```
Last login: Fri Feb 16 13:17:44 on ttys002
carbon:$ utop
────────┬─────────────────────────────────────────────────────────────┬────────
        │ Welcome to utop version 2.0.2 (using OCaml version 4.06.0)! │
        └─────────────────────────────────────────────────────────────┘

Type #utop_help for help about using utop.

─( 13:31:11 )─< command 0 >──────────────────────────────────────{ 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>
val isElem : 'a btree -> 'a -> bool = <fun>
─( 13:31:11 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # isElem t10 13 ;;
- : bool = true
─( 13:31:13 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # isElem t10 12 ;;
- : bool = false
─( 13:31:24 )─< 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>
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>
val isElem : 'a btree -> 'a -> bool = <fun>
─( 13:31:28 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # isElem t10 12 ;;
- : bool = false
─( 13:33:48 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # isElem t10 13 ;;
- : bool = true
─( 13:33:50 )─< command 6 >──────────────────────────────────────{ 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>
val isElem : 'a btree -> 'a -> bool = <fun>
val inc100 : int btree -> int btree = <fun>
─( 13:33:52 )─< command 7 >──────────────────────────────────────{ counter: 0 }─
utop # inc100 t ;;
- : int btree = Node (110, Node (107, Empty, Empty), Node (113, Empty, Empty))
─( 13:35:39 )─< command 8 >──────────────────────────────────────{ counter: 0 }─
utop # List.map ;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
─( 13:35:45 )─< command 9 >──────────────────────────────────────{ 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>
val isElem : 'a btree -> 'a -> bool = <fun>
val inc100 : int btree -> int btree = <fun>
val treeMap : ('a -> 'b) -> 'a btree -> 'b btree = <fun>
─( 13:36:32 )─< command 10 >─────────────────────────────────────{ counter: 0 }─
utop # treeMap (fun x -> x + 100) t ;;
- : int btree = Node (110, Node (107, Empty, Empty), Node (113, Empty, Empty))
─( 13:37:53 )─< command 11 >─────────────────────────────────────{ counter: 0 }─
utop # List.fold_right ;;
- : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b = <fun>
─( 13:38:03 )─< command 12 >─────────────────────────────────────{ 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>
val isElem : 'a btree -> 'a -> bool = <fun>
val inc100 : int btree -> int btree = <fun>
val treeMap : ('a -> 'b) -> 'a btree -> 'b btree = <fun>
val treeReduce : ('a -> 'b -> 'b -> 'b) -> 'b -> 'a btree -> 'b = <fun>
─( 13:41:43 )─< command 13 >──────────────────────────────────────{ counter: 0 }─
utop # t ;;
- : int btree = Node (10, Node (7, Empty, Empty), Node (13, Empty, Empty))
─( 13:48:36 )─< 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>
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>
val isElem : 'a btree -> 'a -> bool = <fun>
val inc100 : int btree -> int btree = <fun>
val treeMap : ('a -> 'b) -> 'a btree -> 'b btree = <fun>
val treeReduce : ('a -> 'b -> 'b -> 'b) -> 'b -> 'a btree -> 'b = <fun>
val add3 : int -> int -> int -> int = <fun>
─( 13:48:58 )─< command 15 >──────────────────────────────────────{ counter: 0 }─
utop # treeReduce add3 0 t ;;
- : int = 30
─( 13:49:16 )─< command 16 >──────────────────────────────────────{ counter: 0 }─
utop # t;;
- : int btree = Node (10, Node (7, Empty, Empty), Node (13, Empty, Empty))
─( 13:49:26 )─< command 17 >──────────────────────────────────────{ 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>
val isElem : 'a btree -> 'a -> bool = <fun>
val inc100 : int btree -> int btree = <fun>
val treeMap : ('a -> 'b) -> 'a btree -> 'b btree = <fun>
val treeReduce : ('a -> 'b -> 'b -> 'b) -> 'b -> 'a btree -> 'b = <fun>
val add3 : int -> int -> int -> int = <fun>
val concat3ints : int -> int -> int -> string = <fun>
─( 13:49:28 )─< command 18 >──────────────────────────────────────{ counter: 0 }─
utop # treeReduce concat3ints "" t ;;
Error: This expression has type int -> int -> int -> string
```

```
      but an expression was expected of type int -> int -> int -> int
      Type string is not compatible with type int
-( 13:52:29 )-< command 19 >------------------------------------{ counter: 0 }-
utop # t ;;
- : int btree = Node (10, Node (7, Empty, Empty), Node (13, Empty, Empty))
-( 13:53:24 )-< 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>
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>
val isElem : 'a btree -> 'a -> bool = <fun>
val inc100 : int btree -> int btree = <fun>
val treeMap : ('a -> 'b) -> 'a btree -> 'b btree = <fun>
val treeReduce : ('a -> 'b -> 'b -> 'b) -> 'b -> 'a btree -> 'b = <fun>
val add3 : int -> int -> int -> int = <fun>
val tstr : string btree =
  Node ("a", Node ("Hello", Empty, Empty), Node ("Why?", Empty, Empty))
val concat3ints : int -> int -> int -> string = <fun>
-( 13:53:48 )-< 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))
val t : int btree = Node (10, Node (7, Empty, Empty), Node (13, Empty, Empty))
val sumTree : int btree -> int = <fun>
val isElem : 'a btree -> 'a -> bool = <fun>
val inc100 : int btree -> int btree = <fun>
val treeMap : ('a -> 'b) -> 'a btree -> 'b btree = <fun>
val treeReduce : ('a -> 'b -> 'b -> 'b) -> 'b -> 'a btree -> 'b = <fun>
val add3 : int -> int -> int -> int = <fun>
val tstr : string btree =
  Node ("a", Node ("Hello", Empty, Empty), Node ("Why?", Empty, Empty))
val concat3ints : int -> string -> string -> string = <fun>
─( 13:53:48 )─< command 22 >────────────────────────────────────{ counter: 0 }─
utop # treeReduce concat3ints "" t ;;
- : string = "10713"
─( 13:54:22 )─< command 23 >────────────────────────────────────{ counter: 0 }─
utop # List.fold_right ;;
- : ('a -> 'b -> 'b) -> 'a list -> 'b -> 'b = <fun>
─( 13:54:29 )─< command 24 >────────────────────────────────────{ counter: 0 }─
utop #
│Arg│Array│ArrayLabels│Assert_failure│Bigarray│Blue│Buffer│Bytes│BytesLabels│Ca│
```