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

Programmierung in der Bioinformatik
Wintersemester 2019/2020
Übungen zur Vorlesung: Ausgabe am 06.11.2019

Aufgabe 3.1 (1 Punkt) Wenn man unter Linux/macOS im Terminal Dateien löscht, kopiert oder ihre Namen ändert, dann muss man sehr vorsichtig mit den entsprechenden Kommandos `rm`, `cp` und `mv` umgehen, damit man nicht versehentlich Dateien löscht. Daher bieten alle drei Kommandos eine Option, die dazu führt, dass vor dem Löschen oder Überschreiben einer Datei der Benutzer diese Aktion mit `y` bestätigen muss. Finden Sie heraus, um welche Option es sich handelt. Mit Hilfe des Kommandos (*bash*-syntax):

```
alias rm='rm <fragebeimloeschenoption>'
```

erreicht man, dass für `rm` diese Option automatisch verwendet wird. Tragen Sie für alle drei Kommandos entsprechende `alias`-Einträge in die Datei `~/.bashrc` ein. Falls diese noch nicht existiert, legen Sie sie neu an. Nach dem Sichern der Datei und dem Befehl

```
. ~/.bashrc
```

bzw. nach jedem Einloggen sind die `alias`-Einträge dann aktiv.

Als Lösung zu dieser Aufgabe geben Sie bitte in einer Datei `interactive.sh` die drei alias Definitionen an, die nun in Ihrer Datei `.bashrc` stehen.

Aufgabe 3.2 (2 Punkte) Die 20 Aminosäuren lassen sich in zwei Gruppen einteilen, nämlich in die Gruppe der hydrophoben und in die Gruppe der hydrophilen Aminosäuren, wie in der folgenden Tabelle dargestellt (in der wir den Einbuchstaben-Code verwenden):

hydrophobe Aminosäuren	hydrophile Aminosäuren
A, F, I, L, M, P, V, W	C, D, E, G, H, K, N, Q, R, S, T, Y

Schreiben Sie ein Python-Script `hydrocount.py`, das über `sys.argv` (siehe Vorlesungsfolien) genau ein String-Argument erhält. Dieses String-Argument enthält beliebig viele Einbuchstaben-Codes von Aminosäuren. Das Programm soll zählen, wieviele hydrophobe und wieviele hydrophile Aminosäuren im String vorkommen. Zeichen im Eingabe-String, die keine Aminosäuren bezeichnen, können ignoriert werden.

Ihr Programm soll als Ergebnis eine Zeile der Form

```
hydrophob=x, hydrophil=y
```

wobei `x` die Anzahl der hydrophoben und `y` die Anzahl der hydrophilen Aminosäuren ist. Beispielsweise soll

```
$ ./hydrocount.py ISDKDLYVAALTNADLN
```

die Ausgabe

```
hydrophobic=8, hydrophilic=9
```

liefern. Um das Programm robust zu machen und dem Benutzer einen Hinweis zu geben, wie es verwendet wird, testen Sie bitte mit einer `if`-Anweisung am Anfang des Programms, ob die Liste `sys.argv` genau zwei Elemente enthält. Wenn das nicht der Fall ist, dann soll eine Fehlermeldung der Form

```
Usage: ./hydrocount.py <aminoacid sequence>
```

mit `sys.stderr.write` auf den Standard-Fehlerkanal ausgegeben werden. Die Funktion `write` kann dabei so wie `print` verwendet werden. Allerdings muss am Ende noch ein \n eingefügt werden, damit nach der Fehlermeldung ein Zeilenvorschub erfolgt. Auf den Namen und Pfad des Programms kann mit `sys.argv[0]` zugegriffen werden.

Wenn `sys.argv` genau zwei Elemente enthält, dann können Sie mit einer Schleife der Form

```
for aa in sys.argv[1]:
```

auf die einzelnen Zeichen des Eingabestrings zugreifen. Die Zeichen sind dann nacheinander in der Variablen `aa` gespeichert und können in einem Block, der gegenüber der obigen `for`-Schleife eingerückt ist, verarbeitet werden. Zum Zählen können Sie zwei Variablen verwenden, die Sie vor der obigen `for`-Schleife mit 0 initialisieren müssen. Sie müssen dann jeweils mit einer `if` bzw. `elif`-Fallunterscheidung entscheiden, ob die aktuell betrachtete Aminosäuren zur einen oder zur anderen Klasse gehört, und die entsprechenden Zähler mit `+= 1` erhöhen.

Nach der `for`-Schleife müssen die Werte der Zähler ausgegeben werden.

