**utop #** let x =45;;

val x : int = 45 **─( 12:21:52 )─< command 6 >───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** x

;;

- : int = 45 **─( 12:23:57 )─< command 7 >───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** let f y = x+y ;;

val f : int -> int = <fun> **─( 12:24:03 )─< command 8 >───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** f 3;;

- : int = 48 **─( 12:24:14 )─< command 9 >───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** let x=12;;

val x : int = 12 **─( 12:24:24 )─< command 10 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** f 3;;

- : int = 48 **─( 12:24:40 )─< command 11 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** x;;

- : int = 12

**utop #** let x = false;;

val x : bool = false **─( 12:25:01 )─< command 13 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** x;;

- : bool = false **─( 12:27:43 )─< command 14 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** f 3;;

- : int = 48

This is called shadowing.

Function saves the environment. That is also why sometimes functions are called closures

**utop #** let add x y = x+y;;

val add : int -> int -> int = <fun>

**─( 12:33:04 )─< command 19 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─**

**utop #** let add = function x -> (function y -> x+y);;

val add : int -> int -> int = <fun> **─( 12:33:09 )─< command 20 >────────**

Use Modules

U can use it with the dot notation.

List.map

**utop #** List.map;;

- : ('a -> 'b) -> 'a list -> 'b list = <fun> **─( 12:34:43 )─< command 21**

**────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** let map = 34;;

val map : int = 34 **─( 12:51:38 )─< command 23**

**counter: 0 }─utop #** map;;

- : int = 34 **─( 12:51:55 )─< command 25 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** List.map(fun x-> x+1) [1;2;3;map];;

- : int list = [2; 3; 4; 35]

**utop #** let f (l:int list) = 3;;

val f : int list -> int = <fun> **─( 12:58:37 )─< command 30 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** f[1;2;3];;

- : int = 3 **─( 13:13:25 )─< command 31 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** f[true;false];;

Error: This expression has type bool but an expression was expected of type int

**utop #** [1;"hi";34;"bye"];;

Error: This expression has type bytes but an expression was expected of type int **─( 13:13:47 )─< command 33 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─**

**utop #** [(1,"hi");(24,"bye")];;

* : (int \* bytes) list = [(1, "hi"); (24, "bye")]

**utop #** type intorstrinf = I of int | S of string;;

type intorstrinf = I of int | S of bytes **─( 13:19:53 )─< command 37 >──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────{ counter: 0 }─utop #** [I 1; S "hi"; S "bye"; I 34];;

- : intorstrinf list = [I 1; S "hi"; S "bye"; I 34]