```
Last login: Wed Feb 14 13:08:40 on ttys003
carbon:$ utop
                Welcome to utop version 2.0.2 (using OCaml version 4.06.0)!

Type #utop_help for help about using utop.

─( 13:38:54 )─< command 0 >──────────────────────────────────{ counter: 0 }─
utop # sqrt ;;
- : float -> float = <fun>
─( 13:38:54 )─< command 1 >──────────────────────────────────{ counter: 0 }─
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
File "inductive.ml", line 39, characters 12-13:
Error: This expression has type int but an expression was expected of type
         float
─( 13:39:00 )─< command 2 >──────────────────────────────────{ counter: 0 }─
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
File "inductive.ml", line 39, characters 7-13:
Error: This expression has type float
       but an expression was expected of type 'a maybe
─( 13:39:23 )─< command 3 >──────────────────────────────────{ counter: 0 }─
```

```
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
val mysqrt : float -> float maybe = <fun>
─( 13:39:29 )─< command 4 >──────────────────────────────────{ counter: 0 }─
utop # mysqrt 45.0 ;;
- : float maybe = Just 6.70820393249936942
─( 13:39:38 )─< command 5 >──────────────────────────────────{ counter: 0 }─
utop # mysqrt (-45.0) ;;
- : float maybe = Nothing
─( 13:39:53 )─< command 6 >──────────────────────────────────{ counter: 0 }─
utop # match mysqrt 45.0 with
| Nothing -> "Oh no!"
| Just _ -> "Yeah!" ;;
- : string = "Yeah!"
─( 13:40:01 )─< command 7 >──────────────────────────────────{ counter: 0 }─
utop # Just 6 ;;
- : int maybe = Just 6
─( 13:40:29 )─< command 8 >──────────────────────────────────{ counter: 0 }─
utop # Just 9.0 ;;
- : float maybe = Just 9.
─( 13:41:51 )─< command 9 >──────────────────────────────────{ counter: 0 }─
utop # Nothing ;;
- : 'a maybe = Nothing
─( 13:42:27 )─< command 10 >─────────────────────────────────{ counter: 0 }─
utop # Just Nothing ;;
- : 'a maybe maybe = Just Nothing
─( 13:45:38 )─< command 11 >─────────────────────────────────{ counter: 0 }─
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
```

```
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
val mysqrt : float -> float maybe = <fun>
val listHd : 'a list -> 'a option = <fun>
─( 13:46:22 )─< command 12 >──────────────────────────────────{ counter: 0 }─
utop # listHd [] ;;
- : 'a option = None
─( 13:50:26 )─< command 13 >──────────────────────────────────{ counter: 0 }─
utop # listHd [1;2;3] ;;
- : int option = Some 1
─( 13:50:30 )─< command 14 >──────────────────────────────────{ counter: 0 }─
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
val mysqrt : float -> float maybe = <fun>
val listHd : 'a list -> 'a option = <fun>
─( 13:50:35 )─< command 15 >──────────────────────────────────{ counter: 0 }─
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
val mysqrt : float -> float maybe = <fun>
val listHd : 'a list -> 'a option = <fun>
type 'a myList = Nil | Cons of 'a * 'a myList
```

```
val empytlist : 'a myList = Nil
val alist : int myList = Cons (3, Cons (2, Cons (1, Nil)))
-( 13:51:15 )-< command 16 >-------------------------------------{ counter: 0 }-
utop # (1, Nil) ;;
- : int * 'a myList = (1, Nil)
-( 13:55:34 )-< command 17 >-------------------------------------{ counter: 0 }-
utop # Cons (1, Nil) ;;
- : int myList = Cons (1, Nil)
-( 13:56:05 )-< command 18 >-------------------------------------{ counter: 0 }-
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
val mysqrt : float -> float maybe = <fun>
val listHd : 'a list -> 'a option = <fun>
type 'a myList = Nil | Cons of 'a * 'a myList
val empytlist : 'a myList = Nil
val alist : int myList = Cons (3, Cons (2, Cons (1, Nil)))
val sumMyList : int myList -> int = <fun>
-( 13:56:44 )-< command 19 >-------------------------------------{ counter: 0 }-
