Informatik I: Einführung in die Programmierung

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

Universität Freiburg
Institut für Informatik

Übungsblatt 7

Abgabe: Freitag, 11. Dezember 2015, 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/ws1516/info1/wiki/hinweise.html`

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

Aufgabe 7.1 (Brainf*ck; Dateien: `bf.py`, `spam.b`; Punkte: 4+4)

In dieser Aufgabe soll die Sprache Brainf*ck um ein zusätzliches Sprachelement erweitert werden. Wir wollen eine einfache Möglichkeit zur bedingten Ausführung von Code realisieren, nämlich eine Verzweigung ohne *else*-Zweig. Dafür sollen die neuen Zeichen `{` und `}` eingeführt werden. Idee dabei ist, dass der Code in den geschweiften Klammern ausgeführt wird, *falls* bei Betreten der Verzweigung der Wert in der aktuellen Zelle gleich 0 ist; andernfalls wird der geklammerte Code übersprungen.
Genauer besitzen die beiden neuen Zeichen folgende Bedeutung:

- `{`: Falls der Wert in der aktuellen Zelle gleich 0 ist (`data[ptr] == 0`), dann setze die Ausführung mit dem Befehl nach der öffnenden Klammer fort. Andernfalls, springe zum Befehl nach der zugehörigen schließenden Klammer.
- `}`: Fahre mit dem Befehl nach der Klammer fort.

(a) Laden Sie den Brainf*ck-Interpreter von der Webseite der Vorlesung herunter und erweitern Sie diesen um das eben beschriebene Sprachelement.

(b) Schreiben Sie ein Brainf*ck-Programm, das für einen beliebigen Eingabestring überprüft, ob dieser mit der Buchstabenfolge `SPAM` (in Großbuchstaben) beginnt. Ist das der Fall, soll das Programm `OK` ausgeben. Andernfalls soll nichts (außer der Mitteilung, dass die Ausführung beendet wurde) ausgegeben werden. Greifen Sie hierbei auf das zuvor implementierte Sprachelement zurück.

Aufgabe 7.2 (Kryptoanalyse; Dateien: `caesar.py`, `crack_caesar.py`; Punkte: 3+7)

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/ws1516/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.

(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
```

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

Aufgabe 7.3 (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, Bezug zur Vorlesung, Interessantes, etc.).