```
Last login: Wed Feb 22 15:36:41 on ttys020
carbon:SamplePrograms$ cd Sec_10_3\:35pm/
carbon:Sec_10_3:35pm$ utop
                Welcome to utop version 1.14 (using OCaml version 4.01.0)!

Type #utop_help for help about using utop.

─( 18:00:00 )─< 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>
─( 15:55:13 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # eval e1 ;;
- : int = 7
─( 15:55:19 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val eval : expr -> int = <fun>
─( 15:55:22 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
val eval : expr -> int = <fun>
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val e2 : expr = Sub (Int 10, Div (Add (Int 1, Mul (Int 2, Int 3)), Int 2))
─( 15:59:06 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # eval e2 ;;
- : int = 7
─( 15:59:12 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
val eval : expr -> int = <fun>
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val e2 : expr = Sub (Int 10, Div (Add (Int 1, Mul (Int 2, Int 3)), Int 3))
─( 15:59:15 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
```

```
utop # eval e2 ;;
- : int = 8
─( 15:59:28 )─< command 7 >──────────────────────────────────{ counter: 0 }─
utop # eval (Div (Int 10, Int 0)) ;;
Exception: Division_by_zero.
─( 15:59:29 )─< command 8 >──────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul
  | Div of expr * expr
File "arithmetic.ml", line 13, characters 4-16:
Error: The constructor Mul expects 0 argument(s),
       but is applied here to 1 argument(s)
─( 16:00:13 )─< command 9 >──────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type exprpair = expr * expr
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of exprpair
File "arithmetic.ml", line 15, characters 25-27:
Error: This expression has type expr/1102
       but an expression was expected of type expr/1117
─( 16:01:03 )─< command 10 >─────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:Sec_10_3:35pm$ utop
──────────────────────────────────────────────────────────────────────────────
        Welcome to utop version 1.14 (using OCaml version 4.01.0)!

Type #utop_help for help about using utop.

─( 18:00:00 )─< command 0 >──────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
File "arithmetic.ml", line 1, characters 16-20:
Error: Unbound type constructor expr
─( 16:03:28 )─< command 1 >──────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type exprpair = expr * expr
and expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of exprpair
val eval : expr -> int = <fun>
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val e2 : expr = Sub (Int 10, Div (Add (Int 1, Mul (Int 2, Int 3)), Int 3))
```

```
─( 16:03:31 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
val eval : expr -> int = <fun>
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val e2 : expr = Sub (Int 10, Div (Add (Int 1, Mul (Int 2, Int 3)), Int 3))
─( 16:03:50 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # let x = x + 1 in x  5 ;;
Error: Unbound value x
─( 16:04:53 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # #use "arithmetic.ml";;
type expr =
    Int of int
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
val eval : expr -> int = <fun>
val e1 : expr = Add (Int 1, Mul (Int 2, Int 3))
val e2 : expr = Sub (Int 10, Div (Add (Int 1, Mul (Int 2, Int 3)), Int 3))
─( 16:19:54 )─< command 5 >──────────────────────────────────────{ 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
type environment = (string * int) list
val lookup : string -> environment -> int = <fun>
val eval : environment -> expr -> int = <fun>
val e2 : expr = Let ("x", Int 5, Add (Id "x", Int 4))
─( 16:21:45 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
utop # eval [] e2 ;;
- : int = 9
─( 16:21:56 )─< command 7 >──────────────────────────────────────{ counter: 0 }─
utop # eval [] (Add (Int 4, Mul( Int 3, Id "x"))) ;;
Exception: Failure "Identifier x is not in scope.".
─( 16:22:04 )─< command 8 >──────────────────────────────────────{ counter: 0 }─
utop # eval [] (Let("x", Int 3,
                Add(Mul (Int 2, Id "x"),
                    Let("x", Int 4,
                        Add(Int 5, Id "x")))));;
- : int = 15
─( 16:24:05 )─< command 9 >──────────────────────────────────────{ counter: 0 }─
utop #
```