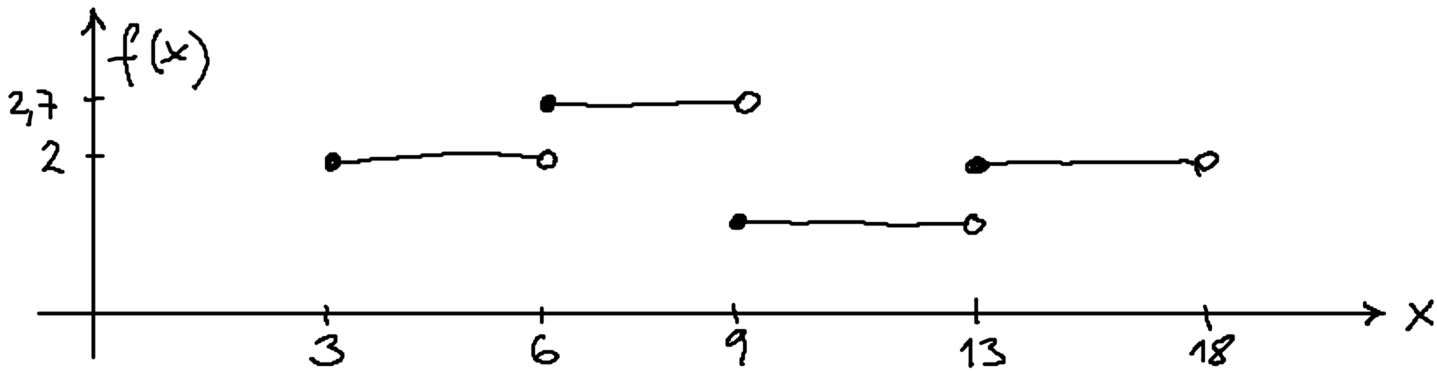
Stückweise konstante Funktionen



$$f(x) = \begin{cases} 2 & 3 \le x < 6 \iff x \in [3,6) \\ 2{,}7 & 6 \le x < 9 \\ \cdot & 9 \le x < 13 \\ \cdot & x < 18 \end{cases}$$

$f : [a,b) \to \mathbb{R}$, bestehend aus endlich vielen konstanten Stücken

Der Einfachheit halber: halboffene Intervalle der Form $[x,y)$.

OPERATIONEN:

- Berechne $f(x)$ für ein gegebenes x.
- Addieren zweier Funktionen f und g. (auf dem gemeinsamen Definitionsbereich)

⋮

[• Einsetzen von f in g: Bestimme $h(x) = g(f(x))$]

Darstellung:

