Stefan Kurtz **Universität Hamburg**
Emanuel Ehmki **Zentrum für Bioinformatik**

Programmierung für Naturwissenschaften 1
Wintersemester 2019/2020
Übungen zur Vorlesung: Ausgabe am 04.12.2019

... und immer wieder am Mittwoch vergügbar:
`https://feedback.informatik.uni-hamburg.de/PfN1/wise2019-2020`

Aufgabe 7.1 (2 Punkte)

Aus Zeitgründen können wir in diesem Jahr in der Vorlesung das Thema Codon-Translation (siehe Folien Seite 333-339) nicht behandeln. Aus der Schule kennen Sie bereits die biologischen Grundlagen und in unserem Modul haben Sie die notwendigen Python-Kenntnisse erworben, um das Thema zu verstehen. Für diese Übungsaufgabe müssen sich einige Studierende vor der Übung anhand der Vorlesungsfolien auf dieses Thema vorbereiten und das erworbene Wissen in Kleingruppen in den ersten 30 Minuten der Übung an die anderen Studiererenden weitergeben.

Die Kleingruppen setzen sich wie folgt zusammen:

1. Molkentin, Myronovych, Hauschild
2. Schenk, Paulsen, Jegminat, Breiholz
3. Jochens, Franke, Flotow, Schuett
4. Stahl, Loewenberg, Kaemmle, Jakobi
5. Grosse, Ehlers, Podolskiy
6. Plesch, Kniep, Moeller, LFranke
7. Gruetzmacher, Eckmann, Harkov, Music
8. Leege, Scheele, Biegemann, Gubernator
9. David, Nabil, Dao, Quante
10. Gruber-Roet, Lehmann, Tang
11. Carlsen, Kuhn, Breker
12. Ebbing, Scheu, Klemm, Lohmann
13. Rahlf, Lindemann, Paffenholz, Liessmann
14. Pluemer, Kaether, Radtke, Witte
15. Froechtling, Fender, Block

Die Namen der Studierenden, die sich auf das Thema vorbereiten müssen, sind jeweils am Beginn jeder Zeile aufgeführt. Falls jemand von diesen Studierenden nicht zur Übung erscheint, verteilen sich die übrigen Mitglieder der Kleingruppe auf die anderen Gruppen.

Diese Aufgabe soll am Anfang der Übung bearbeitet werden. Dazu setzen sich jeweils die einzelnen Kleingruppen an einen Rechner und erarbeiten gemeinsam die Folien zum Thema Codon-Translation. Der oder die Studierende, die sich vorbereitet hat, kann ggf. bei Unklarheiten Fragen, zumindest zu den Grundbegriffen, beantworten oder Erläuterungen mündlich ergänzen. Falls Sie die in den Folien dargestellte auf einen dictionary basierte Implementierung ausprobieren möchten, können Sie auf den Programmcode aus den Materialien zurückgreifen.

Nach der Übung dokumentiert jede Kleingruppe in einer E-mail an `kurtz@zbh.uni-hamburg.de` das Vorgehen bei der Erarbeitung des Themas und ggf. noch bestehende Verständnisfragen oder Hinweise zu Unklarheiten in den Folien. Willkommen sind natürlich auch Bemerkungen zur Lehrform selbst. Dabei soll nicht der Inhalt der Folien repliziert werden. Die E-mail soll die folgenden Eigenschaften haben:

- abgesendet bis zum Abgabetermin der entsprechenden Übung,
- maximal 15 Zeilen mit maximal 80 Zeichen pro Zeile,
- Angabe der Nachnamen aller Mitglieder der Kleingruppe, die teilgenommen haben (alle genannten Personen erhalten die zwei Punkte),
- keine Anhänge.

Die E-mail soll von einer/einem Studierenden erstellt werden, die/der sich nicht auf das Thema vorbereitet hat.

Aufgabe 7.2 (3 Punkte) Ein Labormediziner möchte zukünftig Laborbefunde per Email an die beauftragenden Ärzte verschicken. Sie arbeiten bei dem Labormediziner als studentische Hilfskraft und sollen das Vorhaben technisch umsetzen.

Als Ergebnis einer Laboruntersuchung werden Dateien generiert, die die Daten in einem Tabulator-separierten Format enthalten. Hier ist ein Beispiel mit dem Inhalt einer Datei `patient1.tsv`:

```
_EMAILADDRESSE_ mabuse@yahoo.com
_ANSPRACHE_ geehrter Herr Dr.
_NAME_ Mabuse
_PATIENT_ Ihres Patienten Hans Mueller
_MESSUNG_ Anzahl der Leukozyten (weiße Blutkörperchen)
_WERT_ 2000/mikro l
_BEFUND_ zu niedrig
```

Jede Zeile enthält genau einen Variablen-Bezeichner und einen String, der den Wert darstellt.

