```
Last login: Mon Mar 19 13:24:48 on ttys004
carbon:$ utop
```

```
elcome to utop version 2.0.2 (using OCaml version 4.06.0
```

```
Type #utop_help for help about using utop.

─( 15:46:10 )─< 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>
─( 15:46:10 )─< command 1 >──────────────────{ counter: 0 }─
utop # nats ;;
- : int stream = Cons (1, {contents = Thunk <fun>})
─( 15:46:13 )─< command 2 >──────────────────{ counter: 0 }─
utop # head nats ;;
- : int = 1
─( 15:46:18 )─< command 3 >──────────────────{ counter: 0 }─
utop # head (tail nats) ;;
step 2
- : int = 2
─( 15:46:42 )─< command 4 >──────────────────{ counter: 0 }─
utop # nats ;;
- : int stream =
Cons (1,
```

```
 {contents = Value (Cons (2, {contents = Thunk <fun>}))}
)
-( 15:46:50 )-< command 5 >-----------------{ counter: 0 }-
utop # take 2 nats ;;
step 3
- : int list = [1; 2]
-( 15:47:00 )-< command 6 >-----------------{ counter: 0 }-
utop # take 5 nats ;;
step 4
step 5
step 6
- : int list = [1; 2; 3; 4; 5]
-( 15:47:25 )-< command 7 >-----------------{ counter: 0 }-
utop # nats ;;
- : int stream =
Cons (1,
 {contents =
   Value
    (Cons (2,
      {contents =
        Value
         (Cons (3,
           {contents =
             Value
              (Cons (4,
                {contents =
                  Value
                   (Cons (5,
                     {contents =
                       Value
                        (Cons (6,
                          {contents = Thunk <fun>}))}))}
))}))}))})
-( 15:48:02 )-< 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>
val ones : int stream =
  Cons (1, {contents = Thunk <fun>})
─( 15:48:02 )─< command 9 >──────────────{ counter: 0 }─
utop # take 10 ones ;;
- : int list = [1; 1; 1; 1; 1; 1; 1; 1; 1; 1]
─( 15:53:58 )─< command 10 >─────────────{ counter: 0 }─
utop # ones ;;
- : int stream =
Cons (1,
 {contents =
   Value
    (Cons (1,
      {contents =
        Value
         (Cons (1,
           {contents =
             Value
              (Cons (1,
                {contents =
                  Value
                   (Cons (1,
                     {contents =
                       Value
                        (Cons (1,
                          {contents =
                            Value
                             (Cons (1,
                               {contents =
                                 Value
```

```
                                  (Cons (1,
                                    {contents =
                                      Value
                                       (Cons (1,
                                         {contents =
                                           Value
                                            (Cons (1,
                                              {contents =
                                                Value
                                                 (Cons (1,
                                                   {contents
                                                   =
                                                   Value
                                                   (Cons (1,
                                                   {contents
                                                   =
                                                   Thunk
                                                   <fun>})) })
)})))})))})))})))})))})))})))})
–( 15:54:12 )–< command 11 >————————————{ counter: 0 }–
utop # take 15 ones ;;
- : int list =
[1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1]
–( 15:54:16 )–< command 12 >————————————{ counter: 0 }–
utop # ones ;;
- : int stream =
Cons (1,
 {contents =
   Value
    (Cons (1,
      {contents =
        Value
         (Cons (1,
           {contents =
             Value
              (Cons (1,
                {contents =
```

```
Value
 (Cons (1,
   {contents =
     Value
      (Cons (1,
        {contents =
          Value
           (Cons (1,
             {contents =
               Value
                (Cons (1,
                  {contents =
                    Value
                     (Cons (1,
                       {contents =
                         Value
                          (Cons (1,
                            {contents
                            =
                            Value
                            (Cons (1,
                            {contents
                            =
                            Value
                            (Cons (1,
                            {contents
                            =
                            Value
                            (Cons (1,
                            {contents
                            =
                            Value
                            (Cons (1,
```

```
                                                {contents
                                                =
                                                Value
                                                (Cons (1,
                                                {contents
                                                =
                                                Thunk
                                                <fun>})))})
)})))})))})))})))})))})))})))})))})))})))})))})))})
─( 15:57:12 )─< command 13 >─────────────{ counter: 0 }─
utop # take 5 ones ;;
- : int list = [1; 1; 1; 1; 1]
─( 15:57:14 )─< command 14 >─────────────{ counter: 0 }─
utop # ones ;;
- : int stream =
Cons (1,
 {contents =
   Value
    (Cons (1,
      {contents =
        Value
         (Cons (1,
           {contents =
             Value
              (Cons (1,
                {contents =
                  Value
                   (Cons (1,
                     {contents =
                       Value
                        (Cons (1,
                          {contents =
                            Value
                             (Cons (1,
                               {contents =
                                 Value
                                  (Cons (1,
```

```
                                              {contents =
                                                Value
                                                 (Cons (1,
                                                   {contents =
                                                     Value
                                                      (Cons (1,
                                                        {contents
                                                        =
                                                        Value
                                                        (Cons (1,
                                                        {contents
