```
Last login: Wed Apr 12 15:10:12 on ttys027
carbon:SamplePrograms$ 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 # #use "ourList.ml";;
val map : ('a -> 'b) -> 'a list -> 'b list = <fun>
val filter : ('a -> bool) -> 'a list -> 'a list = <fun>
val foldr : ('a -> 'b -> 'b) -> 'b -> 'a list -> 'b = <fun>
val foldl : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a = <fun>
val is_elem : 'a -> 'a list -> bool = <fun>
val explode : string -> char list = <fun>
val implode : char list -> string = <fun>
─( 15:50:48 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # map ;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
─( 15:50:52 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:SamplePrograms$ 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 "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
─( 15:51:06 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # OurList.map ;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
─( 15:51:27 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # OurList.map (fun x -> x + 1) [1;2;3;4;5] ;;
- : int list = [2; 3; 4; 5; 6]
─( 15:52:13 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:SamplePrograms$ ls
ElemsOfFP_Reade_Chap_8.ml        jan_27.ml
Intervals/                       lab_06.ml
MyCopiesForSecs/                 lazy.ml
```

```
README.md                       ordered_list.ml
Sec_01_1:25pm/                  ourList.ml
Sec_10_3:35pm/                  session_info.byte@
_build/                         session_info.ml
client_server.ml                simple.ml
compare_bintrees.ml             streams.ml
dllist.ml                       usingLists.ml
gcd.ml                          usingLists.native@
generators.py
carbon:SamplePrograms$ corebuild -clean
carbon:SamplePrograms$ ls
ElemsOfFP_Reade_Chap_8.ml       generators.py
Intervals/                      jan_27.ml
MyCopiesForSecs/                lab_06.ml
README.md                       lazy.ml
Sec_01_1:25pm/                  ordered_list.ml
Sec_10_3:35pm/                  ourList.ml
client_server.ml                session_info.ml
compare_bintrees.ml             simple.ml
dllist.ml                       streams.ml
gcd.ml                          usingLists.ml
carbon:SamplePrograms$ corebuild usingLists.byte
ocamlfind ocamldep -syntax camlp4o -package bin_prot.syntax -package sexplib.syn
tax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core -modules
 usingLists.ml > usingLists.ml.depends
ocamlfind ocamldep -syntax camlp4o -package bin_prot.syntax -package sexplib.syn
tax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core -modules
 ourList.ml > ourList.ml.depends
ocamlfind ocamlc -c -w A-4-33-40-41-42-43-34-44 -strict-sequence -g -annot -bin-
annot -short-paths -thread -syntax camlp4o -package bin_prot.syntax -package sex
plib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core
-o ourList.cmo ourList.ml
ocamlfind ocamlc -c -w A-4-33-40-41-42-43-34-44 -strict-sequence -g -annot -bin-
annot -short-paths -thread -syntax camlp4o -package bin_prot.syntax -package sex
plib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core
-o usingLists.cmo usingLists.ml
ocamlfind ocamlc -linkpkg -g -thread -syntax camlp4o -package bin_prot.syntax -p