In einer weiteren Datei `email_template.txt` steht der Text einer Email, in der an den variablen Stellen die genannten Bezeichner stehen:

```
To: _EMAILADDRESSE_
From: labormedizin@hamburg.de

Sehr _ANSPRACHE_ _NAME_,

Sie hatten uns Proben _PATIENT_ geschickt.
Die Untersuchungsergebnisse liegen nun vor.
Die _MESSUNG_ ist mit _WERT_ _BEFUND_.

Mit freundlichen Grüßen,

Ihr Labormediziner
```

Entwickeln Sie ein Python-Programm `gen_email.py`, das beim Aufruf Dateinamen erhält: den Namen einer Datei mit dem Text und den Namen einer Datei mit den Daten einer Laboruntersuchung. Das Programm soll an allen Stellen, an denen im Text ein Variablen-Bezeichner steht, diesen durch den entsprechenden String aus der zweiten Datei ersetzen und den entsprechenden Text zeilenweise ausgeben. Beispiel: der Aufruf von

`./gen_email.py email_template.tsv patient1.tsv`

soll den folgenden Text im Terminal ausgeben:

```
To: mabuse@yahoo.com
From: labormedizin@hamburg.de

Sehr geehrter Herr Dr. Mabuse,

Sie hatten uns Proben Ihres Patienten Hans Mueller geschickt.
Die Untersuchungsergebnisse liegen nun vor.
Die Anzahl der Leukozyten (weiße Blutkörperchen) ist mit 2000/mikro l zu niedrig.

Mit freundlichen Grüßen,

Ihr Labormediziner
```

Das Programm soll beide Dateien nur einmal lesen. Sie müssen also Informationen aus der ersten Datei mit den Variablen-Bezeichnern und den Strings in einer geeigneten Form abspeichern, um beim Lesen der Datei die Substitutionen anzuwenden.

In den Materialien finden Sie ein Makefile mit Beispielaufrufen, Dateien mit Daten und die erwarteten Ausgaben. Durch `make test` verifizieren Sie die Korrektheit Ihres Programms.

Aufgabe 7.3 (4 Punkte) In der Datei `redundant.py` finden Sie ein Python-Programm mit vielen Redundanzen und Anweisungen, die in Python einfacher ausgedrückt werden können. Implementieren Sie in einer Datei `structured.py` eine strukturierte Version des Programms durch Deklaration einer Funktion `compare` mit folgenden Eigenschaften:

- Durch die Funktion sollen möglichst viele Redundanzen im Programmcode der Datei `redundant.py` vermieden werden.
- Die Berechnung der Werte in den Variablen mit dem Suffix `_m` soll mit Hilfe von existierenden Python-Funktionen erfolgen. Dazu müssen Sie sich überlegen, welche Werte in diesen Variablen gespeichert werden.

Ihr Programm `structured.py` soll durch drei Aufrufe der Funktion `compare` mit den passenden Argumenten das gleiche Ergebnis liefern wie `redundant.py`. Das verifizieren Sie durch `make test`. Es empfiehlt sich, den Programmcode für die Funktion schrittweise zu entwickeln, ggf. Zwischenergebnisse auszugeben und diese mit den entsprechenden Zwischenergebnissen aus `redundant.py` zu vergleichen. Sie dürfen in Ihrem Programm nur die Klasse `math` importieren.

Die Musterlösung umfasst nur 24 Zeilen Programmcode, einschließlich der Definition der drei Listen am Anfang. Ihre Lösung sollte nicht mehr als 30 Zeilen umfassen. Bitte beachten Sie, dass keine Zeile des Programms mehr als 80 Zeichen enthalten darf. Diese Eigenschaft können Sie mit Hilfe von `pycheck.py` verifizieren.

Aufgabe 7.4 (1 Punkt) In dieser Aufgabe geht es um die Entwicklung eines Argument-Parsers für eine spätere Aufgabe. Benutzen Sie, wie in der Vorlesung gezeigt, das Python3-Modul `argparse` zur Implementierung einer Funktion `cigarXevalargparse()`. Diese Funktion liefert ein Objekt zurück, dass die gewählten Optionen repräsentiert, und erlaubt Werte von Optionen zu erfragen. Hier ist die Ausgabe der Help-Option, aus der sich die anderen Optionen und Ihre Bedeutung ergeben. Dabei ist `cigarXevalargparse.py` der Name des Hauptprogramms.

```
usage: cigarXevalargparse.py [-h] [-c] [-i] inputfile
```

```
positional arguments:
  inputfile       specify input file

optional arguments:
  -h, --help      show this help message and exit
  -c, --cost      show cost for each cigarXstring
  -i, --identity  show identity for each cigarXstring
```

Die drei Optionen `-h`, `-c` und `-i` sind boolsche Optionen. Das verpflichtende Positionsargument ist `inputfile` und speichert einen String. In den Materialien zu dieser Aufgabe finden Sie eine Datei `cigarXevalargparse_template.py`, die Sie bitte in `cigarXevalargparse.py` umbenennen und um die obige Funktion ergänzen. Durch `make test` verifizieren Sie die Korrektheit Ihrer Funktion.

Bitte die Lösungen zu diesen Aufgaben bis zum 09.12.2019 um 18:00 Uhr an pfn1@zbh.uni-hamburg.de schicken. Die Besprechung der Lösungen erfolgt am 11.12.2019.

Hinweise:

- Die erste Klausur findet einen Tag früher statt als ursprünglich geplant. Die zweite Klausur wurde in einen anderen Hörsaal verlegt:
  - 17. Februar 2020, 9:30–11:00 Uhr, ESA A
  - 17. März 2020, 9:30–11:00 Uhr, Hörsaal C Chemie
- Am Ende des Semesters erhalten Sie eine Liste mit Beispielfragen zur gezielten Vorbereitung auf die Klausur.