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

Programmierung in der Bioinformatik
Wintersemester 2019/2020
Übungen zur Vorlesung: Ausgabe am 13.11.2019

Hinweise zur Nutzung der Umgebungsvariable `PATH`:

Wenn man unter Unix ein Programm `p` ausführen möchte, wird der Installationsort (Pfad) anhand der Umgebungsvariablen `PATH` ermittelt. Es werden alle in der Umgebungsvariablen `PATH` gelisteten Verzeichnisse (Ausgabe mittels `echo $PATH`) durchsucht. Sei `d` der erste Pfad, der eine ausführbare Version des Programms `p` enthält. Dann wird das Programm `p` im Verzeichnis `d`, notiert durch `d/p` ausgeführt. Wenn man ein Programm aufrufen möchte, das in keinem der in `PATH` spezifizierten Pfade vorhanden ist, so muss der entsprechende Pfad dem Programm vorangestellt werden.

Durch

```
export <Variable>=<Wert>
```

lassen sich in der `bash` Werte von Umgebungsvariablen wie `PATH` festlegen. Nutzen Sie `export` um in der Datei `.bashrc` die Pfadliste in `PATH` zu erweitern:

```
export PATH=${PATH}:${HOME}/pfn1_2019/bin
```

Das setzt vorraus, dass das geklonte Repository in Ihrem `HOME`-Verzeichnis liegt. Falls das nicht der Fall ist, muss nach dem Doppelpunkt der Pfad entsprechend angepasst werden.

Nachdem Sie obige Zeilen in `.bashrc` eingetragen haben, muss noch im Terminal

```
. ~/.bashrc
```

ausgeführt werden.

Beim Erweitern der Variablen `PATH` ist zu beachten, dass die bereits in `PATH` definierten Pfade erhalten bleiben. Beachten Sie die Verwendung des Paars von geschweiften Klammern.

Die Reihenfolge der Verzeichnisse in der Variablen `PATH` ist dabei wichtig, wenn es mehrere Verzeichnisse mit gleichen Programmnamen gibt: nur das erste Verzeichnis in der Pfadliste, das das Programm enthält, wird berücksichtigt.

Aufgabe 4.1 (5 Punkte) Schreiben Sie ein Python-Skript `zahlenreihen.py`, das für eine positive ganze Zahl k die folgenden Zahlenreihen und jeweils ihre Summe ausgibt:

a) $2, 4, 6, \cdots, 2k$

b) $\frac{5}{2}, \frac{7}{4}, \frac{9}{8}, \frac{11}{16}, \ldots, \frac{3+2k}{2^k}$

c) $1, -\frac{1}{2}, \frac{1}{4}, -\frac{1}{8}, \cdots, \frac{(-1)^{k-1}}{2^{k-1}}$

d) $\frac{1}{1!}, \frac{1}{2!}, \frac{1}{3!}, \frac{1}{4!}, \ldots, \frac{1}{k!}$

Hinweise:

- Für die Berechnung der Zahlenreihen sollen `for`-Schleifen und die Methode `range` verwendet werden.
- Es sollen jeweils nur die Grundrechenarten `+`, `*`, `-` und `/` verwendet werden, jedoch nicht der Operator `**` und keine Funktion zur Berechnung der Fakultät.
- Die Übergabe von k erfolgt über die Kommandozeile, d.h. über die Liste `sys.argv`. Sie müssen daher überprüfen, ob `sys.argv` die passende Länge 2 hat. Ist das nicht der Fall, dann soll die Zeile

  ```
  Usage: ./zahlenreihen.py <k>
  ```

  ausgegeben werden.
- Die Kommandozeilenparameter in `sys.argv` sind Strings. Daher muss der String in `sys.argv[1]` mit der Methode `int()` in eine ganze Zahl umgewandelt werden. Damit das Programm bei ungültigen Eingaben mit einer verständlichen Fehlermeldung abbricht, muss mit `try/except` eine Ausnahmebehandlung erfolgen, die in etwa so aussehen kann:

  ```
  try:
    k = int(sys.argv[1])
  except ValueError as err:
    sys.stderr.write(formatstring.format(sys.argv[0],sys.argv[1]))
    exit(1)
  ```

  Der Formatstring soll so gebildet werden, dass beim Aufruf von `./zahlenreihen.py abc` die folgende Fehlermeldung ausgegeben wird:

  ```
  ./zahlenreihen.py: cannot convert 'abc' to int
  ```
- Falls der String in eine Zahl k konvertiert wurde, muss noch geprüft werden, dass es es sich bei k um eine positive Zahl handelt. D.h. bei Eingabe von einem Wert ≤ 0 soll eine Fehlermeldung erzeugt. Z.B. soll beim Aufruf `./zahlenreihen.py -3` die Fehlermeldung

  ```
  ./zahlenreihen.py: parameter -3 is not positive int
  ```

  ausgegeben werden.
- Fehlermeldungen werden nach `sys.stderr` ausgegeben und führen zu einem Abbruch des Programms mit `exit(1)`.
- In den Materialien finden Sie die erwartetete Ausgabe des Programms für $k = 10$. Verifizieren Sie durch `make test` die Korrektheit Ihres Programms (inklusive der fehlerhaften Aufrufe entsprechend der obigen Spezifikation).

