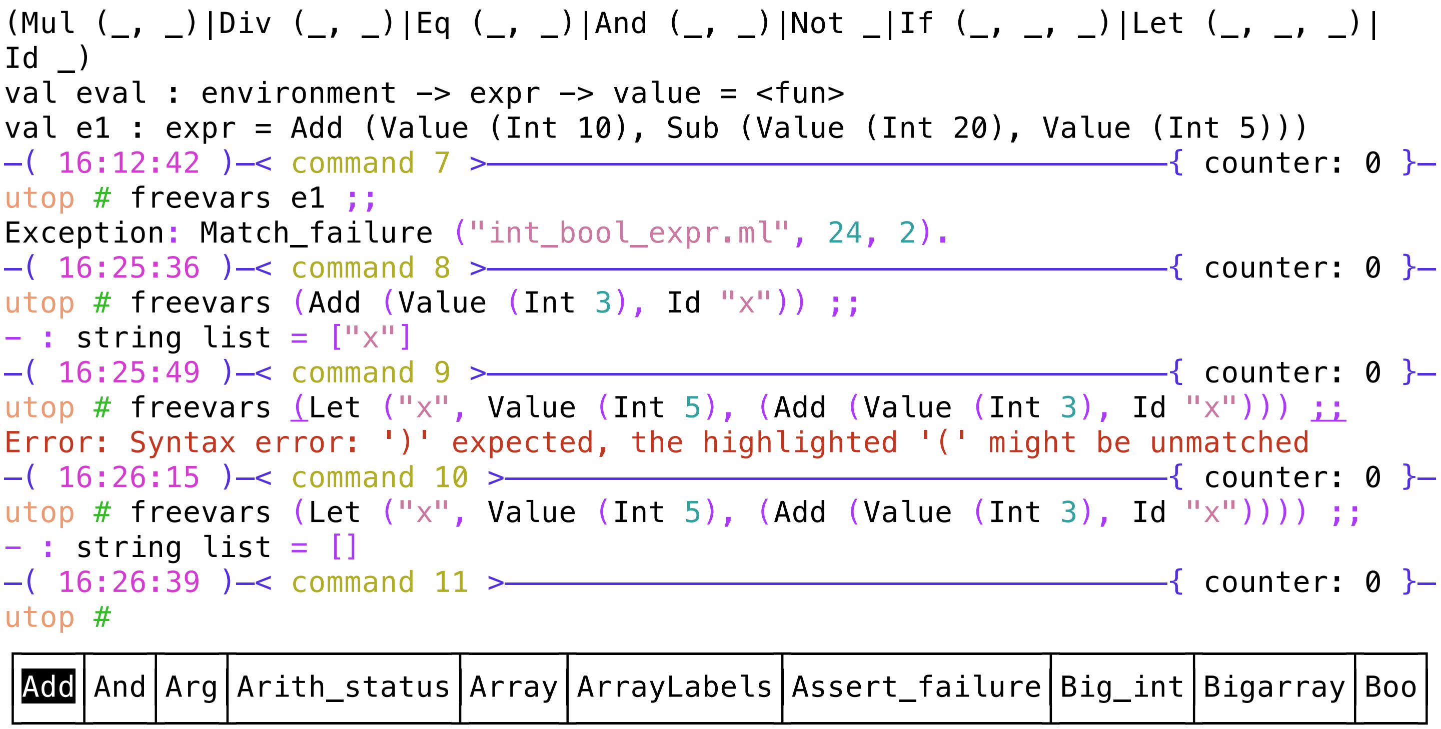
```
Last login: Fri Feb 24 15:34:48 on ttys015
carbon:public-class-repo$ cd SamplePrograms/Sec_10_3\:35pm/
carbon:Sec_10_3:35pm$ utop
───────────────┬─────────────────────────────────────────────────────────────┬──────────
               │ Welcome to utop version 1.14 (using OCaml version 4.01.0)! │
               └─────────────────────────────────────────────────────────────┘

Type #utop_help for help about using utop.

