Stdlib	Containers	Batteries	Base
Set.Make.add	CCSet.Make.add	BatSet.add : 'a -> 'a set -> 'a set	Base.Set.add : ('a, 'cmp) set -> 'a -> ('a, 'cmp) set
	CCSet.Make.add_iter		
	CCSet.Make.add_list		
Set.Make.add_seq	CCSet.Make.add_seq	BatSet.add_seq : 'a Seq.t -> 'a set -> 'a set	
		BatSet.any : 'a set -> 'a	
			Base.Set.are_disjoint : ('a, 'cmp) set -> ('a, 'cmp) set -> bool
		BatSet.at_rank_exn : int -> 'a set -> 'a	
		BatSet.backwards : 'a set -> 'a BatEnum.t	
			Base.Set.binary_search : ('a, 'cmp) set -> compare:('a -> 'key -> int) -> [\`First_equal_to | \`First_greater_than_or_equal_to | \`First_strictly_greater_than | \`Last_equal_to | \`Last_less_than_or_equal_to | \`Last_strictly_less_than] -> 'key -> 'a option
			Base.Set.binary_search_segmented : ('a, 'cmp) set -> segment_of:('a -> [\`Left | \`Right]) -> [\`First_on_right | \`Last_on_left] -> 'a option
Set.Make.cardinal	CCSet.Make.cardinal	BatSet.cardinal : 'a set -> int	
		BatSet.cartesian_product : 'a set -> 'b set -> ('a * 'b) set	
Set.Make.choose	CCSet.Make.choose	BatSet.choose : 'a set -> 'a	Base.Set.choose_exn : ('a, 'b) set -> 'a
Set.Make.choose_opt	CCSet.Make.choose_opt	BatSet.choose_opt : 'a set -> 'a option	Base.Set.choose : ('a, 'b) set -> 'a option
			Base.Set.comparator : ('a, 'cmp) set -> ('a, 'cmp) Base.Comparator.t
			Base.Set.comparator_s : ('a, 'cmp) set -> ('a, 'cmp) Base.Set.comparator
			Base.Set.compare : ('elt -> 'elt -> int) -> ('cmp -> 'cmp -> int) -> ('elt, 'cmp) set -> ('elt, 'cmp) set -> int
Set.Make.compare	CCSet.Make.compare	BatSet.compare : 'a set -> 'a set -> int	Base.Set.compare_direct : ('a, 'cmp) set -> ('a, 'cmp) set -> int
			Base.Set.compare_m__t : (module Base.Set.Compare_m) -> ('elt, 'cmp) set -> ('elt, 'cmp) set -> int
			Base.Set.count : ('a, 'b) set -> f:('a -> bool) -> int
Set.Make.diff	CCSet.Make.diff	BatSet.diff : 'a set -> 'a set -> 'a set	Base.Set.diff : ('a, 'cmp) set -> ('a, 'cmp) set -> ('a, 'cmp) set
Set.Make.disjoint	CCSet.Make.disjoint	BatSet.disjoint : 'a set -> 'a set -> bool	
Set.Make.elements	CCSet.Make.elements	BatSet.elements : 'a set -> 'a list	Base.Set.elements : ('a, 'b) set -> 'a list
Set.Make.empty	CCSet.Make.empty	BatSet.empty : 'a set	Base.Set.empty : ('a, 'cmp) Base.Set.comparator -> ('a, 'cmp) set
		BatSet.enum : 'a set -> 'a BatEnum.t	
Set.Make.equal	CCSet.Make.equal	BatSet.equal : 'a set -> 'a set -> bool	Base.Set.equal : ('a, 'cmp) set -> ('a, 'cmp) set -> bool
			Base.Set.equal_m__t : (module Base.Set.Equal_m) -> ('elt, 'cmp) set -> ('elt, 'cmp) set -> bool
Set.Make.exists	CCSet.Make.exists	BatSet.exists : ('a -> bool) -> 'a set -> bool	Base.Set.exists : ('a, 'b) set -> f:('a -> bool) -> bool
Set.Make.filter	CCSet.Make.filter	BatSet.filter : ('a -> bool) -> 'a set -> 'a set	Base.Set.filter : ('a, 'cmp) set -> f:('a -> bool) -> ('a, 'cmp) set
Set.Make.filter_map	CCSet.Make.filter_map	BatSet.filter_map : ('a -> 'b option) -> 'a set -> 'b set	Base.Set.filter_map : ('b, 'cmp) Base.Set.comparator -> ('a, 'c) set -> f:('a -> 'b option) -> ('b, 'cmp) set
		BatSet.filter_map_endo : ('a -> 'a option) -> 'a set -> 'a set	
Set.Make.find	CCSet.Make.find	BatSet.find : 'a -> 'a set -> 'a	
Set.Make.find_first	CCSet.Make.find_first	BatSet.find_first : ('a -> bool) -> 'a set -> 'a	Base.Set.find_exn : ('a, 'b) set -> f:('a -> bool) -> 'a
