```
Last login: Mon Feb  5 13:23:46 on ttys003
carbon:$ utop
───────┬──────────────────────────────────────────────────────────────┬───────
       │ Welcome to utop version 2.0.2 (using OCaml version 4.06.0)! │
       └──────────────────────────────────────────────────────────────┘

Type #utop_help for help about using utop.

─( 13:23:56 )─< command 0 >────────────────────────────────────{ counter: 0 }─
utop # List.map ;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
─( 13:23:56 )─< command 1 >────────────────────────────────────{ counter: 0 }─
utop # let inc x = x + 1 ;;
val inc : int -> int = <fun>
─( 13:32:13 )─< command 2 >────────────────────────────────────{ counter: 0 }─
utop # List.map inc [1;2;3;4;5] ;;
- : int list = [2; 3; 4; 5; 6]
─( 13:32:25 )─< command 3 >────────────────────────────────────{ counter: 0 }─
utop # List.map Char.code ['e'; 'E'; 'z'; '8'] ;;
- : int list = [101; 69; 122; 56]
─( 13:33:00 )─< command 4 >────────────────────────────────────{ counter: 0 }─
utop # #use "higher_order.ml";;
val implode : char list -> string = <fun>
val explode : string -> char list = <fun>
val map : ('a -> 'b) -> 'a list -> 'b list = <fun>
─( 13:33:00 )─< command 5 >────────────────────────────────────{ counter: 0 }─
utop # map Char.code ['e'; 'E'; 'z'; '8'] ;;
- : int list = [101; 69; 122; 56]
─( 13:35:13 )─< command 6 >────────────────────────────────────{ counter: 0 }─
utop # List.filter ;;
- : ('a -> bool) -> 'a list -> 'a list = <fun>
─( 13:35:26 )─< command 7 >────────────────────────────────────{ counter: 0 }─
utop # List.filter (fun s -> String.length = 6) ["hello"; "planet"; "lane"] ;;
Error: This expression has type int but an expression was expected of type
         string -> int
─( 13:43:35 )─< command 8 >────────────────────────────────────{ counter: 0 }─
utop # List.filter (fun s -> String.length s = 6) ["hello"; "planet"; "lane"]
;;
- : string list = ["planet"]
─( 13:44:30 )─< command 9 >────────────────────────────────────{ counter: 0 }─
utop # 'A' > 'D' ;;
- : bool = false
─( 13:44:43 )─< command 10 >───────────────────────────────────{ counter: 0 }─
utop # 'A' > 99 ;;
Error: This expression has type int but an expression was expected of type
         char
─( 13:52:06 )─< command 11 >───────────────────────────────────{ counter: 0 }─
utop # #use "higher_order.ml";;
val implode : char list -> string = <fun>
val explode : string -> char list = <fun>
val map : ('a -> 'b) -> 'a list -> 'b list = <fun>
val removeABCD : char list -> char list = <fun>
val foldl : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a = <fun>
─( 13:52:12 )─< command 12 >───────────────────────────────────{ counter: 0 }─
utop # foldl (+) 100 [1;2;3] ;;
- : int = 106
─( 14:00:25 )─< command 13 >───────────────────────────────────{ counter: 0 }─
utop # List.fold_right ;;
- : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b = <fun>
─( 14:00:55 )─< command 14 >───────────────────────────────────{ counter: 0 }─
utop # List.fold_right (fun x xs -> x :: xs)  [1;2;3;4;5] [] ;;
- : int list = [1; 2; 3; 4; 5]
─( 14:05:13 )─< command 15 >───────────────────────────────────{ counter: 0 }─
utop # let l = 1 :: (2 :: ()3 :: [])  ;;
Error: The constructor () expects 0 argument(s),
       but is applied here to 1 argument(s)
─( 14:06:08 )─< command 16 >───────────────────────────────────{ counter: 0 }─
utop # let l = 1 :: (2 :: (3 :: []))  ;;
val l : int list = [1; 2; 3]
─( 14:07:12 )─< command 17 >───────────────────────────────────{ counter: 0 }─
utop # let s = 1 +  (2 +  (3 +  0)) ;;
val s : int = 6
─( 14:07:21 )─< command 18 >───────────────────────────────────{ counter: 0 }─
utop # List.fold_right (+) l 0 ;;
- : int = 6
─( 14:07:35 )─< command 19 >───────────────────────────────────{ counter: 0 }─
utop # let s_from_the_left = ((0 + 1) + 2) + 3 ;;
val s_from_the_left : int = 6
─( 14:08:23 )─< command 20 >───────────────────────────────────{ counter: 0 }─
utop #
┌───┬─────┬───────────┬──────────────┬─────────┬──────┬─────┬──────────┬─────┐
│Arg│Array│ArrayLabels│Assert_failure│Bigarray│Buffer│Bytes│BytesLabels│Callb│
└───┴─────┴───────────┴──────────────┴─────────┴──────┴─────┴──────────┴─────┘
```