─( 18:00:00 )─< command 0 >──────────────────────────────────────{ counter: 0 }─
utop # 3 + (4 < 5) ;;
Error: This expression has type bool but an expression was expected of type int
─( 15:48:06 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # #use "int_bool_expr.ml";;
type expr =
    Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Not of expr
  | If of expr * expr * expr
  | Let of string * expr * expr
  | Id of string
  | Value of value
and value = Int of int | Bool of bool
type environment = (string * value) list
File "int_bool_expr.ml", line 31, characters 4-16:
Error: This pattern matches values of type expr
       but a pattern was expected which matches values of type value * value
─( 15:48:14 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # #use "int_bool_expr.ml";;
type expr =
    Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Not of expr
  | If of expr * expr * expr
  | Let of string * expr * expr
  | Id of string
  | Value of value
and value = Int of int | Bool of bool
type environment = (string * value) list
File "int_bool_expr.ml", line 31, characters 4-16:
Error: This pattern matches values of type expr
       but a pattern was expected which matches values of type value * value
─( 16:07:11 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # #use "int_bool_expr.ml";;
```

```
type expr =
    Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Not of expr
  | If of expr * expr * expr
  | Let of string * expr * expr
  | Id of string
  | Value of value
and value = Int of int | Bool of bool
type environment = (string * value) list
File "int_bool_expr.ml", line 25, characters 2-374:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
(Mul (_, _)|Div (_, _)|Lt (_, _)|Eq (_, _)|And (_, _)|Not _|If (_, _, _)|
Let (_, _, _)|Id _)
val eval : environment -> expr -> value = <fun>
val e1 : expr = Add (Value (Int 10), Sub (Value (Int 20), Value (Int 5)))
─( 16:09:45 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # eval [] e1 ;;
- : value = Int 25
─( 16:09:51 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # eval [] (Add (Value (Int 4), Value (Bool true)));;
Exception: Failure "Incompatible types on Add".
─( 16:09:55 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
utop # #use "int_bool_expr.ml";;
type expr =
    Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
  | Lt of expr * expr
  | Eq of expr * expr
  | And of expr * expr
  | Not of expr
  | If of expr * expr * expr
  | Let of string * expr * expr
  | Id of string
  | Value of value
and value = Int of int | Bool of bool
File "int_bool_expr.ml", line 24, characters 2-235:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
(Sub (_, _)|Mul (_, _)|Div (_, _)|Lt (_, _)|Eq (_, _)|And (_, _)|Not _|
If (_, _, _))
val freevars : expr -> string list = <fun>
type environment = (string * value) list
File "int_bool_expr.ml", line 37, characters 2-544:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
```



```
(Mul (_, _)|Div (_, _)|Eq (_, _)|And (_, _)|Not _|If (_, _, _)|Let (_, _, _)|
Id _)
val eval : environment -> expr -> value = <fun>
val e1 : expr = Add (Value (Int 10), Sub (Value (Int 20), Value (Int 5)))
─( 16:12:42 )─< command 7 >──────────────────────────────────────{ counter: 0 }─
utop # freevars e1 ;;
Exception: Match_failure ("int_bool_expr.ml", 24, 2).
─( 16:25:36 )─< command 8 >──────────────────────────────────────{ counter: 0 }─
utop # freevars (Add (Value (Int 3), Id "x")) ;;
- : string list = ["x"]
─( 16:25:49 )─< command 9 >──────────────────────────────────────{ counter: 0 }─
utop # freevars (Let ("x", Value (Int 5), (Add (Value (Int 3), Id "x"))) ;;
Error: Syntax error: ')' expected, the highlighted '(' might be unmatched
─( 16:26:15 )─< command 10 >─────────────────────────────────────{ counter: 0 }─
utop # freevars (Let ("x", Value (Int 5), (Add (Value (Int 3), Id "x")))) ;;
- : string list = []
─( 16:26:39 )─< command 11 >─────────────────────────────────────{ counter: 0 }─
utop #
```

Add│And│Arg│Arith_status│Array│ArrayLabels│Assert_failure│Big_int│Bigarray│Boo