Batteries
BatArray.Labels.LExceptionless.find : f:('a -> bool) -> 'a BatArray.t -> 'a option
BatArray.Labels.LExceptionless.findi : f:('a -> bool) -> 'a BatArray.t -> int option
BatArray.Labels.blit : src:'a array -> src_pos:int -> dst:'a array -> dst_pos:int -> len:int -> unit
BatArray.Labels.count_matching : f:('a -> bool) -> 'a BatArray.t -> int
BatArray.Labels.create : int -> init:'a -> 'a array
BatArray.Labels.create_matrix : dimx:int -> dimy:int -> 'a -> 'a array array
BatArray.Labels.exists : f:('a -> bool) -> 'a BatArray.t -> bool
BatArray.Labels.fast_sort : cmp:('a -> 'a -> int) -> 'a array -> unit
BatArray.Labels.fill : 'a array -> pos:int -> len:int -> 'a -> unit
BatArray.Labels.filter : f:('a -> bool) -> 'a BatArray.t -> 'a BatArray.t
BatArray.Labels.filter_map : f:('a -> 'b option) -> 'a BatArray.t -> 'b BatArray.t
BatArray.Labels.find : f:('a -> bool) -> 'a BatArray.t -> 'a
BatArray.Labels.find_map : f:('a -> 'b option) -> 'a array -> 'b option
BatArray.Labels.find_opt : f:('a -> bool) -> 'a BatArray.t -> 'a option
BatArray.Labels.findi : f:('a -> bool) -> 'a BatArray.t -> int
BatArray.Labels.fold : f:('a -> 'b -> 'a) -> init:'a -> 'b array -> 'a
BatArray.Labels.fold_left : f:('a -> 'b -> 'a) -> init:'a -> 'b array -> 'a
BatArray.Labels.fold_right : f:('b -> 'a -> 'a) -> 'b array -> init:'a -> 'a
BatArray.Labels.fold_while : p:('acc -> 'a -> bool) -> f:('acc -> 'a -> 'acc) -> init:'acc -> 'a array -> 'acc * int
BatArray.Labels.for_all : f:('a -> bool) -> 'a BatArray.t -> bool
BatArray.Labels.init : int -> f:(int -> 'a) -> 'a array
BatArray.Labels.iter : f:('a -> unit) -> 'a array -> unit
BatArray.Labels.iter2 : f:('a -> 'b -> unit) -> 'a BatArray.t -> 'b BatArray.t -> unit
BatArray.Labels.iter2i : f:(int -> 'a -> 'b -> unit) -> 'a BatArray.t -> 'b BatArray.t -> unit
BatArray.Labels.iteri : f:(int -> 'a -> unit) -> 'a array -> unit
BatArray.Labels.make_matrix : dimx:int -> dimy:int -> 'a -> 'a array array
BatArray.Labels.map : f:('a -> 'b) -> 'a BatArray.t -> 'b BatArray.t
BatArray.Labels.mapi : f:(int -> 'a -> 'b) -> 'a BatArray.t -> 'b BatArray.t
BatArray.Labels.modify : f:('a -> 'a) -> 'a array -> unit
BatArray.Labels.modifyi : f:(int -> 'a -> 'a) -> 'a array -> unit
BatArray.Labels.sort : cmp:('a -> 'a -> int) -> 'a array -> unit
BatArray.Labels.stable_sort : cmp:('a -> 'a -> int) -> 'a array -> unit
BatArray.Labels.sub : 'a array -> pos:int -> len:int -> 'a array
BatArray.Cap.Exceptionless.find : ('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> 'a option

Batteries
BatArray.Cap.Exceptionless.findi : ('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> int option
BatArray.Cap.Labels.blit : src:('a, [< `Read | `Write > `Read ]) t -> src_pos:int -> dst:('a, [< `Read | `Write > `Write]) t -> dst_pos:int -> len:int -> unit
BatArray.Cap.Labels.count_matching : f:('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> int
BatArray.Cap.Labels.create : int -> init:'a -> ('a, [< `Read | `Write]) t
BatArray.Cap.Labels.create_matrix : dimx:int -> dimy:int -> 'a -> (('a, [< `Read | `Write]) t, [< `Read | `Write]) t
BatArray.Cap.Labels.exists : f:('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> bool
BatArray.Cap.Labels.fast_sort : cmp:('a -> 'a -> int) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.Labels.fill : ('a, [< `Read | `Write > `Write]) t -> pos:int -> len:int -> 'a -> unit
BatArray.Cap.Labels.filter : f:('a -> bool) -> ('a, [< `Read | `Write > `Read ]) t -> ('a, [< `Read | `Write]) t
BatArray.Cap.Labels.filter_map : f:('a -> 'b option) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write]) t
BatArray.Cap.Labels.find : f:('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> 'a
BatArray.Cap.Labels.find_map : f:('a -> 'b option) -> ('a, [< `Read | `Write > `Read]) t -> 'b option
BatArray.Cap.Labels.find_opt : f:('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> 'a option
BatArray.Cap.Labels.fold : f:('a -> 'b -> 'a) -> init:'a -> ('b, [< `Read | `Write > `Read]) t -> 'a
BatArray.Cap.Labels.fold_left : f:('a -> 'b -> 'a) -> init:'a -> ('b, [< `Read | `Write > `Read]) t -> 'a
BatArray.Cap.Labels.fold_right : f:('b -> 'a -> 'a) -> ('b, [< `Read | `Write > `Read]) t -> init:'a -> 'a
BatArray.Cap.Labels.fold_while : p:('acc -> 'a -> bool) -> f:('acc -> 'a -> 'acc) -> init:'acc -> 'a array -> 'acc * int
BatArray.Cap.Labels.for_all : f:('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> bool
BatArray.Cap.Labels.init : int -> f:(int -> 'a) -> ('a, [< `Read | `Write]) t
BatArray.Cap.Labels.iter : f:('a -> unit) -> ('a, [< `Read | `Write > `Read]) t -> unit
BatArray.Cap.Labels.iter2 : f:('a -> 'b -> unit) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write > `Read]) t -> unit
BatArray.Cap.Labels.iter2i : f:(int -> 'a -> 'b -> unit) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write > `Read]) t -> unit
BatArray.Cap.Labels.iteri : f:(int -> 'a -> unit) -> ('a, [< `Read | `Write > `Read]) t -> unit
BatArray.Cap.Labels.make : int -> init:'a -> ('a, [< `Read | `Write]) t
BatArray.Cap.Labels.make_matrix : dimx:int -> dimy:int -> 'a -> (('a, [< `Read | `Write]) t, [< `Read | `Write]) t
BatArray.Cap.Labels.map : f:('a -> 'b) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write]) t
BatArray.Cap.Labels.mapi : f:(int -> 'a -> 'b) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write]) t
BatArray.Cap.Labels.modify : f:('a -> 'a) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.Labels.modifyi : f:(int -> 'a -> 'a) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.Labels.sort : cmp:('a -> 'a -> int) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.Labels.stable_sort : cmp:('a -> 'a -> int) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.Labels.sub : ('a, [< `Read | `Write > `Read ]) t -> pos:int -> len:int -> ('a, [< `Read | `Write]) t
BatArray.Cap.append : ('a, [< `Read | `Write > `Read ]) t -> ('a, [< `Read | `Write > `Read]) t -> ('a, [< `Read | `Write]) t
BatArray.Cap.backwards : ('a, [< `Read | `Write > `Read]) t -> 'a BatEnum.t

Batteries
BatArray.Cap.blit : ('a, [< `Read | `Write > `Read ]) t -> int -> ('a, [< `Read | `Write > `Write]) t -> int -> int -> unit
BatArray.Cap.combine : ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write > `Read]) t -> ('a * 'b, [< `Read | `Write > `Read]) t
BatArray.Cap.compare : 'a BatOrd.comp -> ('a, [< `Read | `Write > `Read]) t BatOrd.comp
BatArray.Cap.concat : ('a, [< `Read | `Write > `Read ]) t list -> ('a, [< `Read | `Write]) t
BatArray.Cap.copy : ('a, [< `Read | `Write > `Read]) t -> 'a array
BatArray.Cap.count_matching : ('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> int
BatArray.Cap.create_matrix : int -> int -> 'a -> (('a, [< `Read | `Write]) t, [< `Read | `Write]) t
BatArray.Cap.enum : ('a, [< `Read | `Write > `Read]) t -> 'a BatEnum.t
BatArray.Cap.equal : 'a BatOrd.eq -> ('a, [< `Read | `Write > `Read]) t BatOrd.eq
BatArray.Cap.exists : ('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> bool
BatArray.Cap.fast_sort : ('a -> 'a -> int) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.fill : ('a, [< `Read | `Write > `Write]) t -> int -> int -> 'a -> unit
BatArray.Cap.filter : ('a -> bool) -> ('a, [< `Read | `Write > `Read ]) t -> ('a, [< `Read | `Write]) t
BatArray.Cap.filter_map : ('a -> 'b option) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write]) t
BatArray.Cap.find : ('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> 'a
BatArray.Cap.find_all : ('a -> bool) -> ('a, [< `Read | `Write > `Read ]) t -> ('a, [< `Read | `Write]) t
BatArray.Cap.find_map : ('a -> 'b option) -> ('a, [< `Read | `Write > `Read]) t -> 'b option
BatArray.Cap.find_opt : ('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> 'a option
BatArray.Cap.findi : ('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> int
BatArray.Cap.fold : ('a -> 'b -> 'a) -> 'a -> ('b, [< `Read | `Write > `Read]) t -> 'a
BatArray.Cap.fold_left : ('a -> 'b -> 'a) -> 'a -> ('b, [< `Read | `Write > `Read]) t -> 'a
BatArray.Cap.fold_right : ('b -> 'a -> 'a) -> ('b, [< `Read | `Write > `Read]) t -> 'a -> 'a
BatArray.Cap.fold_while : ('acc -> 'a -> bool) -> ('acc -> 'a -> 'acc) -> 'acc -> ('a, [< `Read | `Write > `Read]) t -> 'acc * int
BatArray.Cap.for_all : ('a -> bool) -> ('a, [< `Read | `Write > `Read]) t -> bool
BatArray.Cap.init : int -> (int -> 'a) -> ('a, [< `Read | `Write]) t
BatArray.Cap.iter : ('a -> unit) -> ('a, [< `Read | `Write > `Read]) t -> unit
BatArray.Cap.iter2 : ('a -> 'b -> unit) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write > `Read]) t -> unit
BatArray.Cap.iter2i : (int -> 'a -> 'b -> unit) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write > `Read]) t -> unit
BatArray.Cap.iteri : (int -> 'a -> unit) -> ('a, [< `Read | `Write > `Read]) t -> unit
BatArray.Cap.make_matrix : int -> int -> 'a -> (('a, [< `Read | `Write]) t, [< `Read | `Write]) t
BatArray.Cap.map : ('a -> 'b) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write]) t
BatArray.Cap.mapi : (int -> 'a -> 'b) -> ('a, [< `Read | `Write > `Read ]) t -> ('b, [< `Read | `Write]) t
BatArray.Cap.mem : 'a -> ('a, [< `Read | `Write > `Read]) t -> bool
BatArray.Cap.memq : 'a -> ('a, [< `Read | `Write > `Read]) t -> bool

Batteries
BatArray.Cap.modify : ('a -> 'a) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.modifyi : (int -> 'a -> 'a) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.of_backwards : 'a BatEnum.t -> ('a, [< `Read | `Write]) t
BatArray.Cap.of_enum : 'a BatEnum.t -> ('a, [< `Read | `Write]) t
BatArray.Cap.of_list : 'a list -> ('a, [< `Read | `Write]) t
BatArray.Cap.ord : 'a BatOrd.ord -> ('a, [< `Read | `Write > `Read]) t BatOrd.ord
BatArray.Cap.partition : ('a -> bool) -> ('a, [< `Read | `Write > `Read ]) t -> ('a, [< `Read | `Write ]) t * ('a, [< `Read | `Write]) t
BatArray.Cap.pivot_split : 'a BatOrd.ord -> ('a, [< `Read | `Write > `Read]) t -> 'a -> int * int
BatArray.Cap.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatIO.output -> 'b -> unit) -> 'a BatIO.output -> ('b, [< `Read | `Write > `Read]) t -> unit
BatArray.Cap.rev : ('a, [< `Read | `Write > `Read ]) t -> ('a, [< `Read | `Write]) t
BatArray.Cap.rev_in_place : ('a, [`Read | `Write]) t -> unit
BatArray.Cap.sort : ('a -> 'a -> int) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.split : ('a * 'b, [< `Read | `Write > `Read ]) t -> ('a, [< `Read | `Write ]) t * ('b, [< `Read | `Write]) t
BatArray.Cap.stable_sort : ('a -> 'a -> int) -> ('a, [`Read | `Write]) t -> unit
BatArray.Cap.sub : ('a, [< `Read | `Write > `Read ]) t -> int -> int -> ('a, [< `Read | `Write]) t
BatArray.Cap.to_list : ('a, [< `Read | `Write > `Read]) t -> 'a list
BatArray.Exceptionless.find : ('a -> bool) -> 'a t -> 'a option
BatArray.Exceptionless.findi : ('a -> bool) -> 'a t -> int option
BatArray.Labels.LExceptionless.find : f:('a -> bool) -> 'a t -> 'a option
BatArray.Labels.LExceptionless.findi : f:('a -> bool) -> 'a t -> int option
BatArray.Labels.blit : src:'a array -> src_pos:int -> dst:'a array -> dst_pos:int -> len:int -> unit
BatArray.Labels.count_matching : f:('a -> bool) -> 'a t -> int
BatArray.Labels.create : int -> init:'a -> 'a array
BatArray.Labels.create_matrix : dimx:int -> dimy:int -> 'a -> 'a array array
BatArray.Labels.exists : f:('a -> bool) -> 'a t -> bool
BatArray.Labels.fast_sort : cmp:('a -> 'a -> int) -> 'a array -> unit
BatArray.Labels.fill : 'a array -> pos:int -> len:int -> 'a -> unit
BatArray.Labels.filter : f:('a -> bool) -> 'a t -> 'a t
BatArray.Labels.filter_map : f:('a -> 'b option) -> 'a t -> 'b t
BatArray.Labels.find : f:('a -> bool) -> 'a t -> 'a
BatArray.Labels.find_map : f:('a -> 'b option) -> 'a array -> 'b option
BatArray.Labels.find_opt : f:('a -> bool) -> 'a t -> 'a option
BatArray.Labels.findi : f:('a -> bool) -> 'a t -> int
BatArray.Labels.fold : f:('a -> 'b -> 'a) -> init:'a -> 'b array -> 'a

