Batteries
BatArray.avg : int array -> float
BatArray.backwards : 'a array -> 'a BatEnum.t
BatArray.cartesian_product : 'a array -> 'b array -> ('a * 'b) array
BatArray.count_matching : ('a -> bool) -> 'a array -> int
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.favg : float array -> float
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.findi : ('a -> bool) -> 'a array -> int
BatArray.fold_lefti : ('a -> int -> 'b -> 'a) -> 'a -> 'b array -> 'a
BatArray.fold_righti : (int -> 'b -> 'a -> 'a) -> 'b array -> 'a -> 'a
BatArray.fsum : float array -> float
BatArray.head : 'a array -> int -> 'a array
BatArray.insert : 'a array -> 'a -> int -> 'a array
BatArray.is_sorted_by : ('a -> 'b) -> 'a array -> bool
BatArray.iter2i : (int -> 'a -> 'b -> unit) -> 'a array -> 'b array -> unit
BatArray.kahan_sum : float array -> float
BatArray.Labels.count_matching : f:('a -> bool) -> 'a t -> int
BatArray.Labels.create : int -> init:'a -> 'a array
BatArray.Labels.find : f:('a -> bool) -> 'a t -> 'a
BatArray.Labels.findi : f:('a -> bool) -> 'a t -> int
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.iter2i : f:(int -> 'a -> 'b -> unit) -> 'a t -> 'b t -> unit
BatArray.Labels.modifyi : f:(int -> 'a -> 'a) -> 'a array -> unit
BatArray.left : 'a array -> int -> 'a array
BatArray.max : 'a array -> 'a
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

Batteries
BatArray.of_backwards : 'a BatEnum.t -> 'a array
BatArray.of_enum : 'a BatEnum.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.right : 'a array -> int -> 'a array
BatArray.singleton : 'a -> 'a array
BatArray.sum : int array -> int
BatArray.tail : 'a array -> int -> 'a array
BatList.assoc_inv : 'b -> ('a * 'b) list -> 'a
BatList.assq_inv : 'b -> ('a * 'b) list -> 'a
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.count_matching : ('a -> bool) -> 'a list -> int
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.favg : float list -> float
BatList.filteri_map : (int -> 'a -> 'b option) -> 'a list -> 'b list
BatList.find_exn : ('a -> bool) -> exn -> 'a list -> 'a
BatList.find_map_opt : ('a -> 'b option) -> 'a list -> 'b option
BatList.findi : (int -> 'a -> bool) -> 'a list -> int * 'a
BatList.first : 'a list -> 'a
BatList.fold : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a
BatList.fold_lefti : ('a -> int -> 'b -> 'a) -> 'a -> 'b list -> 'a
BatList.fold_righti : (int -> 'b -> 'a -> 'a) -> 'b list -> 'a -> 'a
BatList.frange : float -> [< `Downto | `To] -> float -> int -> float list
BatList.fsum : float list -> float

Batteries
BatList.group : ('a -> 'a -> int) -> 'a list -> 'a list list
BatList.index_of : 'a -> 'a list -> int option
BatList.index_ofq : 'a -> 'a list -> int option
BatList.kahan_sum : float list -> float
BatList.Labels.count_matching : f:('a -> bool) -> 'a list -> int
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.remove_if : f:('a -> bool) -> 'a list -> 'a list
BatList.Labels.rfind : f:('a -> bool) -> 'a list -> 'a
BatList.make : int -> 'a -> 'a list
BatList.map2i : (int -> 'a -> 'b -> 'c) -> 'a list -> 'b list -> 'c list
BatList.max : ?cmp:('a -> 'a -> int) -> 'a list -> 'a
BatList.mem_cmp : ('a -> 'a -> int) -> 'a -> 'a list -> bool
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.nsplit : ('a -> bool) -> 'a list -> 'a list list
BatList.ntake : int -> 'a list -> 'a list list
BatList.of_backwards : 'a BatEnum.t -> 'a list
BatList.of_enum : 'a BatEnum.t -> 'a list
BatList.ord : 'a BatOrd.ord -> 'a list BatOrd.ord
BatList.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatInnerIO.output -> 'b -> unit) -> 'a BatInnerIO.output -> 'b list -> unit
BatList.remove_all : 'a list -> 'a -> 'a list
BatList.remove_if : ('a -> bool) -> 'a list -> 'a 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