Set.Make.find_first_opt	CCSet.Make.find_first_opt	BatSet.find_first_opt : ('a -> bool) -> 'a set -> 'a option	Base.Set.find : ('a, 'b) set -> f:('a -> bool) -> 'a option
Set.Make.find_last	CCSet.Make.find_last	BatSet.find_last : ('a -> bool) -> 'a set -> 'a	
Set.Make.find_last_opt	CCSet.Make.find_last_opt	BatSet.find_last_opt : ('a -> bool) -> 'a set -> 'a option	
			Base.Set.find_map : ('a, 'c) set -> f:('a -> 'b option) -> 'b option
Set.Make.find_opt	CCSet.Make.find_opt	BatSet.find_opt : 'a -> 'a set -> 'a option	
Set.Make.fold	CCSet.Make.fold	BatSet.fold : ('a -> 'b -> 'b) -> 'a set -> 'b -> 'b	Base.Set.fold : ('a, 'b) set -> init:'accum -> f:('accum -> 'a -> 'accum) -> 'accum
			Base.Set.fold_result : ('a, 'b) set -> init:'accum -> f:('accum -> 'a -> ('accum, 'e) Base.Result.t) -> ('accum, 'e) Base.Result.t

Stdlib	Containers	Batteries	Base
			Base.Set.fold_right : ('a, 'b) set -> init:'accum -> f:('a -> 'accum -> 'accum) -> 'accum
			Base.Set.fold_until : ('a, 'b) set -> init:'accum -> f:('accum -> 'a -> ('accum, 'final) Base.Set_intf.Continue_or_stop.t) -> finish:('accum -> 'final) -> 'final
Set.Make.for_all	CCSet.Make.for_all	BatSet.for_all : ('a -> bool) -> 'a set -> bool	Base.Set.for_all : ('a, 'b) set -> f:('a -> bool) -> bool
			Base.Set.group_by : ('a, 'cmp) set -> equiv:('a -> 'a -> bool) -> ('a, 'cmp) set list
			Base.Set.hash_fold_direct : 'a Base.Hash.folder -> ('a, 'cmp) set Base.Hash.folder
			Base.Set.hash_fold_m__t : (module Base.Set.Hash_fold_m with type t = 'elt) -> Base.Hash.state -> ('elt, 'a) set -> Base.Hash.state
			Base.Set.hash_m__t : (module Base.Set.Hash_fold_m with type t = 'elt) -> ('elt, 'a) set -> int
Set.Make.inter	CCSet.Make.inter	BatSet.intersect : 'a set -> 'a set -> 'a set	Base.Set.inter : ('a, 'cmp) set -> ('a, 'cmp) set -> ('a, 'cmp) set
			Base.Set.invariants : ('a, 'b) set -> bool
Set.Make.is_empty	CCSet.Make.is_empty	BatSet.is_empty : 'a set -> bool	Base.Set.is_empty : ('a, 'b) set -> bool
			Base.Set.is_subset : ('a, 'cmp) set -> of_:('a, 'cmp) set -> bool
Set.Make.iter	CCSet.Make.iter	BatSet.iter : ('a -> unit) -> 'a set -> unit	Base.Set.iter : ('a, 'b) set -> f:('a -> unit) -> unit
			Base.Set.iter2 : ('a, 'cmp) set -> ('a, 'cmp) set -> f:([`Both of 'a * 'a | `Left of 'a | `Right of 'a] -> unit) -> unit
			Base.Set.length : ('a, 'b) set -> int
			Base.Set.m__t_of_sexp : (module Base.Set.M_of_sexp with type comparator_witness = 'cmp and type t = 'elt) -> Base.Sexp.t -> ('elt, 'cmp) set
Set.Make.map	CCSet.Make.map	BatSet.map : ('a -> 'b) -> 'a set -> 'b set	Base.Set.map : ('b, 'cmp) Base.Set.comparator -> ('a, 'c) set -> f:('a -> 'b) -> ('b, 'cmp) set
		BatSet.map_endo : ('a -> 'a) -> 'a set -> 'a set	
Set.Make.max_elt	CCSet.Make.max_elt	BatSet.max_elt : 'a set -> 'a	Base.Set.max_elt_exn : ('a, 'b) set -> 'a
Set.Make.max_elt_opt	CCSet.Make.max_elt_opt	BatSet.max_elt_opt : 'a set -> 'a option	Base.Set.max_elt : ('a, 'b) set -> 'a option
Set.Make.mem	CCSet.Make.mem	BatSet.mem : 'a -> 'a set -> bool	Base.Set.mem : ('a, 'b) set -> 'a -> bool
			Base.Set.merge_to_sequence : ?order:[`Decreasing | `Increasing] -> ?greater_or_equal_to:'a -> ?less_or_equal_to:'a -> ('a, 'cmp) set -> ('a, 'cmp) set -> ('a, 'a) Base.Set.Merge_to_sequence_element.t Base.Sequence.t
Set.Make.min_elt	CCSet.Make.min_elt	BatSet.min_elt : 'a set -> 'a	Base.Set.min_elt_exn : ('a, 'b) set -> 'a