utop # sumMyList alist ;;
- : int = 6
-( 13:58:04 )-< command 20 >-------------------------------------{ counter: 0 }-
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
val mysqrt : float -> float maybe = <fun>
```

```
val listHd : 'a list -> 'a option = <fun>
type 'a myList = Nil | Cons of 'a * 'a myList
val empytlist : 'a myList = Nil
val alist : int myList = Cons (3, Cons (2, Cons (1, Nil)))
val sumMyList : int myList -> int = <fun>
-( 13:58:10 )-< command 21 >-------------------------------------{ counter: 0 }-
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
val mysqrt : float -> float maybe = <fun>
val listHd : 'a list -> 'a option = <fun>
type 'a myList = Nil | Cons of 'a * 'a myList
val empytlist : 'a myList = Nil
val alist : int myList = Cons (3, Cons (2, Cons (1, Nil)))
val sumMyList : int myList -> int = <fun>
type 'a btree = Empty | Node of 'a * 'a btree * 'a btree
val t7 : int btree = Node (7, Empty, Empty)
val t13 : int btree = Node (13, Empty, Empty)
val t10 : int btree =
  Node (10, Node (7, Empty, Empty), Node (13, Empty, Empty))
-( 13:59:52 )-< command 22 >-------------------------------------{ counter: 0 }-
utop # #use "inductive.ml";;
File "inductive.ml", line 65, characters 0-0:
Error: Syntax error: ')' expected
File "inductive.ml", line 64, characters 13-14:
Error: This '(' might be unmatched
-( 14:08:25 )-< command 23 >-------------------------------------{ counter: 0 }-
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
```

```
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
val mysqrt : float -> float maybe = <fun>
val listHd : 'a list -> 'a option = <fun>
type 'a myList = Nil | Cons of 'a * 'a myList
val empytlist : 'a myList = Nil
val alist : int myList = Cons (3, Cons (2, Cons (1, Nil)))
val sumMyList : int myList -> int = <fun>
type 'a btree = Empty | Node of 'a * 'a btree * 'a btree
val t7 : int btree = Node (7, Empty, Empty)
val t13 : int btree = Node (13, Empty, Empty)
val t10 : int btree =
  Node (10, Node (7, Empty, Empty), Node (13, Empty, Empty))
val t : int btree =
  Node (10, Node (7, Empty, Empty), Node (13, Empty, Empty))
─( 14:08:55 )─< command 24 >──────────────────────────────────{ counter: 0 }─
utop # t10 = t ;;
- : bool = true
─( 14:08:59 )─< command 25 >──────────────────────────────────{ counter: 0 }─
utop # #use "inductive.ml";;
type color = Red | Green | Blue
val isRed : color -> bool = <fun>
type weekday = Mon | Tue | Wed | Thr | Fri | Sat | Sun
val isWorkDay : weekday -> bool = <fun>
type intorstr = Int of int | Str of string
type coord = float * float
type circ_desc = coord * float
type tri_desc = coord * coord * coord
type sqr_desc = coord * coord * coord * coord
type shape =
    Circle of circ_desc
  | Triangle of tri_desc
  | Square of sqr_desc
val area : shape -> float = <fun>
type 'a maybe = Nothing | Just of 'a
val mysqrt : float -> float maybe = <fun>
val listHd : 'a list -> 'a option = <fun>
type 'a myList = Nil | Cons of 'a * 'a myList
val empytlist : 'a myList = Nil
val alist : int myList = Cons (3, Cons (2, Cons (1, Nil)))
val sumMyList : int myList -> int = <fun>
type 'a btree = Empty | Node of 'a * 'a btree * 'a btree
val t7 : int btree = Node (7, Empty, Empty)
val t13 : int btree = Node (13, Empty, Empty)
val t10 : int btree =
  Node (10, Node (7, Empty, Empty), Node (13, Empty, Empty))
val t : int btree =
  Node (10, Node (7, Empty, Empty), Node (13, Empty, Empty))
val sumTree : int btree -> int = <fun>
─( 14:09:05 )─< command 26 >──────────────────────────────────{ counter: 0 }─
```

```
utop # sumTree t ;;
- : int = 30
—( 14:13:53 )—< command 27 >————————————————————————————{ counter: 0 }—
utop #
```

Arg	Array	ArrayLabels	Assert_failure	Bigarray	Blue	Buffer	Bytes	BytesLabel