```
Last login: Mon Feb 26 15:58:58 on ttys007
carbon:$ utop
────────────────────────────────────────────────────────────────────────────────
        │ Welcome to utop version 2.0.2 (using OCaml version 4.06.0)! │

Type #utop_help for help about using utop.

─( 15:59:11 )─< command 0 >──────────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr = Int of int | Add of expr * expr | Mul of expr * expr
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val eval : expr -> int = <fun>
─( 15:59:11 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # eval e1 ;;
- : int = 7
─( 15:59:13 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # #use "expr_let.ml";;
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Let of string * expr * expr
  | Id of string
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val lookup : string -> (string * int) list -> int = <fun>
val eval : expr -> (string * int) list -> int = <fun>
val e2 : expr = Id "x"
val e3 : expr = Let ("x", Add (Int 2, Int 3), Mul (Id "x", Int 2))
─( 15:59:17 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # eval e3 [] ;;
- : int = 10
─( 16:15:34 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # eval e2 [] ;;
Exception: Failure "unbound name \"x\"".
─( 16:15:57 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # eval e2 ( ("x", 4)::[] ) ;;
- : int = 4
─( 16:16:07 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
utop # eval e1 [] ;;
- : int = 7
─( 16:16:40 )─< command 7 >──────────────────────────────────────{ counter: 0 }─
utop # eval e3 ( ("x", 4000)::[] ) ;;
- : int = 10
─( 16:17:01 )─< command 8 >──────────────────────────────────────{ counter: 0 }─
utop # #use "expr_let.ml";;
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Let of string * expr * expr
  | Id of string
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val lookup : string -> (string * int) list -> int = <fun>
val eval : expr -> (string * int) list -> int = <fun>
val e2 : expr = Id "x"
val e3 : expr = Let ("x", Add (Int 2, Int 3), Mul (Id "x", Int 2))
File "expr_let.ml", line 31, characters 30-33:
Error: This expression has type int but an expression was expected of type expr
─( 16:17:34 )─< command 9 >─────────────────────────────────────{ counter: 0 }─
utop # #use "expr_let.ml";;
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Let of string * expr * expr
  | Id of string
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val lookup : string -> (string * int) list -> int = <fun>
val eval : expr -> (string * int) list -> int = <fun>
val e2 : expr = Id "x"
val e3 : expr = Let ("x", Add (Int 2, Int 3), Mul (Id "x", Int 2))
val e4 : expr =
  Let ("x", Int 5,
   Add (Let ("x", Int 100, Add (Id "x", Int 1)), Mul (Id "x", Int 7)))
─( 16:24:11 )─< command 10 >────────────────────────────────────{ counter: 0 }─
utop # eval e4 [] ;;
- : int = 136
─( 16:24:22 )─< command 11 >────────────────────────────────────{ counter: 0 }─
utop #
┌───┬───┬─────┬───────────┬──────────────┬────────┬──────┬─────┬───────────┬───┐
│Add│Arg│Array│ArrayLabels│Assert_failure│Bigarray│Buffer│Bytes│BytesLabels│Cal│
└───┴───┴─────┴───────────┴──────────────┴────────┴──────┴─────┴───────────┴───┘
```