Batteries
BatList.singleton : 'a -> 'a list
BatList.sort_unique : ('a -> 'a -> int) -> 'a list -> 'a list
BatList.span : ('a -> bool) -> 'a list -> 'a list * 'a list
BatList.split_at : int -> 'a list -> 'a list * 'a list
BatList.split_nth : int -> 'a list -> 'a list * 'a list
BatList.sum : int list -> int
BatList.takewhile : ('a -> bool) -> 'a list -> 'a list
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_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.add_carry : 'a -> 'b -> ('a, 'b) t -> ('a, 'b) t * 'b option
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.diff : ('a, 'b) t -> ('a, 'b) t -> ('a, 'b) t
BatMap.enum : ('a, 'b) t -> ('a * 'b) BatEnum.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.extract : 'a -> ('a, 'b) t -> 'b * ('a, 'b) t
BatMap.filterv : ('a -> bool) -> ('key, 'a) t -> ('key, 'a) t
BatMap.find_default : 'b -> 'a -> ('a, 'b) t -> 'b
BatMap.foldi : ('a -> 'b -> 'c -> 'c) -> ('a, 'b) t -> 'c -> 'c
BatMap.intersect : ('b -> 'c -> 'd) -> ('a, 'b) t -> ('a, 'c) t -> ('a, 'd) t
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
BatMap.of_enum : ('a * 'b) BatEnum.t -> ('a, 'b) t
BatMap.pop : ('a, 'b) t -> ('a * 'b) * ('a, 'b) t

Batteries
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_exn : 'a -> ('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_stdlib : 'a -> ('b option -> 'b option) -> ('a, 'b) t -> ('a, 'b) t
BatOption.(|?) : 'a option -> 'a -> 'a
BatOption.apply : ('a -> 'a) option -> 'a -> 'a
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
BatOption.get : 'a option -> 'a
BatOption.Infix.(>>=) : 'a option -> ('a -> 'b option) -> 'b option
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.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.Monad.bind : 'a m -> ('a -> 'b m) -> 'b m
BatOption.Monad.return : 'a -> 'a m
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
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

Batteries
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.bind : ('a, 'e) t -> ('a -> ('b, 'e) t) -> ('b, 'e) t
BatResult.catch2 : ('a -> 'b -> 'c) -> 'a -> 'b -> ('c, exn) t
BatResult.catch3 : ('a -> 'b -> 'c -> 'd) -> 'a -> 'b -> 'c -> ('d, exn) t
BatResult.default : 'a -> ('a, 'b) t -> 'a
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_exn : exn -> ('a, exn) t -> bool
BatResult.map_default : 'b -> ('a -> 'b) -> ('a, 'c) t -> 'b
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.value : ('a, 'e) t -> default:'a -> 'a
BatSeq.(---) : int -> int -> int t
BatSeq.(--.) : float * float -> float -> float t
BatSeq.(--~) : char -> char -> char t
BatSeq.(@/) : ('a -> 'b) -> 'a t -> 'b t
BatSeq.(@//) : ('a -> 'b option) -> 'a t -> 'b t
BatSeq.(/@) : 'a t -> ('a -> 'b) -> 'b t
BatSeq.(//) : 'a t -> ('a -> bool) -> 'a t
BatSeq.(//@) : 'a t -> ('a -> 'b option) -> 'b 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.concat : 'a t t -> 'a t
BatSeq.concat_map : ('a -> 'b t) -> 'a t -> 'b t
BatSeq.enum : 'a t -> 'a BatEnum.t