Batteries
BatArray.Labels.fold_left : f:('a -> 'b -> 'a) -> init:'a -> 'b array -> 'a
BatArray.Labels.fold_right : f:('b -> 'a -> 'a) -> 'b array -> init:'a -> 'a
BatArray.Labels.fold_while : p:('acc -> 'a -> bool) -> f:('acc -> 'a -> 'acc) -> init:'acc -> 'a array -> 'acc * int
BatArray.Labels.for_all : f:('a -> bool) -> 'a t -> bool
BatArray.Labels.init : int -> f:(int -> 'a) -> 'a array
BatArray.Labels.iter : f:('a -> unit) -> 'a array -> unit
BatArray.Labels.iter2 : f:('a -> 'b -> unit) -> 'a t -> 'b t -> unit
BatArray.Labels.iter2i : f:(int -> 'a -> 'b -> unit) -> 'a t -> 'b t -> unit
BatArray.Labels.iteri : f:(int -> 'a -> unit) -> 'a array -> unit
BatArray.Labels.make_matrix : dimx:int -> dimy:int -> 'a -> 'a array array
BatArray.Labels.map : f:('a -> 'b) -> 'a t -> 'b t
BatArray.Labels.mapi : f:(int -> 'a -> 'b) -> 'a t -> 'b t
BatArray.Labels.modify : f:('a -> 'a) -> 'a array -> unit
BatArray.Labels.modifyi : f:(int -> 'a -> 'a) -> 'a array -> unit
BatArray.Labels.sort : cmp:('a -> 'a -> int) -> 'a array -> unit
BatArray.Labels.stable_sort : cmp:('a -> 'a -> int) -> 'a array -> unit
BatArray.Labels.sub : 'a array -> pos:int -> len:int -> 'a array
BatArray.append : 'a array -> 'a array -> 'a array
BatArray.avg : int array -> float
BatArray.backwards : 'a array -> 'a BatEnum.t
BatArray.blit : 'a array -> int -> 'a array -> int -> int -> unit
BatArray.bsearch : 'a BatOrd.ord -> 'a array -> 'a -> [`All_bigger | `All_lower | `At of int | `Empty | `Just_after of int]
BatArray.cartesian_product : 'a array -> 'b array -> ('a * 'b) array
BatArray.combine : 'a array -> 'b array -> ('a * 'b) array
BatArray.compare : 'a BatOrd.comp -> 'a array BatOrd.comp
BatArray.concat : 'a array list -> 'a array
BatArray.copy : 'a array -> 'a array
BatArray.count_matching : ('a -> bool) -> 'a array -> int
BatArray.create_matrix : int -> int -> 'a -> 'a array array
BatArray.decorate_fast_sort : ('a -> 'b) -> 'a array -> 'a array
BatArray.decorate_stable_sort : ('a -> 'b) -> 'a array -> 'a array
BatArray.enum : 'a array -> 'a BatEnum.t
BatArray.equal : 'a BatOrd.eq -> 'a array BatOrd.eq
BatArray.exists : ('a -> bool) -> 'a array -> bool

Batteries
BatArray.exists2 : ('a -> 'b -> bool) -> 'a array -> 'b array -> bool
BatArray.fast_sort : ('a -> 'a -> int) -> 'a array -> unit
BatArray.favg : float array -> float
BatArray.fill : 'a array -> int -> int -> 'a -> unit
BatArray.filter : ('a -> bool) -> 'a array -> 'a array
BatArray.filter_map : ('a -> 'b option) -> 'a array -> 'b array
BatArray.filteri : (int -> 'a -> bool) -> 'a array -> 'a array
BatArray.find : ('a -> bool) -> 'a array -> 'a
BatArray.find_all : ('a -> bool) -> 'a array -> 'a array
BatArray.find_map : ('a -> 'b option) -> 'a array -> 'b option
BatArray.find_opt : ('a -> bool) -> 'a array -> 'a option
BatArray.findi : ('a -> bool) -> 'a array -> int
BatArray.fold : ('a -> 'b -> 'a) -> 'a -> 'b array -> 'a
BatArray.fold_left : ('a -> 'b -> 'a) -> 'a -> 'b array -> 'a
BatArray.fold_left_map : ('a -> 'b -> 'a * 'c) -> 'a -> 'b array -> 'a * 'c array
BatArray.fold_lefti : ('a -> int -> 'b -> 'a) -> 'a -> 'b array -> 'a
BatArray.fold_right : ('b -> 'a -> 'a) -> 'b array -> 'a -> 'a
BatArray.fold_righti : (int -> 'b -> 'a -> 'a) -> 'b array -> 'a -> 'a
BatArray.fold_while : ('acc -> 'a -> bool) -> ('acc -> 'a -> 'acc) -> 'acc -> 'a array -> 'acc * int
BatArray.for_all : ('a -> bool) -> 'a array -> bool
BatArray.for_all2 : ('a -> 'b -> bool) -> 'a array -> 'b array -> bool
BatArray.fsum : float array -> float
BatArray.head : 'a array -> int -> 'a array
BatArray.init : int -> (int -> 'a) -> 'a array
BatArray.insert : 'a array -> 'a -> int -> 'a array
BatArray.is_sorted_by : ('a -> 'b) -> 'a array -> bool
BatArray.iter : ('a -> unit) -> 'a array -> unit
BatArray.iter2 : ('a -> 'b -> unit) -> 'a array -> 'b array -> unit
BatArray.iter2i : (int -> 'a -> 'b -> unit) -> 'a array -> 'b array -> unit
BatArray.iteri : (int -> 'a -> unit) -> 'a array -> unit
BatArray.kahan_sum : float array -> float
BatArray.left : 'a array -> int -> 'a array
BatArray.make_float : int -> float array
BatArray.make_matrix : int -> int -> 'a -> 'a array array

Batteries
BatArray.map : ('a -> 'b) -> 'a array -> 'b array
BatArray.map2 : ('a -> 'b -> 'c) -> 'a array -> 'b array -> 'c array
BatArray.mapi : (int -> 'a -> 'b) -> 'a array -> 'b array
BatArray.max : 'a array -> 'a
BatArray.mem : 'a -> 'a array -> bool
BatArray.memq : 'a -> 'a array -> bool
BatArray.min : 'a array -> 'a
BatArray.min_max : 'a array -> 'a * 'a
BatArray.modify : ('a -> 'a) -> 'a array -> unit
BatArray.modifyi : (int -> 'a -> 'a) -> 'a array -> unit
BatArray.of_backwards : 'a BatEnum.t -> 'a array
BatArray.of_enum : 'a BatEnum.t -> 'a array
BatArray.of_list : 'a list -> 'a array
BatArray.of_seq : 'a Seq.t -> 'a array
BatArray.ord : 'a BatOrd.ord -> 'a array BatOrd.ord
BatArray.partition : ('a -> bool) -> 'a array -> 'a array * 'a array
BatArray.pivot_split : 'a BatOrd.ord -> 'a array -> 'a -> int * int
BatArray.print : ?first:string -> ?last:string -> ?sep:string -> ('a, 'b) BatIO.printer -> ('a t, 'b) BatIO.printer
BatArray.range : 'a array -> int BatEnum.t
BatArray.reduce : ('a -> 'a -> 'a) -> 'a array -> 'a
BatArray.remove_at : int -> 'a array -> 'a array
BatArray.rev : 'a array -> 'a array
BatArray.rev_in_place : 'a array -> unit
BatArray.right : 'a array -> int -> 'a array
BatArray.shuffle : ?state:Random.State.t -> 'a array -> unit
BatArray.singleton : 'a -> 'a array
BatArray.sort : ('a -> 'a -> int) -> 'a array -> unit
BatArray.split : ('a * 'b) array -> 'a array * 'b array
BatArray.stable_sort : ('a -> 'a -> int) -> 'a array -> unit
BatArray.sub : 'a array -> int -> int -> 'a array
BatArray.sum : int array -> int
BatArray.tail : 'a array -> int -> 'a array
BatArray.to_list : 'a array -> 'a list
BatArray.to_seq : 'a array -> 'a Seq.t

Batteries
BatArray.to_seqi : 'a array -> (int * 'a) Seq.t
BatList.Labels.LExceptionless.assoc : 'a -> ('a * 'b) list -> 'b option
BatList.Labels.LExceptionless.assoc_inv : 'b -> ('a * 'b) list -> 'a option
BatList.Labels.LExceptionless.assq : 'a -> ('a * 'b) list -> 'b option
BatList.Labels.LExceptionless.at : 'a list -> int -> [`Invalid_argument of string | `Ok of 'a]
BatList.Labels.LExceptionless.find : f:('a -> bool) -> 'a list -> 'a option
BatList.Labels.LExceptionless.findi : f:(int -> 'a -> bool) -> 'a list -> (int * 'a) option
BatList.Labels.LExceptionless.rfind : f:('a -> bool) -> 'a list -> 'a option
BatList.Labels.LExceptionless.split_at : int -> 'a list -> [`Invalid_argument of string | `Ok of 'a list * 'a list]
BatList.Labels.concat_map : f:('a -> 'b list) -> 'a list -> 'b list
BatList.Labels.count_matching : f:('a -> bool) -> 'a list -> int
BatList.Labels.drop_while : f:('a -> bool) -> 'a list -> 'a list
BatList.Labels.exists : f:('a -> bool) -> 'a list -> bool
BatList.Labels.exists2 : f:('a -> 'b -> bool) -> 'a list -> 'b list -> bool
BatList.Labels.fast_sort : ?cmp:('a -> 'a -> int) -> 'a list -> 'a list
BatList.Labels.filter : f:('a -> bool) -> 'a list -> 'a list
BatList.Labels.filter_map : f:('a -> 'b option) -> 'a list -> 'b list
BatList.Labels.find : f:('a -> bool) -> 'a list -> 'a
BatList.Labels.find_all : f:('a -> bool) -> 'a list -> 'a list
BatList.Labels.find_exn : f:('a -> bool) -> exn -> 'a list -> 'a
BatList.Labels.find_map_opt : f:('a -> 'b option) -> 'a list -> 'b option
BatList.Labels.findi : f:(int -> 'a -> bool) -> 'a list -> int * 'a
BatList.Labels.fold : f:('a -> 'b -> 'a) -> init:'a -> 'b list -> 'a
BatList.Labels.fold_left : f:('a -> 'b -> 'a) -> init:'a -> 'b list -> 'a
BatList.Labels.fold_left2 : f:('a -> 'b -> 'c -> 'a) -> init:'a -> 'b list -> 'c list -> 'a
BatList.Labels.fold_right : f:('a -> 'b -> 'b) -> 'a list -> init:'b -> 'b
BatList.Labels.fold_right2 : f:('a -> 'b -> 'c -> 'c) -> 'a list -> 'b list -> init:'c -> 'c
BatList.Labels.for_all : f:('a -> bool) -> 'a list -> bool
BatList.Labels.for_all2 : f:('a -> 'b -> bool) -> 'a list -> 'b list -> bool
BatList.Labels.init : int -> f:(int -> 'a) -> 'a list
BatList.Labels.iter : f:('a -> unit) -> 'a list -> unit
BatList.Labels.iter2 : f:('a -> 'b -> unit) -> 'a list -> 'b list -> unit
BatList.Labels.iteri : f:(int -> 'a -> unit) -> 'a list -> unit
BatList.Labels.map : f:('a -> 'b) -> 'a list -> 'b list

Batteries
BatList.Labels.map2 : f:('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
BatList.Labels.mapi : f:(int -> 'a -> 'b) -> 'a list -> 'b list
BatList.Labels.merge : ?cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
BatList.Labels.partition : f:('a -> bool) -> 'a list -> 'a list * 'a list
BatList.Labels.partition_map : f:('a -> ('b, 'c) BatEither.t) -> 'a list -> 'b list * 'c list
BatList.Labels.remove_if : f:('a -> bool) -> 'a list -> 'a list
BatList.Labels.rev_map : f:('a -> 'b) -> 'a list -> 'b list
BatList.Labels.rev_map2 : f:('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
BatList.Labels.rfind : f:('a -> bool) -> 'a list -> 'a
BatList.Labels.stable_sort : ?cmp:('a -> 'a -> int) -> 'a list -> 'a list
BatList.Labels.subset : cmp:('a -> 'b -> int) -> 'a list -> 'b list -> bool
BatList.Labels.take_while : f:('a -> bool) -> 'a list -> 'a list
BatList.(@) : 'a list -> 'a list -> 'a list
BatList.Exceptionless.assoc : 'a -> ('a * 'b) list -> 'b option
BatList.Exceptionless.assoc_inv : 'b -> ('a * 'b) list -> 'a option
BatList.Exceptionless.assq : 'a -> ('a * 'b) list -> 'b option
BatList.Exceptionless.at : 'a list -> int -> [`Invalid_argument of string | `Ok of 'a]
BatList.Exceptionless.find : ('a -> bool) -> 'a list -> 'a option
BatList.Exceptionless.find_map : ('a -> 'b option) -> 'a list -> 'b option
BatList.Exceptionless.findi : (int -> 'a -> bool) -> 'a list -> (int * 'a) option
BatList.Exceptionless.hd : 'a list -> 'a option
BatList.Exceptionless.last : 'a list -> 'a option
BatList.Exceptionless.max : ?cmp:('a -> 'a -> int) -> 'a list -> 'a option
BatList.Exceptionless.min : ?cmp:('a -> 'a -> int) -> 'a list -> 'a option
BatList.Exceptionless.min_max : ?cmp:('a -> 'a -> int) -> 'a list -> ('a * 'a) option
BatList.Exceptionless.reduce : ('a -> 'a -> 'a) -> 'a list -> 'a option
BatList.Exceptionless.rfind : ('a -> bool) -> 'a list -> 'a option
BatList.Exceptionless.split_at : int -> 'a list -> [`Invalid_argument of string | `Ok of 'a list * 'a list]
BatList.Exceptionless.tl : 'a list -> 'a list option
BatList.Labels.LExceptionless.assoc : 'a -> ('a * 'b) list -> 'b option
BatList.Labels.LExceptionless.assoc_inv : 'b -> ('a * 'b) list -> 'a option
BatList.Labels.LExceptionless.assq : 'a -> ('a * 'b) list -> 'b option
BatList.Labels.LExceptionless.at : 'a list -> int -> [`Invalid_argument of string | `Ok of 'a]
BatList.Labels.LExceptionless.find : f:('a -> bool) -> 'a list -> 'a option

