir wollen im Folgenden ein Programm schreiben, das einen Verzeichnisbaum nach allen Vorkommnissen eines gesuchten Wortes in Dateien mit einer bestimmten Datei-Extension durchsucht und die Suchergebnisse auf der Konsole ausgibt. Dazu nehmen wir an, dass alle Dateien in dem Verzeichnisbaum Textdateien sind.

Wir wollen dieses Ziel in drei Schritten realisieren:

(a) Schreiben Sie zunächst ein Modul `find.py` mit folgender Funktionalität: beim Aufruf von `python3 find.py` in der Konsole wird vom Benutzer abgefragt, welcher Teilbaum des gegenwärtigen Arbeitsverzeichnisses nach Dateien mit welcher Datei-Extension durchsucht werden soll. Dies kann zum Beispiel wie folgt aussehen:

```
$ python3 find.py
Directory to search through [./]: test/
Files to search for [.txt]: .tex
```

– hinter den Doppelpunkten stehen hierbei die Benutzereingaben auf der Konsole. In den eckigen Klammern stehen ferner Default-Werte, die verwendet werden, falls der Benutzer keine eigene Eingabe macht, sondern nur die Return-Taste drückt. Nach der Benutzereingabe soll Ihr Programm den Verzeichnisbaum im angegebenen Pfad nach allen Dateien mit der angegeben Datei-Extension durchsuchen und die jeweiligen Dateinamen (mit relativem Pfad zum gegenwärtigen Arbeitsverzeichnis) auf der Konsole ausgeben. Die Ausgabe könnte dann also wie folgt aussehen:

```
test/blatt06.tex
test/exdirA/loremipsum.tex
test/exdirB/subdirC/loremipsum_long.tex
```

Falls sich keine solchen Dateien finden lassen, soll auf der Konsole ein Hinweis ausgegeben werden, wie etwa:

```
Nothing found.
```

Zur Implementierung: Verwenden Sie den Python-Befehl `input`, um die Benutzer-Abfragen auf der Konsole durchzuführen. Definieren Sie ferner in diesem Modul eine Funktion `find_files` mit einem positionalen Argument `path` für den durchsuchten Teilbaum und einem *Keyword*-Argument `extension` für die Datei-Extension: die Funktion soll dann das angegebene Verzeichnis durchlaufen und die Dateinamen der gefundenen Dateien ausgeben. Hilfreich sind hierbei die Funktionen, die vom Modul `os.path` bereit gestellt werden. Zum Testen müssen Sie natürlich einen eigenen kleinen Verzeichnisbaum im Verzeichnis `sheet06` anlegen.

(b) Während wir in (a) angenommen haben, dass der Benutzer stets „vernünfige“, erwartete Eingaben tätigt, wollen wir nun auch nicht erwartete Eingaben, also Ausnahmen, behandeln. Wenn der Benutzer zum Beispiel anstelle eines Verzeichnisnamens den Pfad zu einer Datei eingegeben hat, so soll sich das Programm kontrolliert (*ohne Fehlermeldung* auf der Konsole) beenden. Gibt der Benutzer also beispielsweise `ReadMe.txt` als Verzeichnisnamen ein (und ist dieser Pfad kein Verzeichnis), so soll Folgendes auf der Konsole ausgegeben werden:

```
ReadMe.txt is not a directory.
```

Weitere mögliche Ausnahmen können entstehen, wenn das eingegebene Verzeichnis nicht existiert oder aber der Benutzer die Ausführung des Programms abbricht (z.B. mit `<CTRL>-C`).

Erweitern Sie dazu Ihre Implementierung aus Aufgabe 6.2(a) um eine geeignete Behandlung dieser Ausnahmen.

(c) Schreiben Sie ein Modul `grep.py`, das die Funktionalität aus 6.2(b) erweitert. Vom Benutzer soll nun zusätzlich ein Wort abgefragt werden, nach dem in den Dateien mit der eingegebenen Extension gesucht werden soll. Ein Suchergebnis besteht dann aus einem Dateinamen und der Zeilennummer, in der das Wort in der Datei vorkommt. Jedes solche Suchergebnis soll auf der Konsole ausgegeben werden (Pfadnamen relativ zum gegenwärtigen Arbeitsverzeichnis), z.B.:

```
test/exdirA/loremipsum.tex: 5
test/exdirA/loremipsum.tex: 8
test/exdirB/subdirC/loremipsum_long.tex: 5
<viele Ausgaben>
test/exdirB/subdirD/loremipsum_long.tex: 101
```

Auch hier sollen vernünftige Ausgaben auf der Konsole erfolgen, wenn der angegebene Pfad nicht existiert oder keine Suchergebnisse vorliegen. In Gegensatz zu 6.2(b) ist es nun aber sinnvoll, als Suchpfade auch einzelne Datei zuzulassen.

Zur Implementierung: Definieren Sie in diesem Modul eine Funktion `grep_word` analog zur Funktion `find_files` aus 6.2(a) mit einem weiteren positionalen Argument `word` für ein Wort, nach dem in den gefundenen Dateien gesucht werden soll. Falls der Benutzer kein Wort eingibt, soll die Funktionalität aus 6.2(a) erzielt werden.

Aufgabe 6.3 (Erfahrungen; 2 Punkte)

Legen Sie im Unterverzeichnis `sheet06` 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.).