Fakultät

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

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

iterative Version

$$5! = \underbrace{1 \cdot 2 \cdot 3 \cdot 4}_{= 4!} \cdot 5$$

$$= 4! \cdot 5$$

rekursive Version

$$fak(0) = 1$$

$$fak(n) = n \cdot fak(n-1)$$

Listen-Sichtweise

Betrachte Liste der Zahlen von 1 bis n

$[1, 2, 3, 4, 5] \rightsquigarrow$ Produkt

Berechne das Produkt aller Zahlen in dieser Liste

reduce()

Erwartet eine Liste...

$[1, 2, 3, 4, 5]$

$f: \mathbb{R} \to \mathbb{R}$

$x \to x^2$

... und eine Funktion

$mult: \mathbb{N} \times \mathbb{N} \longmapsto \mathbb{N}$

$x, y \to x \cdot y$

$\lambda x, y : x \cdot y$

```
def mult(x, y):
    return x * y
lambda x, y : x * y
```

reduce()

Erwartet eine Liste...

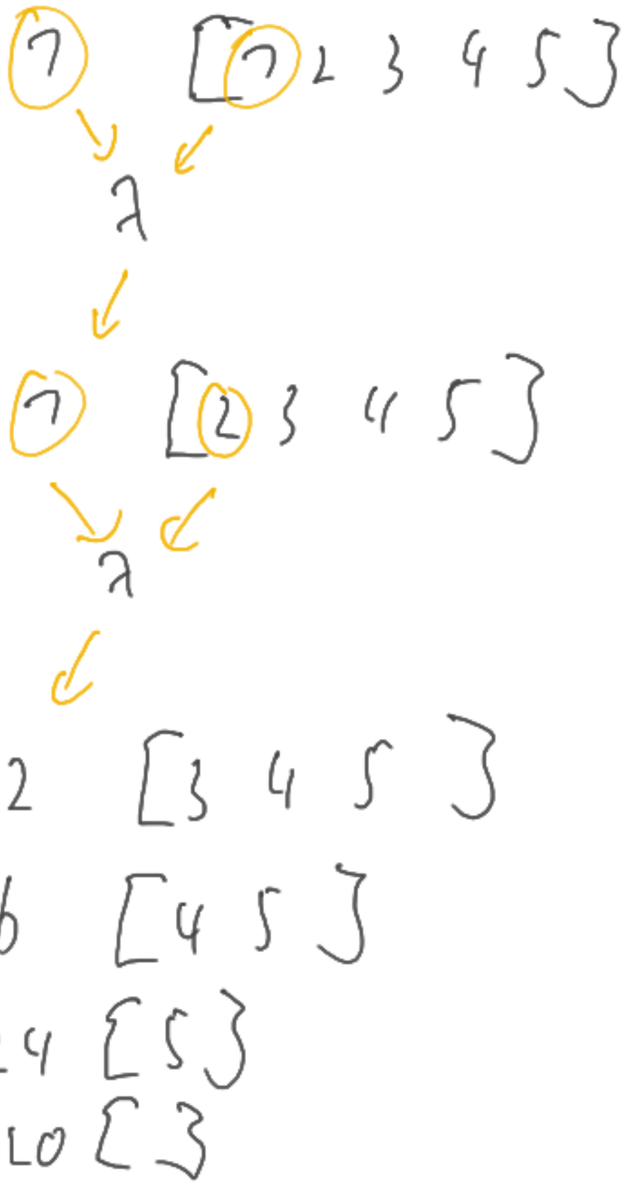
[1, 2, 3, 4, 5]

... und eine Funktion

lambda x, y : x · y

... und einen Startwert:

1



List Comprehension

```
[i for i in range(1,n+1) if teilbar(n,i)]
```

Liste aller Zahlen i ...

... im Bereich von 1 bis n ...

... durch die n teilbar ist.

Menge aller Teiler von n

$$\{ i \in \{1, \ldots, n\} \mid i \mid n \}$$

List Comprehension

Menge aller geraden Zahlen

$$\{x \in \mathbb{N} \mid x = 2y,\ y \in \mathbb{N}\}$$

Generatoren

```
def primes():
    '''Liefert einen Generator für alle Primzahlen.'''
    i = 2
    while True:
        if istPrimzahl(i):
            yield i
        i += 1
```

Für die Vorstellung:

Die Funktion wird an dieser Stelle unterbrochen.

Tatsächlich liefert die gesamte Funktion einen Generator.

