```
Last login: Wed Mar 21 13:17:48 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.

─( 13:43:00 )─< command 0 >──────────────────────{ counter: 0 }─
utop # #mod_use "ourList.ml";;
module OurList :
  sig
    val map : ('a -> 'b) -> 'a list -> 'b list
    val filter : ('a -> bool) -> 'a list -> 'a list
    val foldr : ('a -> 'b -> 'b) -> 'b -> 'a list -> 'b
    val foldl : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a
    val is_elem : 'a -> 'a list -> bool
    val explode : string -> char list
    val implode : char list -> string
  end
─( 13:43:00 )─< command 1 >──────────────────────{ counter: 0 }─
utop # OurList.map (fun x -> x+1) [1;2;3] ;;
- : int list = [2; 3; 4]
─( 13:43:46 )─< command 2 >──────────────────────{ counter: 0 }─
utop # #use "usingLists.ml";;
val length : 'a list -> int = <fun>
val sum : int list -> int = <fun>
Hello
10
─( 13:45:03 )─< command 3 >──────────────────────{ counter: 0 }─
utop # #quit;;
carbon:$ ocamlbuild usingLists.byte
Finished, 5 targets (0 cached) in 00:00:00.
carbon:$ ls
Intervals/                      generators.py
_build/                         group_by_3.ml
buffer.ml                       higher_order.ml
compare_bintrees.ml             inductive.ml
cond.ml                         lazy.ml
dllist.ml                       map.ml
estrings.ml                     ordered_btree.ml
expr/                           ordered_list.ml
```

```
filter.ml                       ourList.ml
find_and_lookup-backup.ml       simple.ml
find_and_lookup.ml              usingLists.byte@
fold.ml                         usingLists.ml
carbon:$ ls -l usingLists.byte
lrwxr-xr-x  1 evw  wheel  120 Mar 21 13:48 usingLists.byte@ -> /
project/evw/Teaching/18_Spring_2041/carbon-repos/public-class-re
po/Sample Programs/Sec_01_1-25pm/_build/usingLists.byte
carbon:$ ls _build/
_digests                ourList.ml.depends
_log                    usingLists.byte*
ocamlc.where            usingLists.cmi
ourList.cmi             usingLists.cmo
ourList.cmo             usingLists.ml
ourList.ml              usingLists.ml.depends
carbon:$ ./usingLists.byte
Hello
10
carbon:$ mv usingLists.byte foo
carbon:$ ./foo
Hello
10
carbon:$ ls -l
total 160
drwxr-xr-x  10 evw  wheel   320 Mar 19 13:12 Intervals/
drwxr-xr-x  14 evw  wheel   448 Mar 21 13:48 _build/
-rw-r--r--   1 evw  wheel   792 Mar 19 13:19 buffer.ml
-rw-r--r--   1 evw  wheel  3170 Mar  7 13:17 compare_bintrees.ml
-rw-r--r--   1 evw  wheel   157 Mar  9 17:10 cond.ml
-rw-r--r--   1 evw  wheel  1755 Mar  5 13:38 dllist.ml
-rw-r--r--   1 evw  wheel   353 Feb  5 12:38 estrings.ml
drwxr-xr-x  12 evw  wheel   384 Mar  5 17:31 expr/
-rw-r--r--   1 evw  wheel    49 Feb  5 12:41 filter.ml
-rw-r--r--   1 evw  wheel  1372 Jan 31 13:15 find_and_lookup-bac
kup.ml
-rw-r--r--@  1 evw  wheel  1502 Feb  2 14:00 find_and_lookup.ml
-rw-r--r--   1 evw  wheel    49 Feb  5 12:41 fold.ml
lrwxr-xr-x   1 evw  wheel   120 Mar 21 13:48 foo@ -> /project/ev
w/Teaching/18_Spring_2041/carbon-repos/public-class-repo/Sample
Programs/Sec_01_1-25pm/_build/usingLists.byte
-rw-r--r--   1 evw  wheel   707 Mar  9 13:19 generators.py
-rw-r--r--   1 evw  wheel  1022 Feb 12 12:50 group_by_3.ml
```

```
-rw-r--r--   1 evw  wheel  1970 Feb  7 13:56 higher_order.ml
-rw-r--r--@  1 evw  wheel  2303 Feb 16 13:54 inductive.ml
-rw-r--r--   1 evw  wheel  2469 Mar 19 14:15 lazy.ml
-rw-r--r--   1 evw  wheel    49 Feb  5 12:41 map.ml
-rw-r--r--   1 evw  wheel   528 Feb 23 13:20 ordered_btree.ml
-rw-r--r--   1 evw  wheel   334 Feb 23 13:20 ordered_list.ml
-rw-r--r--   1 evw  wheel   769 Mar 21 13:09 ourList.ml
-rw-r--r--   1 evw  wheel  2339 Jan 29 13:40 simple.ml
