BILAN - Ocaml

[Session 1](#_cokxr072yjod)

[1 - Manipulations des entiers](#_gcce9we66a94)

[2 - Manipulations des strings](#_q5tmea8co7y2)

[3 - List et Arrays](#_vy7pv6r1aqfo)

[4 - Manipulations des listes](#_4hhi7m198jei)

[List.rev[....] : retourner une liste dans l’ordre inverse](#_10xmtexcnd3m)

[List.hd[....] : retourner le premier élément de la liste](#_nkur4dtemgs3)

[5 - Manipulation des tableaux](#_7ycehlq9kd5o)

[Array.append[|....|][|....|] : concaténer deux tableaux de même type](#_ahm6du2bd071)

[Array.get[|....|] i : obtenir le i-ième élément du tableau](#_4n12x4r6g8ja)

[[|....|].(i) : obtenir le i-ième élément du tableau](#_y41s0fo0iwgu)

[6 - Manipulations des “String”](#_nagqigc289rz)

[String.make n ‘a’ : retourner n fois le caractère a](#_f3hfey8ea2u0)

[^ : relier deux chaînes](#_hmlzmg9fkdne)

[String.length : retourner la longueur d’une chaîne](#_f4u2dj4d5wro)

[String.lowercase “AA” : transformer la chaîne en minuscule](#_hl8wpvja75uo)

[String.concat “aa” [....] : mettre “aa” avant chaque élément de la liste](#_rkp0d67r7k6g)

[int\_of\_string : transformer le type string en integer](#_y9ildxtmbmeq)

[7 - Manipulations des lignes](#_rhjw1t2gf2xh)

[(....) : retourner une ligne contenante de différentes types d’éléments](#_4qa1jjtd1av)

[fst(....) : retourner le premier élément de la ligne](#_puujll9f7wic)

[snd(....) : retourner le deuxième éléments de la ligne](#_p8i9k3wsubtw)

[Session 2](#_gxhtpqw8bh25)

[1 - Variables](#_ph7s56sl9erg)

[2 - Mutables variables](#_wr960qk2u3xg)

[let x = ref nb : Changer la valeur qui est déjà définie](#_sbnlnhfj31zm)

[x:=nb\*nb ou nb/nb ou …. : Changer la valeur qui est déjà définie](#_5o6a90i947og)

[! : transformer la valeur de type réf en int](#_euvrw9gbaaaq)

[print\_int nb : afficher la valeur de nb](#_w2m3zhuaxgh0)

[print\_string “aa” : afficher la valeur de la chaîne](#_45k9u6skhmcx)

[print\_newline : afficher une nouvelle ligne](#_mkfh4fx4gv0v)

[Printf.printf : afficher une chaîne](#_8fu8x17htxzw)

[3 - Manipulations des boucles](#_cb7nj650mwd8)

[let xl = ref[] : mettre xl en référence d’une liste vide](#_ej9qe1ieilsz)

[for i = 1 to 10 do ….done : syntaxe d’une boucle](#_1xlpwb3ayy9u)

[downto : écrire une boucle avec i dans l’ordre décroisante](#_jatn6pla940s)

[4 - Conditions](#_j9gvf4y2ftsn)

[<> : comparer deux entiers sont différents ou pas](#_543krmb14qyt)

[if condition then a else b : syntaxe de if then else](#_nh7959wi1zh5)

[6 - While loop](#_u19wfz2ui4s8)

[while condition do ….done : syntaxe de while loop](#_7f9i2zwko0tb)

[Session 3](#_rh27fptv0s7r)

[1 - Fonction avec un argument](#_j99y4wpvd2j1)

[2 - Fonction avec plusieurs arguments](#_tkqhwsyejfrt)

[3 - Retourner plusieurs valeurs](#_ntwv42ib3qvf)

[4 - Application partielle](#_xj7xrdqwkoeh)

[4 - Fonction anonyme](#_hxribv2oo5cd)

[5 - Iterateurs](#_m046lsssetra)

[map : appliquer une fonction ) tous les éléments d’une liste pour donner la liste des résultats](#_nyb8hz1ueajo)