Generator

```
p = primes()

primzahlen_bis_10 = [next(p) for i in range(10)]

print(primzahlen_bis_10)

print(p)

print(next(p))
```

nach Bedarf next() auf p anwenden

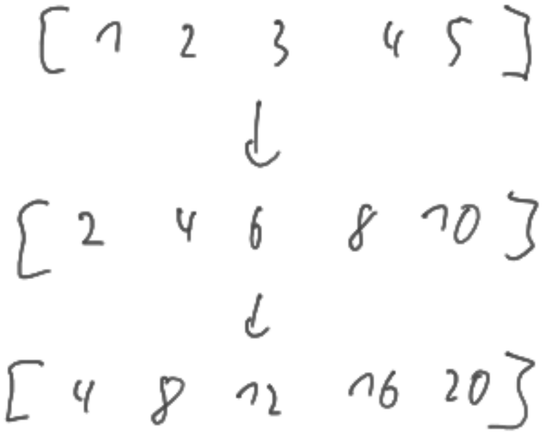
```python
def verdoppeln(gen):
    '''Nimmt einen Generator und liefert einen neuen Generator,
       bei dem jedes Element des alten verdoppelt wird.

       Beispiel: [1,2,3,4,5] --> [2,4,6,8,10]
    '''
    return [2 * i for i in gen]
```

```python
print(verdoppeln([1,2,3,4,5]))
```

Erzeugt eine Liste mit den jeweils doppelten Zahlen.

```python
print(verdoppeln(verdoppeln([1,2,3,4,5])))
```

Erzeugt zwei Listen:



Beide belegen Speicher.

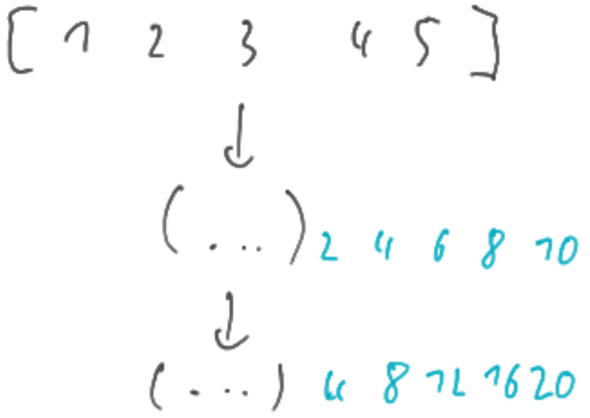
```python
def verdoppeln(gen):
    '''Nimmt einen Generator und liefert einen neuen Generator,
       bei dem jedes Element des alten verdoppelt wird.

       Beispiel: [1,2,3,4,5] --> [2,4,6,8,10]
    '''
    return (2 * i for i in gen)
```

```python
print(verdoppeln([1,2,3,4,5]))
```

Erzeugt eine Liste mit den jeweils doppelten Zahlen.

```python
print(verdoppeln(verdoppeln([1,2,3,4,5])))
```

Erzeugt zwei Listen:



Beide werden erstmal nicht ausgewertet, belegen keinen Speicher.

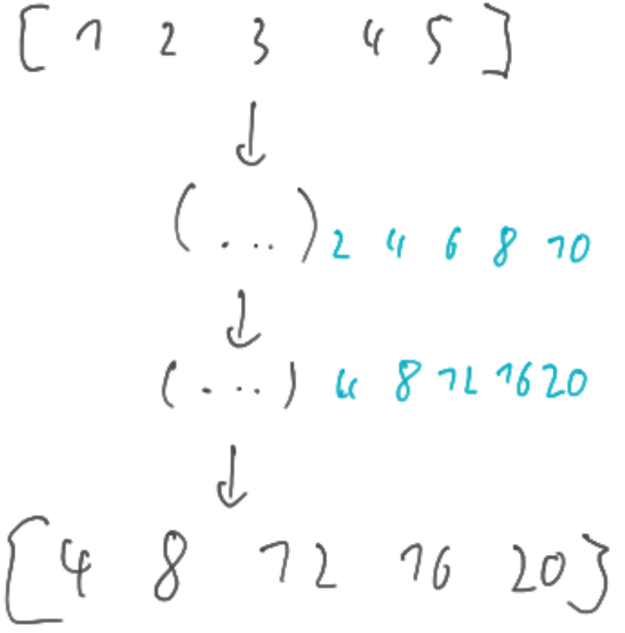
```python
def verdoppeln(gen):
    '''Nimmt einen Generator und liefert einen neuen Generator,
       bei dem jedes Element des alten verdoppelt wird.

       Beispiel: [1,2,3,4,5] --> [2,4,6,8,10]
    '''
    return (2 * i for i in gen)
```

```python
print(verdoppeln([1,2,3,4,5]))
```

Erzeugt eine Liste mit den jeweils doppelten Zahlen.

```python
print(list(verdoppeln(verdoppeln([1,2,3,4,5]))))
```

Erzeugt zwei Listen:



Beide werden erstmal nicht ausgewertet, belegen keinen Speicher.