```
Last login: Wed Feb 22 13:11:21 on ttys018
carbon:public-class-repo$ cd SamplePrograms/Sec_01_1\:25pm/
carbon:Sec_01_1:25pm$ 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 = Const of int | Add of expr * expr | Mul of expr * expr
val eval : expr -> int = <fun>
-( 13:45:00 )-< command 1 >------------------------------------{ counter: 0 }-
utop # let e1 = Add (Const 1, Mul (Const 2, Const 3)) ;;
val e1 : expr = Add (Const 1, Mul (Const 2, Const 3))
-( 13:45:03 )-< command 2 >------------------------------------{ counter: 0 }-
utop # eval e1 ;;
- : int = 7
-( 13:45:42 )-< command 3 >------------------------------------{ counter: 0 }-
utop # #use "arithmetic.ml";;
type expr =
    Const 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 (Const 1, Mul (Const 2, Const 3))
val e2 : expr =
  Sub (Const 10, Div (Add (Const 1, Mul (Const 2, Const 3)), Const 2))
-( 13:45:44 )-< command 4 >------------------------------------{ counter: 0 }-
utop # eval e2 ;;
- : int = 7
-( 13:48:27 )-< command 5 >------------------------------------{ counter: 0 }-
utop # eval (Div (Const 3, Const 0));;
Exception: Division_by_zero.
-( 13:48:32 )-< command 6 >------------------------------------{ counter: 0 }-
utop # #use "arithmetic.ml";;
type expr =
    Const 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 (Const 1, Mul (Const 2, Const 3))
val e2 : expr =
  Sub (Const 10, Div (Add (Const 1, Mul (Const 2, Const 3)), Const 2))
-( 13:49:24 )-< command 7 >------------------------------------{ counter: 0 }-
```

```
utop # #use "expr_let.ml";;
type expr =
    Const 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 e1 : expr = Add (Const 1, Mul (Const 2, Const 3))
val e2 : expr =
  Sub (Const 10, Div (Add (Const 1, Mul (Const 2, Const 3)), Const 2))
val e3 : expr = Let ("x", Const 5, Add (Id "x", Const 4))
-( 14:09:43 )-< command 8 >------------------------------------{ counter: 0 }-
utop # eval e3 ;;
Error: This expression has type expr but an expression was expected of type
         environment = (string * int) list
-( 14:09:52 )-< command 9 >------------------------------------{ counter: 0 }-
utop # eval [] e3 ;;
- : int = 9
-( 14:10:01 )-< command 10 >-----------------------------------{ counter: 0 }-
utop # #use "expr_let.ml";;
type expr =
    Const 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 e1 : expr = Add (Const 1, Mul (Const 2, Const 3))
val e2 : expr =
  Sub (Const 10, Div (Add (Const 1, Mul (Const 2, Const 3)), Const 2))
val e3 : expr = Let ("x", Const 5, Add (Id "x", Const 4))
val e4 : expr =
  Let ("y", Const 5, Let ("x", Add (Id "y", Const 5), Add (Id "x", Id "y")))
-( 14:10:23 )-< command 11 >-----------------------------------{ counter: 0 }-
utop # eval [] e4 ;;
- : int = 15
-( 14:11:16 )-< command 12 >-----------------------------------{ counter: 0 }-
utop # eval [] (Let("x", Int 3,
                 Add(Mul (Int 2, Id "x"),
                     Let("x", Int 4,
                         Add(Int 5, Id "x"))))) ;;
```

```
Error: The variant type expr has no constructor Int
-( 14:11:20 )-< command 13 >-------------------------------------{ counter: 0 }-
utop # eval [] (Let("x", Const 3,
                 Add(Mul (Const 2, Id "x"),
                     Let("x", Const 4,
                         Add(Const 5, Id "x"))))) ;;
- : int = 15
-( 14:15:31 )-< command 14 >-------------------------------------{ counter: 0 }-
utop # #use "expr_let.ml";;
```

;;	