-rw-r--r--   1 evw  wheel   413 Mar 21 13:09 usingLists.ml
carbon:$ ocamlbuild usingLists.byte -o foo2
ocamlbuild: unknown option '-o'.
Usage ocamlbuild [options] <target>
  -version                     Display the version
  --version                    same as -version
  -vnum                        Display the version number
  --vnum                       same as -vnum
  -quiet                       Make as quiet as possible
  -verbose <level>             Set the verbosity level on a scale
 from 0 to 8 (included)
  -documentation               Show rules and flags
  -log <file>                  Set log file
  -no-log                      No log file
  -clean                       Remove build directory and other f
iles, then exit
  -r                           Traverse directories by default (t
rue: traverse)
  -I <path>                    Add to include directories
  -Is <path,...>               (same as above, but accepts a (com
ma or blank)-separated list)
  -X <path>                    Directory to ignore
  -Xs <path,...>               (idem)
  -lib <flag>                  Link to this ocaml library
  -libs <flag,...>             (idem)
  -mod <module>                Link to this ocaml module
  -mods <module,...>           (idem)
  -pkg <package>               Link to this ocaml findlib package
  -pkgs <package,...>          (idem)
  -package <package>           (idem)
  -syntax <syntax>             Specify syntax using ocamlfind
  -lflag <flag>                Add to ocamlc link flags
  -lflags <flag,...>           (idem)
  -cflag <flag>                Add to ocamlc compile flags
  -cflags <flag,...>          (idem)
  -docflag <flag>             Add to ocamldoc flags
  -docflags <flag,...>        (idem)
  -yaccflag <flag>            Add to ocamlyacc flags
  -yaccflags <flag,...>       (idem)
  -lexflag <flag>             Add to ocamllex flags
  -lexflags <flag,...>        (idem)
  -ppflag <flag>              Add to ocaml preprocessing flags
  -pp <flag,...>              (idem)
  -tag <tag>                  Add to default tags
  -tags <tag,...>             (idem)
  -plugin-tag <tag>           Use this tag when compiling the my
ocamlbuild.ml plugin
  -plugin-tags <tag,...>      (idem)
  -tag-line <tag>             Use this line of tags (as in _tags
)
  -show-tags <path>           Show tags that applies on that pat
hname
  -ignore <module,...>        Don't try to build these modules
  -no-links                   Don't make links of produced final
 targets
  -no-skip                    Don't skip modules that are reques
ted by ocamldep but cannot be built
  -no-hygiene                 Don't apply sanity-check rules
  -no-plugin                  Don't build myocamlbuild.ml
  -no-stdlib                  Don't ignore stdlib modules
  -dont-catch-errors          Don't catch and display exceptions
 (useful to display the call stack)
  -just-plugin                Just build myocamlbuild.ml
  -byte-plugin                Don't use a native plugin but byte
code
  -plugin-option              Use the option only when plugin is
 run
  -sanitization-script        Change the file name for the gener
ated sanitization script
  -no-sanitize                Do not generate sanitization scrip
t
  -nothing-should-be-rebuilt  Fail if something needs to be rebu
ilt
  -classic-display            Display executed commands the old-
fashioned way
  -use-menhir                 Use menhir instead of ocamlyacc
  -use-jocaml                 Use jocaml compilers instead of oc
aml ones
  -use-ocamlfind              Use the 'ocamlfind' wrapper instea
d of using Findlib directly to determine command-line arguments.
