```
Last login: Mon Mar 19 13:24:44 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.

─( 13:33:03 )─< command 0 >──────────────────────────────────────{ counter: 0 }─
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
val ones : int = 9999
─( 13:33:03 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # head nats ;;
- : int = 1
─( 13:33:08 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # head (tail nats) ;;
step 2
- : int = 2
─( 13:33:20 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # take 5 nats ;;
step 3
step 4
step 5
step 6
- : int list = [1; 2; 3; 4; 5]
─( 13:33:30 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # take 5 nats ;;
- : int list = [1; 2; 3; 4; 5]
─( 13:33:43 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 nats ;;
step 7
step 8
step 9
step 10
step 11
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
─( 13:35:09 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
```

```
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
File "lazy.ml", line 47, characters 13-36:
Error: This expression has type int lazee = int hidden ref
       but an expression was expected of type
         int stream lazee = int stream hidden ref
       Type int is not compatible with type int stream
-( 13:35:22 )-< command 7 >-------------------------------------{ counter: 0 }-
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
File "lazy.ml", line 47, characters 13-36:
Error: This expression has type int lazee = int hidden ref
       but an expression was expected of type
         int stream lazee = int stream hidden ref
       Type int is not compatible with type int stream
-( 13:41:35 )-< command 8 >-------------------------------------{ counter: 0 }-
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
File "lazy.ml", line 47, characters 4-38:
Error: This kind of expression is not allowed as right-hand side of `let rec'
-( 13:43:45 )-< command 9 >-------------------------------------{ counter: 0 }-
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
```

```
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
File "lazy.ml", line 47, characters 4-38:
Error: This kind of expression is not allowed as right-hand side of `let rec'
-( 13:44:28 )-< command 10 >-------------------------------------{ counter: 0 }-
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
val ones : int stream = Cons (1, {contents = Thunk <fun>})
-( 13:45:15 )-< command 11 >-------------------------------------{ counter: 0 }-
utop # take 100 ones ;;
- : int list =
[1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1;
 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1;
 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1;
 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1]
-( 13:46:31 )-< command 12 >-------------------------------------{ counter: 0 }-
utop # take (-10) ones ;;
Stack overflow during evaluation (looping recursion?).
-( 13:46:37 )-< command 13 >-------------------------------------{ counter: 0 }-
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
val ones : int stream = Cons (1, {contents = Thunk <fun>})
File "lazy.ml", line 58, characters 20-40:
Error: This expression has type 'b stream
       but an expression was expected of type
         'a stream lazee = 'a stream hidden ref
-( 13:47:12 )-< command 14 >-------------------------------------{ counter: 0 }-
utop # #use "lazy.ml";;
File "lazy.ml", line 58, characters 5-7:
```

```
Error: Syntax error
─( 13:51:58 )─< command 15 >──────────────────────────────────{ counter: 0 }─
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
val ones : int stream = Cons (1, {contents = Thunk <fun>})
val filter : ('a -> bool) -> 'a stream -> 'a stream = <fun>
─( 13:55:51 )─< command 16 >──────────────────────────────────{ counter: 0 }─
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
val ones : int stream = Cons (1, {contents = Thunk <fun>})
val filter : ('a -> bool) -> 'a stream -> 'a stream = <fun>
val even : int -> bool = <fun>
─( 13:56:00 )─< command 17 >──────────────────────────────────{ counter: 0 }─
utop # even 4 ;;
- : bool = true
─( 13:56:15 )─< command 18 >──────────────────────────────────{ counter: 0 }─
utop # filter even nats ;;
step 2
- : int stream = Cons (2, {contents = Thunk <fun>})
