Batteries	Base
BatArray.cartesian_product : 'a array -> 'b array -> ('a * 'b) array	Base.Array.cartesian_product : 'a array -> 'b array -> ('a * 'b) array
BatArray.Labels.count_matching : f:('a -> bool) -> 'a array -> int	Base.Array.count : 'a array -> f:('a -> bool) -> int
BatArray.Labels.find : f:('a -> bool) -> 'a array -> 'a	Base.Array.find_exn : 'a array -> f:('a -> bool) -> 'a
BatArray.partition : ('a -> bool) -> 'a array -> 'a array * 'a array	Base.Array.partition_tf : 'a array -> f:('a -> bool) -> 'a array * 'a array
BatArray.reduce : ('a -> 'a -> 'a) -> 'a array -> 'a	Base.Array.reduce_exn : 'a array -> f:('a -> 'a -> 'a) -> 'a
BatArray.split : ('a * 'b) array -> 'a array * 'b array	Base.Array.unzip : ('a * 'b) array -> 'a array * 'b array
BatList.cartesian_product : 'a list -> 'b list -> ('a * 'b) list	Base.List.cartesian_product : 'a list -> 'b list -> ('a * 'b) list
BatList.Labels.fold : f:('a -> 'b -> 'a) -> init:'a -> 'b list -> 'a	Base.List.fold : 'a list -> init:'accum -> f:('accum -> 'a -> 'accum) -> 'accum
BatList.last : 'a list -> 'a	Base.List.last_exn : 'a list -> 'a
BatList.transpose : 'a list list -> 'a list list	Base.List.transpose_exn : 'a list list -> 'a list list
BatOption.some : 'a -> 'a option	Base.Option.some : 'a -> 'a option
BatPrintf.bprintf : Buffer.t -> ('a, Buffer.t, unit) BatPrintf.t -> 'a	Base.Printf.bprintf : Base.Import0.Caml.Buffer.t -> ('r, Base.Import0.Caml.Buffer.t, unit) format -> 'r
BatPrintf.eprintf : ('b, 'a BatInnerIO.output, unit) BatPrintf.t -> 'b	(* Base.Printf.eprintf *)
BatPrintf.fprintf : 'a BatInnerIO.output -> ('b, 'a BatInnerIO.output, unit) BatPrintf.t -> 'b	(* Base.Printf.fprintf *)
BatPrintf.ifprintf : 'c -> ('b, 'a BatInnerIO.output, unit) BatPrintf.t -> 'b	Base.Printf.ifprintf : 'a -> ('r, 'a, 'c, unit) format4 -> 'r
BatPrintf.kbprintf : (Buffer.t -> 'a) -> Buffer.t -> ('b, Buffer.t, unit, 'a) format4 -> 'b	Base.Printf.kbprintf : (Base.Import0.Caml.Buffer.t -> 'a) -> Base.Import0.Caml.Buffer.t -> ('r, Base.Import0.Caml.Buffer.t, unit, 'a) format4 -> 'r
BatPrintf.kfprintf : ('a BatInnerIO.output -> 'b) -> 'a BatInnerIO.output -> ('c, 'a BatInnerIO.output, unit, 'b) format4 -> 'c	(* Base.Printf.kfprintf *)
BatPrintf.kprintf : (string -> 'a) -> ('b, unit, string, 'a) format4 -> 'b	(* Base.Printf.kprintf *)
BatPrintf.ksprintf : (string -> 'a) -> ('b, unit, string, 'a) format4 -> 'b	Base.Printf.ksprintf : (string -> 'a) -> ('r, unit, string, 'a) format4 -> 'r
BatPrintf.printf : ('b, 'a BatInnerIO.output, unit) BatPrintf.t -> 'b	(* Base.Printf.printf *)
BatPrintf.sprintf : ('a, unit, string) BatPrintf.t -> 'a	Base.Printf.sprintf : ('r, unit, string) format -> 'r
BatResult.bind : ('a, 'e) result -> ('a -> ('b, 'e) result) -> ('b, 'e) result	Base.Result.bind : ('a, 'e) result-> f:('a -> ('b, 'e) result) -> ('b, 'e) result
BatSeq.concat : 'a Seq.t Seq.t -> 'a Seq.t	Base.Sequence.concat : 'a Base.Sequence.t Base.Sequence.t -> 'a Base.Sequence.t
BatSeq.find : ('a -> bool) -> 'a Seq.t -> 'a option	Base.Sequence.find : 'a Base.Sequence.t -> f:('a -> bool) -> 'a option
BatSeq.find_map : ('a -> 'b option) -> 'a Seq.t -> 'b option	Base.Sequence.find_map : 'a Base.Sequence.t -> f:('a -> 'b option) -> 'b option
BatSeq.init : int -> (int -> 'a) -> 'a Seq.t	Base.Sequence.init : int -> f:(int -> 'a) -> 'a Base.Sequence.t
BatSeq.reduce : ('a -> 'a -> 'a) -> 'a Seq.t -> 'a	Base.Sequence.reduce_exn : 'a Base.Sequence.t -> f:('a -> 'a -> 'a) -> 'a
BatSeq.tl : 'a Seq.t -> 'a Seq.t	Base.Sequence.tl_eagerly_exn : 'a Base.Sequence.t -> 'a Base.Sequence.t
BatSet.of_array : 'a array -> 'a set	Base.Set.of_array : ('a, 'cmp) Base.Set.comparator -> 'a array -> ('a, 'cmp) set
BatSet.split_opt : 'a -> 'a set -> 'a set * 'a option * 'a set	Base.Set.split : ('a, 'cmp) set -> 'a -> ('a, 'cmp) set * 'a option * ('a, 'cmp) set
BatSet.sym_diff : 'a set -> 'a set -> 'a set	Base.Set.symmetric_diff : ('a, 'cmp) set -> ('a, 'cmp) set -> ('a, 'a) Base.Either.t Base.Sequence.t
BatSet.to_array : 'a set -> 'a array	Base.Set.to_array : ('a, 'b) set -> 'a array
BatString.strip : ?chars:string -> string -> string	Base.String.strip : ?drop:(char -> bool) -> string -> string