                                                        =
                                                        Value
                                                        (Cons (1,
                                                        {contents
                                                        =
                                                        Value
                                                        (Cons (1,
                                                        {contents
                                                        =
                                                        Value
                                                        (Cons (1,
                                                        {contents
                                                        =
                                                        Value
                                                        (Cons (1,
                                                        {contents
                                                        =
                                                        Value
                                                        (Cons (1,
                                                        {contents
                                                        =
                                                        Thunk
                                                        <fun>}))})
)}))}))}))}))}))}))}))}))}))}))}))}))})
─( 15:57:27 )─< 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>
─( 15:57:30 )─< 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>
```

```
step 2
val all_evens : int stream =
  Cons (2, {contents = Thunk <fun>})
─( 16:02:04 )─< command 17 >──────────────{ counter: 0 }─
utop # take 5 all_evens ;;
step 3
step 4
step 5
step 6
step 7
step 8
step 9
step 10
step 11
step 12
- : int list = [2; 4; 6; 8; 10]
─( 16:02:56 )─< command 18 >──────────────{ counter: 0 }─
utop # nats ;;
- : int stream =
Cons (1,
 {contents =
   Value
    (Cons (2,
      {contents =
        Value
         (Cons (3,
           {contents =
             Value
              (Cons (4,
                {contents =
                  Value
                   (Cons (5,
                     {contents =
                       Value
                        (Cons (6,
                          {contents =
                            Value
```

```
                                (Cons (7,
                                  {contents =
                                    Value
                                     (Cons (8,
                                       {contents =
                                         Value
                                          (Cons (9,
                                            {contents =
                                              Value
                                               (Cons (10,
                                                 {contents
                                                 =
                                                 Value
                                                 (Cons
                                                 (11,
                                                 {contents
                                                 =
                                                 Value
                                                 (Cons
                                                 (12,
                                                 {contents
                                                 =
                                                 Thunk
                                                 <fun>}))})
)}))}))}))}))}))}))}))}))}))})
─( 16:03:38 )─< command 19 >─────────────{ 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_evens : int stream =
  Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
─( 16:03:53 )─< command 20 >──────────────{ counter: 0 }─
utop # take 5 (map (fun x -> x + 100) nats) ;;
step 3
step 4
step 5
step 6
- : int list = [101; 102; 103; 104; 105]
─( 16:12:53 )─< 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_evens : int stream =
  Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
val zip :
  ('a -> 'b -> 'c) ->
  'a stream -> 'b stream -> 'c stream = <fun>
─( 16:13:07 )─< command 22 >──────────────{ 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_evens : int stream =
  Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
val zip :
  ('a -> 'b -> 'c) ->
  'a stream -> 'b stream -> 'c stream = <fun>
val all_evens_v2 : int stream =
```

```
  Cons (2, {contents = Thunk <fun>})
-( 16:15:16 )-< command 23 >----------------{ counter: 0 }-
utop # take 10 all_evens_v2 ;;
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]
-( 16:16:33 )-< 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_evens : int stream =
  Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
```

```
val zip :
  ('a -> 'b -> 'c) ->
  'a stream -> 'b stream -> 'c stream = <fun>
val all_evens_v2 : int stream =
  Cons (2, {contents = Thunk <fun>})
val all_evens_v3 : int stream =
  Cons (2, {contents = Thunk <fun>})
─( 16:16:38 )─< command 25 >──────────────{ 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]
─( 16:17:10 )─< command 26 >──────────────{ counter: 0 }─
utop # #use "lazy.ml";;
File "lazy.ml", line 100, characters 2-4:
Error: Syntax error: ')' expected
File "lazy.ml", line 99, characters 11-12:
Error: This '(' might be unmatched
─( 16:17:13 )─< command 27 >──────────────{ 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_evens : int stream =
  Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
val zip :
  ('a -> 'b -> 'c) ->
  'a stream -> 'b stream -> 'c stream = <fun>
val all_evens_v2 : int stream =
  Cons (2, {contents = Thunk <fun>})
val all_evens_v3 : int stream =
  Cons (2, {contents = Thunk <fun>})
File "lazy.ml", line 99, characters 15-47:
Error: This expression has type int stream
       but an expression was expected of type
         int stream lazee = int stream hidden ref
─( 16:25:09 )─< 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_evens : int stream =
  Cons (2, {contents = Thunk <fun>})
val map : ('a -> 'b) -> 'a stream -> 'b stream = <fun>
val zip :
  ('a -> 'b -> 'c) ->
  'a stream -> 'b stream -> 'c stream = <fun>
val all_evens_v2 : int stream =
  Cons (2, {contents = Thunk <fun>})
val all_evens_v3 : int stream =
  Cons (2, {contents = Thunk <fun>})
val factorials : int stream =
  Cons (1, {contents = Thunk <fun>})
─( 16:25:21 )─< 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]
─( 16:25:46 )─< command 30 >──────────────{ counter: 0 }─
utop #
```

Arg	Array	ArrayLabels	Assert_failure	Bigarray	Buffer	B