─( 13:56:18 )─< command 19 >──────────────────────────────────{ counter: 0 }─
utop # take 2 (filter even nats) ;;
step 3
step 4
step 5
step 6
- : int list = [2; 4]
─( 13:56:26 )─< command 20 >──────────────────────────────────{ counter: 0 }─
utop # take 12 (filter even nats) ;;
step 7
step 8
step 9
step 10
step 11
```

```
step 12
step 13
step 14
step 15
step 16
step 17
step 18
step 19
step 20
step 21
step 22
step 23
step 24
step 25
step 26
- : int list = [2; 4; 6; 8; 10; 12; 14; 16; 18; 20; 22; 24]
─( 13:56:50 )─< command 21 >──────────────────────────────────────{ counter: 0 }─
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
val ones : int stream = Cons (1, {contents = Thunk <fun>})
val filter : ('a -> bool) -> 'a stream -> 'a stream = <fun>
val even : int -> bool = <fun>
step 2
val all_even : int stream = Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
─( 13:57:11 )─< command 22 >──────────────────────────────────────{ counter: 0 }─
utop # map (fun x -> x * x) nats ;;
- : int stream = Cons (1, {contents = Thunk <fun>})
─( 14:02:10 )─< command 23 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 (map (fun x -> x * x) nats) ;;
step 3
step 4
step 5
step 6
step 7
step 8
step 9
step 10
step 11
- : int list = [1; 4; 9; 16; 25; 36; 49; 64; 81; 100]
─( 14:02:19 )─< command 24 >──────────────────────────────────────{ counter: 0 }─
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
```

```
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
val ones : int stream = Cons (1, {contents = Thunk <fun>})
val filter : ('a -> bool) -> 'a stream -> 'a stream = <fun>
val even : int -> bool = <fun>
step 2
val all_even : int stream = Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
val all_evens_v2 : int stream = Cons (2, {contents = Thunk <fun>})
val zip : ('a -> 'b -> 'c) -> 'a stream -> 'b stream -> 'c stream = <fun>
─( 14:02:29 )─< command 25 >──────────────────────────────────────{ counter: 0 }─
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
val ones : int stream = Cons (1, {contents = Thunk <fun>})
val filter : ('a -> bool) -> 'a stream -> 'a stream = <fun>
val even : int -> bool = <fun>
step 2
val all_even : int stream = Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
val all_evens_v2 : int stream = Cons (2, {contents = Thunk <fun>})
val zip : ('a -> 'b -> 'c) -> 'a stream -> 'b stream -> 'c stream = <fun>
val all_evens_v3 : int stream = Cons (2, {contents = Thunk <fun>})
─( 14:08:33 )─< command 26 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 all_even_v3 ;;
Error: Unbound value all_even_v3
Hint: Did you mean all_evens_v3?
─( 14:09:51 )─< command 27 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 all_evens_v3 ;;
step 3
step 4
step 5
step 6
step 7
step 8
step 9
step 10
```

```
step 11
- : int list = [2; 4; 6; 8; 10; 12; 14; 16; 18; 20]
─( 14:09:56 )─< command 28 >──────────────────────────────────────{ counter: 0 }─
utop # #use "lazy.ml";;
type 'a lazee = 'a hidden ref
and 'a hidden = Value of 'a | Thunk of (unit -> 'a)
val delay : (unit -> 'a) -> 'a lazee = <fun>
val force : 'a lazee -> unit = <fun>
val demand : 'a lazee -> 'a = <fun>
type 'a stream = Cons of 'a * 'a stream lazee
val from : int -> int stream = <fun>
step 1
val nats : int stream = Cons (1, {contents = Thunk <fun>})
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val take : int -> 'a stream -> 'a list = <fun>
val ones : int stream = Cons (1, {contents = Thunk <fun>})
val filter : ('a -> bool) -> 'a stream -> 'a stream = <fun>
val even : int -> bool = <fun>
step 2
val all_even : int stream = Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
val all_evens_v2 : int stream = Cons (2, {contents = Thunk <fun>})
val zip : ('a -> 'b -> 'c) -> 'a stream -> 'b stream -> 'c stream = <fun>
val all_evens_v3 : int stream = Cons (2, {contents = Thunk <fun>})
val factorials : int stream = Cons (1, {contents = Thunk <fun>})
─( 14:10:00 )─< command 29 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 factorials ;;
step 3
step 4
step 5
step 6
step 7
step 8
step 9
step 10
- : int list = [1; 1; 2; 6; 24; 120; 720; 5040; 40320; 362880]
─( 14:15:20 )─< command 30 >──────────────────────────────────────{ counter: 0 }─
utop #
```

Arg	Array	ArrayLabels	Assert_failure	Bigarray	Buffer	Bytes	BytesLabels	Callbac