III. Funktionale Programmierung

- 1. Prinzipien der funktionalen Programmierung
- 2. Deklarationen
- 3. Ausdrücke
- 4. Muster (Patterns)
- 5. Typen und Datenstrukturen
- **6. Funktionale Programmiertechniken:** Unendliche Datenobjekte

Nicht-strikte Auswertung

```
infinity :: Int
infinity = infinity + 1

mult :: Int -> Int -> Int
mult 0     y = 0
mult (x+1) y = y + mult x y
```

```
mult 0 infinity          = 0

mult infinity 0          Terminiert nicht!

0 * infinity             Terminiert nicht!
```

Unendliche Datenobjekte

```
from :: Int -> [Int]
from x = x : from (x+1)
```

```
take :: Int -> [a] -> [a]
take 0     _      = []
take _     []     = []
take (n+1)(x:xs) = x : take n xs
```

```
   take 2 (from 5)
=  take 2 (5 : from 6)
=  5 : take 1 (from 6)
=  5 : take 1 (6 : from 7)
=  5 : 6 : take 0 (from 7)
=  5 : 6 : []
=  [5,6]
```

Sieb des Eratosthenes

1. Erstelle Liste aller natürlichen Zahlen ab 2.
2. Markiere die erste unmarkierte Zahl in der Liste.
3. Streiche alle Vielfachen der letzten markierten Zahl.
4. Gehe zurück zu Schritt 2.

```
drop_mult :: Int -> [Int] -> [Int]
drop_mult x xs = filter (\y -> mod y x /= 0) xs
```

```
dropall :: [Int] -> [Int]
dropall (x:xs) = x : dropall (drop_mult x xs)
```

```
primes :: [Int]
primes = dropall (from 2)
```

Sieb des Eratosthenes

```
primes = [2,3,5,7,11,13,17,19,23,29,31,...
```

```
take 5 primes = [2,3,5,7,11]
```

```
drop_mult :: Int -> [Int] -> [Int]
drop_mult x xs = filter (\y -> mod y x /= 0) xs
```

```
dropall :: [Int] -> [Int]
dropall (x:xs) = x : dropall (drop_mult x xs)
```

```
primes :: [Int]
primes = dropall (from 2)
```