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 6

Abgabe: Freitag, 05. Dezember 2014, 18:00 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 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 6.1 (Brainf*ck; Datei: oct2dec.b; Punkte: 2+2+4)

In dieser Teilaufgabe sollen Sie ein Programm in Brainfuck (BF; `http://esolangs.org/wiki/Brainfuck` und `http://www.iamcal.com/misc/bf_debug/` für einen Interpreter mit Debugger) schreiben, welches eine beliebige zweistellige Oktalzahl als Eingabe akzeptiert und in Dezimaldarstellung umrechnet und ausgibt. Zum Beispiel müsste Ihr Programm für die Eingabe `31` die Ausgabe `25` liefern. Zur Vereinfachung dürfen Sie davon ausgehen, dass die Eingabe immer zweistellig ist und tatsächlich eine Oktalzahl repräsentiert.
Wir bitten Sie (1) ihr Programm in Blöcke anhand der Teilaufgaben aufzuteilen und (2) Ihr gesamtes Programm gut zu dokumentieren.
Nun zu den konkreten Teilaufgaben.

(a) Einlesen einer zweistelligen Oktalzahl: Zuerst soll Ihr Programm die Eingabe mittels des Befehlszeichens "," einlesen und je Stelle in eine eigene Zelle auf dem Band schreiben. Gibt der Benutzer zum Beispiel `31` ein, so soll diese Zahl wie folgt auf dem Band repräsentiert werden:

`[..,3,1,..]`

und der Zeiger soll am Ende auf die "Achterstelle" (hier also die Zelle mit der Zahl `3`) zeigen.

(b) Erweitern Sie das Programm nun so, dass der Zeiger am Ende auf eine Zelle zeigt, in der die äquivalente Dezimalzahl steht. Zum Beispiel für die Eingabe der Zahl `31` sollte auf dem Band an beliebiger Stelle [..,`25`,..] erscheinen. Ihr Programm sollte natürlich jede beliebige Eingabe einer höchstens zweistelligen Oktalzahl entsprechend umwandeln können.

(c) Wir gehen davon aus, dass der Zeiger auf eine Zelle mit einer Dezimalzahl zeigt, wie in Aufgabenteil (b) realisiert. Geben Sie die beiden Ziffern dieser Zahl einzeln unter Verwendung des Befehls "." so an den Benutzer aus, dass dieser das Ergebnis ablesen kann.

Testen Sie ihr Programm unter Verwendung des hier verfügbaren Debuggers `http://www.iamcal.com/misc/bf_debug/` ausgiebig mit verschiedenen Eingaben.

Aufgabe 6.2 (Kryptologie I; Datei: caesar.py; Punkte: 5)

Wir befassen uns mit der Implementation kryptografischer Verschlüsselungsalgorithmen, siehe dazu auch die Wikipedia-Seite zum Thema "Polyalphabetische Substitution"[1]. Laden Sie bitte die Datei `caesar.py` mittels folgendem URL herunter und speichern Sie sie lokal ab:
`http://gki.informatik.uni-freiburg.de/teaching/ws1415/info1/python/caesar.py`

In dieser Datei wird die Funktion `caesar(text, shift, charset=ascii_letters)` definiert, deren docstring Sie entnehmen können, dass sie auf dem Eingabetext eine Cäsarverschiebung vornimmt. Das Argument `shift` gibt dabei an, um wieviele Stellen all jene Zeichen verschoben werden, die im Zeichensatz `charset` vorkommen. Alle anderen Zeichen bleiben in der Rückgabezeichenfolge unverändert erhalten.
Modifizieren Sie den Programmcode unterhalb der Zeile, die mit `if __name__` beginnt so, dass:

1. der Eingabetext aus einer Datei eingelesen und an die Funktion `caesar` übergeben wird,
2. der Dateiname der einzulesenden Datei sowie der Parameter `shift` per Kommandozeilenargumente dem Programm übergeben werden müssen und
3. das verschlüsselte Resultat in eine Datei geschrieben wird, deren Name sich aus dem Eingabedateinamen plus der Endung `.enc` ergibt.

Zum Beispiel schreibt folgender Aufruf:

```
python caesar.py text.txt 3
```

den Inhalt der Eingabedatei `text.txt` um drei Zeichen verschoben in die neu zu erzeugende oder, falls vorhanden, zu überschreibende Ausgabedatei `text.txt.enc`.

Behandeln Sie Fehler (wie z.B. ungenügende Anzahl von Kommandozeilenparametern oder nicht vorhandene Eingabdatei) auf für den Benutzer sinnvolle Weise.

[1] `http://de.wikipedia.org/wiki/Polyalphabetische_Substitution`

Aufgabe 6.3 (Kryptologie II; Datei: vigenere.py; Punkte: 5)

Beachten Sie die allgemeinen Hinweise zum Thema Kryptologie aus dem vorherigen Aufgabenteil. Laden Sie dann bitte die Datei `vigenere.py` mittels folgendem URL herunter und speichern Sie sie lokal ab:
`http://gki.informatik.uni-freiburg.de/teaching/ws1415/info1/python/vigenere.py`
Ergänzen Sie nun die darin definierte Funktion

```
vigenere(text, secret, charset=ascii_letters)
```

so, dass diese den Eingabetext `text` mittels des geheimen Wortes `secret` anhand des Vigenère-Verfahrens verschlüsselt und als Zeichenkette zurückgibt. Folgende Eigenschaften muss Ihre Implementierung erfüllen:

1. Alle Zeichen des Eingabetextes, die nicht im Zeichensatz `charset` vorkommen, werden unverändert in die Rückgabezeichenkette übernommen. Dabei wird trotzdem ein Zeichen des geheimen Wortes verbraucht.
2. Falls das geheime Wort Zeichen enthält, die nicht im Zeichensatz `charset` enthalten sind, wird `None` zurückgegeben.

Schreiben Sie eine oder mehrere Testfunktionen, die die Funktionalität von `vigenere` mit insgesamt 4 Aufrufen der Funktion testen.

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.).