

Informatik I: Einführung in die Programmierung

9. Rekursion

Albert-Ludwigs-Universität Freiburg



Bernhard Nebel

06. November 2015

1 Rekursion verstehen



Rekursion
verstehen

Fakultäts-
funktion

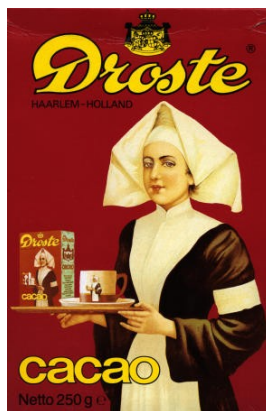
Fibonacci-
Folge

06. November 2015

B. Nebel – Info I

3 / 20

Rekursion verstehen



Rekursion
verstehen

Fakultäts-
funktion

Fibonacci-
Folge

Um Rekursion zu verstehen, muss man zuerst einmal
Rekursion verstehen.

Abb. in Public Domain, Quelle Wikipedia

06. November 2015

B. Nebel – Info I

4 / 20

2 Fakultätsfunktion



Rekursion
verstehen

Fakultäts-
funktion

Rekursive
Definition
Fakultät in Python
Einfache Rekursion
und Iteration

Fibonacci-
Folge

- Rekursive Definition
- Fakultät in Python
- Einfache Rekursion und Iteration

06. November 2015

B. Nebel – Info I

6 / 20

Rekursion als Definitionstechnik: Fakultätsfunktion



- Bei einer **rekursiven Definition** wird das zu Definierende unter Benutzung desselben (normalerweise in einer einfacheren Version) definiert.
- Beispiel **Fakultätsfunktion**
 - Auf wie viele Arten kann man n Dinge sequentiell anordnen?
 - Berechne, auf wie viele Arten man $n - 1$ Dinge anordnen kann. Für jede dieser Anordnungen können wir das „letzte“ Ding auf n Arten einfügen.
 - D.h. wir können die Fakultätsfunktion $n!$ wie folgt definieren:

$$n! = \begin{cases} 1, & \text{falls } n = 0; \\ n \cdot (n-1)!, & \text{sonst.} \end{cases}$$

- Berechne 4!:
 $4! = 4 \cdot 3! = 4 \cdot 3 \cdot 2! = 4 \cdot 3 \cdot 2 \cdot 1! = 4 \cdot 3 \cdot 2 \cdot 1 \cdot 0! = 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 = 24$

Rekursion verstehen
Fakultätsfunktion
Rekursive Definition
Fakultät in Python
Einfache Rekursion und Iteration
Fibonacci-Folge

Fakultätsfunktion in Python



- Wir können in Funktionsdefinitionen bisher undefinierte (z.B. die gerade zu definierende) Funktion benutzen:

Python-Interpreter

```
>>> def fak(n):  
...     if n <= 1:  
...         return 1  
...     else:  
...         return n*fak(n-1)  
...  
>>> fak(4)  
24  
>>> fak(50)  
304140932017133780436126081660647688443776415689605  
12000000000000
```

Rekursion verstehen
Fakultätsfunktion
Rekursive Definition
Fakultät in Python
Einfache Rekursion und Iteration
Fibonacci-Folge

Rekursive Aufrufe



- Was passiert genau?

Aufrufsequenz

→ fak(4) wählt else-Zweig und ruft auf:
→ fak(3) wählt else-Zweig und ruft auf:
→ fak(2) wählt else-Zweig und ruft auf:
→ fak(1) wählt if-Zweig und:
← fak(1) gibt 1 zurück
← fak(2) gibt $(2 \times 1) = 2$ zurück
← fak(3) gibt $(3 \times 2) = 6$ zurück
← fak(4) gibt $(4 \times 6) = 24$ zurück

Visualisierung

Rekursion verstehen
Fakultätsfunktion
Rekursive Definition
Fakultät in Python
Einfache Rekursion und Iteration
Fibonacci-Folge

Einfache Rekursion und Iteration



- Die rekursive Definition der Fakultätsfunktion ist eine **einfache Rekursion**.
- Solche Rekursionen können einfach in **Iterationen** (while-Schleife) umgewandelt werden:

Python-Interpreter

```
>>> def ifak(n):  
...     result = 1  
...     while n >= 1:  
...         result = result * n  
...         n -= 1  
...     return result  
...
```

Visualisierung

Rekursion verstehen
Fakultätsfunktion
Rekursive Definition
Fakultät in Python
Einfache Rekursion und Iteration
Fibonacci-Folge

3 Fibonacci-Folge



- Definition der Fibonacci-Zahlen
- Fibonacci in Python
- Fibonacci iterativ

Rekursion verstehen
Fakultätsfunktion
Fibonacci-Folge
Definition der Fibonacci-Zahlen
Fibonacci in Python
Fibonacci iterativ

Fibonacci-Folge: Das Kanninchenproblem



- Manchmal sind kompliziertere Formen der Rekursion notwendig, z.B. bei der Definition der **Fibonacci-Folge**
- Eingeführt von *Leonardo da Pisa*, genannt *Fibonacci*, in seinem Buch *Liber abbaci* (1202), das u.a. die arabischen Ziffern und den Bruchstrich in die westliche Welt einführten.
- Anzahl der Kanninchen-Paare, die man erhält, wenn jedes Paar ab dem zweiten Monat ein weiteres Kanninchen-Paar erzeugt (und kein Kanninchen stirbt). Wir beginnen im Monat 0, in dem das erste Paar geboren wird:

Monat vorhanden geboren gesamt

0.	0	1	1
1.	1	0	1
2.	1	1	2
3.	2	1	3
4.	3	2	5

Rekursion verstehen
Fakultätsfunktion
Fibonacci-Folge
Definition der Fibonacci-Zahlen
Fibonacci in Python
Fibonacci iterativ

Fibonacci-Zahlen



- Die Fibonacci-Zahlen werden normalerweise wie folgt definiert (und beschreiben damit die vorhandenen Kanninchen-Paare am Anfang des Monats):

$$\text{fib}(n) = \begin{cases} 0, & \text{falls } n = 0; \\ 1, & \text{falls } n = 1; \\ \text{fib}(n-1) + \text{fib}(n-2), & \text{sonst.} \end{cases}$$

- D.h. die Folge beginnt mit 0 und nicht mit 1.
- Beachte: Hier gibt es zwei rekursive Verwendungen der Definition.
- Die Fibonacci-Zahlen spielen in vielen anderen Kontexten eine wichtige Rolle (z.B. Goldener Schnitt).

Rekursion verstehen
Fakultätsfunktion
Fibonacci-Folge
Definition der Fibonacci-Zahlen
Fibonacci in Python
Fibonacci iterativ

Fibonacci-Zahlen in Python



- Umsetzung in Python folgt direkt der mathematischen Definition:

Python-Interpreter

```
>>> def fib(n):  
...     if n <= 1:  
...         return n  
...     else:  
...         return fib(n-1) + fib(n-2)  
...  
>>> fib(35)  
9227465
```

Rekursion verstehen
Fakultätsfunktion
Fibonacci-Folge
Definition der Fibonacci-Zahlen
Fibonacci in Python
Fibonacci iterativ

Aufrufsequenz

→ fib(3) wählt else-Zweig und ruft auf:
| → fib(2) wählt else-Zweig und ruft auf:
| | → fib(1) wählt if-Zweig und
| | ← fib(1) gibt 1 zurück
| fib(2) ruft jetzt auf:
| | → fib(0) wählt if-Zweig und
| | ← fib(0) gibt 0 zurück
| ← fib(2) gibt 1 zurück
fib(3) ruft jetzt auf:
| → fib(1) wählt if-Zweig und
| ← fib(1) gibt 1 zurück
← fib(3) gibt 2 zurück

Rekursion verstehen
Fakultätsfunktion
Fibonacci-Folge
Definition der Fibonacci-Zahlen
Fibonacci in Python
Fibonacci iterativ

Visualisierung

- Es gibt komplexere Formen der Rekursion: **mehrfach**, *indirekt*, *durch Argumente*.
- Es ist nicht ganz einfach, den Verlauf der Ausführung der fib-Funktion nachzuvollziehen.
- Dies ist aber auch nicht notwendig! Es reicht aus, sich zu vergegenwärtigen, dass:
 - falls die Funktion alles richtig macht für Argumente mit dem Wert $< n$,
 - dann berechnet sie das Geforderte→ Prinzip der vollständigen Induktion
- Die mehrfachen rekursiven Aufrufe führen zu **sehr hoher Laufzeit!**
- Auch hier ist eine iterative Lösung (while-Schleife) möglich.

Rekursion verstehen
Fakultätsfunktion
Fibonacci-Folge
Definition der Fibonacci-Zahlen
Fibonacci in Python
Fibonacci iterativ

Iterative Lösung I

- Im Allgemeinen ist es schwierig, Mehrfachrekursion in Iteration umzuwandeln.
- Bei fib hilft die Beobachtung, dass man den Wert immer durch die Addition der letzten beiden Werte berechnen kann. Das geht auch **bei 0 startend!**
- Generiere die Werte aufsteigend, bis die Anzahl der erzeugten Werte den Parameter n erreicht.

Rekursion verstehen
Fakultätsfunktion
Fibonacci-Folge
Definition der Fibonacci-Zahlen
Fibonacci in Python
Fibonacci iterativ

Iterative Lösung II

Python-Interpreter

```
>>> def ifib(n):  
...     if n <= 1:  
...         return n  
...     a = 0  
...     b = 1  
...     i = 2  
...     while i < n:  
...         new = a + b  
...         a = b  
...         b = new  
...         i += 1  
...     return a + b
```

Visualisierung

Rekursion verstehen
Fakultätsfunktion
Fibonacci-Folge
Definition der Fibonacci-Zahlen
Fibonacci in Python
Fibonacci iterativ

- **Rekursion** ist eine bekannte Definitionstechnik aus der Mathematik.
- Rekursion erlaubt es, bestimmte Funktion sehr kurz und **elegant** anzugeben.
- Dies ist nicht immer die **effizienteste** Form! Das ist aber abhängig von der Programmiersprache.
- **Einfachrekursion** kann meist einfach in **Iteration** umgewandelt werden.
- **Mehrfachrekursion** ist komplizierter.
- Es gibt noch komplexere Formen der Rekursion.
- Interessant werden rekursive Funktionen bei **rekursiven Datenstrukturen**.

Rekursion
verstehen

Fakultäts-
funktion

Fibonacci-
Folge

Definition der
Fibonacci-Zahlen
Fibonacci in
Python
Fibonacci iterativ