Batteries
BatSeq.equal_stdlib : ('a -> 'b -> bool) -> 'a t -> 'b t -> bool
BatSeq.find : ('a -> bool) -> 'a t -> 'a option
BatSeq.find_map : ('a -> 'b option) -> 'a t -> 'b option
BatSeq.first : 'a t -> 'a
BatSeq.fold_lefti : ('b -> int -> 'a -> 'b) -> 'b -> 'a t -> 'b
BatSeq.fold_right : ('a -> 'b -> 'b) -> 'a t -> 'b -> 'b
BatSeq.forever : (unit -> 'a) -> 'a t
BatSeq.init : int -> (int -> 'a) -> 'a t
BatSeq.ints : int -> int t
BatSeq.iterate : ('a -> 'a) -> 'a -> 'a t
BatSeq.last : 'a t -> 'a
BatSeq.make : int -> 'a -> 'a t
BatSeq.map_product : ('a -> 'b -> 'c) -> 'a t -> 'b t -> 'c t
BatSeq.max : 'a t -> 'a
BatSeq.mem : 'a -> 'a t -> bool
BatSeq.min : 'a t -> 'a
BatSeq.of_dispenser : (unit -> 'a option) -> 'a t
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.reduce : ('a -> 'a -> 'a) -> 'a t -> 'a
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.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
BatSet.any : 'a t -> 'a
BatSet.at_rank_exn : int -> 'a t -> 'a
BatSet.backwards : 'a t -> 'a BatEnum.t
BatSet.cartesian_product : 'a t -> 'b t -> ('a * 'b) t

Batteries
BatSet.enum : 'a t -> 'a BatEnum.t
BatSet.filter_map_endo : ('a -> 'a option) -> 'a t -> 'a t
BatSet.is_singleton : 'a t -> bool
BatSet.map_endo : ('a -> 'a) -> 'a t -> 'a t
BatSet.of_array : 'a array -> 'a t
BatSet.of_enum : 'a BatEnum.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_exn : 'a -> 'a t -> '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.sym_diff : 'a t -> 'a t -> 'a t
BatSet.to_array : 'a t -> 'a array
BatString.backwards : string -> char BatEnum.t
BatString.chop : ?l:int -> ?r:int -> string -> string
BatString.count_char : string -> char -> int
BatString.count_string : string -> string -> int
BatString.cut_on_char : char -> int -> string -> string
BatString.ends_with_stdlib : suffix:string -> string -> bool
BatString.enum : string -> char BatEnum.t
BatString.exists_stdlib : (char -> bool) -> string -> bool
BatString.explode : string -> char list
BatString.find_from : string -> int -> string -> int
BatString.fold_lefti : ('a -> int -> char -> 'a) -> 'a -> string -> 'a
BatString.fold_righti : (int -> char -> 'a -> 'a) -> string -> 'a -> 'a
BatString.head : string -> int -> string
BatString.icompare : t -> t -> int
BatString.implode : char list -> string
BatString.in_place_mirror : Bytes.t -> unit
BatString.index_after_n : char -> int -> string -> int
BatString.join : string -> string list -> string

Batteries
BatString.lchop : ?n:int -> string -> string
BatString.left : string -> int -> 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_enum : char BatEnum.t -> string
BatString.of_float : float -> string
BatString.of_int : int -> string
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.replace_chars : (char -> string) -> string -> string
BatString.rev_in_place : Bytes.t -> unit
BatString.rfind_from : string -> int -> string -> int
BatString.right : string -> int -> string
BatString.rsplit : string -> by:string -> string * string
BatString.slice : ?first:int -> ?last:int -> string -> string
BatString.splice : string -> int -> int -> string -> string
BatString.split_on_string : by:string -> string -> string list
BatString.starts_with_stdlib : prefix:string -> string -> bool
BatString.strip : ?chars:string -> string -> string
BatString.tail : string -> int -> string
BatString.to_float : string -> float
BatString.to_int : string -> int