Batteries
BatList.Labels.LExceptionless.findi : f:(int -> 'a -> bool) -> 'a list -> (int * 'a) option
BatList.Labels.LExceptionless.rfind : f:('a -> bool) -> 'a list -> 'a option
BatList.Labels.LExceptionless.split_at : int -> 'a list -> [`Invalid_argument of string | `Ok of 'a list * 'a list]
BatList.Labels.concat_map : f:('a -> 'b list) -> 'a list -> 'b list
BatList.Labels.count_matching : f:('a -> bool) -> 'a list -> int
BatList.Labels.drop_while : f:('a -> bool) -> 'a list -> 'a list
BatList.Labels.exists : f:('a -> bool) -> 'a list -> bool
BatList.Labels.exists2 : f:('a -> 'b -> bool) -> 'a list -> 'b list -> bool
BatList.Labels.fast_sort : ?cmp:('a -> 'a -> int) -> 'a list -> 'a list
BatList.Labels.filter : f:('a -> bool) -> 'a list -> 'a list
BatList.Labels.filter_map : f:('a -> 'b option) -> 'a list -> 'b list
BatList.Labels.find : f:('a -> bool) -> 'a list -> 'a
BatList.Labels.find_all : f:('a -> bool) -> 'a list -> 'a list
BatList.Labels.find_exn : f:('a -> bool) -> exn -> 'a list -> 'a
BatList.Labels.find_map_opt : f:('a -> 'b option) -> 'a list -> 'b option
BatList.Labels.findi : f:(int -> 'a -> bool) -> 'a list -> int * 'a
BatList.Labels.fold : f:('a -> 'b -> 'a) -> init:'a -> 'b list -> 'a
BatList.Labels.fold_left : f:('a -> 'b -> 'a) -> init:'a -> 'b list -> 'a
BatList.Labels.fold_left2 : f:('a -> 'b -> 'c -> 'a) -> init:'a -> 'b list -> 'c list -> 'a
BatList.Labels.fold_right : f:('a -> 'b -> 'b) -> 'a list -> init:'b -> 'b
BatList.Labels.fold_right2 : f:('a -> 'b -> 'c -> 'c) -> 'a list -> 'b list -> init:'c -> 'c
BatList.Labels.for_all : f:('a -> bool) -> 'a list -> bool
BatList.Labels.for_all2 : f:('a -> 'b -> bool) -> 'a list -> 'b list -> bool
BatList.Labels.init : int -> f:(int -> 'a) -> 'a list
BatList.Labels.iter : f:('a -> unit) -> 'a list -> unit
BatList.Labels.iter2 : f:('a -> 'b -> unit) -> 'a list -> 'b list -> unit
BatList.Labels.iteri : f:(int -> 'a -> unit) -> 'a list -> unit
BatList.Labels.map : f:('a -> 'b) -> 'a list -> 'b list
BatList.Labels.map2 : f:('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
BatList.Labels.mapi : f:(int -> 'a -> 'b) -> 'a list -> 'b list
BatList.Labels.merge : ?cmp:('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
BatList.Labels.partition : f:('a -> bool) -> 'a list -> 'a list * 'a list
BatList.Labels.partition_map : f:('a -> ('b, 'c) BatEither.t) -> 'a list -> 'b list * 'c list
BatList.Labels.remove_if : f:('a -> bool) -> 'a list -> 'a list

Batteries
BatList.Labels.rev_map : f:('a -> 'b) -> 'a list -> 'b list
BatList.Labels.rev_map2 : f:('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
BatList.Labels.rfind : f:('a -> bool) -> 'a list -> 'a
BatList.Labels.stable_sort : ?cmp:('a -> 'a -> int) -> 'a list -> 'a list
BatList.Labels.subset : cmp:('a -> 'b -> int) -> 'a list -> 'b list -> bool
BatList.Labels.take_while : f:('a -> bool) -> 'a list -> 'a list
BatList.append : 'a list -> 'a list -> 'a list
BatList.assoc : 'a -> ('a * 'b) list -> 'b
BatList.assoc_inv : 'b -> ('a * 'b) list -> 'a
BatList.assoc_opt : 'a -> ('a * 'b) list -> 'b option
BatList.assq : 'a -> ('a * 'b) list -> 'b
BatList.assq_inv : 'b -> ('a * 'b) list -> 'a
BatList.assq_opt : 'a -> ('a * 'b) list -> 'b option
BatList.at : 'a list -> int -> 'a
BatList.at_opt : 'a list -> int -> 'a option
BatList.backwards : 'a list -> 'a BatEnum.t
BatList.cartesian_product : 'a list -> 'b list -> ('a * 'b) list
BatList.combine : 'a list -> 'b list -> ('a * 'b) list
BatList.compare : 'a BatOrd.comp -> 'a list BatOrd.comp
BatList.compare_length_with : 'a list -> int -> int
BatList.compare_lengths : 'a list -> 'b list -> int
BatList.concat : 'a list list -> 'a list
BatList.concat_map : ('a -> 'b list) -> 'a list -> 'b list
BatList.cons : 'a -> 'a list -> 'a list
BatList.count_matching : ('a -> bool) -> 'a list -> int
BatList.drop : int -> 'a list -> 'a list
BatList.drop_while : ('a -> bool) -> 'a list -> 'a list
BatList.dropwhile : ('a -> bool) -> 'a list -> 'a list
BatList.enum : 'a list -> 'a BatEnum.t
BatList.eq : 'a BatOrd.eq -> 'a list BatOrd.eq
BatList.equal : ('a -> 'a -> bool) -> 'a list -> 'a list -> bool
BatList.exists : ('a -> bool) -> 'a list -> bool
BatList.exists2 : ('a -> 'b -> bool) -> 'a list -> 'b list -> bool
BatList.fast_sort : ('a -> 'a -> int) -> 'a list -> 'a list

Batteries
BatList.favg : float list -> float
BatList.filter : ('a -> bool) -> 'a list -> 'a list
BatList.filter_map : ('a -> 'b option) -> 'a list -> 'b list
BatList.filteri : (int -> 'a -> bool) -> 'a list -> 'a list
BatList.filteri_map : (int -> 'a -> 'b option) -> 'a list -> 'b list
BatList.find : ('a -> bool) -> 'a list -> 'a
BatList.find_all : ('a -> bool) -> 'a list -> 'a list
BatList.find_exn : ('a -> bool) -> exn -> 'a list -> 'a
BatList.find_map : ('a -> 'b option) -> 'a list -> 'b
BatList.find_map_opt : ('a -> 'b option) -> 'a list -> 'b option
BatList.find_opt : ('a -> bool) -> 'a list -> 'a option
BatList.findi : (int -> 'a -> bool) -> 'a list -> int * 'a
BatList.first : 'a list -> 'a
BatList.flatten : 'a list list -> 'a list
BatList.fold : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a
BatList.fold_left : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a
BatList.fold_left2 : ('a -> 'b -> 'c -> 'a) -> 'a -> 'b list -> 'c list -> 'a
BatList.fold_left_map : ('a -> 'b -> 'a * 'c) -> 'a -> 'b list -> 'a * 'c list
BatList.fold_lefti : ('a -> int -> 'b -> 'a) -> 'a -> 'b list -> 'a
BatList.fold_right : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b
BatList.fold_right2 : ('a -> 'b -> 'c -> 'c) -> 'a list -> 'b list -> 'c -> 'c
BatList.fold_righti : (int -> 'b -> 'a -> 'a) -> 'b list -> 'a -> 'a
BatList.fold_while : ('acc -> 'a -> bool) -> ('acc -> 'a -> 'acc) -> 'acc -> 'a list -> 'acc * 'a list
BatList.for_all : ('a -> bool) -> 'a list -> bool
BatList.for_all2 : ('a -> 'b -> bool) -> 'a list -> 'b list -> bool
BatList.frange : float -> [< `Downto | `To] -> float -> int -> float list
BatList.fsum : float list -> float
BatList.group : ('a -> 'a -> int) -> 'a list -> 'a list list
BatList.group_consecutive : ('a -> 'a -> bool) -> 'a list -> 'a list list
BatList.hd : 'a list -> 'a
BatList.index_of : 'a -> 'a list -> int option
BatList.index_ofq : 'a -> 'a list -> int option
BatList.init : int -> (int -> 'a) -> 'a list
BatList.interleave : ?first:'a -> ?last:'a -> 'a -> 'a list -> 'a list

Batteries
BatList.is_empty : 'a list -> bool
BatList.iter : ('a -> unit) -> 'a list -> unit
BatList.iter2 : ('a -> 'b -> unit) -> 'a list -> 'b list -> unit
BatList.iter2i : (int -> 'a -> 'b -> unit) -> 'a list -> 'b list -> unit
BatList.iteri : (int -> 'a -> unit) -> 'a list -> unit
BatList.kahan_sum : float list -> float
BatList.last : 'a list -> 'a
BatList.length : 'a list -> int
BatList.make : int -> 'a -> 'a list
BatList.map : ('a -> 'b) -> 'a list -> 'b list
BatList.map2 : ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
BatList.map2i : (int -> 'a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
BatList.mapi : (int -> 'a -> 'b) -> 'a list -> 'b list
BatList.max : ?cmp:('a -> 'a -> int) -> 'a list -> 'a
BatList.mem : 'a -> 'a list -> bool
BatList.mem_assoc : 'a -> ('a * 'b) list -> bool
BatList.mem_assq : 'a -> ('a * 'b) list -> bool
BatList.mem_cmp : ('a -> 'a -> int) -> 'a -> 'a list -> bool
BatList.memq : 'a -> 'a list -> bool
BatList.merge : ('a -> 'a -> int) -> 'a list -> 'a list -> 'a list
BatList.min : ?cmp:('a -> 'a -> int) -> 'a list -> 'a
BatList.min_max : ?cmp:('a -> 'a -> int) -> 'a list -> 'a * 'a
BatList.modify : 'a -> ('b -> 'b) -> ('a * 'b) list -> ('a * 'b) list
BatList.modify_at : int -> ('a -> 'a) -> 'a list -> 'a list
BatList.modify_def : 'b -> 'a -> ('b -> 'b) -> ('a * 'b) list -> ('a * 'b) list
BatList.modify_opt : 'a -> ('b option -> 'b option) -> ('a * 'b) list -> ('a * 'b) list
BatList.modify_opt_at : int -> ('a -> 'a option) -> 'a list -> 'a list
BatList.n_cartesian_product : 'a list list -> 'a list list
BatList.nsplit : ('a -> bool) -> 'a list -> 'a list list
BatList.ntake : int -> 'a list -> 'a list list
BatList.nth : 'a list -> int -> 'a
BatList.nth_opt : 'a list -> int -> 'a option
BatList.of_backwards : 'a BatEnum.t -> 'a list
BatList.of_enum : 'a BatEnum.t -> 'a list

Batteries
BatList.of_seq : 'a Seq.t -> 'a list
BatList.ord : 'a BatOrd.ord -> 'a list BatOrd.ord
BatList.partition : ('a -> bool) -> 'a list -> 'a list * 'a list
BatList.partition_map : ('a -> ('b, 'c) BatEither.t) -> 'a list -> 'b list * 'c list
BatList.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatInnerIO.output -> 'b -> unit) -> 'a BatInnerIO.output -> 'b list -> unit
BatList.range : int -> [< `Downto | `To] -> int -> int list
BatList.reduce : ('a -> 'a -> 'a) -> 'a list -> 'a
BatList.remove : 'a list -> 'a -> 'a list
BatList.remove_all : 'a list -> 'a -> 'a list
BatList.remove_assoc : 'a -> ('a * 'b) list -> ('a * 'b) list
BatList.remove_assq : 'a -> ('a * 'b) list -> ('a * 'b) list
BatList.remove_at : int -> 'a list -> 'a list
BatList.remove_if : ('a -> bool) -> 'a list -> 'a list
BatList.rev : 'a list -> 'a list
BatList.rev_append : 'a list -> 'a list -> 'a list
BatList.rev_map : ('a -> 'b) -> 'a list -> 'b list
BatList.rev_map2 : ('a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
BatList.rfind : ('a -> bool) -> 'a list -> 'a
BatList.rindex_of : 'a -> 'a list -> int option
BatList.rindex_ofq : 'a -> 'a list -> int option
BatList.shuffle : ?state:Random.State.t -> 'a list -> 'a list
BatList.singleton : 'a -> 'a list
BatList.sort : ('a -> 'a -> int) -> 'a list -> 'a list
BatList.sort_uniq : ('a -> 'a -> int) -> 'a list -> 'a list
BatList.sort_unique : ('a -> 'a -> int) -> 'a list -> 'a list
BatList.span : ('a -> bool) -> 'a list -> 'a list * 'a list
BatList.split : ('a * 'b) list -> 'a list * 'b list
BatList.split_at : int -> 'a list -> 'a list * 'a list
BatList.split_nth : int -> 'a list -> 'a list * 'a list
BatList.stable_sort : ('a -> 'a -> int) -> 'a list -> 'a list
BatList.subset : ('a -> 'b -> int) -> 'a list -> 'b list -> bool
BatList.sum : int list -> int
BatList.take : int -> 'a list -> 'a list
BatList.take_while : ('a -> bool) -> 'a list -> 'a list

