Informatik für Schüler, Foliensatz 18
Rekursion

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal
26. März 2009

Was ist eine Rekursion

- Rekursion (lat. recurrere „zurücklaufen“) bedeutet in Mathematik, Logik und Informatik, eine Funktion durch sich selbst zu definieren.
- Ein rekursives Programm ist ein Unterprogramm, das sich selbst aufruft.
- zählt zu den mächtigsten und gefährlichsten Beschreibungsmitteln der Informatik
- Beispiel Fakultät

$$n! = \begin{cases} n \cdot (n-1)! & \text{für} \quad n > 1 \\ 1 & \text{sonst} \end{cases}$$

Achtung: Jede Rekursion benötigt eine Abbruchbedingung, sonst Endlosprogramm

Rekursive Berechnung der Fakultät

```
def fakultaet(n):
  print 'berechne ', n, '!'
  if n>1:
   f=n*fakultaet(n-1)
  else:
   f=1
  print n, '! = ', f
  return f

fakultaet(5)
```

- Welche Ausgabefolge erzeugt das Programm?
- Was passiert wenn die Fallunterscheidung für die Abbruchbedingung fehlt?

Was passiert, wenn sich ein Programm mehrfach selbst aufruft (Bsp. Berechnung Fibonacci-Zahl)

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{für} \quad n > 2 \\ n & \text{sonst} \end{cases}$$

```
def fib(n):
  print 'berechne fib( ', n, ')'
  if n>2:
   f=fib(n-1)+fib(n-2)
  else:
   f=n
  print n, 'fib(n) = ', f
  return f

fib(5)
```

- Welche Ausgabefolge erzeugt das Programm?

Aufgabe 18.1: Rekursive Berechnung der Fibonacci-Zahl

Bestimmen Sie für die rekusive Berechnung der Fibonacci-Zahl experimentell, wie oft die Funktion »fib(n)« insgesamt in Abhängigkeit von n aufgerufen wird.

n	5	6	7	8	9	10	11	12	13	14	15	16
fib(n)												

Hilfestellung: Man kann innerhalb eines Unterprogramms mit dem Schlüsselwort »global« eine globale Variable als Zähler vereinbaren:

```
def fib(n):
  global Zaehler
  Zaehler = Zaehler +1
  ...
```

Der Zähler muss vor dem ersten Unterprogrammaufruf existieren.

- Was bedeutet »globale Variable«?

Aufgabe 18.2: Multiplikation rekursiv

Beschreiben Sie die Multiplikation von zwei natürlichen Zahlen rekursiv, zuerst auf dem Papier und dann als Programm. Das Unterprogramm

```
def mult(a, b):
 ...
 return Produkt
```

soll intern nur aus Fallunterscheidungen, Additionen, Konstantenzuweisungen, einem Aufruf von sich selbst und print-Anweisungen zur Kontrolle, ob es richtig arbeitet, bestehen.

Die Testausgaben zur Funktionskontrolle können wie folgt aussehen:

```
berechne 5*7
berechne 4*7
berechne 3*7
berechne 2*7
berechne 1*7
1*7 = 7
2*7 = 14
3*7 = 21
4*7 = 28
5*7 = 35
```

Was passiert, wenn die Aufrufargumente keine positiven ganzen Zahlen sind? (Null, negativ, Nachkommastellen)

Aufgabe 18.3: Umwandlung einer rekursiven Liste in einen String

Die Eingabe sei eine Liste, deren Elemente entweder Zeichen des Zeichensatzes oder selbst Listen aus Zeichen und Listen sind, z.B.:

```
x1 = ['H', 'a', 'l', 'l', 'o'];
x2 = ['W', 'e', 'l', 't'];
x3 = ['g', 'e', 'h', 't'];
x4 = [x3, ' ', 'e', 's', ' ', 'd', 'i', 'r'];
x5 = [x1, ' ', x2, ',', ' ','W', 'i', 'e', x4,'?'];
```

Schreiben Sie ein rekusives Programm mit einer Liste als Aufrufparameter und einem String als Rückgabewert:

```
def Liste2Str(l):
  ...
  return s
```

das für jedes Element der Liste, wenn es ein Zeichen ist, das Zeichen an einen zum Beginn leeren String anhängt, undf wenn es eine Liste ist, sich selbst mit der Teilliste aufruft und den Ergebnisstring anhängt. Kontrollieren Sie die Abarbeitung mit geeigneten Testausgaben.

In welche Zeichenkette müsste das Programm die folgende Liste umwandeln:

```
a = ['p', 'l', 'a', ' ', a];
```

Wie schützt sich ein Python-Programm vor rekursiven Listen? (ausprobieren)