```
> rlwrap sml
Standard ML of New Jersey v110.74 [built: Thu Aug 16 11:25:45 2012]
- (* Tutorial of basic ML types, values, and operators *)
- (* ML comments go between parentheses and asterisks. *)

- 5+5;
val it = 10 : int

- 3-5;  (* notice negative sign in result is written with a tilde *)
val it = ~2 : int

- #"m";
val it = #"m" : char

- "yo" ^ "yo";
val it = "yoyo" : string

- (* plus and minus operators are only defined on ints and reals *)
- "yo" + "yo";
stdIn:6.6 Error: overloaded variable not defined at type
  symbol: +
  type: string

- 3 + 3.5; (* both operands must be ints or both must be reals *)
stdIn:1.1-2.3 Error: operator and operand don't agree [literal]
  operator domain: int * int
  operand:         int * real
  in expression:
    3 + 3.5

- 4.2 + ~1.0;
val it = 3.2 : real

- 3 + #"A"; (* these sorts of expressions work in C but not ML *)
stdIn:9.1-9.9 Error: operator and operand don't agree [literal]
  operator domain: int * int
  operand:         int * char
  in expression:
    3 + #"A"

- true andalso false;
val it = false : bool

- not(5=3);  (* notice equality test is only one equal sign *)
val it = true : bool

- 3>5 orelse 5<=3;
val it = false : bool

- 3<>4;  (* inequality test *)
val it = true : bool

- if 3=5 then false<>false else not true;
val it = false : bool
```

```
- (* parentheses can be put around any expression *)
- ((if (3=(5)) then (false<>false) else (not true)));
val it = false : bool

- if 5 then 4 else 3; (* "if" expression must have boolean type *)
stdIn:1.1-9.7 Error: test expression in if is not of type bool
[literal]
  test expression: int
  in expression:
    if 5 then 4 else 3

- (* "then" and "else" expressions can have any type, *)
- (*   but they must have the same type *)
- if true then true else 4;
stdIn:1.1-10.5 Error: types of if branches do not agree [literal]
  then branch: bool
  else branch: int
  in expression:
    if true then true else 4

- (* "if" expressions must have both "then" and "else" expressions *)
- (* there is no such thing as if-then expressions in ML *)
- if true then 3;
= (* SML/NJ responds with '=' because it expects more input *)
= (* this is a mistake, so kill this expression with ctrl-c *)
= <ctrl-c>
Interrupt
- (* we are now back, ready to input more expressions *)


- (* expressions can be nested *)
- 5 + (if true then 3 else 4);
Question for class: How does SML/NJ respond at this point?




- if (if 2=2 then 2=3 else 2=2) then (if 2=2 then 4 else 5)
= (* SML/NJ responds with '=' because it expects more input*)
= else (if 2=3 then 6 else 7);
Question for class: How does SML/NJ respond at this point?





- (* quit with ctrl-d *)
- <ctrl-d>
>
```

```
> rlwrap sml
Standard ML of New Jersey v110.74 [built: Thu Aug 16 11:25:45 2012]
- (* Tutorial of top-level variables, tuples, and lists in ML *)

- (* Define top-level variables (i.e., globals) with "val" keyword *)
- val v1 = "hi ";
val v1 = "hi " : string

- (* Called "top-level" because not defined within another construct *)
- (* E.g., a var defined within a function is not a top-level var *)

- val v2 = "there";
val v2 = "there" : string

- v1 ^ v2;
val it = "hi there" : string

- (* Think of the following as creating a new variable called v1 *)
- (* Don't think of the following as updating the value of the old v1*)
- val v1 = 5;
val v1 = 5 : int

- (* Technically, we have two variables called v1 defined now *)
- (* But the new definition overshadows the old one *)
- v1 ^ v2;
stdIn:5.1-5.8 Error: operator and operand don't agree [tycon mismatch]
  operator domain: string * string
  operand:         int * string
  in expression:
    v1 ^ v2

- (* Forgetting the "val" keyword changes the expression's meaning *)
- v1 = 3;
val it = false : bool

- (* A tuple is a comma-separated, finite sequence of expressions
     between parentheses (must have at least two expressions).
     The order of expressions within a tuple matters:
        (3,4) is different than (4,3).
     Expressions in a tuple can have different types:
        (3, 4.5, true) is a tuple of type int*real*bool   *)
- val t1 = (3, 4.5, true);
val t1 = (3,4.5,true) : int * real * bool

- (* Can put general expressions in tuples and have nested tuples *)
- val t2 = (if 2=2 then 3 else 4, (false, 5.6));
val t2 = (3,(false,5.6)) : int * (bool * real)

- (* There are no tuples with zero components. *)
- (* However, () is a special value of type unit *)
- ();
val it = () : unit
- (* Unit is an interesting type; only one value has type unit *)
- (* A value is anything that can be a final result of a program *)
- if true then (if false then () else ()) else ();
val it = () : unit
```

```
- (* Even the bool type is inhabited by two values, true and false *)
Question for class: How many values does type int have?


- (* A list is a comma-separated, finite sequence of expressions
     between brackets.
     Lists, unlike tuples, may have only 0 or 1 elements.
     The order of expressions within a list matters:
        [3,4] is different than [4,3].
     Unlike tuples, expressions in a list must have the same type:
        [3, 4, 5] is a list of type int list *)
- val L = [3,4,5];
val L = [3,4,5] : int list

- val L2 = [3,4.5,true];
stdIn:4.10-4.22 Error: operator and operand don't agree [tycon
mismatch]
  operator domain: real * real list
  operand:         real * bool list
  in expression:
    4.5 :: true :: nil

- (* list concatenation *)
- val L = L @ [2];
val L = [3,4,5,2] : int list

- val L = [2] @ L;
val L = [2,3,4,5,2] : int list

- (* prepending to a list with the cons operator *)
- 1 :: L;
val it = [1,2,3,4,5,2] : int list

- (* Empty list can be written in two ways *)
- (* Empty list has type 'a list, meaning that ML knows it's a list,
=    but it could be any type of list (e.g., int list or bool list) *)
- [];
val it = [] : 'a list

- nil;
val it = [] : 'a list

- val L = 1::2::3::8::[];
val L = [1,2,3,8] : int list

- (* First element in a list is the head; all others are the tail. *)

- (* Make L the head of some lists. *)
- L::[9,10]::[11]::[];
val it = [[1,2,3,8],[9,10],[11]] : int list list

- L::nil;
val it = [[1,2,3,8]] : int list list
```