Batteries
BatList.takedrop : int -> 'a list -> 'a list * 'a list
BatList.takewhile : ('a -> bool) -> 'a list -> 'a list
BatList.tl : 'a list -> 'a list
BatList.to_seq : 'a list -> 'a Seq.t
BatList.transpose : 'a list list -> 'a list list
BatList.unfold : 'b -> ('b -> ('a * 'b) option) -> 'a list
BatList.unfold_exc : (unit -> 'a) -> 'a list * exn
BatList.unfold_exn : (unit -> 'a) -> 'a list * exn
BatList.unique : ?eq:('a -> 'a -> bool) -> 'a list -> 'a list
BatList.unique_cmp : ?cmp:('a -> 'a -> int) -> 'a list -> 'a list
BatList.unique_hash : ?hash:('a -> int) -> ?eq:('a -> 'a -> bool) -> 'a list -> 'a list
BatMap.(-->) : ('a, 'b) t -> 'a -> 'b
BatMap.(<--) : ('a, 'b) t -> 'a * 'b -> ('a, 'b) t
BatMap.Exceptionless.any : ('a, 'b) t -> ('a * 'b) option
BatMap.Exceptionless.choose : ('a, 'b) t -> ('a * 'b) option
BatMap.Exceptionless.find : 'a -> ('a, 'b) t -> 'b option
BatMap.PMap.(-->) : ('a, 'b) t -> 'a -> 'b
BatMap.PMap.(<--) : ('a, 'b) t -> 'a * 'b -> ('a, 'b) t
BatMap.PMap.Exceptionless.any : ('a, 'b) t -> ('a * 'b) option
BatMap.PMap.Exceptionless.choose : ('a, 'b) t -> ('a * 'b) option
BatMap.PMap.Exceptionless.find : 'a -> ('a, 'b) t -> 'b option
BatMap.PMap.add : 'a -> 'b -> ('a, 'b) t -> ('a, 'b) t
BatMap.PMap.add_carry : 'a -> 'b -> ('a, 'b) t -> ('a, 'b) t * 'b option
BatMap.PMap.add_seq : ('key * 'a) BatSeq.t -> ('key, 'a) t -> ('key, 'a) t
BatMap.PMap.any : ('key, 'a) t -> 'key * 'a
BatMap.PMap.at_rank_exn : int -> ('a, 'b) t -> 'a * 'b
BatMap.PMap.backwards : ('a, 'b) t -> ('a * 'b) BatEnum.t
BatMap.PMap.bindings : ('key, 'a) t -> ('key * 'a) list
BatMap.PMap.cardinal : ('a, 'b) t -> int
BatMap.PMap.choose : ('key, 'a) t -> 'key * 'a
BatMap.PMap.choose_opt : ('key, 'a) t -> ('key * 'a) option
BatMap.PMap.compare : ('b -> 'b -> int) -> ('a, 'b) t -> ('a, 'b) t -> int
BatMap.PMap.create : ('a -> 'a -> int) -> ('a, 'b) t
BatMap.PMap.diff : ('a, 'b) t -> ('a, 'b) t -> ('a, 'b) t

Batteries
BatMap.PMap.empty : ('a, 'b) t
BatMap.PMap.enum : ('a, 'b) t -> ('a * 'b) BatEnum.t
BatMap.PMap.equal : ('b -> 'b -> bool) -> ('a, 'b) t -> ('a, 'b) t -> bool
BatMap.PMap.exists : ('a -> 'b -> bool) -> ('a, 'b) t -> bool
BatMap.PMap.extract : 'a -> ('a, 'b) t -> 'b * ('a, 'b) t
BatMap.PMap.filter : ('key -> 'a -> bool) -> ('key, 'a) t -> ('key, 'a) t
BatMap.PMap.filter_map : ('key -> 'a -> 'b option) -> ('key, 'a) t -> ('key, 'b) t
BatMap.PMap.filterv : ('a -> bool) -> ('key, 'a) t -> ('key, 'a) t
BatMap.PMap.find : 'a -> ('a, 'b) t -> 'b
BatMap.PMap.find_default : 'b -> 'a -> ('a, 'b) t -> 'b
BatMap.PMap.find_first : ('a -> bool) -> ('a, 'b) t -> 'a * 'b
BatMap.PMap.find_first_opt : ('a -> bool) -> ('a, 'b) t -> ('a * 'b) option
BatMap.PMap.find_last : ('a -> bool) -> ('a, 'b) t -> 'a * 'b
BatMap.PMap.find_last_opt : ('a -> bool) -> ('a, 'b) t -> ('a * 'b) option
BatMap.PMap.fold : ('b -> 'c -> 'c) -> ('a, 'b) t -> 'c -> 'c
BatMap.PMap.foldi : ('a -> 'b -> 'c -> 'c) -> ('a, 'b) t -> 'c -> 'c
BatMap.PMap.for_all : ('a -> 'b -> bool) -> ('a, 'b) t -> bool
BatMap.PMap.get_cmp : ('a, 'b) t -> 'a -> 'a -> int
BatMap.PMap.intersect : ('b -> 'c -> 'd) -> ('a, 'b) t -> ('a, 'c) t -> ('a, 'd) t
BatMap.PMap.is_empty : ('a, 'b) t -> bool
BatMap.PMap.iter : ('a -> 'b -> unit) -> ('a, 'b) t -> unit
BatMap.PMap.keys : ('a, 'b) t -> 'a BatEnum.t
BatMap.PMap.map : ('b -> 'c) -> ('a, 'b) t -> ('a, 'c) t
BatMap.PMap.mapi : ('a -> 'b -> 'c) -> ('a, 'b) t -> ('a, 'c) t
BatMap.PMap.max_binding : ('key, 'a) t -> 'key * 'a
BatMap.PMap.max_binding_opt : ('key, 'a) t -> ('key * 'a) option
BatMap.PMap.mem : 'a -> ('a, 'b) t -> bool
BatMap.PMap.merge : ('key -> 'a option -> 'b option -> 'c option) -> ('key, 'a) t -> ('key, 'b) t -> ('key, 'c) t
BatMap.PMap.merge_unsafe : ('key -> 'a option -> 'b option -> 'c option) -> ('key, 'a) t -> ('key, 'b) t -> ('key, 'c) t
BatMap.PMap.min_binding : ('key, 'a) t -> 'key * 'a
BatMap.PMap.min_binding_opt : ('key, 'a) t -> ('key * 'a) option
BatMap.PMap.modify : 'a -> ('b -> 'b) -> ('a, 'b) t -> ('a, 'b) t
BatMap.PMap.modify_def : 'b -> 'a -> ('b -> 'b) -> ('a, 'b) t -> ('a, 'b) t
BatMap.PMap.modify_opt : 'a -> ('b option -> 'b option) -> ('a, 'b) t -> ('a, 'b) t

Batteries
BatMap.PMap.of_enum : ?cmp:('a -> 'a -> int) -> ('a * 'b) BatEnum.t -> ('a, 'b) t
BatMap.PMap.of_seq : ?cmp:('key -> 'key -> int) -> ('key * 'a) BatSeq.t -> ('key, 'a) t
BatMap.PMap.partition : ('a -> 'b -> bool) -> ('a, 'b) t -> ('a, 'b) t * ('a, 'b) t
BatMap.PMap.pop : ('a, 'b) t -> ('a * 'b) * ('a, 'b) t
BatMap.PMap.pop_max_binding : ('key, 'a) t -> ('key * 'a) * ('key, 'a) t
BatMap.PMap.pop_min_binding : ('key, 'a) t -> ('key * 'a) * ('key, 'a) t
BatMap.PMap.print : ?first:string -> ?last:string -> ?sep:string -> ?kvsep:string -> ('a BatInnerIO.output -> 'b -> unit) -> ('a BatInnerIO.output -> 'c -> unit) -> 'a BatInnerIO.output -> ('b, 'c) t -> unit
BatMap.PMap.remove : 'a -> ('a, 'b) t -> ('a, 'b) t
BatMap.PMap.remove_exn : 'a -> ('a, 'b) t -> ('a, 'b) t
BatMap.PMap.singleton : ?cmp:('a -> 'a -> int) -> 'a -> 'b -> ('a, 'b) t
BatMap.PMap.split : 'key -> ('key, 'a) t -> ('key, 'a) t * 'a option * ('key, 'a) t
BatMap.PMap.to_rev_seq : ('key, 'a) t -> ('key * 'a) BatSeq.t
BatMap.PMap.to_seq : ('key, 'a) t -> ('key * 'a) BatSeq.t
BatMap.PMap.to_seq_from : 'key -> ('key, 'a) t -> ('key * 'a) BatSeq.t
BatMap.PMap.union : ('a, 'b) t -> ('a, 'b) t -> ('a, 'b) t
BatMap.PMap.union_stdlib : ('key -> 'a -> 'a -> 'a option) -> ('key, 'a) t -> ('key, 'a) t -> ('key, 'a) t
BatMap.PMap.update : 'a -> 'a -> 'b -> ('a, 'b) t -> ('a, 'b) t
BatMap.PMap.update_stdlib : 'a -> ('b option -> 'b option) -> ('a, 'b) t -> ('a, 'b) t
BatMap.PMap.values : ('a, 'b) t -> 'b BatEnum.t
BatMap.S.Exceptionless.any : 'a t -> (key * 'a) option
BatMap.S.Exceptionless.choose : 'a t -> (key * 'a) option
BatMap.S.Exceptionless.find : key -> 'a t -> 'a option
BatMap.S.Infix.(-->) : 'a t -> key -> 'a
BatMap.S.Infix.(<--) : 'a t -> key * 'a -> 'a t
BatMap.S.Labels.add : key:key -> data:'a -> 'a t -> 'a t
BatMap.S.Labels.compare : cmp:('a -> 'a -> int) -> 'a t -> 'a t -> int
BatMap.S.Labels.equal : cmp:('a -> 'a -> bool) -> 'a t -> 'a t -> bool
BatMap.S.Labels.filter : f:(key -> 'a -> bool) -> 'a t -> 'a t
BatMap.S.Labels.filterv : f:('a -> bool) -> 'a t -> 'a t
BatMap.S.Labels.fold : f:(key:key -> data:'a -> 'b -> 'b) -> 'a t -> init:'b -> 'b
BatMap.S.Labels.iter : f:(key:key -> data:'a -> unit) -> 'a t -> unit
BatMap.S.Labels.map : f:('a -> 'b) -> 'a t -> 'b t
BatMap.S.Labels.mapi : f:(key:key -> data:'a -> 'b) -> 'a t -> 'b t
BatMap.S.add : key -> 'a -> 'a t -> 'a t

Batteries
BatMap.S.add_seq : (key * 'a) BatSeq.t -> 'a t -> 'a t
BatMap.S.any : 'a t -> key * 'a
BatMap.S.backwards : 'a t -> (key * 'a) BatEnum.t
BatMap.S.bindings : 'a t -> (key * 'a) list
BatMap.S.cardinal : 'a t -> int
BatMap.S.choose : 'a t -> key * 'a
BatMap.S.choose_opt : 'a t -> (key * 'a) option
BatMap.S.compare : ('a -> 'a -> int) -> 'a t -> 'a t -> int
BatMap.S.empty : 'a t
BatMap.S.enum : 'a t -> (key * 'a) BatEnum.t
BatMap.S.equal : ('a -> 'a -> bool) -> 'a t -> 'a t -> bool
BatMap.S.exists : (key -> 'a -> bool) -> 'a t -> bool
BatMap.S.extract : key -> 'a t -> 'a * 'a t
BatMap.S.filter : (key -> 'a -> bool) -> 'a t -> 'a t
BatMap.S.filter_map : (key -> 'a -> 'b option) -> 'a t -> 'b t
BatMap.S.filterv : ('a -> bool) -> 'a t -> 'a t
BatMap.S.find : key -> 'a t -> 'a
BatMap.S.find_default : 'a -> key -> 'a t -> 'a
BatMap.S.find_first : (key -> bool) -> 'a t -> key * 'a
BatMap.S.find_first_opt : (key -> bool) -> 'a t -> (key * 'a) option
BatMap.S.find_last : (key -> bool) -> 'a t -> key * 'a
BatMap.S.find_last_opt : (key -> bool) -> 'a t -> (key * 'a) option
BatMap.S.find_opt : key -> 'a t -> 'a option
BatMap.S.fold : (key -> 'a -> 'b -> 'b) -> 'a t -> 'b -> 'b
BatMap.S.for_all : (key -> 'a -> bool) -> 'a t -> bool
BatMap.S.is_empty : 'a t -> bool
BatMap.S.iter : (key -> 'a -> unit) -> 'a t -> unit
BatMap.S.keys : 'a t -> key BatEnum.t
BatMap.S.map : ('a -> 'b) -> 'a t -> 'b t
BatMap.S.mapi : (key -> 'a -> 'b) -> 'a t -> 'b t
BatMap.S.max_binding : 'a t -> key * 'a
BatMap.S.max_binding_opt : 'a t -> (key * 'a) option
BatMap.S.mem : key -> 'a t -> bool
BatMap.S.merge : (key -> 'a option -> 'b option -> 'c option) -> 'a t -> 'b t -> 'c t

Batteries
BatMap.S.min_binding : 'a t -> key * 'a
BatMap.S.min_binding_opt : 'a t -> (key * 'a) option
BatMap.S.modify : key -> ('a -> 'a) -> 'a t -> 'a t
BatMap.S.modify_def : 'a -> key -> ('a -> 'a) -> 'a t -> 'a t
BatMap.S.modify_opt : key -> ('a option -> 'a option) -> 'a t -> 'a t
BatMap.S.of_enum : (key * 'a) BatEnum.t -> 'a t
BatMap.S.of_seq : (key * 'a) BatSeq.t -> 'a t
BatMap.S.partition : (key -> 'a -> bool) -> 'a t -> 'a t * 'a t
BatMap.S.pop : 'a t -> (key * 'a) * 'a t
BatMap.S.pop_max_binding : 'a t -> (key * 'a) * 'a t
BatMap.S.pop_min_binding : 'a t -> (key * 'a) * 'a t
BatMap.S.print : ?first:string -> ?last:string -> ?sep:string -> ?kvsep:string -> ('a BatInnerIO.output -> key -> unit) -> ('a BatInnerIO.output -> 'c -> unit) -> 'a BatInnerIO.output -> 'c t -> unit
BatMap.S.remove : key -> 'a t -> 'a t
BatMap.S.remove_exn : key -> 'a t -> 'a t
BatMap.S.singleton : key -> 'a -> 'a t
BatMap.S.split : key -> 'a t -> 'a t * 'a option * 'a t
BatMap.S.to_rev_seq : 'a t -> (key * 'a) BatSeq.t
BatMap.S.to_seq : 'a t -> (key * 'a) BatSeq.t
BatMap.S.to_seq_from : key -> 'a t -> (key * 'a) BatSeq.t
BatMap.S.union : (key -> 'a -> 'a -> 'a option) -> 'a t -> 'a t -> 'a t
BatMap.S.update : key -> key -> 'a -> 'a t -> 'a t
BatMap.S.update_stdlib : key -> ('a option -> 'a option) -> 'a t -> 'a t
BatMap.S.values : 'a t -> 'a BatEnum.t
BatMap.add : 'a -> 'b -> ('a, 'b) t -> ('a, 'b) t
BatMap.add_carry : 'a -> 'b -> ('a, 'b) t -> ('a, 'b) t * 'b option
BatMap.add_seq : ('key * 'a) BatSeq.t -> ('key, 'a) t -> ('key, 'a) t
BatMap.any : ('key, 'a) t -> 'key * 'a
BatMap.at_rank_exn : int -> ('key, 'a) t -> 'key * 'a
BatMap.backwards : ('a, 'b) t -> ('a * 'b) BatEnum.t
BatMap.bindings : ('key, 'a) t -> ('key * 'a) list
BatMap.cardinal : ('a, 'b) t -> int
BatMap.choose : ('key, 'a) t -> 'key * 'a
BatMap.choose_opt : ('key, 'a) t -> ('key * 'a) option
BatMap.compare : ('b -> 'b -> int) -> ('a, 'b) t -> ('a, 'b) t -> int

