Anki

q:

Librairie standard:

```
List.hd
```

a:

Signature

```
val hd : 'a list -> 'a
```

Description
Return the first element of the given list

Raises
- `Failure` if list is empty

Anki

q:

Librairie standard:

```
List.nth
```

a:

Signature

```
val nth : 'a list -> int -> 'a
```

Description

Return the `n`-th element of the given list. The first element (head of the list) is at position 0.

Raises

- `Failure` if the list is too short
- `Invalid_argument` if `n` is negative

Remarque:
See also: `List.nth_opt` : retourne une option

Anki

q:

Librairie standard:

```
List.rev
```

a:

Signature

```
val rev : 'a list -> 'a list
```

Description

List reversal.

Anki

q:

Librairie standard:

```
Array.init / List.init
```

a:

Signature

```
val init : int -> (int -> 'a) -> 'a array
```

Description

`init n f` returns a fresh array of length `n`, with element number `i` initialized to the result of `f i`

Raises

- `Invalid_argument` if `n<0`

Anki

q:

Librairie standard:

```
List.concat
```

a:

Signature

```
val concat : 'a list list -> 'a list
```

Description

Concatenate a list of lists. The elements of the argument are all concatenated together (in the same order) to give the result. Not tail-recursive (length of the argument + length of the longest sub-list).

Remarque:
See also: `List.flatten` : same thing

Anki

q:

Librairie standard:

```
List.iter / Array.iter
```

a:

Signature

```
val iter : ('a -> unit) -> 'a list -> unit
```

Description

`iter f [a1; ...; an]` applies function `f` in turn to `[a1; ...; an]`.

It is equivalent to `f a1; f a2; ...; f an`.

Example

```
let print_number n =
  print_endline (string_of_int n)

let () =
  let numbers = [1; 2; 3; 4; 5] in
  List.iter print_number numbers
```

Anki

q:

Librairie standard:

```
Array.iteri / List.iteri
```

a:

Signature

```
val iteri : (int -> 'a -> unit) -> 'a array -> unit
```

Description

Same as `Array.iter`, but the function is applied to the index of the element as first argument, and the element itself as second argument

Example

```
let print_index_value i v =
  Printf.printf "Index %d: Value %d\n" i v

let () =
  let numbers = [|10; 20; 30; 40; 50|] in
  Array.iteri print_index_value numbers
```

Anki

q:

Librairie standard:

`List.map / Array.map`

a:

Signature

```
val map : ('a -> 'b) -> 'a list -> 'b list
```

Description

`map f [a1; ...; an]` applies function `f` to `a1, ..., an`, and builds the list `[f a1; ...; f an]` with the results returned by `f`

Example

```
let double x = x * 2

let () =
  let numbers = [1; 2; 3; 4; 5] in
  let doubled_numbers = List.map double numbers in
  List.iter (Printf.printf "%d ") doubled_numbers;
  print_newline ()
```

Remarque:
See also:

- `List.mapi / Array.mapi` : same, but the first argumen that's passed is the index.
- `List.rev_map` : `rev_map f l` gives the same result as `List.rev (List.map f l)`, but is more efficient.

  Doesn't exist on Array
- `List.filter_map` :

  ```
  val filter_map : ('a -> 'b option) -> 'a list -> 'b
  list
  ```

  `filter_map f l` applies `f` to every element of `l`, filters out the `None` elements and returns the list of the arguments of the `Some` elements **Doesn't exist on Array**

Anki

q:

Librairie standard

```
Array.map_inplace
```

a:

Signature

```
val map_inplace : ('a -> 'a) -> 'a array -> unit
```

Description

`map_inplace f a` applies function `f` to all elements of `a`, and updates their values in place.

