Der Cons-Operator[1]

Der : Operator (genannt Cons-Operator) kann sowohl zum Aufbau einer Liste als auch zum Abbau einer Liste verwendet werden, je nachdem, an welcher Stelle er eingesetzt wird. Wenn er in einem Ausdruck verwendet wird, also auf der rechten Seite des Gleichheitszeichens, in der Funktionsdefinition, dann wird er zum Aufbau einer Liste verwendet.
Wenn er in einem Muster, Pattern Matching, linke Seite des Gleichheitszeichens, verwendet wird bewirkt er das Gegenteil - er dekonstruiert (zerlegt) eine Liste.
Eine Liste **konstruieren**:
am Prompt

```
> 1:2:[3, 4]
[1,2,3,4]
```

Im Skript:

```
neue_liste = 123:25:[87,13] -- beachte: der letzte Operand des : Operators ist immer eine Liste!
```

Eine Liste **zerlegen**:

```
zerlege_liste_first (first:second:rest) = first -- Klammerung ist zwingend!
zerlege_liste_second (first:second:rest) = second
zerlege_liste_rest (first:second:rest) = rest
-- Ende Skript
Aufruf am Prompt:
>zerlege_liste_first [15..99]
15
>zerlege_liste_second [15..99]
16
>zerlege_liste_rest [15..99]
[17,18,19,20,21,22 ……. 97,98,99]
```

Beispiel-Aufgabe:
Es soll die Länge einer Liste (= Anzahl der Listenelemente) ermittelt werden.
Problem: Wie können wir auf die Listenelemente rekursiv zugreifen? Oder: wie lässt sich eine Liste entsprechend zerlegen? --> Das geht mit dem :-Operator auf der **linken** Seite des Gleichheitszeichens. Diesen benutzen wir als Trenner / Aufspalter.

Informelle Beschreibung des Algorithmus zur Bestimmung der Länge einer Liste

1. Eine Liste hat die Länge 0, wenn sie leer ist
2. sonst addieren wir die Anzahl des aktuellen Elements (=1) mit der Länge der Restliste

Vorübung in einem Skript:

```
test (x:xs) = x – welchen Typ gibt die Funktion test zurück?
test2 (x:xs) = xs – welchen Typ gibt die Funktion test2 zurück?
```

Idee: Verwendung von Pattern-Matching für Listen zum Trennen von Element und Restliste zur späteren Verwendung:

```
laenge [] = 0
laenge (x:xs) = 1 + laenge xs – Klammern sind notwendig!
```

Weiteres Beispiel: Dezimal nach Dual Umwandlung

Cons-Operator **rechts** vom Gleichheitszeichen zum **Zusammensetzen** von Listenelement und Liste:

```
dezdual::Integer -> [Integer]
dezdual 0 = [0] -- Rekursionsanker
dezdual n = reverse (rest n) -- reverse ist eine intern definierte Listenfunktion

rest:: Integer -> [Integer]
-- rest zerlegt die Zahl in ganzzahligen Anteil und Rest, der Rest wird in
-- einer Liste mittels des cons-Operators gesammelt
rest 0 = []
rest n
   | mod n 2 == 1 = 1 : rest (div n 2)
   | mod n 2 == 0 = 0 : rest (div n 2)
>rest 74
[0,1,0,1,0,0,1]
-- wie liesse sich aus der Ergebnisliste die darstellungskonforme Dualzahl 1001010 als String generieren??
```

Aufgabe: Worin liegt der Unterschied zwischen : und ++ in einer Funktionsdefinition?

[1] nach https://stackoverflow.com/questions/18312535/how-does-cons-work-in-haskell