Batteries
BatMap.diff : ('a, 'b) t -> ('a, 'b) t -> ('a, 'b) t
BatMap.empty : ('a, 'b) t
BatMap.enum : ('a, 'b) t -> ('a * 'b) BatEnum.t
BatMap.equal : ('b -> 'b -> bool) -> ('a, 'b) t -> ('a, 'b) t -> bool
BatMap.exists : ('a -> 'b -> bool) -> ('a, 'b) t -> bool
BatMap.extract : 'a -> ('a, 'b) t -> 'b * ('a, 'b) t
BatMap.filter : ('key -> 'a -> bool) -> ('key, 'a) t -> ('key, 'a) t
BatMap.filter_map : ('key -> 'a -> 'b option) -> ('key, 'a) t -> ('key, 'b) t
BatMap.filterv : ('a -> bool) -> ('key, 'a) t -> ('key, 'a) t
BatMap.find : 'a -> ('a, 'b) t -> 'b
BatMap.find_default : 'b -> 'a -> ('a, 'b) t -> 'b
BatMap.find_first : ('a -> bool) -> ('a, 'b) t -> 'a * 'b
BatMap.find_first_opt : ('a -> bool) -> ('a, 'b) t -> ('a * 'b) option
BatMap.find_last : ('a -> bool) -> ('a, 'b) t -> 'a * 'b
BatMap.find_last_opt : ('a -> bool) -> ('a, 'b) t -> ('a * 'b) option
BatMap.find_opt : 'a -> ('a, 'b) t -> 'b option
BatMap.fold : ('b -> 'c -> 'c) -> ('a, 'b) t -> 'c -> 'c
BatMap.foldi : ('a -> 'b -> 'c -> 'c) -> ('a, 'b) t -> 'c -> 'c
BatMap.for_all : ('a -> 'b -> bool) -> ('a, 'b) t -> bool
BatMap.intersect : ('b -> 'c -> 'd) -> ('a, 'b) t -> ('a, 'c) t -> ('a, 'd) t
BatMap.is_empty : ('a, 'b) t -> bool
BatMap.iter : ('a -> 'b -> unit) -> ('a, 'b) t -> unit
BatMap.keys : ('a, 'b) t -> 'a BatEnum.t
BatMap.map : ('b -> 'c) -> ('a, 'b) t -> ('a, 'c) t
BatMap.mapi : ('a -> 'b -> 'c) -> ('a, 'b) t -> ('a, 'c) t
BatMap.max_binding : ('key, 'a) t -> 'key * 'a
BatMap.max_binding_opt : ('key, 'a) t -> ('key * 'a) option
BatMap.mem : 'a -> ('a, 'b) t -> bool
BatMap.merge : ('key -> 'a option -> 'b option -> 'c option) -> ('key, 'a) t -> ('key, 'b) t -> ('key, 'c) t
BatMap.min_binding : ('key, 'a) t -> 'key * 'a
BatMap.min_binding_opt : ('key, 'a) t -> ('key * 'a) option
BatMap.modify : 'a -> ('b -> 'b) -> ('a, 'b) t -> ('a, 'b) t
BatMap.modify_def : 'b -> 'a -> ('b -> 'b) -> ('a, 'b) t -> ('a, 'b) t
BatMap.modify_opt : 'a -> ('b option -> 'b option) -> ('a, 'b) t -> ('a, 'b) t

Batteries
BatMap.of_enum : ('a * 'b) BatEnum.t -> ('a, 'b) t
BatMap.of_seq : ('key * 'a) BatSeq.t -> ('key, 'a) t
BatMap.partition : ('a -> 'b -> bool) -> ('a, 'b) t -> ('a, 'b) t * ('a, 'b) t
BatMap.pop : ('a, 'b) t -> ('a * 'b) * ('a, 'b) t
BatMap.pop_max_binding : ('key, 'a) t -> ('key * 'a) * ('key, 'a) t
BatMap.pop_min_binding : ('key, 'a) t -> ('key * 'a) * ('key, 'a) t
BatMap.print : ?first:string -> ?last:string -> ?sep:string -> ?kvsep:string -> ('a BatInnerIO.output -> 'b -> unit) -> ('a BatInnerIO.output -> 'c -> unit) -> 'a BatInnerIO.output -> ('b, 'c) t -> unit
BatMap.remove : 'a -> ('a, 'b) t -> ('a, 'b) t
BatMap.remove_exn : 'a -> ('a, 'b) t -> ('a, 'b) t
BatMap.singleton : 'a -> 'b -> ('a, 'b) t
BatMap.split : 'key -> ('key, 'a) t -> ('key, 'a) t * 'a option * ('key, 'a) t
BatMap.to_rev_seq : ('key, 'a) t -> ('key * 'a) BatSeq.t
BatMap.to_seq : ('key, 'a) t -> ('key * 'a) BatSeq.t
BatMap.to_seq_from : 'key -> ('key, 'a) t -> ('key * 'a) BatSeq.t
BatMap.union : ('a, 'b) t -> ('a, 'b) t -> ('a, 'b) t
BatMap.union_stdlib : ('key -> 'a -> 'a -> 'a option) -> ('key, 'a) t -> ('key, 'a) t -> ('key, 'a) t
BatMap.update : 'a -> 'a -> 'b -> ('a, 'b) t -> ('a, 'b) t
BatMap.update_stdlib : 'a -> ('b option -> 'b option) -> ('a, 'b) t -> ('a, 'b) t
BatMap.values : ('a, 'b) t -> 'b BatEnum.t
BatOption.(|?) : 'a option -> 'a -> 'a
BatOption.Infix.(>>=) : 'a option -> ('a -> 'b option) -> 'b option
BatOption.Infix.(|?) : 'a option -> 'a -> 'a
BatOption.Labels.map : f:('a -> 'b) -> 'a option -> 'b option
BatOption.Labels.map_default : f:('a -> 'b) -> 'b -> 'a option -> 'b
BatOption.Labels.may : f:('a -> unit) -> 'a option -> unit
BatOption.Monad.bind : 'a m -> ('a -> 'b m) -> 'b m
BatOption.Monad.return : 'a -> 'a m
BatOption.apply : ('a -> 'a) option -> 'a -> 'a
BatOption.bind : 'a option -> ('a -> 'b option) -> 'b option
BatOption.compare : ?cmp:('a -> 'a -> int) -> 'a option -> 'a option -> int
BatOption.default : 'a -> 'a option -> 'a
BatOption.default_delayed : (unit -> 'a) -> 'a option -> 'a
BatOption.enum : 'a option -> 'a BatEnum.t
BatOption.eq : ?eq:('a -> 'a -> bool) -> 'a option -> 'a option -> bool

Batteries
BatOption.filter : ('a -> bool) -> 'a option -> 'a option
BatOption.get : 'a option -> 'a
BatOption.get_exn : 'a option -> exn -> 'a
BatOption.is_none : 'a option -> bool
BatOption.is_some : 'a option -> bool
BatOption.map : ('a -> 'b) -> 'a option -> 'b option
BatOption.map_default : ('a -> 'b) -> 'b -> 'a option -> 'b
BatOption.map_default_delayed : ('a -> 'b) -> (unit -> 'b) -> 'a option -> 'b
BatOption.may : ('a -> unit) -> 'a option -> unit
BatOption.of_enum : 'a BatEnum.t -> 'a option
BatOption.ord : 'a BatOrd.ord -> 'a option BatOrd.ord
BatOption.print : ('a BatInnerIO.output -> 'b -> unit) -> 'a BatInnerIO.output -> 'b t -> unit
BatOption.some : 'a -> 'a option
BatPrintf.bprintf : Buffer.t -> ('a, Buffer.t, unit) t -> 'a
BatPrintf.bprintf2 : Buffer.t -> ('b, 'a BatInnerIO.output, unit) t -> 'b
BatPrintf.eprintf : ('b, 'a BatInnerIO.output, unit) t -> 'b
BatPrintf.fprintf : 'a BatInnerIO.output -> ('b, 'a BatInnerIO.output, unit) t -> 'b
BatPrintf.ifprintf : 'c -> ('b, 'a BatInnerIO.output, unit) t -> 'b
BatPrintf.kbprintf : (Buffer.t -> 'a) -> Buffer.t -> ('b, Buffer.t, unit, 'a) format4 -> 'b
BatPrintf.kbprintf2 : (Buffer.t -> 'b) -> Buffer.t -> ('c, 'a BatInnerIO.output, unit, 'b) format4 -> 'c
BatPrintf.kfprintf : ('a BatInnerIO.output -> 'b) -> 'a BatInnerIO.output -> ('c, 'a BatInnerIO.output, unit, 'b) format4 -> 'c
BatPrintf.kprintf : (string -> 'a) -> ('b, unit, string, 'a) format4 -> 'b
BatPrintf.ksprintf : (string -> 'a) -> ('b, unit, string, 'a) format4 -> 'b
BatPrintf.ksprintf2 : (string -> 'b) -> ('c, 'a BatInnerIO.output, unit, 'b) format4 -> 'c
BatPrintf.printf : ('b, 'a BatInnerIO.output, unit) t -> 'b
BatPrintf.sprintf : ('a, unit, string) t -> 'a
BatPrintf.sprintf2 : ('a, 'b BatInnerIO.output, unit, string) format4 -> 'a
BatResult.Infix.(>>=) : ('a, 'e) t -> ('a -> ('c, 'e) t) -> ('c, 'e) t
BatResult.Monad.(>>=) : ('a, 'e) t -> ('a -> ('c, 'e) t) -> ('c, 'e) t
BatResult.Monad.bind : ('a, 'e) t -> ('a -> ('c, 'e) t) -> ('c, 'e) t
BatResult.Monad.return : 'a -> ('a, 'b) t
BatResult.bind : ('a, 'e) t -> ('a -> ('b, 'e) t) -> ('b, 'e) t
BatResult.catch : ('a -> 'e) -> 'a -> ('e, exn) t
BatResult.catch2 : ('a -> 'b -> 'c) -> 'a -> 'b -> ('c, exn) t

Batteries
BatResult.catch3 : ('a -> 'b -> 'c -> 'd) -> 'a -> 'b -> 'c -> ('d, exn) t
BatResult.compare : ok:('a -> 'a -> int) -> error:('e -> 'e -> int) -> ('a, 'e) t -> ('a, 'e) t -> int
BatResult.default : 'a -> ('a, 'b) t -> 'a
BatResult.equal : ok:('a -> 'a -> bool) -> error:('e -> 'e -> bool) -> ('a, 'e) t -> ('a, 'e) t -> bool
BatResult.error : 'e -> ('a, 'e) t
BatResult.fold : ok:('a -> 'c) -> error:('e -> 'c) -> ('a, 'e) t -> 'c
BatResult.get : ('a, exn) t -> 'a
BatResult.get_error : ('a, 'e) t -> 'e
BatResult.get_ok : ('a, 'e) t -> 'a
BatResult.is_bad : ('a, 'e) t -> bool
BatResult.is_error : ('a, 'e) t -> bool
BatResult.is_exn : exn -> ('a, exn) t -> bool
BatResult.is_ok : ('a, 'e) t -> bool
BatResult.iter : ('a -> unit) -> ('a, 'e) t -> unit
BatResult.iter_error : ('e -> unit) -> ('a, 'e) t -> unit
BatResult.join : (('a, 'e) t, 'e) t -> ('a, 'e) t
BatResult.map : ('a -> 'b) -> ('a, 'e) t -> ('b, 'e) t
BatResult.map_both : ('a1 -> 'a2) -> ('b1 -> 'b2) -> ('a1, 'b1) t -> ('a2, 'b2) t
BatResult.map_default : 'b -> ('a -> 'b) -> ('a, 'c) t -> 'b
BatResult.map_error : ('e -> 'f) -> ('a, 'e) t -> ('a, 'f) t
BatResult.of_option : 'a option -> ('a, unit) t
BatResult.ok : 'a -> ('a, 'b) t
BatResult.print : ('b BatInnerIO.output -> 'a -> unit) -> 'b BatInnerIO.output -> ('a, exn) t -> unit
BatResult.to_list : ('a, 'e) t -> 'a list
BatResult.to_option : ('a, 'b) t -> 'a option
BatResult.to_seq : ('a, 'e) t -> 'a BatSeq.t
BatResult.value : ('a, 'e) t -> default:'a -> 'a
BatSeq.(--) : int -> int -> int t
BatSeq.(---) : int -> int -> int t
BatSeq.(--.) : float * float -> float -> float t
BatSeq.(--^) : int -> int -> int t
BatSeq.(--~) : char -> char -> char t
BatSeq.(//) : 'a t -> ('a -> bool) -> 'a t
BatSeq.(//@) : 'a t -> ('a -> 'b option) -> 'b t

