```
Last login: Mon Apr  3 15:34:45 on ttys006
carbon:Sec_01_1:25pm$ 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 "interpreter.ml";;
type value = Int of int | Bool of bool
 type expr =
      Add of expr * expr

  | Mul of expr * expr
  | Sub of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Var of string
  | Value of value
type environment = (string * value) list
val lookup : string -> (string * 'a) list -> 'a = <fun>
val eval : expr -> environment -> value = <fun>
type state = environment
type stmt =
    Assign of string * expr
  | While of expr * stmt
  | IfThen of expr * stmt
  | Seq of stmt * stmt
  | ReadNum of string
  | WriteNum of expr
val program_1 : stmt =
  Seq (Assign ("x", Value (Int 2)),
   Seq (Assign ("y", Add (Var "x", Value (Int 3))),
    Seq (Assign ("z", Add (Var "y", Value (Int 2))), WriteNum (Var "z"))))
val program_2 : stmt =
  Seq (ReadNum "x",
   Seq (Assign ("i", Value (Int 0)),
    Seq (Assign ("sum", Value (Int 0)),
     Seq
      (While (Lt (Var "i", Var "x"),
        Seq (WriteNum (Var "i"),
         Seq (Assign ("sum", Add (Var "sum", Var "i")),
          Assign ("i", Add (Var "i", Value (Int 1)))))),
```

```
      WriteNum (Var "sum")))))
val read_number : unit -> int = <fun>
val write_number : int -> unit = <fun>
File "interpreter.ml", line 128, characters 30-44:
Error: This expression has type unit but an expression was expected of type
         state = (string * value) list
-( 15:42:27 )-< command 1 >-------------------------------------{ counter: 0 }
-
utop # #quit ;;
carbon:Sec_10_3:35pm$ cd ../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 "interpreter.ml";;
type value = Int of int | Bool of bool
 type expr =
      Add of expr * expr

  | Mul of expr * expr
  | Sub of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Var of string
  | Value of value
type environment = (string * value) list
val lookup : string -> (string * 'a) list -> 'a = <fun>
val eval : expr -> environment -> value = <fun>
type state = environment
type stmt =
    Assign of string * expr
  | While of expr * stmt
  | IfThen of expr * stmt
  | IfThenElse of expr * stmt * stmt
  | Seq of stmt * stmt
  | WriteNum of expr
  | ReadNum of string
val program_1 : stmt =
  Seq (Assign ("x", Value (Int 1)),
   Seq (Assign ("y", Add (Var "x", Value (Int 2))),
    Seq (Assign ("z", Add (Var "y", Value (Int 3))), WriteNum (Var "z"))))
```

```
val program_2 : stmt =
  Seq (ReadNum "x",
   Seq (Assign ("i", Value (Int 0)),
    Seq (Assign ("sum", Value (Int 0)),
     Seq
      (While (Lt (Var "i", Var "x"),
        Seq (WriteNum (Var "i"),
         Seq (Assign ("sum", Add (Var "sum", Var "i")),
          Assign ("i", Add (Var "i", Value (Int 1)))))),
      WriteNum (Var "sum")))))
val read_number : unit -> int = <fun>
val write_number : int -> unit = <fun>
File "interpreter.ml", line 140, characters 5-260:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
Int _
File "interpreter.ml", line 127, characters 2-637:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
(IfThen (_, _)|IfThenElse (_, _, _))
val exec : stmt -> state -> state = <fun>
─( 15:43:16 )─< command 1 >───────────────────────────────────────{ counter: 0 }
─
utop # exec program_2 [] ;;
Enter an integer value:
 5
  0

1
2
3
4
10
- : state =
[("i", Int 5); ("sum", Int 10); ("i", Int 4); ("sum", Int 6); ("i", Int 3);
 ("sum", Int 3); ("i", Int 2); ("sum", Int 1); ("i", Int 1); ("sum", Int 0);
 ("sum", Int 0); ("i", Int 0); ("x", Int 5)]
─( 15:43:18 )─< command 2 >──────────────────────( 15:44:11 )─< command 2 >─────{
 counter: 0 }─
utop # #use "interpreter.ml";;
type value = Int of int | Bool of bool         type expr =
                  Add of expr * expr                           | Mul of expr * e
xpr_int│Bigarr│
  | Sub of expr * expr───┴─────┴───────────┴───  | Div of expr * expr────┴───────
┘
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Var of string
  | Value of value
type environment = (string * value) list
```

```
val lookup :
  string -> (string * 'a) list -> 'a = <fun>
val eval : expr -> environment -> value =
  <fun>
type state = environment
type stmt =
    Assign of string * expr
  | While of expr * stmt
  | IfThen of expr * stmt
  | IfThenElse of expr * stmt * stmt
  | Seq of stmt * stmt
  | WriteNum of expr
  | ReadNum of string
val program_1 : stmt =
  Seq (Assign ("x", Value (Int 1)),
   Seq
    (Assign ("y",
      Add (Var "x", Value (Int 2))),
    Seq
     (Assign ("z",
       Add (Var "y", Value (Int 3))),
     WriteNum (Var "z"))))
val program_2 : stmt =
  Seq (ReadNum "x",
   Seq (Assign ("i", Value (Int 0)),
    Seq (Assign ("sum", Value (Int 0)),
     Seq
      (While (Lt (Var "i", Var "x"),
        Seq (WriteNum (Var "i"),
         Seq
          (Assign ("sum",
            Add (Var "sum", Var "i")),
          Assign ("i",
           Add (Var "i", Value (Int 1)))))),
      WriteNum (Var "sum")))))
val read_number : unit -> int = <fun>
val write_number : int -> unit = <fun>
File "interpreter.ml", line 140, characters 5-260:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
Int _
File "interpreter.ml", line 127, characters 2-637:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
(IfThen (_, _)|IfThenElse (_, _, _))
val exec : stmt -> state -> state = <fun>
-( 15:44:11 )-< command 3 >------{ counter: 0 }-
utop # #quit ;;
carbon:Sec_01_1:25pm$
```