```
Last login: Mon Feb 26 13:23:13 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.

─( 13:51:51 )─< command 0 >──────────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr =
    Plus of expr * expr
  | Mult of expr * expr
  | Neg of expr
  | Int of int
val eval : expr -> int = <fun>
val e1 : expr = Plus (Int 1, Mult (Int 2, Int 3))
─( 13:51:52 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # e1 ;;
- : expr = Plus (Int 1, Mult (Int 2, Int 3))
─( 13:51:55 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # eval e1 ;;
- : int = 7
─( 13:51:58 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr =
    Plus of expr * expr
  | Mult of expr * expr
  | Sub of expr * expr
  | Div of expr * expr
  | Neg of expr
  | Int of int
val eval : expr -> int = <fun>
val e1 : expr = Plus (Int 1, Mult (Int 2, Int 3))
val e2 : expr = Div (Mult (Int 2, Sub (Int 5, Int 3)), Int 2)
─( 13:52:00 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # eval e2 ;;
- : int = 2
─( 13:54:59 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr =
    Plus of expr * expr
  | Mult of expr * expr
  | Sub of expr * expr
  | Div of expr * expr
  | Neg of expr
  | Int of int
val eval : expr -> int = <fun>
val e1 : expr = Plus (Int 1, Mult (Int 2, Int 3))
val e2 : expr = Div (Mult (Int 2, Sub (Int 5, Int 3)), Int 2)
─( 13:55:05 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
```

```
utop # #use "expr_let.ml";;
type expr =
    Plus of expr * expr
  | Mult of expr * expr
  | Sub of expr * expr
  | Div of expr * expr
  | Neg of expr
  | Int of int
  | Let of string * expr * expr
  | Id of string
val lookup : string -> (string * int) list -> int = <fun>
val eval : expr -> (string * int) list -> int = <fun>
val e3 : expr =
  Plus (Int 3, Let ("x", Plus (Int 4, Int 5), Mult (Id "x", Int 5)))
─( 14:10:10 )─< command 7 >──────────────────────────────────────{ counter: 0 }─
utop # eval e3 ;;
- : (string * int) list -> int = <fun>
─( 14:10:19 )─< command 8 >──────────────────────────────────────{ counter: 0 }─
utop # eval e3 [] ;;
- : int = 48
─( 14:10:26 )─< command 9 >──────────────────────────────────────{ counter: 0 }─
utop #
```

Arg	Array	ArrayLabels	Assert_failure	Bigarray	Buffer	Bytes	BytesLabels	Callbac