Stefan Kurtz **Universität Hamburg**
Jeanine Liebold **Zentrum für Bioinformatik**

Programmierung für Naturwissenschaften 1
Wintersemester 2022/2023
Übungen zur Vorlesung: Ausgabe am 02.11.2022

Hinweis der Woche: Im Repository finden Sie eine Datei `Vorlesung/for_python_tutor.html`, die Sie im Browser öffnen können. Dort sind viele der Python-Programme aus der Vorlesung mit dem Web-Server `http://www.pythontutor.com` verlinkt. Wenn Sie dem Hyperlink folgen, erscheint eine Web-Seite mit dem entsprechenden Python-Programm, das Sie dann schrittweise ausführen können. Dabei sehen Sie die Werte der einzelnen Variablen des Programms sowie die jeweils entstehende Ausgabe. Sie können diesen Web-Server auch zur Entwicklung eigener Python-Programme verwenden. Die gleichzeitige Anzeige der ausgeführten Anweisung und dadurch entstehenden Effekte könnte für Studierende, die mit dem Programmieren beginnen, hilfreich sein.

Wie schon auf Blatt01 wird zur Verifikation Ihrer Lösung das Programm `diff` verwendet (siehe `Makefile`) in `Aufgabe2/Leapyear` und `Aufgabe3/UmlautConvert`. Ein Aufruf von `diff file1 file2` für zwei Dateien `file1` und `file2` liefert die Zeilen, die sich in den Dateien unterscheiden. Eine Zeile, die in `file1` steht, aber nicht in `file2`, wird durch das Zeichen < am Anfang der Zeile gekennzeichnet. Eine Zeile, die in `file2` steht, aber nicht in `file1`, wird durch das Zeichen > am Anfang der Zeile gekennzeichnet. Häufig wird statt eines Dateinamen das Zeichen `-` verwendet, wenn `diff` nach dem Pipezeichen steht. Dann liest `diff` über die Pipe das Ergebnis eines anderen Programms und vergleicht es mit der Datei, die beim Aufruf neben `-` angegeben wurde. Daher liefert `cat file1 | diff - file2` das gleiche Ergebnis wie `diff file1 file2`.

Aufgabe 2.1 (2 Punkte)

In der Vorlesung wurde ein Programm gezeigt, dass `Hallo Welt` auf die Standardausgabe (also im Terminal) ausgibt. Schreiben Sie nun ein Python-Skript, in dem nicht festgelegt ist, wer gegrüßt werden soll. Das soll der Benutzer entscheiden.

In den Materialien zu dieser Aufgabe finden Sie eine Datei `helloWorld_template.py`, die Sie in `helloWorld.py` umbenennen. Diese Datei hat dann schon Ausführungsrechte. Fügen Sie nach der Umbenennung Ihren Python-Code ein, so dass nach Eingabe von

```
./helloWorld.py Tick Trick Track
```

die folgende Ausgabe im Terminal erscheint:

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

Das können Sie dadurch erreichen, dass Ihr Programm auf die Liste `sys.argv` zugreift. Dieses enthält die Argumente des Programms beim Aufruf. Damit diese Liste 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 und mit Hilfe der Slice-Notation können Sie nacheinander auf diese Strings zugreifen und die geforderte Ausgabe erzeugen.

Hinweise zur Lösung: `python-slides.pdf`, Abschnitt *Lists*, frame 11 und frame 15

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.

In den Materialien zu dieser Aufgabe finden Sie eine Datei `leapyear_template.py`, die Sie in `leapyear.py` umbenennen. Diese Datei hat dann schon Ausführungsrechte. Fügen Sie nach der Umbenennung Ihren Python-Code ein.

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.

Offensichtlich besteht also die Bedingung für ein Schaltjahr aus mehreren Teilbedingungen. Diese müssen Sie durch die Operatoren `and` und `or` miteinander verknüpfen.

Die Strings aus `sys.argv` erhalten Sie mit einer for-Schleife. Sie müssen aus jedem String `year_string` allerdings noch durch `year = int(year_string)` eine ganze Zahl `year` extrahieren. Z.B. liefert `int('42')` die ganze Zahl 42. 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 `Leapyear`) finden Sie ein Makefile mit Spezifikationen von Tests. Durch `make test` verifizieren Sie, dass Ihr Programm korrekt funktioniert.

Um zu Testen, ob eine Jahreszahl Vielfaches von 100 bzw. von 400 ist, können Sie den Modulo (engl. Modulus) Operator % verwenden. Dieser liefert den Restwert bei ganzzahliger Teilung, d.h. für ganze Zahlen p und q ist `p = (p//q)* q + p % q` wobei `//` für die ganzzahlige Teilung steht. So ist z.B. `15 = (15//7)* 7 + 15 % 7 = 2 * 7 + 1`. Wenn die Jahreszahl in einer Variable `y` steht, dann liefert der Ausdruck `y % 100 == 0` genau dann wahr, wenn die Jahrezahl ohne Rest durch 100 teilbar ist.