Remarque:
See also: `Array.mapi_inplace`

Anki

q:

Librairie standard

```
List.fold_left / Array.fold_left
```

a:

Signature

```
val fold_left : ('acc -> 'a -> 'acc) -> 'acc -> 'a list ->
'acc
```

Description

`fold_left f init [b1; ...; bn]` is
`f (... (f (f init b1) b2) ...) bn`

Examples

```
let () =
  let numbers = [1; 2; 3; 4; 5] in
  let sum = List.fold_left (fun acc x -> acc + x) 0
numbers in
```

```
let () =
  let numbers = [|1; 2; 3; 4; 5|] in
  (* product *)
  Array.fold_left (fun acc x -> acc * x) 1 numbers in
```

Anki

q:

Librairie standard

```
Array.fold_right / List.fold_right
```

a:

Signature

```
val fold_right : ('a -> 'acc -> 'acc) -> 'a array -> 'acc
-> 'acc
```

Description

`Array.fold_right f a` init computes `f a.(0) (f a.(1) ( ... (f a.(n-1) init) ...))`[s], where `n` is the length of the array `a`

Example

```
let subtract x acc = x - acc

let () =
  let numbers = [|10; 20; 30|] in
  let result = Array.fold_right subtract numbers 0 in
  Printf.printf "Result: %d\n" result
```

Remarque:
Attention à l'ordre dans la signature!

- Sur les listes, ça ne fait pas de TCO, c'est donc extremement lent, à ne **pas utiliser sur les listes**!

Anki

q:

Librairie standard

Fonctions sur deux listes et arrays:

```
List.iter2 / Array.iter2
List.map2 / Array.map2
```

a:

iter2

Signature

```
val iter2 : ('a -> 'b -> unit) -> 'a list -> 'b list ->
unit
```

Description

`iter2 f a b` applies function `f` to all the elements of `a` and `b`

Raises

- `Invalid_argument` if the two lists are determined to have different lengths

map2

Signature

```
val map2 : ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c
list
```

Description

`map2 f [a1; ...; an] [b1; ...; bn]` is `[f a1 b1; ...; f an bn]`

Raises

- `Invalid_argument` if the two lists are determined to have different lengths

Anki

q:

Librairie standard

```
List.for_all / Array.for_all
```

a:

Signature

```
val for_all : ('a -> bool) -> 'a list -> bool
```

Description

`for_all f [a1; ...; an]` checks if all elements of the list satisfy the predicate `f`. That is, it returns `(f a1) && (f a2) && ... && (f an)` for a non-empty list and `true` if the list is empty

Remarque:
See also

- `Array.for_all2` :
  ```
  val for_all2 : ('a -> 'b -> bool) -> 'a array -> 'b array -> bool
  ```
- `Array.exists / List.exists` :
  ```
  val exists : ('a -> bool) -> 'a array -> bool
  ```
  which checks if **at least one element** satisfies the predicate (stops when it finds one)

Anki

q:

Librairie standard

```
List.mem / Array.mem
```

a:

Signature

```
val mem : 'a -> 'a list -> bool
```

Description

`mem a set` is `true` if and only if `a` is structurally equal to an element of set (i.e. there is an `x` in set such that `compare a x = 0`)

Remarque:

See also: `Array.memq / List.memq` : same, but uses physical equality instead of structural equality to compare array/list elements

Anki

q:

Librairie standard

```
List.find / Array.find
List.find_opt / Array.find_opt
List.find_index / Array.find_index
```

a:

find

Signature

```
val find : ('a -> bool) -> 'a list -> 'a
```

Description

`find f l` returns the first element of the list `l` that satisfies the predicate `f`

Raises

- `Not_found` if there is no value that satisfies `f` in the list `l`

find_opt

Signature

```
val find_opt : ('a -> bool) -> 'a list -> 'a option
```

Description

Same as `find`, but as an option

find_index

Signature

```
val find_index : ('a -> bool) -> 'a list -> int option
```

Description

Same as `find_opt`, but returns index instead.

Remarque:
See also:

- `List.find_map / Array.find_map` :
  `val find_map : ('a -> 'b option) -> 'a list -> 'b option`

  `find_map f l` applies `f` to the elements of `l` in order, and returns the first result of the form `Some v`, or `None` if none exist.

