```
Last login: Wed Jan 24 15:43:51 on ttys004
carbon:$ utop
────────┬─────────────────────────────────────────────────────────────┬────────
        │ Welcome to utop version 2.0.2 (using OCaml version 4.06.0)! │
        └─────────────────────────────────────────────────────────────┘

Type #utop_help for help about using utop.

─( 15:49:47 )─< command 0 >──────────────────────────────────{ counter: 0 }─
utop # #use "simple.ml";;
val inc_v1 : int -> int = <fun>
val inc_v2 : int -> int = <fun>
val square : int -> int = <fun>
val cube : int -> int = <fun>
val add : int -> int -> int = <fun>
val inc_v3 : int -> int = <fun>
val add3 : int -> int -> int -> int = <fun>
val greater : 'a -> 'a -> 'a = <fun>
val circle_area : float -> float = <fun>
val power : int -> float -> float = <fun>
val power_v2 : int -> float -> float = <fun>
val cube : float -> float = <fun>
val foo : float = 13.824
val bar : float = 13.824
val gcd : int -> int -> int = <fun>
val sum : int list -> int = <fun>
val all : bool list -> bool = <fun>
val even2ways : int list -> bool = <fun>
val even : int -> bool = <fun>
val string_concat : string -> string list -> string = <fun>
─( 15:49:47 )─< command 1 >──────────────────────────────────{ counter: 0 }─
utop # string_concat ","  ["a"; "b"; "c"] ;;
- : string = "a,b,c"
─( 15:49:49 )─< command 2 >──────────────────────────────────{ counter: 0 }─
utop # string_concat ","  ["a"] ;;
- : string = "a"
─( 15:50:15 )─< command 3 >──────────────────────────────────{ counter: 0 }─
utop # string_concat ","  [] ;;
- : string = ""
─( 15:50:19 )─< command 4 >──────────────────────────────────{ counter: 0 }─
utop # let x1::rest = 1::2::3::[] ;;
Characters 4-12:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a case that is not matched:
[]
val x1 : int = 1
val rest : int list = [2; 3]
─( 15:50:22 )─< command 5 >──────────────────────────────────{ counter: 0 }─
utop # x1  ;;
- : int = 1
```

```
─( 15:59:27 )─< command 6 >──────────────────────────────────{ counter: 0 }─
utop # let m = (1::2::3::[]) :: (4::5::6::[]) :: [] ;;
val m : int list list = [[1; 2; 3]; [4; 5; 6]]
─( 15:59:56 )─< command 7 >──────────────────────────────────{ counter: 0 }─
utop # let (x1::x2)::rest = m ;;
Characters 4-18:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a case that is not matched:
([]::_|[])
val x1 : int = 1
val x2 : int list = [2; 3]
val rest : int list list = [[4; 5; 6]]
─( 16:00:31 )─< command 8 >──────────────────────────────────{ counter: 0 }─
utop # #use "simple.ml";;
val inc_v1 : int -> int = <fun>
val inc_v2 : int -> int = <fun>
val square : int -> int = <fun>
val cube : int -> int = <fun>
val add : int -> int -> int = <fun>
val inc_v3 : int -> int = <fun>
val add3 : int -> int -> int -> int = <fun>
val greater : 'a -> 'a -> 'a = <fun>
val circle_area : float -> float = <fun>
val power : int -> float -> float = <fun>
val power_v2 : int -> float -> float = <fun>
val cube : float -> float = <fun>
val foo : float = 13.824
val bar : float = 13.824
val gcd : int -> int -> int = <fun>
val all : bool list -> bool = <fun>
val even2ways : int list -> bool = <fun>
val even : int -> bool = <fun>
val sum : int list -> int = <fun>
val string_concat : string -> string list -> string = <fun>
─( 16:01:47 )─< command 9 >──────────────────────────────────{ counter: 0 }─
utop # not ;;
- : bool -> bool = <fun>
─( 16:08:03 )─< command 10 >─────────────────────────────────{ counter: 0 }─
utop # #use "simple.ml";;
val inc_v1 : int -> int = <fun>
val inc_v2 : int -> int = <fun>
val square : int -> int = <fun>
val cube : int -> int = <fun>
val add : int -> int -> int = <fun>
val inc_v3 : int -> int = <fun>
val add3 : int -> int -> int -> int = <fun>
val greater : 'a -> 'a -> 'a = <fun>
val circle_area : float -> float = <fun>
val power : int -> float -> float = <fun>
val power_v2 : int -> float -> float = <fun>
val cube : float -> float = <fun>
val foo : float = 13.824
