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

Programmierung in der Bioinformatik
Wintersemester 2019
Übungen zur Vorlesung: Ausgabe am 30.10.2020

Aufgabe 2.1 (2 Punkte) Ihr erstes Python-Programm `hello.py` soll `Hallo Welt` im Terminal ausgeben. Wie man das macht, steht im Vorlesungsskript.

Vergessen Sie nicht, in die erste Zeile Ihres Python-Skripts `hello.py` den magischen String

```
#!/usr/bin/env python3
```

zu schreiben. Machen Sie außerdem die Datei `hello.py` mit dem Shell-Befehl `chmod u+x hello.py` ausführbar.

Schreiben Sie nun ein Python-Skript `helloWorld.py`, in dem nicht festgelegt ist, wer gegrüßt werden soll. Das soll der Benutzer entscheiden. Nach Eingabe von

```
halloWorld.py Tick Trick Track
```

soll folgende Ausgabe im Terminal erscheinen:

```
Hallo Tick
Hallo Trick
Hallo Track
```

Das können Sie dadurch erreichen, dass Ihr Programm auf das Array `sys.argv` zugreift. Dieses enthält die Argumente des Programms beim Aufruf. Damit dieses Array in Ihrem Skript bekannt ist, muss nach dem magischen String die Zeile `import sys` eingefügt werden. Im obigen Beispiel steht der String Tick in `sys.argv[1]`, Trick in `sys.argv[2]` und Track in `sys.argv[3]`. Durch eine `for`-Schleife der Form `for name in sys.argv[1:]:` können Sie im davon abhängigen Block nacheinander auf die Namen über die Variable `name` zugreifen.

Aufgabe 2.2 (2 Punkte) Schreiben Sie ein Python-Programm `leapyear.py`, das für beliebig viele Jahreszahlen jeweils ausgibt, ob das Jahr ein Schaltjahr ist oder nicht. Die Jahreszahlen sollen dabei über `sys.argv` wie bei `helloWorld.py` übergeben werden.

Hinweis: Ein Jahr ist ein Schaltjahr, wenn eine der beiden folgenden Aussagen gilt:

- Die Jahreszahl ist ein Vielfaches von 4 und kein Vielfaches von 100.
- Die Jahreszahl ist ein Vielfaches von 400.

Die Strings aus `sys.argv` erhalten Sie mit einer for-Schleife, wie in `helloWorld.py`. Sie müssen jeden String `year_string` allerdings noch durch `year = int(year_string)` in eine ganze Zahl `year` konvertieren. Beispiel: Beim Aufruf von `leapyear.py 1800 1900 2000 2004 2017` soll im Terminal die folgende Ausgabe erscheinen:

```
1800 ist kein Schaltjahr
1900 ist kein Schaltjahr
2000 ist ein Schaltjahr
2004 ist ein Schaltjahr
2017 ist kein Schaltjahr
```

Im Material zu dieser Übungsaufgabe (Unterverzeichnis `Celsius`) finden Sie ein Makefile mit Spezifikationen von Tests. Durch `make test` verifizieren Sie, dass Ihr Programm korrekt funktioniert.

Aufgabe 2.3 (4 Punkte) Wie betrachten das Python-Programm in der Datei `false_statements.py`.

```
num1 = 7
num2 = 8
ls1 = []
ls2 = [1,2,3]

if num1 + num2 > 15:
  print("1")

if (num1 + num2) % 2 == 0:
  print("2")

if (num1 * num2) % 16 == 0:
  print("3")

if len(ls2) == 2:
  print("4")

if len(ls1) + len(ls2) == 5:
  print("5")

if ls1 != [] and ls1[0] == ls1[len(ls1)-1]:
  print("6")
```

Dabei ist `%` der Modulo-Operator (Restwert-Operator), d.h. für zwei ganze Zahlen `i` und `j` liefert `i % j` den ganzzahligen Rest beim Teilen von `i` durch `j`. Der Operator == vergleicht zwei Werte und liefert genau dann `True`, wenn die Werte gleich sind. Der Operator `!=` vergleicht zwei Werte und liefert genau dann `True`, wenn die Werte ungleich sind. Die eckigen Klammern bei einer Liste stehen für den Indexzugriff, d.h. wenn `ls` eine Liste der Länge n mit $n > 0$ ist, bezeichnet `ls[i]` mit $0 \leq i \leq n-1$ das $i+1$te Element der Liste. Z.B. ist `s[0]` das erste Element des Arrays und `s[n-1]` das $n-$te. Die Funktion `len`, angewandt auf eine Liste, liefert deren Länge. Die Anweisungen, die gegenüber einer `if`-Anweisung eingerückt sind, heissen Blöcke und werden nur dann ausgeführt, wenn die Bedingung in der `if`-Anweisung wahr ist.

Wenn man das Programm ausführt, liefert es keine Ausgabe im Terminal. Überlegen Sie, warum das so ist.

Kopieren Sie das Programm in eine Datei `true_statements.py` und ändern Sie nur die Werte der 4 Variablen `num1`, `num2`, `ls1`, `ls2` so, dass alle `if`-Anweisungen zu `True` auswerten und 1 bis 6 zeilenweise ausgegeben wird. Durch `make test` verifizieren Sie, dass Ihr Programm korrekt funktioniert.

Aufgabe 2.4 (2 Punkte)

Implementieren Sie ein Python-Skript `celsius.py`, das für alle ganzzahligen Temperaturen von 1 bis 100 Grad Celsius die entsprechende Temperatur in Fahrenheit ausgibt. Zur Erinnerung: wenn t_C die Temperatur in Celsius ist, dann ist $t_F = (t_C \cdot 9)/5 + 32$ die entsprechende Temperatur in Fahrenheit. Verwenden Sie die Methode `range()`, um die Werte im genannten Temperatur-Bereich aufzuzählen.

Am Anfang der Ausgabe soll eine Zeile der Form `# celsius fahrenheit` ausgegeben werden, wobei die beiden Bezeichner für die Masseinheiten durch einen Tabulator getrennt werden. Formatieren Sie die Ausgabe mit `print`. Verwenden Sie dazu die Anweisung

```
print("{:.2f}\t{:.2f}".format(celsius, fahrenheit))
```

Im Material zu dieser Übungsaufgabe (Unterverzeichnis `Celsius`) finden Sie ein Makefile, das Tests durchführt. Durch `make test` verifizieren Sie, dass Ihr Programm korrekt funktioniert.

Bitte die Lösungen zu diesen Aufgaben bis zum 04.11.2019 um 18:00 Uhr an pfn1@zbh.uni-hamburg.de schicken. Die Besprechung der Lösungen erfolgt am 06.11.2019.