 Use -no-ocamlfind to disable. Implies -plugin-use-ocamlfind.
  -no-ocamlfind               Don't use ocamlfind. Implies -plug
in-no-ocamlfind.
  -plugin-use-ocamlfind       Use the 'ocamlfind' wrapper for bu
ilding myocamlbuild.ml
  -plugin-no-ocamlfind        Don't use ocamlfind for building m
yocamlbuild.ml
  -toolchain <toolchain>      Set the Findlib toolchain to use.
The default toolchain is always used for building myocamlbuild.m
l.
  -j <N>                      Allow N jobs at once (0 for unlimi
ted)
  -build-dir <path>           Set build directory (implies no-li
nks)
  -install-lib-dir <path>     Set the install library directory
  -install-bin-dir <path>     Set the install binary directory
  -where                      Display the install library direct
ory
  -which <command>            Display path to the tool command
  -ocamlc <command>           Set the OCaml bytecode compiler
  -plugin-ocamlc <command>    Set the OCaml bytecode compiler us
ed when building myocamlbuild.ml (only)
  -ocamlopt <command>         Set the OCaml native compiler
  -plugin-ocamlopt <command>  Set the OCaml native compiler used
 when building myocamlbuild.ml (only)
  -ocamldep <command>         Set the OCaml dependency tool
  -ocamldoc <command>         Set the OCaml documentation genera
tor
  -ocamlyacc <command>        Set the ocamlyacc tool
  -menhir <command>           Set the menhir tool (use it after
-use-menhir)
  -ocamllex <command>         Set the ocamllex tool
  -ocamlmklib <command>       Set the ocamlmklib tool
  -ocamlmktop <command>       Set the ocamlmktop tool
  -ocamlrun <command>         Set the ocamlrun tool
  --                          Stop argument processing, remainin
g arguments are given to the user program
  -help                       Display this list of options
  --help                      Display this list of options

carbon:$ ocamlbuild usingLists.native
Finished, 7 targets (4 cached) in 00:00:03.
carbon:$ ./usingLists.native
Hello
10
carbon:$ ocamldebug usingLists.native
        OCaml Debugger version 4.06.0

(ocd) break @ usingLists 16
Loading program... /project/evw/Teaching/18_Spring_2041/carbon-r
epos/public-class-repo/Sample Programs/Sec_01_1-25pm/usingLists.
native is not a bytecode file.
(ocd) carbon:$ cd Intervals/
carbon:$ ls
README.md       v2/             v4/             v6/
v1/             v3/             v5/             v7/
carbon:$ cd v1
carbon:$ ls
intInterval.ml          useIntInterval.ml
carbon:$ utop
 ┌────────────────────────────────────────────────────────────┬─
 │ Welcome to utop version 2.0.2 (using OCaml version 4.06.0)! │
 └────────────────────────────────────────────────────────────┘


Type #utop_help for help about using utop.

