```
Last login: Fri Apr 14 15:50:20 on ttys017
carbon:SamplePrograms$ cd Intervals/v4/
carbon:v4$ ocamlbuild useInterval.byte
Finished, 9 targets (9 cached) in 00:00:00.
carbon:v4$ ./useInterval.byte
An interval: (3, 4)
Another interval: (3, 6)
Their intresection: (3, 4)
A string interval: (a, d)
carbon:v4$ 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 # #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 Make_interval :
      functor (Endpoint : Comparable) ->
        sig
          type t = Interval of Endpoint.t * Endpoint.t | Empty
          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
─( 16:06:13 )─< command 1 >──────────────────────────────────────────{ counter: 0 }─
utop # #mod_use "intInterval.ml" ;;
module IntInterval :
  sig
    module Int_interval :
      sig
        type t = Interval of int * int | Empty
        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
  end
─( 16:06:26 )─< command 2 >──────────────────────────────────────────{ counter: 0 }─
utop # IntInterval.Empty ;;
Error: Unbound constructor IntInterval.Empty
─( 16:06:45 )─< command 3 >──────────────────────────────────────────{ counter: 0 }─
utop # IntInterval.Int_interval.Empty ;;
- : IntInterval.Int_interval.t = IntInterval.Int_interval.Empty
─( 16:07:25 )─< command 4 >──────────────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:v4$ cd ../v5
carbon:v5$ 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 # #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
─( 16:12:55 )─< command 1 >──────────────────────────────────────────{ counter: 0 }─
utop # #use "intInterval.ml";;
module Int_interval : Intervals.Interval_intf
File "intInterval.ml", line 27, characters 28-29:
Error: This expression has type int but an expression was expected of type
         Int_interval.endpoint
─( 16:13:03 )─< command 2 >──────────────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:v5$ cd ../v6
carbon:v6$ 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 # #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
─( 16:18:17 )─< 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>
─( 16:18:22 )─< command 2 >──────────────────────────────────────────────{ counter: 0 }─
utop # #quit ;;
carbon:v6$ cd ../v7
carbon:v7$ 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 # #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
─( 16:22:54 )─< command 1 >──────────────────────────────────────────────{ counter: 0 }─
utop # #use "intervals.ml";;
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
-( 16:22:58 )-< command 2 >-------------------------------------------------{ counter: 0 }-
utop #
```

Arg	Arith_status	Array	ArrayLabels	Assert_failure	Big_int	Bigarray	Buffer	Callback	Camlinter