Data structures

```
(*** DATA STRUCTURES ***)

(* LIST *)

(* square brackets, items are semi-colon separated *)
(* dynamically allocated space, size can be casually changed *)
(* dynamically allocated space, can store elements of different datat types *)

let my_list = [1;2;3];; (* of type "int list" *)

(* LIST METHODS *)

(* LIST INDEXING *)

List.nth my_list 1;; (* evaluates to integer 2, the second element in the list of index 1 *)

(* LIST MAP *)

(* List.map() calls an anonymous function that is user-defined *)

(* List.map() applies the given function to each iteration variablein the list *)

List.map(fun x -> x * 2)  [1;2;3];; (* this should evaulate to [2;4;6] *)

(* LIST FILTER *)

(* List.filter() also calls an anonymous function that is user-defined *)

(* List.filter() applies the specified conditonal check as a function to the list, and only those that pass said check are remaining in the list *)

List.filter (fun x -> x mod 2 = 0) [1;2;3;4];; (* this should evaulate to [2;4] *)

(* ADDING ELEMENTS *)

(* add an item to the FRONT of a list with the :: constructor which is often referred to as a "cons" *)

1 :: [2;3];; (* evaluates to [1;2;3] *)

(* TUPLES *)

(* (* optionally surrounded by *) round brackets, items are comma separated *)

let my_tuple = 3, 4;; (* of type "int * int" *)
let my_other_tuple = (5,6,7);; (* this is a much clearer more approved syntax *)

(* warning to not separate list items by commas, otherwise you'll accidentally create a list with a tuple inside *)

let bad_list = [1,2];; (* this shit becomes [(1,2)] *)

(* ARRAYS *)

(* "[| |]" surrounded, items are semicolon seperated *)
(* statically allocated space, size of array declared at initialization * )
(* statically allocated space, arrays can only contain same data type *)

let my_array = [| 1;2;3 |];;

(* ARRAY INDEXING *)

my_array.(0);; (* this evaluates to the integer 1, the first element of the array with index 0 *)
```

Strings and Characters