```
Last login: Tue Jan 31 15:09:53 on ttys014
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 "map.ml";;
val map : ('a -> 'b) -> 'a list -> 'b list = <fun>
─( 13:41:06 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # map (fun x -> x + 1)  [2;3;4;5] ;;
- : int list = [3; 4; 5; 6]
─( 13:41:13 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # map ( (+) 1 ) [2;3;4;5] ;;
- : int list = [3; 4; 5; 6]
─( 13:41:36 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # map Char.code ['a'; '3' ];;
- : int list = [97; 51]
─( 13:42:12 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # let add x y = x + y ;;
val add : int -> int -> int = <fun>
─( 13:42:12 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # map (add 3) [2;3;4] ;;
- : int list = [5; 6; 7]
─( 13:46:34 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
utop # #use "estrings.ml";;
type estring = char list
val explode : string -> char list = <fun>
val implode : char list -> string = <fun>
val freshman : estring -> estring = <fun>
File "estrings.ml", line 24, characters 17-57:
Error: This expression has type string but an expression was expected of type
         char list
─( 13:46:42 )─< command 7 >──────────────────────────────────────{ counter: 0 }─
utop # #quit;;
carbon:SamplePrograms$ cd Sec_01_1\:25pm/
carbon:Sec_01_1:25pm$ ocaml
        OCaml version 4.01.0

# #quit ;;
carbon:Sec_01_1:25pm$ 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 "map.ml";;
```

```
val map : ('a -> 'b) -> 'a list -> 'b list = <fun>
─( 13:50:37 )─< command 1 >──────────────────────────────────{ counter: 0 }─
utop # #use "estring.ml";;
type estring = char list
val explode : string -> char list = <fun>
val implode : char list -> string = <fun>
─( 13:50:41 )─< command 2 >──────────────────────────────────{ counter: 0 }─
utop # explode "Hello. World!" ;;
- : char list =
['H'; 'e'; 'l'; 'l'; 'o'; '.'; ' '; 'W'; 'o'; 'r'; 'l'; 'd'; '!']
─( 13:50:46 )─< command 3 >──────────────────────────────────{ counter: 0 }─
utop # #use "estring.ml";;
type estring = char list
val explode : string -> char list = <fun>
val implode : char list -> string = <fun>
val freshperson : char list -> char list = <fun>
─( 13:50:55 )─< command 4 >──────────────────────────────────{ counter: 0 }─
utop # freshperson (explode "Hey, freshman!")  ;;
- : char list =
['H'; 'e'; 'y'; ','; ' '; 'f'; 'r'; 'e'; 's'; 'h'; 'm'; 'a'; 'n'; '?']
─( 13:53:04 )─< command 5 >──────────────────────────────────{ counter: 0 }─
utop # implode (freshperson (explode "Hey, freshman!"))  ;;
- : string = "Hey, freshman?"
─( 13:54:42 )─< command 6 >──────────────────────────────────{ counter: 0 }─
utop # #use "filter.ml";;
val filter : ('a -> bool) -> 'a list -> 'a list = <fun>
─( 13:54:49 )─< command 7 >──────────────────────────────────{ counter: 0 }─
utop # let even x = x mod 2 = 0 ;;
val even : int -> bool = <fun>
─( 14:03:45 )─< command 8 >──────────────────────────────────{ counter: 0 }─
utop # even 3 ;;
- : bool = false
─( 14:03:59 )─< command 9 >──────────────────────────────────{ counter: 0 }─
utop # even 4 ;;
- : bool = true
─( 14:04:02 )─< command 10 >─────────────────────────────────{ counter: 0 }─
utop # filter even [1;2;3;4;5;6];;
- : int list = [2; 4; 6]
─( 14:04:04 )─< command 11 >─────────────────────────────────{ counter: 0 }─
utop # #use "filter.ml";;
val filter : ('a -> bool) -> 'a list -> 'a list = <fun>
val filter' : ('a -> bool) -> 'a list -> 'a list = <fun>
─( 14:04:13 )─< command 12 >─────────────────────────────────{ counter: 0 }─
utop # filter' even [1;2;3;4;5;6];;
- : int list = [2; 4; 6]
─( 14:07:24 )─< command 13 >─────────────────────────────────{ counter: 0 }─
utop # not ;;
- : bool -> bool = <fun>
─( 14:07:30 )─< command 14 >─────────────────────────────────{ counter: 0 }─
utop # #use "filter.ml";;
val filter : ('a -> bool) -> 'a list -> 'a list = <fun>
```

```
File "filter.ml", line 16, characters 22-23:
Error: Unbound value x
─( 14:08:08 )─< command 15 >──────────────────────────────────{ counter: 0 }─
utop # #use "filter.ml";;
val filter : ('a -> bool) -> 'a list -> 'a list = <fun>
File "filter.ml", line 13, characters 2-106:
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
_::_
(However, some guarded clause may match this value.)
val filter' : ('a -> bool) -> 'a list -> 'a list = <fun>
─( 14:08:26 )─< command 16 >──────────────────────────────────{ counter: 0 }─
utop # #use "filter.ml";;
val filter : ('a -> bool) -> 'a list -> 'a list = <fun>
val filter' : ('a -> bool) -> 'a list -> 'a list = <fun>
val filter_out : ('a -> bool) -> 'a list -> 'a list = <fun>
─( 14:08:35 )─< command 17 >──────────────────────────────────{ counter: 0 }─
utop # filter_out even [1;2;3;4;5];;
─( 14:12:26 )─< command 18 >──────────────────────────────────{ counter: 0 }─
utop #
```

Arg	Arith_status	Array	ArrayLabels	Assert_failure	Big_int	Bigarray	Buffer	Call
						garray	Buffer	Call