1.) Folge von Sprungstellen

$$a = a_0 < a_1 < a_2 < \dots < a_n = b$$

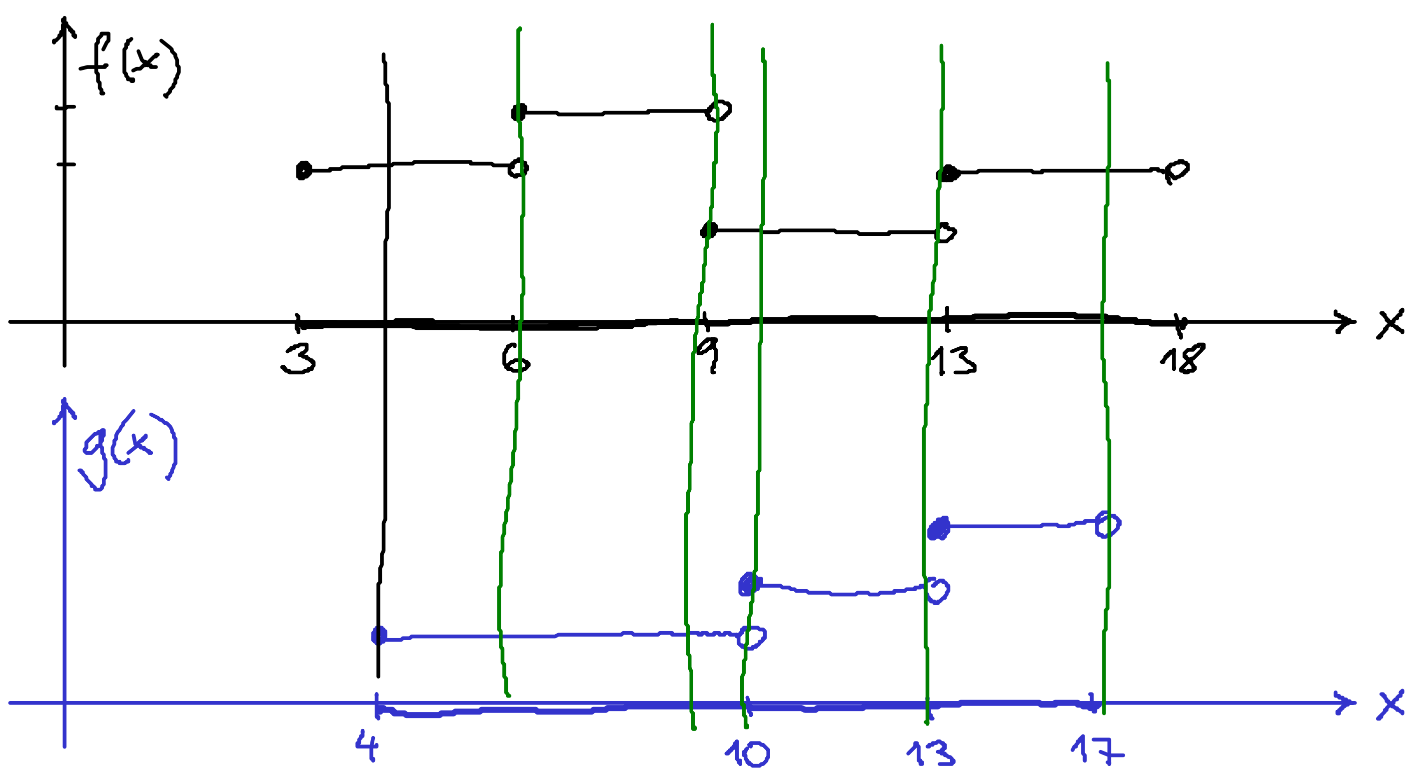
2.) Folge von Werten

$$h_1, h_2, \dots, h_n \in \mathbb{R}$$

für $a_{i-1} \leq x < a_i$: $f(x) = h_i$ $(i = 1, \dots, n)$

$$F = (a_0, h_1, a_1, h_2, \dots, a_{n-1}, h_n, a_n)$$

$F = (a_0)$ (beliebiges a_0) stellt die Funktion mit Definitionsbereich $\emptyset$ dar.



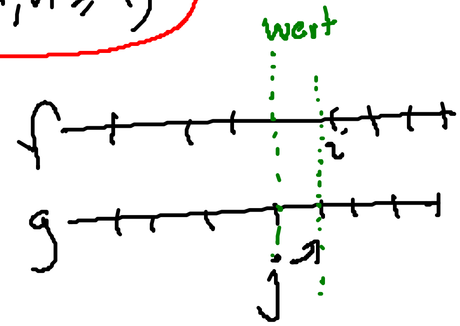
Verschmelzen von $f = (f_0, f_1, \dots, f_{m-1})$ mit $g = (g_0, \dots\ g_{n-1})$ } (beide Sortiert)

als Generatorfunktion mit yield-Anweisung.

ANNAHME $(m, n \geq 1)$

```
def v(f,g):
    i = j = 0; m=len(f); n=len(g)
    wert = min(f_0, g_0)
    while True:
        yield wert
        if i<m and wert==f_i:
            i += 1
        if j<n and wert==g_j:
            j += 1
        if i<m and j<n:
            wert = min(f_i, g_j)
        elif i<m:
            wert = f_i
        elif j<n:
            wert = g_j
        else: return
def verschmelze(f,g):
    return tuple(v(f,g))
```

g[j]



kann auch so $i=m$ oder $j=n$

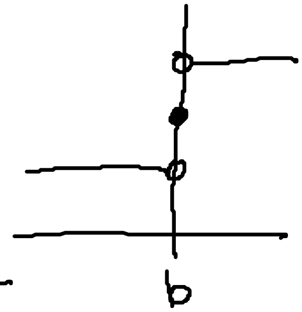
INVARIANTE

$\boxed{\text{wert} = \min(f_i, g_j)}$ und die Element < wert wurden bereits ausgegeben.

"Erweiterte" Zahlen

$[a,b)$ $\quad a \le x < b^+$

$b^+ \le x < c$



$x \le b$

$b < x$

$z = b^+ \qquad b < b^+$

$b^+ < x$ für alle Zahlen $x > b$

gewöhnlichen Zahlen

(b, Seite) $\in \{\text{None}, +1, -1\}$

Wert (normale Zahl)

b^-

$a \le b \iff a = b$ oder $a < b$

$a > b \iff b < a$

$>$

$\neq$

$a \le b^+$

$\iff b^+ \ge a$

$\iff \neg\, b^+ < a \quad \ldots$ __lt__