Aufgabe 4.2 (3 Punkte) Die Quersumme einer ganzen Zahl ist die Summe der Ziffern, aus der diese Zahl besteht. Schreiben Sie ein Python-Skript `quersumme.py`, das die Quersumme einer beliebigen negativen oder positiven ganzen Zahl berechnet und ausgibt. Einer positiven ganzen Zahl kann optional das Zeichen `+` vorangestellt werden. Sie können vorraussetzen, dass das Zeichen `+` und `-` (falls es verwendet wird) jeweils direkt vor den Ziffern vorkommt.

Die Zahl soll als String auf der Kommandozeile übergeben werden und dieser String enthält möglicherweise am linken und rechten Rand Leerzeichen. Für den Fall, dass das Skript nicht mit der korrekten Anzahl von Argumenten aufgerufen wird, soll die Fehlermeldung

```
Usage: ./quersumme.py <integer>
```

auf `sys.stderr` ausgegeben werden.

Überprüfen Sie mit Hilfe eines regulären Ausdrucks die Eingabe auf Korrektheit und geben Sie eine Fehlermeldung auf `sys.stderr` aus, falls das nicht so ist. Der Aufruf `./quersumme.py 1.5` soll die Fehlermeldung

```
./quersumme.py: argument "1.5" is not an integer
```

auf `sys.stderr` liefern. Bei Fehlermeldungen bricht das Programm mit `exit(1)` ab.

In den Materialien finden Sie einige Beispielaufrufe und die erwartete Ausgabe. Durch `make test` verifizieren Sie die Korrektheit Ihres Programms (inklusive der Fehlerbehandlung).

Aufgabe 4.3 (2 Punkte) Schreiben Sie ein Programm `datetonumber.py`, das genau ein Argument von der Kommandozeile erhält, und zwar den Namen einer Datei. Diese Datei soll in jeder Zeile ausschließlich ein Datum in der Form `DD.MM.JJJJ` enthalten. Ihr Programm soll eine Datei in einem solchen Format einlesen und zu jedem Datum nach einem Tabulatorzeichen die Nummer des Tages im gesamten Jahr angeben.

Beispiel: Nehmen wir an, die Datei `randomdates.csv` enthält folgende Zeilen:

```
05.03.2017
27.09.2006
09.11.2010
24.05.2011
17.11.2000
```

Dann soll `./datetonumber.py randomdates.csv` die folgende Ausgabe liefern:

```
05.03.2017 64
27.09.2006 270
09.11.2010 313
24.05.2011 144
17.11.2000 322
```

Bitte beachten Sie Schaltjahre. Um zu ermitteln, ob ein Jahr ein Schaltjahr ist, können Sie Teile der Lösung einer früheren Aufgabe wiederverwenden.

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. Durch den Befehl `make test` in der Linux Shell verifizieren Sie die Korrektheit Ihres Programms.

Bitte die Lösungen zu diesen Aufgaben bis zum 18.11.2019 um 18:00 Uhr an pfn1@zbh.uni-hamburg.de schicken. Die Besprechung der Lösungen erfolgt am 20.11.2019.

Bitte beachten Sie die Hinweise zur Anfertigung von Lösungen der Übungsaufgaben, die Ihnen ausgehändigt wurden. Achten Sie insbesondere auf Folgendes:

- Erfolgreiche Tests durch Aufruf von `make test`. Wenn ein Test nicht erfolgreich ist, dann muss das dokumentiert werden.

- korrektes Format der abgegebenen Dateien, d.h. insbesondere keine CRLF Zeilentrenner, die entstehen, wenn man die Dateien unter MS-Windows speichert. Wenn das für Sie gilt, dann können Sie durch den Aufruf des Skriptes `pfn1_2019/bin/from_MS_Winwindows.sh` die Konvertierung in ein Linux-kompatibles Format vornehmen.
- maximale Zeilenlänge von 80 in den Python-Dateien. Das Skript `pfn1_2019/bin/pycheck.py`, angewendet auf eine Liste von Dateien, sucht nach Zeilen, die länger als 80 Zeichen lang sind und liefert die Zeilennummer der ersten solchen Zeile.

Wenn Sie die obigen Schritte zur Ergänzung der Umgebungsvariable `PATH` durchgeführt haben, dann können Sie `from_MS_Winwindows.sh` und `pycheck.py` ohne Angabe des Pfades aufrufen.

Hinweis: Einigen Studierenden fällt es noch leicht, die Lösungen der Übungsaufgaben zu erstellen und Sie brauchen keine Unterstützung in den Übungen am Mittwoch. Falls Sie also vor Beginn der Übungen bereits die vollständige Lösung per E-mail abgegeben haben, informieren Sie den Übungsleiter[1] bitte vor Beginn der Übungen darüber. In einem solchen Fall brauchen Sie nicht zur Übung erscheinen.

[1] `kurtz@zbh.uni-hamburg.de`, `ehmki@zbh.uni-hamburg.de`