Batteries
BatSeq.(/@) : 'a t -> ('a -> 'b) -> 'b t
BatSeq.(@/) : ('a -> 'b) -> 'a t -> 'b t
BatSeq.(@//) : ('a -> 'b option) -> 'a t -> 'b t
BatSeq.Exceptionless.at : 'a t -> int -> 'a option
BatSeq.Exceptionless.combine : 'a t -> 'b t -> ('a * 'b) t
BatSeq.Exceptionless.first : 'a t -> 'a option
BatSeq.Exceptionless.hd : 'a t -> 'a option
BatSeq.Exceptionless.last : 'a t -> 'a option
BatSeq.Exceptionless.max : 'a t -> 'a option
BatSeq.Exceptionless.min : 'a t -> 'a option
BatSeq.Exceptionless.reduce : ('a -> 'a -> 'a) -> 'a t -> 'a option
BatSeq.Exceptionless.tl : 'a t -> 'a t option
BatSeq.append : 'a t -> 'a t -> 'a t
BatSeq.assoc : 'a -> ('a * 'b) t -> 'b option
BatSeq.at : 'a t -> int -> 'a
BatSeq.combine : 'a t -> 'b t -> ('a * 'b) t
BatSeq.compare : ('a -> 'b -> int) -> 'a t -> 'b t -> int
BatSeq.concat : 'a t t -> 'a t
BatSeq.concat_map : ('a -> 'b t) -> 'a t -> 'b t
BatSeq.cons : 'a -> 'a t -> 'a t
BatSeq.cycle : 'a t -> 'a t
BatSeq.drop : int -> 'a t -> 'a t
BatSeq.drop_while : ('a -> bool) -> 'a t -> 'a t
BatSeq.empty : 'a t
BatSeq.enum : 'a t -> 'a BatEnum.t
BatSeq.equal : ?eq:('a -> 'a -> bool) -> 'a t -> 'a t -> bool
BatSeq.equal_stdlib : ('a -> 'b -> bool) -> 'a t -> 'b t -> bool
BatSeq.exists : ('a -> bool) -> 'a t -> bool
BatSeq.exists2 : ('a -> 'b -> bool) -> 'a t -> 'b t -> bool
BatSeq.filter : ('a -> bool) -> 'a t -> 'a t
BatSeq.filter_map : ('a -> 'b option) -> 'a t -> 'b t
BatSeq.find : ('a -> bool) -> 'a t -> 'a option
BatSeq.find_map : ('a -> 'b option) -> 'a t -> 'b option
BatSeq.first : 'a t -> 'a

Batteries
BatSeq.flat_map : ('a -> 'b t) -> 'a t -> 'b t
BatSeq.flatten : 'a t t -> 'a t
BatSeq.fold_left : ('a -> 'b -> 'a) -> 'a -> 'b t -> 'a
BatSeq.fold_left2 : ('a -> 'b -> 'c -> 'a) -> 'a -> 'b t -> 'c t -> 'a
BatSeq.fold_lefti : ('b -> int -> 'a -> 'b) -> 'b -> 'a t -> 'b
BatSeq.fold_right : ('a -> 'b -> 'b) -> 'a t -> 'b -> 'b
BatSeq.for_all : ('a -> bool) -> 'a t -> bool
BatSeq.for_all2 : ('a -> 'b -> bool) -> 'a t -> 'b t -> bool
BatSeq.forever : (unit -> 'a) -> 'a t
BatSeq.group : ('a -> 'a -> bool) -> 'a t -> 'a t t
BatSeq.hd : 'a t -> 'a
BatSeq.init : int -> (int -> 'a) -> 'a t
BatSeq.interleave : 'a t -> 'a t -> 'a t
BatSeq.ints : int -> int t
BatSeq.is_empty : 'a t -> bool
BatSeq.iter : ('a -> unit) -> 'a t -> unit
BatSeq.iter2 : ('a -> 'b -> unit) -> 'a t -> 'b t -> unit
BatSeq.iterate : ('a -> 'a) -> 'a -> 'a t
BatSeq.iteri : (int -> 'a -> unit) -> 'a t -> unit
BatSeq.last : 'a t -> 'a
BatSeq.length : 'a t -> int
BatSeq.make : int -> 'a -> 'a t
BatSeq.map : ('a -> 'b) -> 'a t -> 'b t
BatSeq.map2 : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
BatSeq.map_product : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
BatSeq.mapi : (int -> 'a -> 'b) -> 'a t -> 'b t
BatSeq.max : 'a t -> 'a
BatSeq.mem : 'a -> 'a t -> bool
BatSeq.memoize : 'a t -> 'a t
BatSeq.min : 'a t -> 'a
BatSeq.nil : 'a t
BatSeq.of_dispenser : (unit -> 'a option) -> 'a t
BatSeq.of_list : 'a list -> 'a t
BatSeq.of_string : ?first:string -> ?last:string -> ?sep:string -> (string -> 'a) -> string -> 'a t

Batteries
BatSeq.once : 'a t -> 'a t
BatSeq.partition : ('a -> bool) -> 'a t -> 'a t * 'a t
BatSeq.partition_map : ('a -> ('b, 'c) Either.t) -> 'a t -> 'b t * 'c t
BatSeq.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatInnerIO.output -> 'b -> unit) -> 'a BatInnerIO.output -> 'b t -> unit
BatSeq.product : 'a t -> 'b t -> ('a * 'b) t
BatSeq.reduce : ('a -> 'a -> 'a) -> 'a t -> 'a
BatSeq.repeat : 'a -> 'a t
BatSeq.return : 'a -> 'a t
BatSeq.scan : ('b -> 'a -> 'b) -> 'b -> 'a t -> 'b t
BatSeq.sorted_merge : ('a -> 'a -> int) -> 'a t -> 'a t -> 'a t
BatSeq.split : ('a * 'b) t -> 'a t * 'b t
BatSeq.take : int -> 'a t -> 'a t
BatSeq.take_while : ('a -> bool) -> 'a t -> 'a t
BatSeq.tl : 'a t -> 'a t
BatSeq.to_buffer : ?first:string -> ?last:string -> ?sep:string -> ('a -> string) -> Buffer.t -> (unit -> 'a node) -> unit
BatSeq.to_dispenser : 'a t -> unit -> 'a option
BatSeq.to_string : ?first:string -> ?last:string -> ?sep:string -> ('a -> string) -> 'a t -> string
BatSeq.transpose : 'a t t -> 'a t t
BatSeq.uncons : 'a t -> ('a * 'a t) option
BatSeq.unfold : ('b -> ('a * 'b) option) -> 'b -> 'a t
BatSeq.unzip : ('a * 'b) t -> 'a t * 'b t
BatSeq.zip : 'a t -> 'b t -> ('a * 'b) t
BatSet.Make.Exceptionless.any : t -> elt option
BatSet.Make.Exceptionless.choose : t -> elt option
BatSet.Make.Exceptionless.find : elt -> t -> elt option
BatSet.Make.Exceptionless.max_elt : t -> elt option
BatSet.Make.Exceptionless.min_elt : t -> elt option
BatSet.Make.Labels.exists : f:(elt -> bool) -> t -> bool
BatSet.Make.Labels.filter : f:(elt -> bool) -> t -> t
BatSet.Make.Labels.filter_map : f:(elt -> elt option) -> t -> t
BatSet.Make.Labels.fold : f:(elt -> 'a -> 'a) -> t -> init:'a -> 'a
BatSet.Make.Labels.for_all : f:(elt -> bool) -> t -> bool
BatSet.Make.Labels.iter : f:(elt -> unit) -> t -> unit
BatSet.Make.Labels.map : f:(elt -> elt) -> t -> t

Batteries
BatSet.Make.Labels.partition : f:(elt -> bool) -> t -> t * t
BatSet.Make.add : elt -> t -> t
BatSet.Make.add_seq : elt BatSeq.t -> t -> t
BatSet.Make.any : t -> elt
BatSet.Make.at_rank_exn : int -> t -> elt
BatSet.Make.backwards : t -> elt BatEnum.t
BatSet.Make.cardinal : t -> int
BatSet.Make.choose : t -> elt
BatSet.Make.choose_opt : t -> elt option
BatSet.Make.compare : t -> t -> int
BatSet.Make.compare_subset : t -> t -> int
BatSet.Make.diff : t -> t -> t
BatSet.Make.disjoint : t -> t -> bool
BatSet.Make.elements : t -> elt list
BatSet.Make.empty : t
BatSet.Make.enum : t -> elt BatEnum.t
BatSet.Make.equal : t -> t -> bool
BatSet.Make.exists : (elt -> bool) -> t -> bool
BatSet.Make.filter : (elt -> bool) -> t -> t
BatSet.Make.filter_map : (elt -> elt option) -> t -> t
BatSet.Make.find : elt -> t -> elt
BatSet.Make.find_first : (elt -> bool) -> t -> elt
BatSet.Make.find_first_opt : (elt -> bool) -> t -> elt option
BatSet.Make.find_last : (elt -> bool) -> t -> elt
BatSet.Make.find_last_opt : (elt -> bool) -> t -> elt option
BatSet.Make.find_opt : elt -> t -> elt option
BatSet.Make.fold : (elt -> 'a -> 'a) -> t -> 'a -> 'a
BatSet.Make.for_all : (elt -> bool) -> t -> bool
BatSet.Make.inter : t -> t -> t
BatSet.Make.is_empty : t -> bool
BatSet.Make.is_singleton : t -> bool
BatSet.Make.iter : (elt -> unit) -> t -> unit
BatSet.Make.map : (elt -> elt) -> t -> t
BatSet.Make.max_elt : t -> elt

Batteries
BatSet.Make.max_elt_opt : t -> elt option
BatSet.Make.mem : elt -> t -> bool
BatSet.Make.min_elt : t -> elt
BatSet.Make.min_elt_opt : t -> elt option
BatSet.Make.of_array : elt array -> t
BatSet.Make.of_enum : elt BatEnum.t -> t
BatSet.Make.of_list : elt list -> t
BatSet.Make.of_seq : elt BatSeq.t -> t
BatSet.Make.partition : (elt -> bool) -> t -> t * t
BatSet.Make.pop : t -> elt * t
BatSet.Make.pop_max : t -> elt * t
BatSet.Make.pop_min : t -> elt * t
BatSet.Make.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatInnerIO.output -> elt -> unit) -> 'a BatInnerIO.output -> t -> unit
BatSet.Make.remove : elt -> t -> t
BatSet.Make.remove_exn : elt -> t -> t
BatSet.Make.singleton : elt -> t
BatSet.Make.split : elt -> t -> t * bool * t
BatSet.Make.split_le : elt -> t -> t * t
BatSet.Make.split_lt : elt -> t -> t * t
BatSet.Make.split_opt : elt -> t -> t * elt option * t
BatSet.Make.subset : t -> t -> bool
BatSet.Make.sym_diff : t -> t -> t
BatSet.Make.to_array : t -> elt array
BatSet.Make.to_list : t -> elt list
BatSet.Make.to_rev_seq : t -> elt BatSeq.t
BatSet.Make.to_seq : t -> elt BatSeq.t
BatSet.Make.to_seq_from : elt -> t -> elt BatSeq.t
BatSet.Make.union : t -> t -> t
BatSet.Make.update : elt -> elt -> t -> t
BatSet.Make2.Product.Exceptionless.any : t -> elt option
BatSet.Make2.Product.Exceptionless.choose : t -> elt option
BatSet.Make2.Product.Exceptionless.find : elt -> t -> elt option
BatSet.Make2.Product.Exceptionless.max_elt : t -> elt option
BatSet.Make2.Product.Exceptionless.min_elt : t -> elt option

Batteries
BatSet.Make2.Product.Labels.exists : f:(elt -> bool) -> t -> bool
BatSet.Make2.Product.Labels.filter : f:(elt -> bool) -> t -> t
BatSet.Make2.Product.Labels.filter_map : f:(elt -> elt option) -> t -> t
BatSet.Make2.Product.Labels.fold : f:(elt -> 'a -> 'a) -> t -> init:'a -> 'a
BatSet.Make2.Product.Labels.for_all : f:(elt -> bool) -> t -> bool
BatSet.Make2.Product.Labels.iter : f:(elt -> unit) -> t -> unit
BatSet.Make2.Product.Labels.map : f:(elt -> elt) -> t -> t
BatSet.Make2.Product.Labels.partition : f:(elt -> bool) -> t -> t * t
BatSet.Make2.Product.add : elt -> t -> t
BatSet.Make2.Product.add_seq : elt BatSeq.t -> t -> t
BatSet.Make2.Product.any : t -> elt
BatSet.Make2.Product.at_rank_exn : int -> t -> elt
BatSet.Make2.Product.backwards : t -> elt BatEnum.t
BatSet.Make2.Product.cardinal : t -> int
BatSet.Make2.Product.choose : t -> elt
BatSet.Make2.Product.choose_opt : t -> elt option
BatSet.Make2.Product.compare : t -> t -> int
BatSet.Make2.Product.compare_subset : t -> t -> int
BatSet.Make2.Product.diff : t -> t -> t
BatSet.Make2.Product.disjoint : t -> t -> bool
BatSet.Make2.Product.elements : t -> elt list
BatSet.Make2.Product.empty : t
BatSet.Make2.Product.enum : t -> elt BatEnum.t
BatSet.Make2.Product.equal : t -> t -> bool
BatSet.Make2.Product.exists : (elt -> bool) -> t -> bool
BatSet.Make2.Product.filter : (elt -> bool) -> t -> t
BatSet.Make2.Product.filter_map : (elt -> elt option) -> t -> t
BatSet.Make2.Product.find : elt -> t -> elt
BatSet.Make2.Product.find_first : (elt -> bool) -> t -> elt
BatSet.Make2.Product.find_first_opt : (elt -> bool) -> t -> elt option
BatSet.Make2.Product.find_last : (elt -> bool) -> t -> elt
BatSet.Make2.Product.find_last_opt : (elt -> bool) -> t -> elt option
BatSet.Make2.Product.find_opt : elt -> t -> elt option
BatSet.Make2.Product.fold : (elt -> 'a -> 'a) -> t -> 'a -> 'a

Batteries
BatSet.Make2.Product.for_all : (elt -> bool) -> t -> bool
BatSet.Make2.Product.inter : t -> t -> t
BatSet.Make2.Product.is_empty : t -> bool
BatSet.Make2.Product.is_singleton : t -> bool
BatSet.Make2.Product.iter : (elt -> unit) -> t -> unit
BatSet.Make2.Product.map : (elt -> elt) -> t -> t
BatSet.Make2.Product.max_elt : t -> elt
BatSet.Make2.Product.max_elt_opt : t -> elt option
BatSet.Make2.Product.mem : elt -> t -> bool
BatSet.Make2.Product.min_elt : t -> elt
BatSet.Make2.Product.min_elt_opt : t -> elt option
BatSet.Make2.Product.of_array : elt array -> t
BatSet.Make2.Product.of_enum : elt BatEnum.t -> t
BatSet.Make2.Product.of_list : elt list -> t
BatSet.Make2.Product.of_seq : elt BatSeq.t -> t
BatSet.Make2.Product.partition : (elt -> bool) -> t -> t * t
BatSet.Make2.Product.pop : t -> elt * t
BatSet.Make2.Product.pop_max : t -> elt * t
BatSet.Make2.Product.pop_min : t -> elt * t
BatSet.Make2.Product.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatInnerIO.output -> elt -> unit) -> 'a BatInnerIO.output -> t -> unit
BatSet.Make2.Product.remove : elt -> t -> t
BatSet.Make2.Product.remove_exn : elt -> t -> t
BatSet.Make2.Product.singleton : elt -> t
BatSet.Make2.Product.split : elt -> t -> t * bool * t
BatSet.Make2.Product.split_le : elt -> t -> t * t
BatSet.Make2.Product.split_lt : elt -> t -> t * t
BatSet.Make2.Product.split_opt : elt -> t -> t * elt option * t
BatSet.Make2.Product.subset : t -> t -> bool
BatSet.Make2.Product.sym_diff : t -> t -> t
BatSet.Make2.Product.to_array : t -> elt array
BatSet.Make2.Product.to_list : t -> elt list
BatSet.Make2.Product.to_rev_seq : t -> elt BatSeq.t
BatSet.Make2.Product.to_seq : t -> elt BatSeq.t
BatSet.Make2.Product.to_seq_from : elt -> t -> elt BatSeq.t

