```
Last login: Mon Mar 27 13:13:35 on ttys017
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 "streams.ml";;
type 'a stream = Cons of 'a * (unit -> 'a stream)
val ones : int stream = Cons (1, <fun>)
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val from : int -> int stream = <fun>
val nats : int stream = Cons (1, <fun>)
val take : int -> 'a stream -> 'a list = <fun>
val filter : ('a -> bool) -> 'a stream -> 'a stream = <fun>
val even : int -> bool = <fun>
val squares_from : int -> int stream = <fun>
val t1 : int list = [1; 4; 9; 16; 25; 36; 49; 64; 81; 100]
val squares : int stream = Cons (1, <fun>)
val zip : ('a -> 'b -> 'c) -> 'a stream -> 'b stream -> 'c stream = <fun>
val nats2 : int stream = Cons (1, <fun>)
val factorials : int stream = Cons (1, <fun>)
val non : ('a -> bool) -> 'a -> bool = <fun>
val multiple_of : int -> int -> bool = <fun>
val sift : int -> int stream -> int stream = <fun>
val sieve : int stream -> int stream = <fun>
val primes : int stream = Cons (2, <fun>)
─( 13:43:42 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # #use "client_server.ml";;
val initial_value : int = 0
val client : int stream -> int stream = <fun>
val server : int stream -> int stream = <fun>
val requests_f : unit -> int stream = <fun>
val responses_f : unit -> int stream = <fun>
─( 13:43:48 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 (requests_f ()) ;;
server: 0
client: 3
server: 0
client: 3
server: 5
client: 8
server: 0
client: 3
server: 5
client: 8
server: 10
client: 13
server: 0
client: 3
server: 5
client: 8
```

```
server: 10
client: 13
server: 15
client: 18
server: 0
client: 3
server: 5
client: 8
server: 10
client: 13
server: 15
client: 18
server: 20
client: 23
server: 0
client: 3
server: 5
client: 8
server: 10
client: 13
server: 15
client: 18
server: 20
client: 23
server: 25
client: 28
server: 0
client: 3
server: 5
client: 8
server: 10
client: 13
server: 15
client: 18
server: 20
client: 23
server: 25
client: 28
server: 30
client: 33
server: 0
client: 3
server: 5
client: 8
server: 10
client: 13
server: 15
client: 18
server: 20
client: 23
server: 25
client: 28
server: 30
client: 33
server: 35
```

```
client: 38
server: 0
client: 3
server: 5
client: 8
server: 10
client: 13
server: 15
client: 18
server: 20
client: 23
server: 25
client: 28
server: 30
client: 33
server: 35
client: 38
server: 40
client: 43
server: 0
client: 3
server: 5
client: 8
server: 10
client: 13
server: 15
client: 18
server: 20
client: 23
server: 25
client: 28
server: 30
client: 33
server: 35
client: 38
server: 40
client: 43
server: 45
client: 48
- : int list = [0; 5; 10; 15; 20; 25; 30; 35; 40; 45]
─( 13:43:55 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # #quit ;;
carbon:SamplePrograms$ python3 generators.py
generating another square, this time for 0
printing the next square: 0
generating another square, this time for 1
printing the next square: 1
generating another square, this time for 2
printing the next square: 4
generating another square, this time for 3
printing the next square: 9
generating another square, this time for 4
printing the next square: 16
generating another square, this time for 5
printing the next square: 25
```

```
generating another square, this time for 6
printing the next square: 36
generating another square, this time for 7
printing the next square: 49
generating another square, this time for 8
printing the next square: 64
generating another square, this time for 9
printing the next square: 81
generating another square, this time for 10
printing the next square: 100
carbon:SamplePrograms$ utop ;;
-bash: syntax error near unexpected token `;;'
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 "streams.ml" ;;
type 'a stream = Cons of 'a * (unit -> 'a stream)
val ones : int stream = Cons (1, <fun>)
val head : 'a stream -> 'a = <fun>
val tail : 'a stream -> 'a stream = <fun>
val from : int -> int stream = <fun>
val nats : int stream = Cons (1, <fun>)
