Informatik für Schüler, Foliensatz 10
Wiederholung und Listen

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal
17. März 2010

Syntaxfehler (-Beseitigung)

Was bedeuten folgende Fehlermeldungen?

- File "tmp.py", line 3 ... IndentationError: unexpected indent
- File "tmp.py", line 6 SyntaxError: Non-ASCII character '\xc3'
- File "tmp.py", line 6, in <module> print x, y NameError: name 'y' is not defined

---

```
while a<1000:
    a*=2
  print a
```

- Welche Fehlermeldungen sind zu erwarten?
- Wie könnte das richtige Programm aussehen?
- Wie funktioniert das Programm schrittweise?

Ein- und Ausgabe

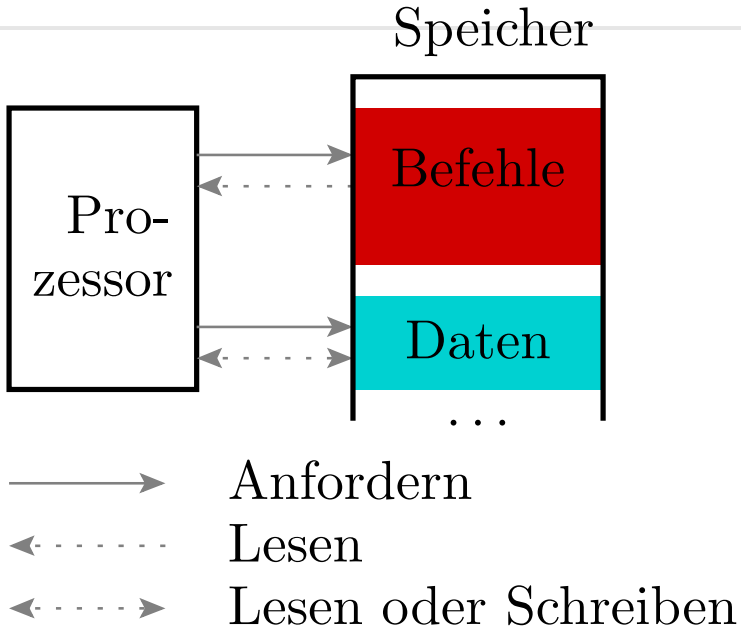
Ein Programm soll eine Zeichenkette in eine Variable t einlesen:

- Beispielprogramm für die Eingabe vom Terminal?
- Beispielprogramm für die Eingabe aus einer Datei?

Ein Programm soll eine Zeichenkette »Hallo« ausgeben:

- Beispielprogramm für die Ausgabe an das Terminal?
- Beispielprogramm für die Ausgabe in einer Datei?

Das Terminal ist auch eine Datei



Dateinamen mit Terminal-Kommando »tty« bestimmen. Ergebnis Pfad+Dateiname, z.B. /dev/pts/0

```
fin = open('/dev/pts/0', 'r')
fout= open('/dev/pts/0', 'w')
fout.write('Eingabe:')
x=fin.readline()
print 'Ergebnis: ', x
fout.close()
print 'Ende'
```

- Ausprobieren und zu beobachtende Probleme beschreiben!

Probleme und Lösungen

- Eine Dateiobjekt speichert Schreibdaten in einem Zwischenspeicher bis:
  - dieser voll ist
  - das Dateiobjekt gelöscht wird
  - der Puffer geleert wird mit:

    ```
    fout.flush()
    ```
- Die erste print-Anweisung erzeugt einen Zeilenvorschub zuviel. Warum?
  - Wie funktioniert readline()? Notfalls ausprobieren!
  - Wie kann der zusätzliche Zeilenumbruch vermieden werden?
- Was passiert, wenn das Programm in einem anderen Terminal gestartet wird?
  - Die Ausgabe erscheint auf dem anderen Terminal

- Die Eingabe funktioniert nicht: Warum?
  - Auf der anderen Konsole läuft der Befehlsinterpreter, der die Eingabedaten schneller holt
  - Umgehung: Start eines Programms, dass nicht auf Eingabedaten wartet im anderen Terminal, z.B.:

```
while True:
    pass
```

- Was tut das kleine Programm?
- Experiment:
  - auf Terminal 2 mit »tty« zugeordneten Dateinamen bestimmen und dann das Programm mit der while-Schleife starten
  - Dateinamen in das Ein-/Ausgabe-Programm einsetzen; anderes und Ein-/Ausgabe-Programm auf Terminal 1 starten
  - Programm läuft auf Terminal 1, aber Ein- und Ausgabe erfolgt über Terminal 2
  - mit einem Ein-/Ausgabe-Programm auf beiden Terminals hat man schon fast ein Chat-Programm