Im Material zu dieser Übungsaufgabe gibt es eine Datei mit Proteinsequenzen und den dafür erwarteten Ergebnissen. Durch `make test` verifizieren Sie, dass Ihr Programm korrekt funktioniert.

Aufgabe 3.3 (2 Punkte) In der Vorlesung haben Sie `while`- und `for`-Schleifen kennengelernt.

In den Materialien zur Übung finden Sie eine Datei `loops.original.py`. Diese enthält 2 `for`-Schleifen und 2 `while`-Schleifen. Erstellen Sie eine Kopie der Datei mit dem Namen `loops.py`. Ersetzen Sie darin alle `for`-Schleifen durch `while`-Schleifen und umgekehrt.

In den Materialien zur Übung finden Sie eine Testdatei und ein `Makefile`. Darin ist ein Test implementiert, der Ihr Programm aufruft und das Ergebnis mit Hilfe des Linux-Tools `diff` mit dem erwarteten Ergebnis vergleicht. Diesen Test können Sie mit dem Befehl `make test` in der Linux Shell ausführen (Achtung: Ein erfolgreicher Test bedeutet nicht unbedingt, dass die Aufgabe korrekt gelöst ist. Die Aufgabe gilt selbstverständlich erst dann als gelöst, wenn die Umwandlung bzgl. der Form der Schleifen auch durchgeführt wurde.)

Aufgabe 3.4 (5 Punkte)

In dieser Aufgabe geht es darum, Daten zu CO2-Emissionen verschiedener Industrieländer zu plotten. In der Datei `data.tsv` findet man die Emissionswerte für jedes Land in einer Zeile. Die Zeile beginnt mit dem Bezeichner des Landes, danach folgen jeweils 10 Emissionswerte (als Fließkommazahlen) in der Einheit Tonnen. Die Reihenfolge der Werte entspricht der Reihenfolge in der Liste `jahre`, die in der Datei `co2_simple_plot_template.py` definiert ist. Benennen Sie diese Datei in `co2_simple_plot.py` um und ergänzen Sie Python-Anweisungen für die folgenden Schritte:

1. Erzeugen zweier leerer Listen `laender` und `data_lists`.
2. Öffnen der Datei mit dem Namen `data.tsv` in einem `try/except`-Block und Zuweisung an eine Variable `stream`.
3. Zeilenweises Einlesen der geöffneten Datei `data.tsv` durch Zugriff auf die Variable `stream`.
4. Extraktion einer Liste `str_list` von Strings aus der aktuellen Zeile `line`. Hierfür wird auf `line` die Methode `rstrip()` gefolgt von `split()` angewendet. Benutzen Sie dazu jeweils den Operator `.`, auf dessen rechter Seite die anzuwendende Operation steht.
5. Anhängen des ersten Strings aus `str_list` an die Liste `laender`.
6. Erzeugen einer leeren Liste `float_list`. Aus allen folgenden Strings `s` aus `str_list` werden mit `float(s)` Fließkommazahlen erzeugt, die jeweils an `float_list` angehängt werden. Nach dem letzten gelesenen Wert wird `float_list` an `data_lists` angehängt. `data_lists` ist also eine Liste von Listen von Fließkommazahlen.
7. Schließen von `stream`.
8. Erzeugen von Objekten `fig` und `ax` mit `plt.subplots`.
9. Spezifikation des Titels und der Markierungen der X- und Y-Achse entsprechend der Vorgaben in `CO2_plot_reference.pdf`.
10. Spezifikation der Jahreszahlen auf der X-Achse durch `ax.set_xticks(jahre)`.
11. Für das i-te Land Aufruf der Methode `ax.plot()` mit drei Argumenten:
    - die Liste `jahre` (enthält die Werte auf der X-Achse)
    - die Liste `data_lists[i]` (enthält die Werte auf der Y-Achse für das i-te Land)
    - die Markierung `label=laender[i]` zur Festlegung der Markierung in der Legende
12. die Werte für i erhalten Sie durch eine `for`-Schleife mit der Funktion `range`.
13. Speichern des Plots in einer Datei `CO2_plot.pdf`.

Durch `make test` verifizieren Sie, dass Ihr Programm ohne Fehler terminiert und eine Datei mit dem geforderten Namen entsteht. Sie müssen natürlich selbst überprüfen, ob Ihr Plot dem Plot in der Datei `CO2_plot_reference.pdf` entspricht.

Bitte die Lösungen zu diesen Aufgaben bis zum 11.11.2019 um 18:00 Uhr an pfn1@zbh.uni-hamburg.de schicken. Die Besprechung der Lösungen erfolgt am 13.11.2019.