[fold\_left : faire ajouter un entier à une liste de gauche à droit](#_8cgz63lv0akj)

[List.iter : appliquer une fonction en itération pour une liste](#_7r1ospxx2rge)

[Session 4](#_qj3vxhk753a2)

[1 - match….with pour les entiers](#_vnl3r1yumv3e)

[2 - match….with pour la caractère](#_mf50xauoydxv)

[Char.uppercase : transformer les minuscules en majuscules](#_jvprlbca43pd)

[failwith “....” : lancer une exception](#_2ud1b8hnud5g)

[as : donner le nom au paramètre](#_jmp8o8vkjvny)

[3 - match….with pour les lignes](#_zhy8bwxvjyz2)

[when : vérifier la condition](#_r0hrbyhv6y4c)

[4 - match….with pour une liste](#_i77bre24or1g)

[h :: t : h est le preimier élément de la liste, t sont les restes éléments](#_slrzm4r0jyui)

[:: : permet de concaténer un élément dans une liste](#_slrzm4r0jyui)

[[ \_ ] : liste contenante un seul élément](#_r0s6d3eja79p)

[\_::e::\_ : deuxième élément de la liste](#_r0s6d3eja79p)

[5 - match….with pour les tableaux](#_3yjuzzjcu3ws)

[[|\_:\_|] : tableau contenant deux éléments](#_rwhnlijg8jo7)

[6 - exhaustivité](#_1ql4qmcdab0a)

[Session 5](#_jfsth26ksuve)

[1 - let keyword](#_qxvwa7cmtojh)

[2 - parenthèses](#_re6zqw2u7d5m)

[begin….end…. : d’abord exécuter la partie begin et exécuter la partie end après](#_qx45o7mj4svu)

# 

# 

# 

# 

# Session 1

## 1 - Manipulations des entiers

# 1;;

- : int = 1

Retourner un entier

# 1+2;;

- : int = 3

Retourner la somme des deux entiers

## 2 - Manipulations des strings

# "Mary";;

- : string = "Mary"

Retourner une chaîne de caractère

# 'a';;

- : char = 'a'

Retourner un caractère

# 'aa';;

Error: Syntax error

‘ ’ ne peut que utiliser pour un caractère, pas pour plusieurs

## 3 - List et Arrays

# [1;2;3;4];;

- : int list = [1; 2; 3; 4]

Retourner une liste des entiers

# ["aa";"bb"];;

- : string list = ["aa"; "bb"]

Retourner une liste de caractères

# [|1;2;3;4|];;

- : int array = [|1; 2; 3; 4|]

Retourner un tableau d’entiers

# [|"aa";"bb"|];;

- : string array = [|"aa"; "bb"|]

Retourner un tableau des chaînes de caractères

## 4 - Manipulations des listes

### **List.rev[....]** : retourner une liste dans l’ordre inverse

# List.rev[1;2;3;4];;

- : int list = [4; 3; 2; 1]

Retourner une liste de la liste dans l’ordre décroissante

### List.hd[....] : retourner le premier élément de la liste

# List.hd[1;2;3;4];;

- : int = 1

Retourner le premier entier de la liste

# List.hd["aa";"bb"];;

- : string = "aa"

Retourner la première chaîne de caractères de la liste

## 5 - Manipulation des tableaux

### Array.append[|....|][|....|] : concaténer deux tableaux de même type

# Array.append[|1;2;3|][|4;5;6|];;

- : int array = [|1; 2; 3; 4; 5; 6|]

Concaténer deux tableaux de type “int”

# Array.append[|1;2;3|][|2;3;4|];;

- : int array = [|1; 2; 3; 2; 3; 4|]

Concaténer deux tableaux de type “int” contenant des répétitions

# Array.append[|"aa";"bb"|][|"cc";"dd"|];;

- : string array = [|"aa"; "bb"; "cc"; "dd"|]

Concaténer deux tableaux de type “string”