Batteries
BatSet.Make2.Product.union : t -> t -> t
BatSet.Make2.Product.update : elt -> elt -> t -> t
BatSet.Make2.cartesian_product : Make(O1).t -> Make(O2).t -> Product.t
BatSet.PSet.add : 'a -> 'a t -> 'a t
BatSet.PSet.add_seq : 'a BatSeq.t -> 'a t -> 'a t
BatSet.PSet.any : 'a t -> 'a
BatSet.PSet.at_rank_exn : int -> 'a t -> 'a
BatSet.PSet.cardinal : 'a t -> int
BatSet.PSet.choose : 'a t -> 'a
BatSet.PSet.choose_opt : 'a t -> 'a option
BatSet.PSet.compare : 'a t -> 'a t -> int
BatSet.PSet.create : ('a -> 'a -> int) -> 'a t
BatSet.PSet.diff : 'a t -> 'a t -> 'a t
BatSet.PSet.disjoint : 'a t -> 'a t -> bool
BatSet.PSet.elements : 'a t -> 'a list
BatSet.PSet.empty : 'a t
BatSet.PSet.enum : 'a t -> 'a BatEnum.t
BatSet.PSet.equal : 'a t -> 'a t -> bool
BatSet.PSet.exists : ('a -> bool) -> 'a t -> bool
BatSet.PSet.filter : ('a -> bool) -> 'a t -> 'a t
BatSet.PSet.filter_map : ('a -> 'b option) -> 'a t -> 'b t
BatSet.PSet.filter_map_endo : ('a -> 'a option) -> 'a t -> 'a t
BatSet.PSet.find : 'a -> 'a t -> 'a
BatSet.PSet.find_first : ('a -> bool) -> 'a t -> 'a
BatSet.PSet.find_first_opt : ('a -> bool) -> 'a t -> 'a option
BatSet.PSet.find_last : ('a -> bool) -> 'a t -> 'a
BatSet.PSet.find_last_opt : ('a -> bool) -> 'a t -> 'a option
BatSet.PSet.find_opt : 'a -> 'a t -> 'a option
BatSet.PSet.fold : ('a -> 'b -> 'b) -> 'a t -> 'b -> 'b
BatSet.PSet.for_all : ('a -> bool) -> 'a t -> bool
BatSet.PSet.get_cmp : 'a t -> 'a -> 'a -> int
BatSet.PSet.intersect : 'a t -> 'a t -> 'a t
BatSet.PSet.is_empty : 'a t -> bool
BatSet.PSet.is_singleton : 'a t -> bool

Batteries
BatSet.PSet.iter : ('a -> unit) -> 'a t -> unit
BatSet.PSet.map : ('a -> 'b) -> 'a t -> 'b t
BatSet.PSet.map_endo : ('a -> 'a) -> 'a t -> 'a t
BatSet.PSet.max_elt : 'a t -> 'a
BatSet.PSet.max_elt_opt : 'a t -> 'a option
BatSet.PSet.mem : 'a -> 'a t -> bool
BatSet.PSet.min_elt : 'a t -> 'a
BatSet.PSet.min_elt_opt : 'a t -> 'a option
BatSet.PSet.of_array : ?cmp:('a -> 'a -> int) -> 'a array -> 'a t
BatSet.PSet.of_enum : ?cmp:('a -> 'a -> int) -> 'a BatEnum.t -> 'a t
BatSet.PSet.of_enum_cmp : cmp:('a -> 'a -> int) -> 'a BatEnum.t -> 'a t
BatSet.PSet.of_list : ?cmp:('a -> 'a -> int) -> 'a list -> 'a t
BatSet.PSet.of_seq : ?cmp:('a -> 'a -> int) -> 'a BatSeq.t -> 'a t
BatSet.PSet.partition : ('a -> bool) -> 'a t -> 'a t * 'a t
BatSet.PSet.pop : 'a t -> 'a * 'a t
BatSet.PSet.pop_max : 'a t -> 'a * 'a t
BatSet.PSet.pop_min : 'a t -> 'a * 'a t
BatSet.PSet.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatInnerIO.output -> 'c -> unit) -> 'a BatInnerIO.output -> 'c t -> unit
BatSet.PSet.remove : 'a -> 'a t -> 'a t
BatSet.PSet.remove_exn : 'a -> 'a t -> 'a t
BatSet.PSet.singleton : ?cmp:('a -> 'a -> int) -> 'a -> 'a t
BatSet.PSet.split : 'a -> 'a t -> 'a t * bool * 'a t
BatSet.PSet.split_le : 'a -> 'a t -> 'a t * 'a t
BatSet.PSet.split_lt : 'a -> 'a t -> 'a t * 'a t
BatSet.PSet.split_opt : 'a -> 'a t -> 'a t * 'a option * 'a t
BatSet.PSet.subset : 'a t -> 'a t -> bool
BatSet.PSet.sym_diff : 'a t -> 'a t -> 'a t
BatSet.PSet.to_array : 'a t -> 'a array
BatSet.PSet.to_list : 'a t -> 'a list
BatSet.PSet.to_rev_seq : 'a t -> 'a BatSeq.t
BatSet.PSet.to_seq : 'a t -> 'a BatSeq.t
BatSet.PSet.to_seq_from : 'a -> 'a t -> 'a BatSeq.t
BatSet.PSet.union : 'a t -> 'a t -> 'a t
BatSet.PSet.update : 'a -> 'a -> 'a t -> 'a t

Batteries
BatSet.S.Exceptionless.any : t -> elt option
BatSet.S.Exceptionless.choose : t -> elt option
BatSet.S.Exceptionless.find : elt -> t -> elt option
BatSet.S.Exceptionless.max_elt : t -> elt option
BatSet.S.Exceptionless.min_elt : t -> elt option
BatSet.S.Labels.exists : f:(elt -> bool) -> t -> bool
BatSet.S.Labels.filter : f:(elt -> bool) -> t -> t
BatSet.S.Labels.filter_map : f:(elt -> elt option) -> t -> t
BatSet.S.Labels.fold : f:(elt -> 'a -> 'a) -> t -> init:'a -> 'a
BatSet.S.Labels.for_all : f:(elt -> bool) -> t -> bool
BatSet.S.Labels.iter : f:(elt -> unit) -> t -> unit
BatSet.S.Labels.map : f:(elt -> elt) -> t -> t
BatSet.S.Labels.partition : f:(elt -> bool) -> t -> t * t
BatSet.S.add : elt -> t -> t
BatSet.S.add_seq : elt BatSeq.t -> t -> t
BatSet.S.any : t -> elt
BatSet.S.at_rank_exn : int -> t -> elt
BatSet.S.backwards : t -> elt BatEnum.t
BatSet.S.cardinal : t -> int
BatSet.S.choose : t -> elt
BatSet.S.choose_opt : t -> elt option
BatSet.S.compare : t -> t -> int
BatSet.S.compare_subset : t -> t -> int
BatSet.S.diff : t -> t -> t
BatSet.S.disjoint : t -> t -> bool
BatSet.S.elements : t -> elt list
BatSet.S.empty : t
BatSet.S.enum : t -> elt BatEnum.t
BatSet.S.equal : t -> t -> bool
BatSet.S.exists : (elt -> bool) -> t -> bool
BatSet.S.filter : (elt -> bool) -> t -> t
BatSet.S.filter_map : (elt -> elt option) -> t -> t
BatSet.S.find : elt -> t -> elt
BatSet.S.find_first : (elt -> bool) -> t -> elt

Batteries
BatSet.S.find_first_opt : (elt -> bool) -> t -> elt option
BatSet.S.find_last : (elt -> bool) -> t -> elt
BatSet.S.find_last_opt : (elt -> bool) -> t -> elt option
BatSet.S.find_opt : elt -> t -> elt option
BatSet.S.fold : (elt -> 'a -> 'a) -> t -> 'a -> 'a
BatSet.S.for_all : (elt -> bool) -> t -> bool
BatSet.S.inter : t -> t -> t
BatSet.S.is_empty : t -> bool
BatSet.S.is_singleton : t -> bool
BatSet.S.iter : (elt -> unit) -> t -> unit
BatSet.S.map : (elt -> elt) -> t -> t
BatSet.S.max_elt : t -> elt
BatSet.S.max_elt_opt : t -> elt option
BatSet.S.mem : elt -> t -> bool
BatSet.S.min_elt : t -> elt
BatSet.S.min_elt_opt : t -> elt option
BatSet.S.of_array : elt array -> t
BatSet.S.of_enum : elt BatEnum.t -> t
BatSet.S.of_list : elt list -> t
BatSet.S.of_seq : elt BatSeq.t -> t
BatSet.S.partition : (elt -> bool) -> t -> t * t
BatSet.S.pop : t -> elt * t
BatSet.S.pop_max : t -> elt * t
BatSet.S.pop_min : t -> elt * t
BatSet.S.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatInnerIO.output -> elt -> unit) -> 'a BatInnerIO.output -> t -> unit
BatSet.S.remove : elt -> t -> t
BatSet.S.remove_exn : elt -> t -> t
BatSet.S.singleton : elt -> t
BatSet.S.split : elt -> t -> t * bool * t
BatSet.S.split_le : elt -> t -> t * t
BatSet.S.split_lt : elt -> t -> t * t
BatSet.S.split_opt : elt -> t -> t * elt option * t
BatSet.S.subset : t -> t -> bool
BatSet.S.sym_diff : t -> t -> t

Batteries
BatSet.S.to_array : t -> elt array
BatSet.S.to_list : t -> elt list
BatSet.S.to_rev_seq : t -> elt BatSeq.t
BatSet.S.to_seq : t -> elt BatSeq.t
BatSet.S.to_seq_from : elt -> t -> elt BatSeq.t
BatSet.S.union : t -> t -> t
BatSet.S.update : elt -> elt -> t -> t
BatSet.add : 'a -> 'a t -> 'a t
BatSet.add_seq : 'a BatSeq.t -> 'a t -> 'a t
BatSet.any : 'a t -> 'a
BatSet.at_rank_exn : int -> 'a t -> 'a
BatSet.backwards : 'a t -> 'a BatEnum.t
BatSet.cardinal : 'a t -> int
BatSet.cartesian_product : 'a t -> 'b t -> ('a * 'b) t
BatSet.choose : 'a t -> 'a
BatSet.choose_opt : 'a t -> 'a option
BatSet.compare : 'a t -> 'a t -> int
BatSet.diff : 'a t -> 'a t -> 'a t
BatSet.disjoint : 'a t -> 'a t -> bool
BatSet.elements : 'a t -> 'a list
BatSet.empty : 'a t
BatSet.enum : 'a t -> 'a BatEnum.t
BatSet.equal : 'a t -> 'a t -> bool
BatSet.exists : ('a -> bool) -> 'a t -> bool
BatSet.filter : ('a -> bool) -> 'a t -> 'a t
BatSet.filter_map : ('a -> 'b option) -> 'a t -> 'b t
BatSet.filter_map_endo : ('a -> 'a option) -> 'a t -> 'a t
BatSet.find : 'a -> 'a t -> 'a
BatSet.find_first : ('a -> bool) -> 'a t -> 'a
BatSet.find_first_opt : ('a -> bool) -> 'a t -> 'a option
BatSet.find_last : ('a -> bool) -> 'a t -> 'a
BatSet.find_last_opt : ('a -> bool) -> 'a t -> 'a option
BatSet.find_opt : 'a -> 'a t -> 'a option
BatSet.fold : ('a -> 'b -> 'b) -> 'a t -> 'b -> 'b

Batteries
BatSet.for_all : ('a -> bool) -> 'a t -> bool
BatSet.intersect : 'a t -> 'a t -> 'a t
BatSet.is_empty : 'a t -> bool
BatSet.is_singleton : 'a t -> bool
BatSet.iter : ('a -> unit) -> 'a t -> unit
BatSet.map : ('a -> 'b) -> 'a t -> 'b t
BatSet.map_endo : ('a -> 'a) -> 'a t -> 'a t
BatSet.max_elt : 'a t -> 'a
BatSet.max_elt_opt : 'a t -> 'a option
BatSet.mem : 'a -> 'a t -> bool
BatSet.min_elt : 'a t -> 'a
BatSet.min_elt_opt : 'a t -> 'a option
BatSet.of_array : 'a array -> 'a t
BatSet.of_enum : 'a BatEnum.t -> 'a t
BatSet.of_list : 'a list -> 'a t
BatSet.of_seq : 'a BatSeq.t -> 'a t
BatSet.partition : ('a -> bool) -> 'a t -> 'a t * 'a t
BatSet.pop : 'a t -> 'a * 'a t
BatSet.pop_max : 'a t -> 'a * 'a t
BatSet.pop_min : 'a t -> 'a * 'a t
BatSet.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatInnerIO.output -> 'c -> unit) -> 'a BatInnerIO.output -> 'c t -> unit
BatSet.remove : 'a -> 'a t -> 'a t
BatSet.remove_exn : 'a -> 'a t -> 'a t
BatSet.singleton : 'a -> 'a t
BatSet.split : 'a -> 'a t -> 'a t * bool * 'a t
BatSet.split_le : 'a -> 'a t -> 'a t * 'a t
BatSet.split_lt : 'a -> 'a t -> 'a t * 'a t
BatSet.split_opt : 'a -> 'a t -> 'a t * 'a option * 'a t
BatSet.subset : 'a t -> 'a t -> bool
BatSet.sym_diff : 'a t -> 'a t -> 'a t
BatSet.to_array : 'a t -> 'a array
BatSet.to_list : 'a t -> 'a list
BatSet.to_rev_seq : 'a t -> 'a BatSeq.t
BatSet.to_seq : 'a t -> 'a BatSeq.t

