```
Last login: Wed Mar 28 13:23:25 on ttys005
carbon:$ cd Intervals/
carbon:$ cd v5/
carbon:$ ls
intInterval.ml  intervals.ml
carbon:$ utop
```

```
        Welcome to utop version 2.0.2 (using OCaml version 4.06.0)!
```

```
Type #utop_help for help about using utop.

─( 13:32:53 )─< command 0 >──────────────────────────────────────{ counter: 0 }─
utop # #mod_use "intervals.ml";;
module Intervals :
  sig
    module type Comparable =
      sig type t val compare : t -> t -> int val to_string : t -> string end
    module type Interval_intf =
      sig
        type t
        type endpoint
        val create : endpoint -> endpoint -> t
        val is_empty : t -> bool
        val contains : t -> endpoint -> bool
        val intersect : t -> t -> t
        val to_string : t -> string
      end
    module Make_interval : functor (Endpoint : Comparable) -> Interval_intf
  end
─( 13:32:54 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # #use "intInterval.ml";;
module Int_comparable :
  sig type t = int val compare : 'a -> 'a -> t val to_string : t -> string end
module Int_interval :
  sig
    type t = Intervals.Make_interval(Int_comparable).t
    type endpoint = Intervals.Make_interval(Int_comparable).endpoint
    val create : endpoint -> endpoint -> t
    val is_empty : t -> bool
    val contains : t -> endpoint -> bool
    val intersect : t -> t -> t
    val to_string : t -> string
  end
File "intInterval.ml", line 35, characters 28-29:
Error: This expression has type int but an expression was expected of type
         Int_interval.endpoint
─( 13:32:59 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:$ cd ../v6
carbon:$ utop
```

```
        Welcome to utop version 2.0.2 (using OCaml version 4.06.0)!
```

```
Type #utop_help for help about using utop.

─( 13:37:05 )─< command 0 >──────────────────────────────────{ counter: 0 }─
utop # #mod_use "intervals.ml";;
module Intervals :
  sig
    module type Comparable =
      sig type t val compare : t -> t -> int val to_string : t -> string end
    module type Interval_intf =
      sig
        type t
        type endpoint
        val create : endpoint -> endpoint -> t
        val is_empty : t -> bool
        val contains : t -> endpoint -> bool
        val intersect : t -> t -> t
        val to_string : t -> string
      end
    module Make_interval :
      functor (Endpoint : Comparable) ->
        sig
          type t
          type endpoint = Endpoint.t
          val create : endpoint -> endpoint -> t
          val is_empty : t -> bool
          val contains : t -> endpoint -> bool
          val intersect : t -> t -> t
          val to_string : t -> string
        end
  end
─( 13:37:05 )─< command 1 >──────────────────────────────────{ counter: 0 }─
utop # #use "intInterval.ml";;
module Int_comparable :
  sig type t = int val compare : t -> t -> t val to_string : t -> string end
module Int_interval :
  sig
    type t = Intervals.Make_interval(Int_comparable).t
    type endpoint = int
    val create : endpoint -> endpoint -> t
    val is_empty : t -> bool
    val contains : t -> endpoint -> bool
    val intersect : t -> t -> t
    val to_string : t -> string
  end
val i : Int_interval.t = <abstr>
─( 13:37:08 )─< command 2 >──────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:$ cd ../v7
carbon:$ utop

        Welcome to utop version 2.0.2 (using OCaml version 4.06.0)!
```

```
Type #utop_help for help about using utop.

─( 13:52:51 )─< command 0 >──────────────────────────────────{ counter: 0 }─
utop # #mod_use "intervals.ml";;
module Intervals :
  sig
    module type Comparable =
      sig type t val compare : t -> t -> int val to_string : t -> string end
    module type Interval_intf =
      sig
        type t
        type endpoint
        val create : endpoint -> endpoint -> t
        val is_empty : t -> bool
        val contains : t -> endpoint -> bool
        val intersect : t -> t -> t
        val to_string : t -> string
      end
    module Make_interval :
      functor (Endpoint : Comparable) ->
        sig
          type t
          val create : Endpoint.t -> Endpoint.t -> t
          val is_empty : t -> bool
          val contains : t -> Endpoint.t -> bool
          val intersect : t -> t -> t
          val to_string : t -> string
        end
  end
─( 13:52:51 )─< command 1 >──────────────────────────────────{ counter: 0 }─
utop # #use "intInterval.ml";;
module Int_comparable :
  sig type t = int val compare : t -> t -> t val to_string : t -> string end
module Int_interval :
  sig
    type t = Intervals.Make_interval(Int_comparable).t
    val create : int -> int -> t
    val is_empty : t -> bool
    val contains : t -> int -> bool
    val intersect : t -> t -> t
    val to_string : t -> string
  end
val i : Int_interval.t = <abstr>
─( 13:52:53 )─< command 2 >──────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:$
```