- Im Modul »sys« gibt es bereits geöffnete Dateiobjekte für die Ein- und Ausgabe auf demselben Terminal:

```
import sys
fin = sys.stdin   # Dateiobjekt Standardeingabe
fout = sys.stdout # Dateiobjekt Standardausgabe
fout.write('Eingabeaufforderung: ')
fout.flush()
x=fin.readline()
print x[:-1]
```

Analyse einer Funktion

```
import sys
def Eingabe(a):
    sys.stdout.write(a)
    sys.stdout.fush()
    return sys.stdin.read()

print Eingabe('Text1')
```

- Welche Datentypen haben der Parameter und der Funktionswert?
- Was passiert schrittweise bei der Abbarbeitung?

Listen

Eine Liste ist ein bearbeitbares Sequenzobjekt.

- Ein Seqenzobjekt ist eine Zusammenfassung von Elementen. Die anderen Sequenzobjekte »string« und »tuple« sind nicht bearbeitbar, d.h. nur erzeug- und lösschbar.
- Liste erzeugen:

```
liste = [1, 'das', 'ist', True]
```

- Elemente verändern:

```
liste[2] = 'war'
liste[0:1] = ('xx', 'yy')
```

- Bearbeitungsmethoden: Elemente einfügen, anhängen, löschen etc.

Methoden

- Eine Methode ist ein Unterprogramme zur Bearbeitung von Objekten. Notation:

  Objektname.Methode(Liste_der-Argumente)

  Das Objekt selbst ist für die Methode ein les- und veränderbarer Parameter.
- Methoden wurden bereits für Objekte vom Typ »file« benutzt:

```
fo=open('Datei', 'w'); # Objekt erzeugen
fo.write('Text') # Schreiben
fo.fush()  # Puffer leeren
fo.close() # Objekt löschen + Datei schließen
```

Bearbeitungsmethoden für Listen

- Anfügen eines Elements x am Ende der Liste:

  ```
  liste.append(x)
  ```

- Anfügen aller Element des Sequenzobjekts x am Ende der Liste:

  ```
  liste.extend(x)
  ```

- Anfügen eines Elements x an der Stelle i:

  ```
  liste.insert(i, x)
  ```

- Löschen von Element i:

  ```
  del liste[i]
  ```

  (keine Methode, sondern allgemeine Löschfunktion

- Sortieren der Liste:

  ```
  liste.sort()
  ```

Aufgabe 10.1: Manuell sortieren

Vereinbaren Sie eine gefüllte und eine leere Liste, z.B.:

```
L1 = ['e1', 'e2', 'a', 'b']
L2 =[]
```

Schreiben Sie ein Programm, dass, solange Liste l1 nicht leer ist:

- die Listen L1 und L2 ausdruckt
- anschließend zur Eingabe einer Zahl mit einem gültigen Listenindex für L1 auffordert
- das Element mit dem Index aus der Liste L1 löscht und an die Liste L2 anhängt.

Aufgabe 10.2: Umwandlung von Zeichenketten in Listen und umgekehrt

Das zu entwickelnde Unterprogramm

```
def str2liste(s):
    ...
    return l
```

sollen eine Zeichenkette in eine Liste, deren Elemente Zeichen sind, umwandeln, und das zu entwickelnde Unterprogramm

```
def liste2str(l):
    ...
    return s
```

soll eine Liste aus Zeichen in eine Zeichenkette umwandeln.

- Testrahmen für die beiden Methoden:

```
s1 = raw_input('Zeichenfolge ohne Hochkommas: ')
x = str2liste(s1)
print 'Liste: ', x
s2 = liste2str(x)
print 'Zeichenfolge: ', s2
```

Wenden Sie die Methode »sort« auf ihr Listenobjekt »x« an:

```
x.sort()
```

Was passiert mit dem Listenobjekt?

Aufgabe 10.3: Wörter sortieren

Schreiben Sie in ähnlicher Weise wie in der Voraufgabe ein Programm, das eine Eingabezeichenkette in eine Liste von Worten umwandelt, diese sortiert und die sortierte Wortliste als Zeichenkette ausgibt.

Hinweis: Es ist zweckmäßig, hierfür zusätzlich eine Funktion »isTrennz()« zu definieren, die als Argument ein Zeichen übernimmt. Wenn das Zeichen ein Leeerzeichen, »\n« oder »\t« (Tabulator) ist, soll die Funktion »True«, sonst »False« zurückgeben.