Batteries
BatSet.to_seq_from : 'a -> 'a t -> 'a BatSeq.t
BatSet.union : 'a t -> 'a t -> 'a t
BatSet.update : 'a -> 'a -> 'a t -> 'a t
BatString.Cap.Exceptionless.find : [> `Read ] t -> [> `Read] t -> int option
BatString.Cap.Exceptionless.find_from : [> `Read ] t -> int -> [> `Read] t -> int option
BatString.Cap.Exceptionless.index : [> `Read] t -> char -> int option
BatString.Cap.Exceptionless.index_from : [> `Read] t -> int -> char -> int option
BatString.Cap.Exceptionless.rfind : [> `Read ] t -> [> `Read] t -> int option
BatString.Cap.Exceptionless.rfind_from : [> `Read ] t -> int -> [> `Read] t -> int option
BatString.Cap.Exceptionless.rindex : [> `Read] t -> char -> int option
BatString.Cap.Exceptionless.rindex_from : [> `Read] t -> int -> char -> int option
BatString.Cap.Exceptionless.rsplit : [> `Read ] t -> by:[> `Read] t -> ('a t * 'b t) option
BatString.Cap.Exceptionless.split : [> `Read ] t -> by:[> `Read] t -> ('a t * 'b t) option
BatString.Cap.Exceptionless.to_float : [> `Read] t -> float option
BatString.Cap.Exceptionless.to_int : [> `Read] t -> int option
BatString.Cap.backwards : [> `Read] t -> char BatEnum.t
BatString.Cap.blit : [> `Read ] t -> int -> [> `Write] t -> int -> int -> unit
BatString.Cap.capitalize : [> `Read] t -> 'a t
BatString.Cap.chop : ?l:int -> ?r:int -> [> `Read] t -> 'a t
BatString.Cap.compare : [> `Read ] t -> [> `Read] t -> int
BatString.Cap.concat : [> `Read ] t -> [> `Read] t list -> 'a t
BatString.Cap.contains : [> `Read] t -> char -> bool
BatString.Cap.contains_from : [> `Read] t -> int -> char -> bool
BatString.Cap.copy : [> `Read] t -> 'a t
BatString.Cap.count_char : [> `Read] t -> char -> int
BatString.Cap.ends_with : [> `Read ] t -> [> `Read] t -> bool
BatString.Cap.enum : [> `Read] t -> char BatEnum.t
BatString.Cap.escaped : [> `Read] t -> 'a t
BatString.Cap.exists : [> `Read ] t -> [> `Read] t -> bool
BatString.Cap.explode : [> `Read] t -> char list
BatString.Cap.fill : [> `Write] t -> int -> int -> char -> unit
BatString.Cap.filter : (char -> bool) -> [> `Read] t -> 'a t
BatString.Cap.filter_map : (char -> char option) -> [> `Read] t -> 'a t
BatString.Cap.find : [> `Read ] t -> [> `Read] t -> int

Batteries
BatString.Cap.find_from : [> `Read ] t -> int -> [> `Read] t -> int
BatString.Cap.fold_left : ('a -> char -> 'a) -> 'a -> [> `Read] t -> 'a
BatString.Cap.fold_lefti : ('a -> int -> char -> 'a) -> 'a -> [> `Read] t -> 'a
BatString.Cap.fold_right : (char -> 'a -> 'a) -> [> `Read] t -> 'a -> 'a
BatString.Cap.fold_righti : (int -> char -> 'a -> 'a) -> [> `Read] t -> 'a -> 'a
BatString.Cap.head : [> `Read] t -> int -> 'a t
BatString.Cap.icompare : [> `Read ] t -> [> `Read] t -> int
BatString.Cap.implode : char list -> 'a t
BatString.Cap.index : [> `Read] t -> char -> int
BatString.Cap.index_from : [> `Read] t -> int -> char -> int
BatString.Cap.init : int -> (int -> char) -> 'a t
BatString.Cap.is_empty : 'a t -> bool
BatString.Cap.iter : (char -> unit) -> [> `Read] t -> unit
BatString.Cap.join : [> `Read ] t -> [> `Read] t list -> 'a t
BatString.Cap.lchop : ?n:int -> [> `Read] t -> 'a t
BatString.Cap.left : [> `Read] t -> int -> 'a t
BatString.Cap.lowercase : [> `Read] t -> 'a t
BatString.Cap.make : int -> char -> 'a t
BatString.Cap.map : (char -> char) -> [> `Read] t -> 'a t
BatString.Cap.mapi : (int -> char -> char) -> [> `Read] t -> 'a t
BatString.Cap.nreplace : str:[> `Read ] t -> sub:[> `Read] t -> by:[> `Read] t -> 'a t
BatString.Cap.nsplit : [> `Read ] t -> by:[> `Read] t -> 'a t list
BatString.Cap.of_backwards : char BatEnum.t -> 'a t
BatString.Cap.of_bytes : Bytes.t -> 'a t
BatString.Cap.of_char : char -> 'a t
BatString.Cap.of_enum : char BatEnum.t -> 'a t
BatString.Cap.of_float : float -> 'a t
BatString.Cap.of_int : int -> 'a t
BatString.Cap.of_list : char list -> 'a t
BatString.Cap.print : 'a BatInnerIO.output -> [> `Read] t -> unit
BatString.Cap.print_quoted : 'a BatInnerIO.output -> [> `Read] t -> unit
BatString.Cap.println : 'a BatInnerIO.output -> [> `Read] t -> unit
BatString.Cap.quote : [> `Read] t -> string
BatString.Cap.rchop : ?n:int -> [> `Read] t -> 'a t

Batteries
BatString.Cap.rcontains_from : [> `Read] t -> int -> char -> bool
BatString.Cap.repeat : [> `Read] t -> int -> 'a t
BatString.Cap.replace : str:[> `Read ] t -> sub:[> `Read] t -> by:[> `Read] t -> bool * 'a t
BatString.Cap.replace_chars : (char -> [> `Read ] t) -> [> `Read] t -> 'a t
BatString.Cap.rfind : [> `Read ] t -> [> `Read] t -> int
BatString.Cap.rfind_from : [> `Read ] t -> int -> [> `Read] t -> int
BatString.Cap.right : [> `Read] t -> int -> 'a t
BatString.Cap.rindex : [> `Read] t -> char -> int
BatString.Cap.rindex_from : [> `Read] t -> int -> char -> int
BatString.Cap.rsplit : [> `Read ] t -> by:[> `Read] t -> 'a t * 'b t
BatString.Cap.slice : ?first:int -> ?last:int -> [> `Read] t -> 'a t
BatString.Cap.splice : [`Read | `Write] t -> int -> int -> [> `Read] t -> 'a t
BatString.Cap.split : [> `Read ] t -> by:[> `Read] t -> 'a t * 'b t
BatString.Cap.starts_with : [> `Read ] t -> [> `Read] t -> bool
BatString.Cap.strip : ?chars:[> `Read ] t -> [> `Read] t -> 'a t
BatString.Cap.sub : [> `Read] t -> int -> int -> 'a t
BatString.Cap.tail : [> `Read] t -> int -> 'a t
BatString.Cap.to_float : [> `Read] t -> float
BatString.Cap.to_int : [> `Read] t -> int
BatString.Cap.to_list : [> `Read] t -> char list
BatString.Cap.trim : [> `Read] t -> 'a t
BatString.Cap.uncapitalize : [> `Read] t -> 'a t
BatString.Cap.uppercase : [> `Read] t -> 'a t
BatString.Exceptionless.find : string -> string -> int option
BatString.Exceptionless.find_from : string -> int -> string -> int option
BatString.Exceptionless.index : string -> char -> int option
BatString.Exceptionless.index_from : string -> int -> char -> int option
BatString.Exceptionless.rfind : string -> string -> int option
BatString.Exceptionless.rfind_from : string -> int -> string -> int option
BatString.Exceptionless.rindex : string -> char -> int option
BatString.Exceptionless.rindex_from : string -> int -> char -> int option
BatString.Exceptionless.rsplit : string -> by:string -> (string * string) option
BatString.Exceptionless.split : string -> by:string -> (string * string) option
BatString.Exceptionless.to_float : string -> float option

Batteries
BatString.Exceptionless.to_int : string -> int option
BatString.backwards : string -> char BatEnum.t
BatString.blit : string -> int -> Bytes.t -> int -> int -> unit
BatString.capitalize : string -> string
BatString.capitalize_ascii : string -> string
BatString.cat : string -> string -> string
BatString.chop : ?l:int -> ?r:int -> string -> string
BatString.compare : t -> t -> int
BatString.concat : string -> string list -> string
BatString.contains : string -> char -> bool
BatString.contains_from : string -> int -> char -> bool
BatString.copy : string -> string
BatString.count_char : string -> char -> int
BatString.count_string : string -> string -> int
BatString.cut_on_char : char -> int -> string -> string
BatString.edit_distance : t -> t -> int
BatString.empty : string
BatString.ends_with : string -> string -> bool
BatString.ends_with_stdlib : suffix:string -> string -> bool
BatString.enum : string -> char BatEnum.t
BatString.equal : t -> t -> bool
BatString.escaped : string -> string
BatString.exists : string -> string -> bool
BatString.exists_stdlib : (char -> bool) -> string -> bool
BatString.explode : string -> char list
BatString.fill : Bytes.t -> int -> int -> char -> unit
BatString.filter : (char -> bool) -> string -> string
BatString.filter_map : (char -> char option) -> string -> string
BatString.find : string -> string -> int
BatString.find_all : string -> string -> int BatEnum.t
BatString.find_from : string -> int -> string -> int
BatString.fold_left : ('a -> char -> 'a) -> 'a -> string -> 'a
BatString.fold_lefti : ('a -> int -> char -> 'a) -> 'a -> string -> 'a
BatString.fold_right : (char -> 'a -> 'a) -> string -> 'a -> 'a

Batteries
BatString.fold_righti : (int -> char -> 'a -> 'a) -> string -> 'a -> 'a
BatString.for_all : (char -> bool) -> string -> bool
BatString.get_int16_be : string -> int -> int
BatString.get_int16_le : string -> int -> int
BatString.get_int16_ne : string -> int -> int
BatString.get_int32_be : string -> int -> int32
BatString.get_int32_le : string -> int -> int32
BatString.get_int32_ne : string -> int -> int32
BatString.get_int64_be : string -> int -> int64
BatString.get_int64_le : string -> int -> int64
BatString.get_int64_ne : string -> int -> int64
BatString.get_int8 : string -> int -> int
BatString.get_uint16_be : string -> int -> int
BatString.get_uint16_le : string -> int -> int
BatString.get_uint16_ne : string -> int -> int
BatString.get_uint8 : string -> int -> int
BatString.get_utf_16be_uchar : t -> int -> Uchar.utf_decode
BatString.get_utf_16le_uchar : t -> int -> Uchar.utf_decode
BatString.get_utf_8_uchar : t -> int -> Uchar.utf_decode
BatString.head : string -> int -> string
BatString.icompare : t -> t -> int
BatString.implode : char list -> string
BatString.in_place_mirror : Bytes.t -> unit
BatString.index : string -> char -> int
BatString.index_after_n : char -> int -> string -> int
BatString.index_from : string -> int -> char -> int
BatString.index_from_opt : string -> int -> char -> int option
BatString.index_opt : string -> char -> int option
BatString.init : int -> (int -> char) -> string
BatString.is_empty : string -> bool
BatString.is_valid_utf_16be : t -> bool
BatString.is_valid_utf_16le : t -> bool
BatString.is_valid_utf_8 : t -> bool
BatString.iter : (char -> unit) -> string -> unit

Batteries
BatString.iteri : (int -> char -> unit) -> string -> unit
BatString.join : string -> string list -> string
BatString.lchop : ?n:int -> string -> string
BatString.left : string -> int -> string
BatString.lowercase : string -> string
BatString.lowercase_ascii : string -> string
BatString.make : int -> char -> string
BatString.map : (char -> char) -> string -> string
BatString.mapi : (int -> char -> char) -> string -> string
BatString.nreplace : str:string -> sub:string -> by:string -> string
BatString.nsplit : string -> by:string -> string list
BatString.numeric_compare : t -> t -> int
BatString.of_backwards : char BatEnum.t -> string
BatString.of_bytes : Bytes.t -> string
BatString.of_char : char -> string
BatString.of_enum : char BatEnum.t -> string
BatString.of_float : float -> string
BatString.of_int : int -> string
BatString.of_list : char list -> string
BatString.of_seq : char Seq.t -> t
BatString.ord : t -> t -> BatOrd.order
BatString.print : 'a BatInnerIO.output -> string -> unit
BatString.print_quoted : 'a BatInnerIO.output -> string -> unit
BatString.println : 'a BatInnerIO.output -> string -> unit
BatString.quote : string -> string
BatString.rchop : ?n:int -> string -> string
BatString.rcontains_from : string -> int -> char -> bool
BatString.repeat : string -> int -> string
BatString.replace : str:string -> sub:string -> by:string -> bool * string
BatString.replace_chars : (char -> string) -> string -> string
BatString.rev : string -> string
BatString.rev_in_place : Bytes.t -> unit
BatString.rfind : string -> string -> int
BatString.rfind_from : string -> int -> string -> int

Batteries
BatString.right : string -> int -> string
BatString.rindex : string -> char -> int
BatString.rindex_from : string -> int -> char -> int
BatString.rindex_from_opt : string -> int -> char -> int option
BatString.rindex_opt : string -> char -> int option
BatString.rsplit : string -> by:string -> string * string
BatString.slice : ?first:int -> ?last:int -> string -> string
BatString.splice : string -> int -> int -> string -> string
BatString.split : string -> by:string -> string * string
BatString.split_on_char : char -> string -> string list
BatString.split_on_string : by:string -> string -> string list
BatString.starts_with : string -> string -> bool
BatString.starts_with_stdlib : prefix:string -> string -> bool
BatString.strip : ?chars:string -> string -> string
BatString.sub : string -> int -> int -> string
BatString.tail : string -> int -> string
BatString.to_bytes : string -> Bytes.t
BatString.to_float : string -> float
BatString.to_int : string -> int
BatString.to_list : string -> char list
BatString.to_seq : t -> char Seq.t
BatString.to_seqi : t -> (int * char) Seq.t
BatString.trim : string -> string
BatString.uncapitalize : string -> string
BatString.uncapitalize_ascii : string -> string
BatString.uppercase : string -> string
BatString.uppercase_ascii : string -> string