```
Last login: Mon Jan 29 15:42:14 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.

─( 15:42:40 )─< 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 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>
val head : 'a list -> 'a = <fun>
val sum_v2 : int list -> int = <fun>
val sum : int list -> int = <fun>
val first : 'a * 'b * 'c -> 'a = <fun>
val first' : 'a * 'b * 'c -> 'a = <fun>
val first'' : 'a * 'b * 'c -> 'a = <fun>
val m : (string * int) list =
  [("dog", 1); ("chicken", 2); ("dog", 3); ("cat", 5)]
File "simple.ml", line 151, characters 40-44:
Error: This expression has type ('a * 'b) list
       but an expression was expected of type 'a
       The type variable 'a occurs inside ('a * 'b) list
```

```
─( 15:42:40 )─< command 1 >──────────────────────────────{ 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>
val head : 'a list -> 'a = <fun>
val sum_v2 : int list -> int = <fun>
val sum : int list -> int = <fun>
val first : 'a * 'b * 'c -> 'a = <fun>
val first' : 'a * 'b * 'c -> 'a = <fun>
val first'' : 'a * 'b * 'c -> 'a = <fun>
val m : (string * int) list =
  [("dog", 1); ("chicken", 2); ("dog", 3); ("cat", 5)]
val lookup_all : 'a -> ('a * 'b) list -> 'b list = <fun>
─( 15:42:42 )─< command 2 >──────────────────────────────{ counter: 0 }─
utop # lookup_all "dog" m ;;
- : int list = [1; 3]
─( 15:43:10 )─< command 3 >──────────────────────────────{ 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>
val head : 'a list -> 'a = <fun>
val sum_v2 : int list -> int = <fun>
val sum : int list -> int = <fun>
val first : 'a * 'b * 'c -> 'a = <fun>
val first' : 'a * 'b * 'c -> 'a = <fun>
val first'' : 'a * 'b * 'c -> 'a = <fun>
val m : (string * int) list =
  [("dog", 1); ("chicken", 2); ("dog", 3); ("cat", 5)]
File "simple.ml", line 151, characters 26-28:
Error: This variant expression is expected to have type unit
       The constructor :: does not belong to type unit
─( 15:43:16 )─< command 4 >─────────────────────────────{ 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>
val head : 'a list -> 'a = <fun>
val sum_v2 : int list -> int = <fun>
val sum : int list -> int = <fun>
val first : 'a * 'b * 'c -> 'a = <fun>
val first' : 'a * 'b * 'c -> 'a = <fun>
val first'' : 'a * 'b * 'c -> 'a = <fun>
val m : (string * int) list =
  [("dog", 1); ("chicken", 2); ("dog", 3); ("cat", 5)]
val lookup_all : 'a -> ('a * 'b) list -> 'b list = <fun>
val lookup_all' : 'a -> ('a * 'b) list -> 'b list = <fun>
─( 15:45:27 )─< command 5 >─────────────────────────────{ counter: 0 }─
utop # lookup_all' "dog" m ;;
- : int list = [1; 3]
─( 15:45:54 )─< command 6 >─────────────────────────────{ counter: 0 }─
utop # 1 + 2 * 3 ;;
- : int = 7
─( 15:45:58 )─< command 7 >─────────────────────────────{ counter: 0 }─
utop #    1 ;;
- : int = 1
─( 16:16:24 )─< command 8 >─────────────────────────────{ counter: 0 }─
utop # inc ;;
Error: Unbound value inc
Hint: Did you mean incr?
─( 16:18:59 )─< command 9 >─────────────────────────────{ counter: 0 }─
utop # let inc x = x + 1 ;;
val inc : int -> int = <fun>
─( 16:19:52 )─< command 10 >────────────────────────────{ counter: 0 }─
utop # ( inc , 3 ) ;;
- : (int -> int) * int = (<fun>, 3)
─( 16:19:56 )─< command 11 >────────────────────────────{ counter: 0 }─
utop # (1 + 2) * 3 ;;
- : int = 9
─( 16:20:01 )─< command 12 >────────────────────────────{ counter: 0 }─
utop # let x = (1,2,3) ;;
val x : int * int * int = (1, 2, 3)
─( 16:21:16 )─< command 13 >────────────────────────────{ counter: 0 }─
utop # let x' = ((1,2), 3) ;;
```

```
val x' : (int * int) * int = ((1, 2), 3)
─( 16:23:21 )─< command 14 >──────────────────────────────{ counter: 0 }─
utop # ( inc , 3 ) ;;
- : (int -> int) * int = (<fun>, 3)
─( 16:23:29 )─< command 15 >──────────────────────────────{ counter: 0 }─
utop # ( fun x -> x + 1, 'c' ) ;;
- : int -> int * char = <fun>
─( 16:33:01 )─< command 16 >──────────────────────────────{ counter: 0 }─
utop # fun x -> (x + 1, 'c') ;;
- : int -> int * char = <fun>
─( 16:33:14 )─< command 17 >──────────────────────────────{ counter: 0 }─
utop # ( (fun x -> x + 1), 'c' ) ;;
- : (int -> int) * char = (<fun>, 'c')
─( 16:33:31 )─< command 18 >──────────────────────────────{ counter: 0 }─
utop # ( fun x ->( x + 1, 'c') ) ;;
- : int -> int * char = <fun>
─( 16:34:09 )─< command 19 >──────────────────────────────{ counter: 0 }─
utop # x ;;
- : int * int * int = (1, 2, 3)
─( 16:35:37 )─< command 20 >──────────────────────────────{ counter: 0 }─
utop # match x with | a,b,c -> c ;;
- : int = 3
─( 16:36:23 )─< command 21 >──────────────────────────────{ counter: 0 }─
utop # ( fun x -> x + 1, 'c' k ;;
Error: Syntax error: ')' expected, the highlighted '(' might be unmatch
ed
─( 16:36:35 )─< command 22 >──────────────────────────────{ counter: 0 }─
utop # 1 + "D" ;;
Error: This expression has type string
       but an expression was expected of type int
─( 16:41:19 )─< command 23 >──────────────────────────────{ counter: 0 }─
utop #

│Arg│Array│ArrayLabels│Assert_failure│Bigarray│Buffer│Bytes│BytesLabel│
```