### Array.get[|....|] i : obtenir le i-ième élément du tableau

# Array.get[|1;2;3|]2;;

- : int = 3

Retourner le deuxième élément du tableau

### [|....|].(i) : obtenir le i-ième élément du tableau

# [|1;2;3|].(2);;

- : int = 3

Retourner le deuxième élément du tableau

## 6 - Manipulations des “String”

### String.make n ‘a’ : retourner n fois le caractère a

# String.make 10 'a';;

- : string = "aaaaaaaaaa"

Retourner 10 fois le caractère a

### ^ : relier deux chaînes

# "aa" ^ "bb";;

- : string = "aabb"

Relier les chaînes “aa” et “bb”

### String.length : retourner la longueur d’une chaîne

# String.length "aaaa";;

- : int = 4

Retourner la longueur de la chaîne “aaaa”

### String.lowercase “AA” : transformer la chaîne en minuscule

# String.lowercase "AA";;

- : string = "aa"

Retourner la chaîne “AA” en minuscule

### String.concat “aa” [....] : mettre “aa” avant chaque élément de la liste

# String.concat "/" ["";"usr";"local";"bin"];;

- : string = "/usr/local/bin"

Mettre la chaîne “/” avant chaque élément de la liste

### int\_of\_string : transformer le type string en integer

# let a = "11";;

val a : string = "11"

# int\_of\_string a;;

- : int = 11

Transformer la chaîne des entiers en integer

## 7 - Manipulations des lignes

### (....) : retourner une ligne contenante de différentes types d’éléments

# ("A",1,true);;

- : string \* int \* bool = ("A", 1, true)

Retourner une ligne contenant de différentes types d’éléments

### fst(....) : retourner le premier élément de la ligne

# fst("aa",true);;

- : string = "aa"

Retourner le première élément

### snd(....) : retourner le deuxième éléments de la ligne

# snd("aa",true);;

- : bool = true

Retourner le deuxième élément

# Session 2

## 1 - Variables

# let a = 3 \* 5;;

val a : int = 15

Retourner la valeur a, let permet de donner un nom au résultat de calculation

# a;;

- : int = 15

Retourner la valeur de a

# let b = a+1;;

val b : int = 16

Calculer le résultat de b qui fait a + 1

## 2 - Mutables variables

# x<-x+1;;

Error: Unbound instance variable x

On ne peut pas changer la valeur qui a été donné un nom après sa définition

### let x = ref nb : Changer la valeur qui est déjà définie

# let x = ref 42;;

val x : int ref = {contents = 42}

Retourner la valeur changée

### x:=nb\*nb ou nb/nb ou …. : Changer la valeur qui est déjà définie

val x : int ref = {contents = 42}

# x := 100/4;;

- : unit = ()

Retourner la valeur changée

# x;;

- : int ref = {contents = 25}

Vérifier que la valeur a été changée

### ! : transformer la valeur de type réf en int

# let y = !x+1;;

val y : int = 26

Calculer la valeur a avec “!”

### print\_int nb : afficher la valeur de nb

# print\_int 3;;

3- : unit = ()

# print\_int !x;;

25- : unit = ()

Retourner la valeur de x en utilisant “!”

### print\_string “aa” : afficher la valeur de la chaîne

# print\_string "Hello";;

Hello- : unit = ()

### print\_newline : afficher une nouvelle ligne

# print\_newline;;

- : unit -> unit = <fun>

### Printf.printf : afficher une chaîne

# Printf.printf "x = %d. Bye %s\n" !x "John";;

x = 25. Bye John

- : unit = ()

## 3 - Manipulations des boucles

### let xl = ref[] : mettre xl en référence d’une liste vide

# let xl = ref[];;

val xl : '\_a list ref = {contents = []}

Retourner la liste ref

### for i = 1 to 10 do ….done : syntaxe d’une boucle

# for i = 1 to 10 do

xl := i::!xl;

done;

xl;;

- : int list ref = {contents = [10; 9; 8; 7; 6; 5; 4; 3; 2; 1]}

Retourner la liste contenante les chiffres concaténer dans l’ordre de 1 à 10

Attention : Élément ajouté dans la liste sont toujours avant les éléments anciens

### downto : écrire une boucle avec i dans l’ordre décroisante

# for i = 10 downto 1 do

xl:=i::!xl;

done;;

