Einschub: Rekursion

Man kann auf Rekursion (in imperativer Programmierung) verzichten.
Allerdings vereinfacht es manchmal die Umsetzung von Algorithmen.

Beispiel: Fakultät

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1$$

Rekursiv formuliert:

$$n! = n \cdot (n-1)! \text{ mit } 0! = 1$$

Einschub: Rekursion

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1$$

```
def fak(n):
  ergebnis = 1
  while n > 0:
    ergebnis = ergebnis*n
    n = n – 1
  return ergebnis
```

$$n! = n \cdot (n-1)! \text{ mit } 0! = 1$$

```
def fak(n):
  if n == 0:
    return 1
  else:
    return n*fak(n-1)
```

Einschub: Rekursion

$$n! = n \cdot (n-1)! \text{ wobei } 0! = 1 \text{ gilt}$$

Zwei Dinge sind notwendig, damit Rekursion funktionieren kann:

1. Abbruchbedingung

2. Rekursiver Aufruf mit veränderten Parameterwerten(!)

```
def fak(n):
  if n == 0:
    return 1
  else:
    return n*fak(n-1)
```