

Objective Caml version 3.06

```
# sqrt;;  
- : float -> float = <fun>  
  
# sin;;  
- : float -> float = <fun>  
  
# float;;  
- : int -> float = <fun>  
  
# +;;  
Characters 0-1:  
  +;;  
  ^  
Syntax error  
  
# (+);;  
- : int -> int -> int = <fun>  
  
# (+) 2 3;;  
- : int = 5  
  
# (+) (2,3);;  
Characters 5-8:  
  (+) (2,3);;  
      ^^^  
This expression has type int * int but is here used with type int  
  
# function (x: int) -> x + 1;;  
- : int -> int = <fun>  
  
# function x -> x + 1;;  
- : int -> int = <fun>  
  
# function (x,y) -> x < y;;  
- : 'a * 'a -> bool = <fun>  
  
# let succ = function x -> x + 1;;  
val succ : int -> int = <fun>  
  
# succ 3;;  
- : int = 4  
  
# succ succ 3;;  
Characters 0-4:  
  succ succ 3;;  
  ^^^^^  
This function is applied to too many arguments  
  
# succ (succ 3);;  
- : int = 5  
  
# let pi = 2. *. asin 1.;;  
val pi : float = 3.14159265359  
  
# let circle_area = function (r: float) -> pi *. r *. r;;  
val circle_area : float -> float = <fun>  
  
# circle_area 4.0;;  
- : float = 50.2654824574  
  
# let circle_area = function r -> pi *. r *. r;;  
val circle_area : float -> float = <fun>  
  
# let circle_area = fun r -> pi *. r *. r;;  
val circle_area : float -> float = <fun>  
  
# let circle_area (r: float) = pi *. r *. r;;  
val circle_area : float -> float = <fun>  
  
# let circle_area r = pi *. r *. r;;  
val circle_area : float -> float = <fun>  
  
# let rectangle_area (x, y) = x *. y;;  
val rectangle_area : float * float -> float = <fun>  
  
# rectangle_area (2.5, 3.);;
```

```
- : float = 7.5

# let rectangle_area = fun x -> fun y -> x *. y;;
val rectangle_area : float -> float -> float = <fun>

# let rectangle_area = fun x y -> x *. y;;
val rectangle_area : float -> float -> float = <fun>

# let rectangle_area x y = x *. y;;
val rectangle_area : float -> float -> float = <fun>

# rectangle_area (2.5, 3.);;
Characters 16-23:
  rectangle_area (2.5, 3.);;
                ^^^^^^^
This expression has type float * float but is here used with type float

# rectangle_area 2.5 3.;;
- : float = 7.5

# rectangle_area 2.5;;
- : float -> float = <fun>

# let even (n: int) = n mod 2 = 0;;
val even : int -> bool = <fun>

# even 3;;
- : bool = false

# type int_set = int -> bool;;
type int_set = int -> bool

# let even: int_set = fun (n: int) -> n mod 2 = 0;;
val even : int_set = <fun>

# let odd: int_set = fun (n: int) -> n mod 2 <> 0;;
val odd : int_set = <fun>

# let positive: int_set = fun (n: int) -> n > 0;;
val positive : int_set = <fun>

# let negative: int_set = fun (n: int) -> n < 0;;
val negative : int_set = <fun>

# let negative = fun (n: int) -> n < 0;;
val negative : int -> bool = <fun>

#
```