ackage sexplib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -pac
kage core ourList.cmo usingLists.cmo -o usingLists.byte
carbon:SamplePrograms$ ./usingLists.byte
Hello
10
carbon:SamplePrograms$ corebuild usingLists.native
ocamlfind ocamlopt -c -w A-4-33-40-41-42-43-34-44 -strict-sequence -g -annot -bi
n-annot -short-paths -thread -syntax camlp4o -package bin_prot.syntax -package s
explib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package cor
e -o ourList.cmx ourList.ml
ocamlfind ocamlopt -c -w A-4-33-40-41-42-43-34-44 -strict-sequence -g -annot -bi
n-annot -short-paths -thread -syntax camlp4o -package bin_prot.syntax -package s
explib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package cor
e -o usingLists.cmx usingLists.ml
ocamlfind ocamlopt -linkpkg -g -thread -syntax camlp4o -package bin_prot.syntax
-package sexplib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -p
```

```
ackage core ourList.cmx usingLists.cmx -o usingLists.native
carbon:SamplePrograms$ ./usingLists.native
Hello
10
carbon:SamplePrograms$ cd Intervals/v1
carbon:v1$ ls
_build/                 useIntInterval.byte@
intInterval.ml          useIntInterval.ml
carbon:v1$ corebuild -clean
carbon:v1$ ls
intInterval.ml          useIntInterval.ml
carbon:v1$ corebuild useIntInterval.byte
ocamlfind ocamldep -syntax camlp4o -package bin_prot.syntax -package sexplib.syn
tax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core -modules
 useIntInterval.ml > useIntInterval.ml.depends
ocamlfind ocamldep -syntax camlp4o -package bin_prot.syntax -package sexplib.syn
tax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core -modules
 intInterval.ml > intInterval.ml.depends
ocamlfind ocamlc -c -w A-4-33-40-41-42-43-34-44 -strict-sequence -g -annot -bin-
annot -short-paths -thread -syntax camlp4o -package bin_prot.syntax -package sex
plib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core
-o intInterval.cmo intInterval.ml
ocamlfind ocamlc -c -w A-4-33-40-41-42-43-34-44 -strict-sequence -g -annot -bin-
annot -short-paths -thread -syntax camlp4o -package bin_prot.syntax -package sex
plib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core
-o useIntInterval.cmo useIntInterval.ml
ocamlfind ocamlc -linkpkg -g -thread -syntax camlp4o -package bin_prot.syntax -p
ackage sexplib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -pac
kage core intInterval.cmo useIntInterval.cmo -o useIntInterval.byte
carbon:v1$ ./useIntInterval.byte
An interval: (3, 4)
Another interval: (3, 6)
Their intresection: (3, 4)
carbon:v1$ cd ../v2
carbon:v2$ ls
_build/                 intInterval.ml          useIntInterval.byte@
intInterval.byte@       intInterval.mli         useIntInterval.ml
carbon:v2$ corebuild -clean
carbon:v2$ corebuild useIntInterval.byte
ocamlfind ocamldep -syntax camlp4o -package bin_prot.syntax -package sexplib.syn
tax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core -modules
 useIntInterval.ml > useIntInterval.ml.depends
ocamlfind ocamldep -syntax camlp4o -package bin_prot.syntax -package sexplib.syn
tax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core -modules
 intInterval.mli > intInterval.mli.depends