val take : int -> 'a stream -> 'a list = <fun>
val filter : ('a -> bool) -> 'a stream -> 'a stream = <fun>
val even : int -> bool = <fun>
val squares_from : int -> int stream = <fun>
val t1 : int list = [1; 4; 9; 16; 25; 36; 49; 64; 81; 100]
val squares : int stream = Cons (1, <fun>)
val zip : ('a -> 'b -> 'c) -> 'a stream -> 'b stream -> 'c stream = <fun>
val nats2 : int stream = Cons (1, <fun>)
val factorials : int stream = Cons (1, <fun>)
val non : ('a -> bool) -> 'a -> bool = <fun>
val multiple_of : int -> int -> bool = <fun>
val sift : int -> int stream -> int stream = <fun>
val sieve : int stream -> int stream = <fun>
val primes : int stream = Cons (2, <fun>)
-( 13:53:43 )-< command 1 >-------------------------------------------{ counter: 0 }-
utop # take 10 nats ;;
step 2
step 3
step 4
step 5
step 6
step 7
step 8
step 9
step 10
step 11
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
-( 13:53:49 )-< command 2 >-------------------------------------------{ counter: 0 }-
```

```
utop # take 10 nats ;;
step 2
step 3
step 4
step 5
step 6
step 7
step 8
step 9
step 10
step 11
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
─( 13:53:54 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 nats ;;
step 2
step 3
step 4
step 5
step 6
step 7
step 8
step 9
step 10
step 11
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
─( 13:54:03 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 nats ;;
step 2
step 3
step 4
step 5
step 6
step 7
step 8
step 9
step 10
step 11
- : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
─( 13:54:04 )─< command 5 >──────────────────────────────────────{ 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 # #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 filter : ('a -> bool) -> 'a stream -> 'a stream = <fun>
val even : int -> bool = <fun>
val squares_from : int -> int stream = <fun>
val squares : int stream = Cons (1, {contents = Thunk <fun>})
val zip : ('a -> 'b -> 'c) -> 'a stream -> 'b stream -> 'c stream = <fun>
val factorials : int stream = Cons (1, {contents = Thunk <fun>})
─( 14:03:40 )─< command 1 >──────────────────────────────────────{ counter: 0 }─
utop # take 4 (from 4) ;;
step 4
step 5
step 6
step 7
step 8
- : int list = [4; 5; 6; 7]
─( 14:03:48 )─< command 2 >──────────────────────────────────────{ counter: 0 }─
utop # take 5 nats ;;
step 2
step 3
step 4
step 5
step 6
- : int list = [1; 2; 3; 4; 5]
─( 14:04:04 )─< command 3 >──────────────────────────────────────{ counter: 0 }─
utop # take 8 nats ;;
step 7
step 8
step 9
- : int list = [1; 2; 3; 4; 5; 6; 7; 8]
─( 14:04:14 )─< command 4 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 primes ;;
Error: Unbound value primes
─( 14:05:00 )─< command 5 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 factorials ;;
step 10
- : int list = [1; 1; 2; 6; 24; 120; 720; 5040; 40320; 362880]
─( 14:05:45 )─< command 6 >──────────────────────────────────────{ counter: 0 }─
utop # take 10 factorials ;;
- : int list = [1; 1; 2; 6; 24; 120; 720; 5040; 40320; 362880]
─( 14:05:55 )─< command 7 >──────────────────────────────────────{ counter: 0 }─
utop # take 20 factorials ;;
step 11
step 12
step 13
step 14
step 15
step 16
step 17
step 18
step 19
```

```
step 20
- : int list =
[1; 1; 2; 6; 24; 120; 720; 5040; 40320; 362880; 3628800; 39916800; 479001600;
 6227020800; 87178291200; 1307674368000; 20922789888000; 355687428096000;
 6402373705728000; 121645100408832000]
─( 14:06:11 )─< command 8 >──────────────────────────────────────────────────(
─( 14:06:11 )─< command 8 >──────────────────────────────────────{ counter: 0 }─
utop #
```

```
┌───┬──────────────┬─────┬──────────┬──────────────┬──────┬─────────┬──────┬────
│Arg│Arith_status  │Array│ArrayLabels│Assert_failure│Big_int│Bigarray│Buffer│Call
└───┴──────────────┴─────┴──────────┴──────────────┴──────┴─────────┴──────┴────
ack│CamlinternalLazy│CamlinternalMod│Camlinterna│
└───┴────────────────┴───────────────┴───────────┘
```