- : unit = ()

# xl;;

- : int list ref = {contents = [1; 2; 3; 4; 5; 6; 7; 8; 9; 10]}

Retourner la liste contenante les chiffres concaténer dans l’ordre de 10 à 1

## 4 - Conditions

# 1>2;;

- : bool = false

Retourner le résultat d’un booléen

# "aaaa"<"aaaaa";;

- : bool = true

Comparer la longueur des deux chaînes

# "3"<"22";;

- : bool = false

# "3"<"12";;

- : bool = false

# "3"<"33";;

- : bool = true

Comparer deux chaînes de caractères des entiers

Attention : comparer tout d’abord la première chiffre de la chaîne, si pareil, comparer la longueur, le plus long est le plus grand

# 1=1;;

- : bool = true

Comparer deux entiers

### <> : comparer deux entiers sont différents ou pas

# 1<>1;;

- : bool = false

1 est différente de 1 -> false

# "1"=1;;

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

string

On ne peut pas conparer deux éléments de types différents

# 1 = int\_of\_string "1";;

- : bool = true

Transformer string en integer

5 - if then else

### if condition then a else b : syntaxe de if then else

# let a = 1 and b = 2;;

val a : int = 1

val b : int = 2

mettre a en valeur 1 et b en valeur 2

# Printf.printf "min(%d,%d) = %d\n" a b

(if a < b then a else b);;

min(1,2) = 1

- : unit = ()

Chercher la valeur le plus petite entre a et b

# let z = if a < 100 then begin

print\_string "return at least 100 !";

print\_newline ();

100

end else a;;

return at least 100 !

val z : int = 100

Si a < 100 retourner “return at least 100” et “100” sinon retourner a

## 6 - While loop

### while condition do ….done : syntaxe de while loop

# let x = ref 24;;

val x : int ref = {contents = 24}

mettre e en valeur ref 24

# while !x > 20 do

print\_int !x; print\_newline ();

x := !x - 2

done ;;

24

22

- : unit = ()

si la valeur de x est > 20 retourner x et faire x - 2 jusqu’à x < 20

# Session 3

## 1 - Fonction avec un argument

# let incr n = n+1;;

val incr : int -> int = <fun>

# let x = 2;;

val x : int = 2

# incr 2;;

- : int = 3

Incrémenter à n

# let plus\_and\_divide (x,y,z) = (x+y)/z;;

val plus\_and\_divide : int \* int \* int -> int = <fun>

# let l = (1,2,3);;

val l : int \* int \* int = (1, 2, 3)

# plus\_and\_divide l;;

- : int = 1

## 2 - Fonction avec plusieurs arguments

# let plus x y = x+y;;

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

# plus 2 2;;

- : int = 4

## 3 - Retourner plusieurs valeurs

# let divide x y = (x/y,x mod y);;

val divide : int -> int -> int \* int = <fun>

# divide 2 3;;

- : int \* int = (0, 2)

## 4 - Application partielle

# let plus x y = x+y;;

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

Fonction permettant de calculer la somme de x et y

# let incr = plus 1;;

val incr : int -> int = <fun>

# incr 42;;

- : int = 43

Fonction permettant de calculer un nb plus 1

# let mul x y = x \* y;;

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

Fonction permettant de calculer le produit de x et y

# let double = mul 2;;

val double : int -> int = <fun>

Fonction partielle permettant de calculer un nb fois 2

# double 2;;

- : int = 4

## 4 - Fonction anonyme

# (fun x->x+1) 42;;

- : int = 43

# let incr = fun x ->x+1;;

val incr : int -> int = <fun>

# incr 42;;

- : int = 43

## 5 - Iterateurs

### map : appliquer une fonction ) tous les éléments d’une liste pour donner la liste des résultats

# List.map;;

- : ('a -> 'b) -> 'a list -> 'b list = <fun>

# List.map(fun x->x+1)[1;2;3;4];;

- : int list = [2; 3; 4; 5]