ocamlfind ocamlc -c -w A-4-33-40-41-42-43-34-44 -strict-sequence -g -annot -bin-
annot -short-paths -thread -syntax camlp4o -package bin_prot.syntax -package sex
plib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core
-o intInterval.cmi intInterval.mli
ocamlfind ocamlc -c -w A-4-33-40-41-42-43-34-44 -strict-sequence -g -annot -bin-
annot -short-paths -thread -syntax camlp4o -package bin_prot.syntax -package sex
plib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core
-o useIntInterval.cmo useIntInterval.ml
```

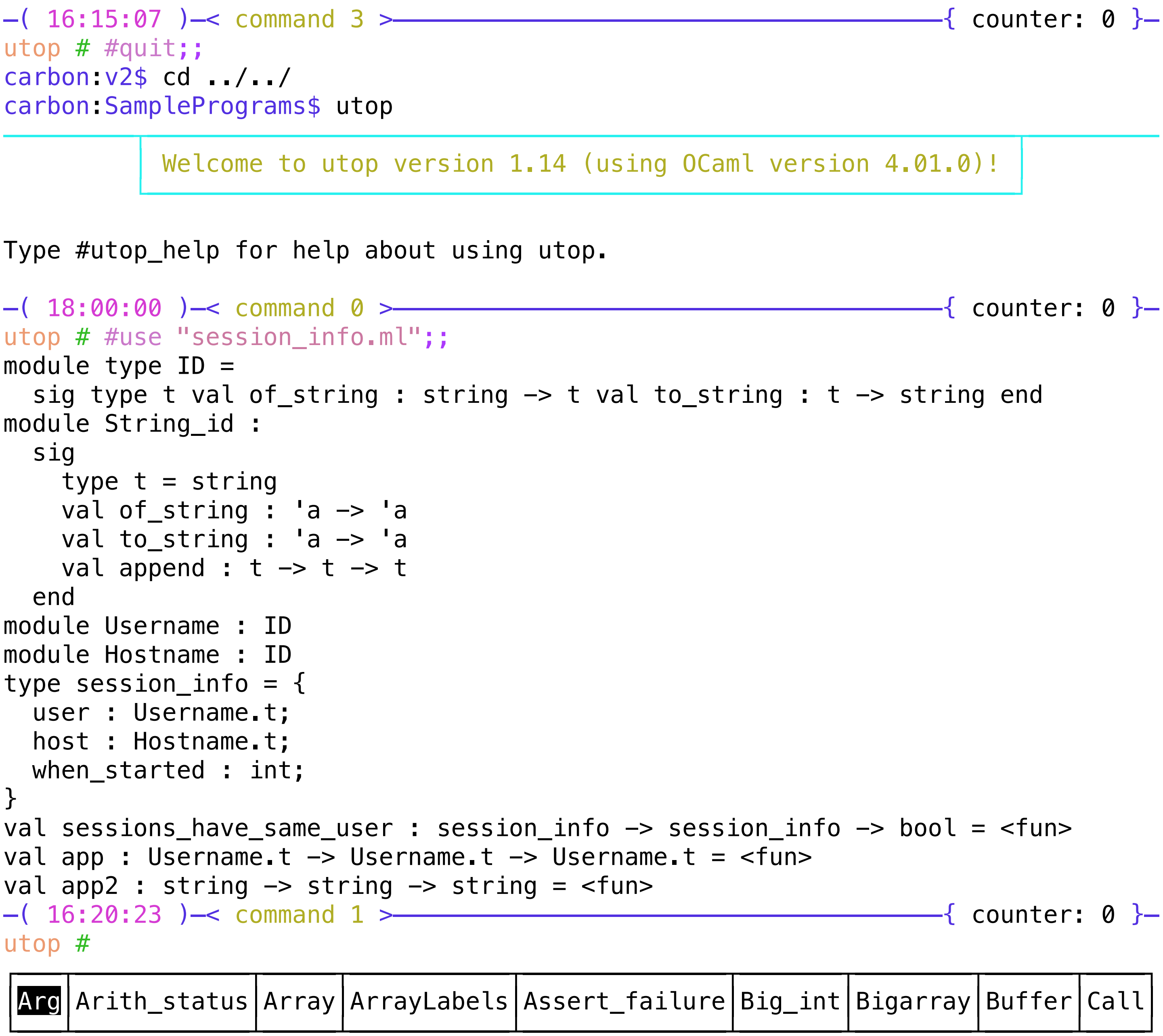
```
ocamlfind ocamldep -syntax camlp4o -package bin_prot.syntax -package sexplib.syn
tax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core -modules
 intInterval.ml > intInterval.ml.depends
ocamlfind ocamlc -c -w A-4-33-40-41-42-43-34-44 -strict-sequence -g -annot -bin-
annot -short-paths -thread -syntax camlp4o -package bin_prot.syntax -package sex
plib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -package core
-o intInterval.cmo intInterval.ml
ocamlfind ocamlc -linkpkg -g -thread -syntax camlp4o -package bin_prot.syntax -p
ackage sexplib.syntax,comparelib.syntax,fieldslib.syntax,variantslib.syntax -pac
kage core intInterval.cmo useIntInterval.cmo -o useIntInterval.byte
carbon:v2$ ./useIntInterval.byte
An interval: (3, 4)
Another interval: (3, 6)
Their intresection: (3, 4)
carbon:v2$ 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 "intInterval.ml" ;;
module IntInterval :
  sig
    type intInterval = Interval of int * int | Empty
    type t = intInterval
    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
─( 16:08:14 )─< command 1 >────────────────────────────────────{ counter: 0 }─
utop # #quit ;;
carbon:v2$ ls
_build/                 intInterval.mli         useIntInterval.ml
intInterval.ml          useIntInterval.byte@
carbon:v2$ 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 # List.map ;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
─( 16:14:52 )─< command 1 >────────────────────────────────────{ counter: 0 }─
utop # open List ;;
─( 16:14:59 )─< command 2 >────────────────────────────────────{ counter: 0 }─
utop # map ;;
- : ('a -> 'b) -> 'a list -> 'b list = <fun>
```



```
─( 16:15:07 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:v2$ cd ../../
carbon:SamplePrograms$ 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 # #use "session_info.ml";;
module type ID =
  sig type t val of_string : string -> t val to_string : t -> string end
module String_id :
  sig
    type t = string
    val of_string : 'a -> 'a
    val to_string : 'a -> 'a
    val append : t -> t -> t
  end
module Username : ID
module Hostname : ID
type session_info = {
  user : Username.t;
  host : Hostname.t;
  when_started : int;
}
val sessions_have_same_user : session_info -> session_info -> bool = <fun>
val app : Username.t -> Username.t -> Username.t = <fun>
val app2 : string -> string -> string = <fun>
─( 16:20:23 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop #
┌───┬────────────┬─────┬───────────┬────────────────┬───────┬────────┬──────┬────┐
│Arg│Arith_status│Array│ArrayLabels│Assert_failure│Big_int│Bigarray│Buffer│Call│
└───┴────────────┴─────┴───────────┴────────────────┴───────┴────────┴──────┴────┘
```