

High-Order function:

val map: ('a → 'b) → 'a list → 'b list:

'a type: contains everything of any type
[1, 'a, "Hello"] *polymorphic*

int list: [1, 2, 3]

string list: ['a', 'b', ...]

> let function (x: 'a) = x

↳ x can be any type

generate 1 → N:

```
let rec gen start limit =  
  if start > limit then  
    [] ← empty when start > limit  
  else  
    start :: gen (start + 1) limit;
```

let my-list = gen 1 10;

stacked → 1 the 1+1

10
max

$\text{map } f [a_1; \dots; a_n]$

list

function f which would be applied to each element in the list

f :
for each element " a_i " in the list, function f is applied

\Rightarrow would change the whole list

$\text{map } f [a_1; \dots; a_n] = [f(a_1); \dots; f(a_n)]$

let $\text{succ } x = x + 1$ in $\text{map succ } [1; 2; 3]$
 $= [2; 3; 4]$

f : is a function that we could replace after

take 'a' only!

let rec map f list:

let $\text{succ } x = x + 1$

let result = map $[1; 2; 3]$

↳ 'a type
return 'b

let rec map f l =
 match l with
 | [] → []
 | hd::tl → (f hd) :: (map f tl)

iii

$$n = m \quad [n]$$

$$n > m$$

$$n \begin{matrix} [n+1] \\ 0 \end{matrix}$$

$$n < m$$

$$\begin{matrix} & \longleftarrow & \\ m & -1 & n \end{matrix}$$

$$[1; 2; 3]$$

$\underbrace{\quad}_{L_1} \quad \underbrace{\quad}_{L_1}$

$$[4; 5]$$

$\underbrace{\quad}_{L_2}$

$$[1, 2, 3, 1]$$

$\underbrace{\quad}_{L_1}$

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