Ajouter 1 à chaque élément de la liste

### fold\_left : faire ajouter un entier à une liste de gauche à droit

# List.fold\_left;;

- : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a = <fun>

# List.fold\_left(fun x y ->x+y) 0 [1;2;3;4];;

- : int = 10

équivalant :

plus(plus(plus(plus 0 1)2)3)4;;

### List.iter : appliquer une fonction en itération pour une liste

# List.iter print\_int [ 1; 2; 3; 4 ];;

1234- : unit = ()

Afficher un par un des éléments dans la liste

# Session 4

## 1 - match….with pour les entiers

# let string\_of\_int x = match x with

| 0 -> "zero"

| 1 -> "one"

| 2 -> "two"

| \_ -> "many";;

val string\_of\_int : int -> string = <fun>

# string\_of\_int 23;;

- : string = "many"

équivalant :

# let string\_of\_int2 = function

| 0 -> "zero"

| 1 -> "one"

| 2 -> "two"

| \_ -> "many";;

val string\_of\_int2 : int -> string = <fun>

# string\_of\_int2 22;;

- : string = "many"

## 2 - match….with pour la caractère

# let is\_capital = function

'a'..'z'->false

|'A'..'Z'->true

|\_-> failwith "not a letters";;

val is\_capital : char -> bool = <fun>

# is\_capital 'A';;

- : bool = true

# is\_capital 'a';;

- : bool = false

# is\_capital '1';;

Exception: Failure "not a letters".

Si lettre est majuscule retourner vrai si en minuscule retourner false sinon retourner exception

### Char.uppercase : transformer les minuscules en majuscules

### failwith “....” : lancer une exception

### as : donner le nom au paramètre

# let capitalize = function

| 'a' .. 'z' as letter -> Char.uppercase letter

| 'A' .. 'Z' as letter -> letter

| \_ -> failwith "Not a valid letter" ;;

val capitalize : char -> char = <fun>

# capitalize 'a';;

- : char = 'A'

Transformer les minuscules en majuscules

## 3 - match….with pour les lignes

# let fit str len = match (str,len) with

|("foo",51)->true

|("bar",51)->true

|(\_,42)->false

|\_-> (String.length str)=len;;

val fit : string -> int -> bool = <fun>

Cette fonction contient 2 paramètres, si str = foo et len = 51, retourner true, si str = bar e len = 51 retourner true, si str est égal à autre chaîne et len = 42, retourner false, si on entre n’importe quel chaîne et si len qu’on entre est égal à la longueur de la chaîne, retourne true sinon retourner false

# fit "foo" 51;;

- : bool = true

# fit "foo" 52;;

- : bool = false

# fit "bar" 51;;

- : bool = true

# fit "bar" 52;;

- : bool = false

# fit "aa" 42;;

- : bool = false

# fit "bb" 42;;

- : bool = false

# fit "abab" 4;;

- : bool = true

équivalant :

# let fit str len = match (str,len) with

| ("foo", 51)

| ("bar", 51) -> true

| (\_ , 42) -> false

| \_ -> (String.length str) = len ;;

val fit : string -> int -> bool = <fun>

Si deux condition provoque true, on peut écrire une seule fois “->true” dans la fonction

équivalant :

### when : vérifier la condition

# let fit str len = match (str,len) with

| (\_,51) when (str="foo" || str="bar") -> true

| (\_, x) when x=42 -> false

| \_ -> (String.length str) = len;;

val fit : string -> int -> bool = <fun>

équivalant :

# let fit str len = match (str,len) with

| (\_,51) when (str="foo" || str="bar") -> true

| (\_, x) when x=42 -> false

| \_ -> if (String.length str)=len then true else false;;

val fit : string -> int -> bool = <fun>

## 4 - match….with pour une liste

### h :: t : h est le preimier élément de la liste, t sont les restes éléments

### :: : permet de concaténer un élément dans une liste

# let head = function

|[]->failwith "empty list"

|h::t->h;;

val head : 'a list -> 'a = <fun>

# let l = [1;2;3;4];;

val l : int list = [1; 2; 3; 4]