  `mapi` also exists, it passed the index as the first argument to `f`

Anki

q:

Librairie standard

```
List.filter
```

a:

Doesn't exist on Array

Signature

```
val filter : ('a -> bool) -> 'a list -> 'a list
```

Description

`filter f l` returns all the elements of the list `l` that satisfy the predicate `f`. The order of the elements in the input list is preserved

Remarque:

- `find_all` does the same thing
- `filteri` also exists, it takes the index as a first parameter

Anki

q:

Librairie standard

```
List.equal
```

a:

Doesn't exist on Array

Signature

```
val equal : ('a -> 'a -> bool) -> 'a list -> 'a list ->
bool
```

Description

`equal eq [a1; ...; an] [b1; ..; bm]` holds when the two input lists have the same length, and for each pair of elements `ai`, `bi` at the same position we have `eq ai bi`

`eq` can be called even if the two lists don't have the same length, if it's costly, you might want to check `List.compare_lengths` first

Anki

q:

Librairie standard

```
List.sort
List.stable_sort
List.fast_sort
List.sort_uniq
List.merge
```

a:

Generally, just passing the standard `compare` function is a good comparison function.

`List.sort`

Signature

```
val sort : ('a -> 'a -> int) -> 'a list -> 'a list
```

Description

- The current implementation uses Merge Sort. It runs in constant heap space and logarithmic stack space.
- $O(n \log n)$ time complexity

As of writing this in April 2025, the `sort` function is just `stable_sort`

`List.stable_sort`

Signature

```
val stable_sort : ('a -> 'a -> int) -> 'a list -> 'a list
```

Description

- sorting algorithm is guaranteed to be stable
- The current implementation uses Merge Sort. It runs in constant heap space and logarithmic stack space.
- $O(n \log n)$ time complexity

`List.fast_sort`

Signature

```
val fast_sort : ('a -> 'a -> int) -> 'a list -> 'a list
```

Description

Same as `List.sort` or `List.stable_sort`, whichever is faster on typical input.

`List.sort_uniq`

Signature

```
val sort_uniq : ('a -> 'a -> int) -> 'a list -> 'a list
```

Description

Same as `List.sort`, but also remove duplicates.

`List.merge`

Signature

```
val merge : ('a -> 'a -> int) -> 'a list -> 'a list -> 'a
list
```

Description

Merge two lists: Assuming that `l1` and `l2` are sorted according to the comparison function `cmp`, merge cmp `l1` `l2` will return a sorted list containing all the elements of `l1` and `l2`.

If several elements compare equal, the elements of `l1` will be before the elements of `l2`. Not tail-recursive (sum of the lengths of the arguments).

Anki

q:

Librairie standard

```
Array.sort
Array.stable_sort
Array.fast_sort
```

a:

All the sorts are done in place

`Array.sort`

Signature

```
val sort : ('a -> 'a -> int) -> 'a array -> unit
```

Description

- The current implementation uses Heap Sort. It runs in constant stack space and (at most) logarithmic stack space
- $O(n \log n)$ time complexity

`Array.stable_sort`

Signature

```
val stable_sort : ('a -> 'a -> int) -> 'a array -> unit
```

Description

- sorting algorithm is guaranteed to be stable
- $O(n \log n)$ time complexity
- The current implementation uses Merge Sort

It uses a temporary array of length `n/2`, where `n` is the length of the array. It is usually faster than the current implementation of `Array.sort`

`Array.fast_sort`

Signature

```
val fast_sort : ('a -> 'a -> int) -> 'a array -> unit
```

Description

Same as `Array.sort` or `Array.stable_sort`, whichever is faster on typical input.

Anki

q:

Librairie standard

```
Array.length / List.length
```

a:

Signature

```
val length : 'a array -> int
```

Description

Return the length (number of elements) of the given array

Anki

q:

Librairie standard

```
Array.make
Array.make_matrix
```

a:

Array.make

Signature

```
val make : int -> 'a -> 'a array
```

Description

- `make n x` returns a fresh array of length `n`, initialized with `x`
- initially all items are physically equal: if `x` is mutable, then it will be shared with all the elements of the array, and modifying `x` through one of the array entries will modify all other entries at the same time

Raises

- `Invalid_argument` if `n < 0` or `n > Sys.max_array_length`. If the value of `x` is a floating-point number, then the maximum size is only `Sys.max_array_length / 2`

Array.make_matrix

Signature

```
val make_matrix : int -> int -> 'a -> 'a array array
```

Description

- `make_matrix dimx dimy e` returns a two-dimensional array (an array of arrays) with first dimension `dimx` and second dimension `dimy`.
- All the elements of this new matrix are initially physically equal to `e`.
- The element `(x,y)` of a matrix `m` is accessed with the notation `m.(x).(y)`