─( 13:56:56 )─< command 0 >──────────────────────{ counter: 0 }─
utop # #mod_use "intInterval.ml";;
module IntInterval :
  sig
    type intInterval = Interval of int * int | Empty
    val is_empty : intInterval -> bool
    val contains : intInterval -> int -> bool
    val intersect : intInterval -> intInterval -> intInterval
    val to_string : intInterval -> string
  end
─( 13:56:56 )─< command 1 >──────────────────────{ counter: 0 }─
utop # #use "useIntInterval.ml";;
val i1 : IntInterval.intInterval = IntInterval.Interval (3, 4)
val i2 : IntInterval.intInterval = IntInterval.Interval (3, 6)
```

```
An interval: (3, 4)
Another interval: (3, 6)
Their intresection: (3, 4)
─( 13:57:02 )─< command 2 >──────────────────────{ counter: 0 }─
utop # #quit;;
carbon:$ up
/project/evw/Teaching/18_Spring_2041/carbon-repos/public-class-r
epo/Sample Programs/Sec_01_1-25pm/Intervals
carbon:$ cd v2
carbon:$ ls
intInterval.ml          useIntInterval.ml
intInterval.mli
carbon:$ more intInterval.ml
(* A module for intervals over integers.

   Here, the type is abstract and hidden from users of the code
because
   the corresponding .mli file does not mention the type 'intInt
erval'.
   Thus it is not visible since it is not in the interface for t
his
   module.

   This code is based on the Interval examples in Chapter 9 of R
eal
   World OCaml by Jason Hickey, Anil Madhavapeddy and Yaron Mins
ky.
 *)

type intInterval = Interval of int * int
                 | Empty

(* Invariant: low > hight in Interval(low,high) *)

type t = intInterval

let create (low: int) (high:int) : t =
  Interval (low, high)

let is_empty (i:intInterval) : bool =
  match i with
  | Empty -> true
  | Interval _ -> false

let contains (i:intInterval) (x:int) : bool =
  match i with
  | Empty -> false
  | Interval (l,h) -> l <= x && x <= h

let intersect (i1:intInterval) (i2:intInterval) : intInterval =
  match i1, i2 with
  | Empty, _ | _, Empty -> Empty
  | Interval (l1, h1), Interval (l2, h2) ->
     Interval (max l1 l2, min h1 h2)

let to_string (i:intInterval) : string =
  match i with
  | Empty -> "Empty"
  | Interval (l,h) -> "(" ^ string_of_int l ^ ", " ^ string_of_i
nt h ^ ")"


carbon:$
carbon:$
carbon:$
carbon:$ more intInterval.mli
(* An interface file for the intInterval that hides the implemen
tation
   type.
 *)

type 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
carbon:$ up
/project/evw/Teaching/18_Spring_2041/carbon-repos/public-class-r
```

```
epo/Sample Programs/Sec_01_1-25pm/Intervals
carbon:$ cd v1
carbon:$ ls
intInterval.ml          useIntInterval.ml
carbon:$ ls
intInterval.ml          useIntInterval.ml
carbon:$ pwd
/project/evw/Teaching/18_Spring_2041/carbon-repos/public-class-r
epo/Sample Programs/Sec_01_1-25pm/Intervals/v1
carbon:$ ocamlbuild useIntInterval.byte
Finished, 5 targets (0 cached) in 00:00:00.
carbon:$ ls
_build/                 useIntInterval.byte@
intInterval.ml          useIntInterval.ml
carbon:$ ls _build/
_digests                        ocamlc.where
_log                            useIntInterval.byte*
intInterval.cmi                 useIntInterval.cmi
intInterval.cmo                 useIntInterval.cmo
intInterval.ml                  useIntInterval.ml
intInterval.ml.depends          useIntInterval.ml.depends
carbon:$ more _build/intInterval.cmi
"_build/intInterval.cmi" may be a binary file.  See it anyway?
carbon:$ up
/project/evw/Teaching/18_Spring_2041/carbon-repos/public-class-r
epo/Sample Programs/Sec_01_1-25pm/Intervals
carbon:$ cd v3
carbon:$ ls
intervals.ml            useIntInterval.ml
carbon:$ ocamlbuild useIntInterval.byte
Finished, 5 targets (0 cached) in 00:00:00.
carbon:$ ./useIntInterval.byte
An interval: (3, 4)
Another interval: (3, 6)
Their intresection: (3, 4)
carbon:$
```