Hinweise zur Lösung: `python-slides.pdf`, Abschnitt *Flow of control*, frame 3-13

Aufgabe 2.3 (6 Punkte)

In dieser Aufgabe geht es darum, einen Text mit vielen Umlauten in zwei verschiedene Formate zu konvertieren. Der Text stammt aus dem Artikel

`https://www.spiegel.de/kultur/gesellschaft/`
`post-an-den-zwuebelfuesch-uebel-wuetet-der-guertelwuerger-a-739388.html`

vom 19.01.2011 und wurde von einer Leserin Stefanie Gresens aus Buxtehude verfasst.

In der Version des Textes in der Datei `post_an_den_zwiebelfisch_latex.txt` (siehe Material) werden Umlaute durch Voranstellen des Zeichens `"` vor einem Zeichen A, O, U, a, o, u, s codiert, z.B. in der ersten Zeile dieser Datei:

```
"Uberm"utige H"uhner br"uten
```

Diese Notation findet man häufig im Kontext von LaTeX, der „Programmiersprache“ zur Formatierung von wissenschaftlichen Texten. Daher sprechen wir hier von LaTeX-Codierung.

In der Version des Textes in der Datei `post_an_den_zwiebelfisch_utf8.txt` werden Umlaute durch ein Zeichen mit einer Nummer entsprechend der folgenden Tabelle codiert:

Umlaut	Nummer des Zeichens
Ä	196
Ö	214
Ü	220
ä	228
ö	246
ü	252
ß	223

Da das Linux-Programm `file` diese Datei als `UTF-8 Unicode text` identifiziert, sprechen wir von UTF8-Codierung.

In Python3 kann die Konvertierung einer Zeichennummer in das entsprechende Zeichen durch die Funktion `chr` erfolgen, wie die folgende Auswertung einer Liste mit Aufrufen der Funktion `chr` zeigt:

```
[chr(196),chr(214),chr(220),chr(228),chr(246),chr(252),chr(223)]
  ⇒ ['Ä', 'Ö', 'Ü', 'ä', 'ö', 'ü', 'ß']
```

Benennen Sie die Datei `umlaut_latex_to_utf8_template.py` in `umlaut_latex_to_utf8.py` um und implementieren Sie in dieser Datei ein Python3-Programm zur Konvertierung von Umlauten in LaTeX-Codierung in Umlaute in UTF8-Codierung. In der umbenannten Datei finden Sie bereits einen Test, ob an das Programm genau ein Argument (nämlich ein Dateiname) übergeben wird.

Falls der Test nicht fehlschlägt, wird das Argument als Dateiname interpretiert und die entsprechende Datei soll geöffnet und zeilenweise eingelesen werden. In jeder Zeile sollen alle Umlaute in LaTeX-Codierung durch die entsprechende UTF8-Codierung ersetzt werden. Nachdem alle Ersetzungen erfolgt sind, wird die Zeile auf der Standardausgabe (also im Terminal) ausgegeben. Wenn man das Programm mit der ersten oben genannten Datei als Argument aufruft, dann entsteht der Inhalt der zweiten Datei. Das können Sie durch den Aufruf von `make to_utf8` im Terminal testen. Bitte vergessen Sie nicht, am Ende des Programms das beim Öffnen erzeugte `stream`-Objekt wieder zu schliessen. 3 Punkte

Schreiben Sie ein Programm `umlaut_utf8_to_latex.py` in Python3, das genau ein Argument erhält, nämlich den Namen einer Datei mit einem Text, der Umlaute in der UTF8-Codierung enthält.

Die Datei soll, wie im ersten Programm auch, geöffnet und zeilenweise eingelesen werden. In jeder Zeilen sollen alle Umlaute in UTF8-Codierung durch die entsprechende LaTeX-Codierung ersetzt werden. Nachdem alle Ersetzungen erfolgt sind, wird die Zeile auf der Standardausgabe (also im Terminal) ausgegeben. Wenn man das Programm mit der zweiten oben genannten Datei aufruft, dann entsteht der Inhalt der ersten Datei. Das können Sie durch den Aufruf von `make to_latex` im Terminal testen. Auch in diesem Programm muss das `stream`-Objekt am Ende wieder geschlossen werden. 3 Punkte

Die Ersetzung von einer Umlaut-Codierung in die andere kann jeweils durch sieben aufeinanderfolgende Aufrufe der Methode `re.sub` pro Zeile erfolgen.[1]

Beachten Sie, dass beide zu entwickelnden Skripte in der ersten Zeile den magischen String

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

enthalten müssen. Ebenso müssen Sie die Dateien durch den Shell-Befehl `chmod u+x *.py` ausführbar machen.

Nachdem Sie beide Programme entwickelt haben, verifizieren Sie durch `make test`, dass Ihr Programm für die Testdaten korrekt funktioniert.

Hinweise zur Lösung: `python-slides.pdf`, Abschnitt *Sequences and Strings*, frame 17 und 30

Bitte die Lösungen zu diesen Aufgaben bis zum 07.11.2022 um 18:00 Uhr an pfn1@zbh.uni-hamburg.de schicken.

[1] Es ist möglich, die Aufgabe mit nur einem Aufruf von `re.sub` pro Zeile zu bewältigen. Dafür sind aber Techniken notwendig, die Sie erst später in der Vorlesung kennen lernen.