Raises

- `Invalid_argument` if `dimx` or `dimy` is negative or greater than `Sys.max_array_length`. If the value of `e` is a floating-point number, then the maximum size is only `Sys.max_array_length / 2`

Anki

q:

Librairie standard

```
Array.append
Array.concat
```

a:

Array.append

Signature

```
val append : 'a array -> 'a array -> 'a array
```

Description

`append v1 v2` returns a fresh array containing the concatenation of the arrays `v1` and `v2`

Raises

- `Invalid_argument` if `length v1 + length v2 > Sys.max_array_length`

Array.concat

Signature

```
val concat : 'a array list -> 'a array
```

Description

Same as `Array.append`, but concatenates a list of arrays

Anki

q:

Librairie standard

```
Array.copy
```

a:

Signature

```
val copy : 'a array -> 'a array
```

Description

`copy a` returns a copy of `a`, that is, a fresh array containing the same elements as `a`.

Remarque:
Pour cloner une liste, faire `List.take n l` avec `n > length l`

Anki

q:

Librairie standard

```
Array.sub
```

a:

Signature

```
val sub : 'a array -> int -> int -> 'a array
```

Description

`sub a pos len` returns a fresh array of length `len`, containing the elements number `pos` to `pos + len - 1` of array `a`

Raises

- `Invalid_argument` if `pos` and `len` do not designate a valid subarray of `a`; that is, if `pos < 0`, or `len < 0`, or `pos + len > length a`

Anki

q:

Librairie standard

```
Array.fill
```

a:

Signature

```
val fill : 'a array -> int -> int -> 'a -> unit
```

Description

`fill a pos len x` modifies the array `a` in place, storing `x` in elements number `pos` to `pos + len - 1`

Raises

- `Invalid_argument` if `pos` and `len` do not designate a valid subarray of `a`

Anki

q:

Librairie standard

Comment convertir une liste en array, ou inversement?

a:

```
Array.to_list
Array.of_list
```

Anki

q:

Librairie standard

```
String.empty
String.init
```

a:

Signature

```
val empty : string
```

Description

The empty string

Signature

```
val init : int -> (int -> char) -> string
```

Description

Init a string of length `n` with index `i` holding the character `f i`

Anki

q:

Librairie standard

```
^
String.cat
String.concat
```

a:

Signature

```
val cat : string -> string -> string
```

Description

Concatenates string 1 and string 2

Signature

```
val concat : string -> string list -> string
```

Description

`concat sep ss` concatenates the list of strings `ss` inserting `sep` between them

Anki

q:

Librairie standard

```
String.starts_with
String.ends_with
```

a:

Signature

```
val starts_with : prefix:string -> string -> bool
```

Description

Returns true if the string starts with the given prefix

Example

```
let s = "ehello world" in
String.starts_with ~prefix:"hello" s
```

Signature

```
val ends_with : suffix:string -> string -> bool
```

Description

Returns true if the string ends with the given prefix

Example

```
String.ends_with ~suffix:"perdu" "J'ai perdu";;
- : bool = true
```

Anki

q:

Librairie standard

```
String.split_on_char
```

a:

Signature

```
val split_on_char : char -> string -> string list
```

Description

All possibly empty substrings of `s` delimited by the character `sep`

- If there are no results, returns the singleton list `[""]`

Two invariants

- The list is not empty
- concatenating the elements with `sep` returns the original string
- no string in the result contains `sep`

Anki

q:

Librairie standard

```
String.trim
```

a:

Signature

```
val trim : string -> string
```

Description

`trim s` is `s` without leading and trailing whitespace. Whitespace characters are: `' '`, `'\x0C'` (form feed), `'\n'`, `'\r'`, and `'\t'`.

Anki

q:

Librairie standard

```
String.iter
```

a:

Signature

```
val iter : (char -> unit) -> string -> unit
```

Description

`iter f s` applies function `f` in turn to all the characters of `s`. It is equivalent to `f s.[0]; f s.[1]; ...; f s.[length s - 1]; ()`.

Remarque:
`String.iteri` also exists