Set.Make.min_elt_opt	CCSet.Make.min_elt_opt	BatSet.min_elt_opt : 'a set -> 'a option	Base.Set.min_elt : ('a, 'b) set -> 'a option
			Base.Set.nth : ('a, 'b) set -> int -> 'a option
		BatSet.of_array : 'a array -> 'a set	Base.Set.of_array : ('a, 'cmp) Base.Set.comparator -> 'a array -> ('a, 'cmp) set
		BatSet.of_enum : 'a BatEnum.t -> 'a set	
			Base.Set.of_increasing_iterator_unchecked : ('a, 'cmp) Base.Set.comparator -> len:int -> f:(int -> 'a) -> ('a, 'cmp) set
	CCSet.Make.of_iter		
Set.Make.of_list	CCSet.Make.of_list	BatSet.of_list : 'a list -> 'a set	Base.Set.of_list : ('a, 'cmp) Base.Set.comparator -> 'a list -> ('a, 'cmp) set
Set.Make.of_seq	CCSet.Make.of_seq	BatSet.of_seq : 'a Seq.t -> 'a set	
			Base.Set.of_sorted_array : ('a, 'cmp) Base.Set.comparator -> 'a array -> ('a, 'cmp) set Base.Or_error.t
			Base.Set.of_sorted_array_unchecked : ('a, 'cmp) Base.Set.comparator -> 'a array -> ('a, 'cmp) set
Set.Make.partition	CCSet.Make.partition	BatSet.partition : ('a -> bool) -> 'a set -> 'a set * 'a set	Base.Set.partition_tf : ('a, 'cmp) set -> f:('a -> bool) -> ('a, 'cmp) set * ('a, 'cmp) set
		BatSet.pop : 'a set -> 'a * 'a set	
		BatSet.pop_max : 'a set -> 'a * 'a set	
		BatSet.pop_min : 'a set -> 'a * 'a set	
	CCSet.Make.pp		
		BatSet.print : ?first:string -> ?last:string -> ?sep:string -> ('a BatInnerIO.output -> 'c -> unit) -> 'a BatInnerIO.output -> 'c set -> unit	
Set.Make.remove	CCSet.Make.remove	BatSet.remove : 'a -> 'a set -> 'a set	Base.Set.remove : ('a, 'cmp) set -> 'a -> ('a, 'cmp) set
		BatSet.remove_exn : 'a -> 'a set -> 'a set	

Stdlib	Containers	Batteries	Base
			Base.Set.remove_index : ('a, 'cmp) set -> int -> ('a, 'cmp) set
			Base.Set.sexp_of_m__t : (module Base.Set.Sexp_of_m with type t = 'elt) -> ('elt, 'cmp) set -> Base.Sexp.t
Set.Make.singleton	CCSet.Make.singleton	BatSet.singleton : 'a -> 'a set	Base.Set.singleton : ('a, 'cmp) Base.Set.comparator -> 'a -> ('a, 'cmp) set
Set.Make.split	CCSet.Make.split	BatSet.split : 'a -> 'a set -> 'a set * bool * 'a set	
		BatSet.split_le : 'a -> 'a set -> 'a set * 'a set	
		BatSet.split_lt : 'a -> 'a set -> 'a set * 'a 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
			Base.Set.stable_dedup_list : ('a, 'b) Base.Set.comparator -> 'a list -> 'a list
Set.Make.subset	CCSet.Make.subset	BatSet.subset : 'a set -> 'a set -> bool	
			Base.Set.sum : (module Base.Container.Summable with type t = 'sum) -> ('a, 'b) set -> f:('a -> 'sum) -> 'sum
		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
	CCSet.Make.to_iter		
	CCSet.Make.to_list	BatSet.to_list : 'a set -> 'a list	Base.Set.to_list : ('a, 'b) set -> 'a list
Set.Make.to_rev_seq	CCSet.Make.to_rev_seq	BatSet.to_rev_seq : 'a set -> 'a Seq.t	
Set.Make.to_seq	CCSet.Make.to_seq	BatSet.to_seq : 'a set -> 'a Seq.t	
Set.Make.to_seq_from	CCSet.Make.to_seq_from	BatSet.to_seq_from : 'a -> 'a set -> 'a Seq.t	
			Base.Set.to_sequence : ?order:[`Decreasing | `Increasing] -> ?greater_or_equal_to:'a -> ?less_or_equal_to:'a -> ('a, 'cmp) set -> 'a Base.Sequence.t
	CCSet.Make.to_string		
Set.Make.union	CCSet.Make.union	BatSet.union : 'a set -> 'a set -> 'a set	Base.Set.union : ('a, 'cmp) set -> ('a, 'cmp) set -> ('a, 'cmp) set
			Base.Set.union_list : ('a, 'cmp) Base.Set.comparator -> ('a, 'cmp) set list -> ('a, 'cmp) set
		BatSet.update : 'a -> 'a -> 'a set -> 'a set	