**Ocaml**

**Variable**

let *var* = …;;

let *var*=

let *var1*= [*element1*; *element2*;…]

and *var2*= …

in …;;

**List**

List.rev *list*];;

List.flatten *list*;;

List.map *function list*;;

List.filter *function list*;;

**Type Define**

type *name* = *value1*|*value2*|*value3*…;;

type *name* = *name* of *value\_contructor1*|*value\_contructor2*|…;;

type binary\_tree = Leaf | Node of int \* binary\_tree \* binary\_tree;;

ex\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

type point =

Cartesian of (float \* float)

| Polar of (float \* float);;

let x = Cartesian (0.5, 0.5);;

match x with

Cartesian (0.5, 0.5) -> true

| \_ -> false;;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Control Flow**

if .. then …

else if … then

else …

**Match**

match … with

… -> …

| … -> …

|(*element1*, *element2*) -> …

|*head*::*tail* -> …

| \_ -> …

match … with

… when *conditions* -> …

| …

**Function**

let *var* = fun *para* -> …;;

anonymous function

let *function\_name para1* *para2*= …;;

let *function\_name* = function

|0 -> ..

| \_ -> … ;;

**Recursion**

let rec *function\_name* x = …;;

let rec fact (x:int) : int =

if x = 0 then 1

else x \* (fact (x-1));;

**Conditional**

let *FunctionName Para* =

If … then …

else …;;

= -- Traditional equality

== -- Compares physical location (and deprecated)

**General**

**Compiler vs Interpreter**

**Compiler**

Faster

**Interpreter**

Portable

Simpler

Better error checking

**Just in time compiling**

Compile each function when it is actually called

Compile only those pieces seem to be called frequently

**Profiling**

Run program first, find the piece that is called frequently, compile it