val bar : float = 13.824
val gcd : int -> int -> int = <fun>
val all : bool list -> bool = <fun>
val even2ways : int list -> bool = <fun>
val even : int -> bool = <fun>
val sum : int list -> int = <fun>
val string_concat : string -> string list -> string = <fun>
val is_empty : 'a list -> bool = <fun>
val is_empty' : 'a list -> bool = <fun>
val not_empty : 'a list -> bool = <fun>
val not_empty'' : 'a list -> bool = <fun>
─( 16:11:44 )─< command 11 >─────────────────────────────────{ counter: 0 }─
utop # sum ;;
- : int list -> int = <fun>
─( 16:12:35 )─< command 12 >─────────────────────────────────{ counter: 0 }─
utop # #use "simple.ml";;
val inc_v1 : int -> int = <fun>
val inc_v2 : int -> int = <fun>
val square : int -> int = <fun>
val cube : int -> int = <fun>
val add : int -> int -> int = <fun>
val inc_v3 : int -> int = <fun>
val add3 : int -> int -> int -> int = <fun>
val greater : 'a -> 'a -> 'a = <fun>
val circle_area : float -> float = <fun>
val power : int -> float -> float = <fun>
val power_v2 : int -> float -> float = <fun>
val cube : float -> float = <fun>
val foo : float = 13.824
val bar : float = 13.824
val gcd : int -> int -> int = <fun>
val all : bool list -> bool = <fun>
val even2ways : int list -> bool = <fun>
val even : int -> bool = <fun>
val sum : int list -> int = <fun>
val string_concat : string -> string list -> string = <fun>
val is_empty : 'a list -> bool = <fun>
val is_empty' : 'a list -> bool = <fun>
val not_empty : 'a list -> bool = <fun>
val not_empty'' : 'a list -> bool = <fun>
val sum : int list -> int = <fun>
val length : 'a list -> int = <fun>
─( 16:14:23 )─< command 13 >─────────────────────────────────{ counter: 0 }─
utop # #use "simple.ml";;
val inc_v1 : int -> int = <fun>
val inc_v2 : int -> int = <fun>
val square : int -> int = <fun>
val cube : int -> int = <fun>
val add : int -> int -> int = <fun>
val inc_v3 : int -> int = <fun>
val add3 : int -> int -> int -> int = <fun>
val greater : 'a -> 'a -> 'a = <fun>
val circle_area : float -> float = <fun>
val power : int -> float -> float = <fun>
val power_v2 : int -> float -> float = <fun>
val cube : float -> float = <fun>
val foo : float = 13.824
val bar : float = 13.824
val gcd : int -> int -> int = <fun>
val all : bool list -> bool = <fun>
val even2ways : int list -> bool = <fun>
val even : int -> bool = <fun>
val sum : int list -> int = <fun>
val string_concat : string -> string list -> string = <fun>
val is_empty : 'a list -> bool = <fun>
val is_empty' : 'a list -> bool = <fun>
val not_empty : 'a list -> bool = <fun>
val not_empty'' : 'a list -> bool = <fun>
val sum : int list -> int = <fun>
val length : 'a list -> int = <fun>
─( 16:15:09 )─< command 14 >─────────────────────────────────{ counter: 0 }─
utop # String.length ;;
- : string -> int = <fun>
─( 16:18:18 )─< command 15 >─────────────────────────────────{ counter: 0 }─
utop # m ;;
- : int list list = [[1; 2; 3]; [4; 5; 6]]
─( 16:18:49 )─< command 16 >─────────────────────────────────{ counter: 0 }─
utop # length ;;
- : 'a list -> int = <fun>
─( 16:21:02 )─< command 17 >─────────────────────────────────{ counter: 0 }─
utop # length m ;;
- : int = 2
─( 16:21:07 )─< command 18 >─────────────────────────────────{ counter: 0 }─
utop # [] ;;
- : 'a list = []
─( 16:21:56 )─< command 19 >─────────────────────────────────{ counter: 0 }─
utop # [[]] ;;
- : 'a list list = [[]]
─( 16:28:57 )─< command 20 >─────────────────────────────────{ counter: 0 }─
utop # List.fold_left ;;
- : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a = <fun>
─( 16:29:12 )─< command 21 >─────────────────────────────────{ counter: 0 }─
utop # List.fold ;;
Error: Unbound value List.fold
─( 16:29:51 )─< command 22 >─────────────────────────────────{ counter: 0 }─
utop #
┌───┬─────┬───────────┬──────────────┬────────┬──────┬─────┬──────────┬───┐
│Arg│Array│ArrayLabels│Assert_failure│Bigarray│Buffer│Bytes│BytesLabels│Cal│
```