# head l ;;

- : int = 1

Retourner le premier élément de la liste

### [ \_ ] : liste contenante un seul élément

### \_::e::\_ : deuxième élément de la liste

# let second\_element = function

|[]->failwith "liste vide"

|[\_]->failwith "liste contenante juste un élément"

|\_::e::\_->e;;

val second\_element : 'a list -> 'a = <fun>

# second\_element l;;

- : int = 2

Retourner le deuxième élément de la liste

## 5 - match….with pour les tableaux

### [|\_:\_|] : tableau contenant deux éléments

# let l = [|1;2|];;

val l : int array = [|1; 2|]

# let have\_size\_two = function

|[|\_;\_|] ->true

|\_->false;;

val have\_size\_two : 'a array -> bool = <fun>

# have\_size\_two l;;

- : bool = true

Retourner vrai si la taille du tableau est 2

## 6 - exhaustivité

# let head\_partial = function

|h::\_->h;;

Warning 8: this pattern-matching is not exhaustive.

Here is an example of a value that is not matched:

[]

val head\_partial : 'a list -> 'a = <fun>

# let l = [1;2;3];;

val l : int list = [1; 2; 3]

# head\_partial l;;

- : int = 1

Retourner le premier élément d’une liste

# let head = function

|[]->failwith "liste vide"

|h::\_->h

|[h]->h;;

Warning 11: this match case is unused.

val head : 'a list -> 'a = <fun>

# head l ;;

- : int = 1

# head [];;

Exception: Failure "liste vide".

# head [1];;

- : int = 1

Si liste est vide retourner exception si liste contenante plusieurs éléments retourner le premier, si liste contenante un seul élément retourner cet élément

# Session 5

## 1 - let keyword

# let x = "aaaa";;

val x : string = "aaaa"

# let x = 41 in x + 1;;

- : int = 42

# x;;

- : string = "aaaa"

Même si on donne x un valeur provisoire mais x a toujours la valeur qu’on a défini au début.

## 2 - parenthèses

### begin….end…. : d’abord exécuter la partie begin et exécuter la partie end après

# begin 1 + 2 end \* 3;;

- : int = 9

Faire (1+2)\*3

Exercice : Retrouver des parenthèse pour que le programme fonctionne

1) - let ten = let double x = x+x in double 3 + 2;;

val ten : int = 8

Mais ce programme demande à retourner dix, donc la correction :

# let ten =

let double x = x+x in

double(3+2);;

val ten : int = 10

2) - let hundred =

if true or false then

print\_string "May I help you?\n" ;

100

else 0;; ;;

Error: Syntax error

Ce programme demande à retourner 100, donc la correction :

# let hundred =

if (true or false) then

(print\_string "May I help you?\n" ;

100)

else 0 ;;

May I help you?

val hundred : int = 100

3) - let one = let accum = ref -54 in

for i = 1 to ten do accum := !accum + i done ;

!accum ;;

Ce programme demande à retourner 1, donc la correction :

# let one =

let accum = ref (-54) in

for i = 1 to ten do

accum := !accum + i done ;

!accum;;

val one : int = 1

4) - one + match hundred with

| 42 -> match ten with 10 -> 52 | \_ -> 0

| 100 -> match ten with 10 -> 110 | \_ -> 0

| \_ -> -1;; ;;

Warning 8: this pattern-matching is not exhaustive.

Here is an example of a value that is not matched:

0

Warning 11: this match case is unused.

Warning 11: this match case is unused.

Exception: Match\_failure ("//toplevel//", 43, -113).

Ce programme demande = retourner 1 et 100 plus d’autres chiffres, donc la correction :

# one + match hundred with

| 42 -> (match ten with 10 -> 52) | \_ -> 0

| 100 -> (match ten with 10 -> 110) | \_ -> 0

| \_ -> -1;;

Warning 8: this pattern-matching is not exhaustive.

Here is an example of a value that is not matched:

0

Warning 8: this pattern-matching is not exhaustive.

Here is an example of a value that is not matched:

0

Warning 11: this match case is unused.

Warning 11: this match case is unused.

Warning 11: this match case is unused.

- : int = 1

<http://caml.inria.fr/pub/old_caml_site/FAQ/exemples-fra.html>