# Aide-mémoire OCaml

## Déclarations et instructions

| | |
|---|---|
| commentaires | (* ... *) |
| définition d'une valeur | let v = expression |
| récursive | let rec v = ... |
| locale | let v = ... in expression |
| définitions simultanées | let v = ... and w = ... |
| successives | let v = ... in let w = ... |
| référence (modifiable) | let v = ref expression |
| valeur d'une référence | !v |
| modifier une référence | v := ... |
| fonction sans argument | let f () = ... |
| à un argument | let f x = ... |
| à n arguments | let f $x_1$ ... $x_n$ = ... |
| expression conditionnelle | if condition then $expression_1$ else $expression_2$ |
| filtrage simple | match valeur with<br>\| $motif_1$ -> $expression_1$<br>...<br>\| $motif_k$ -> $expression_k$<br>\| _ -> expression-par-défaut |
| filtrage avec garde | match valeur with<br>\| $motif_1$ when $condition_1$ -> ...<br>... |
| ne rien faire | () |
| suite d'instructions | begin ... end |
| boucle croissante | for i = début to fin do ... done |
| conditionnelle | while condition do ... done |
| exception personnalisée | failwith "message d'erreur" |
| fonction anonyme | function x -> ... |
| à plusieurs arguments | fun x y -> ... |

## Expressions booléennes

| | |
|---|---|
| vrai, faux | true, false (en minuscules) |
| et, ou, non | &&, \|\|, not |
| comparaison | <, <=, =, <>, >=, > |
| booléen ⟼ chaîne | string_of_bool |
| chaîne ⟼ booléen | bool_of_string |

## Expressions entières

| | |
|---|---|
| opérations arithmétiques | +, -, *, /, mod |
| valeur absolue | abs |
| minimum, maximum | min a b, max a b |
| entier ⟼ chaîne | string_of_int |
| chaîne ⟼ entier | int_of_string |
| entier aléatoire de {0,...,n-1} | Random.int n |
| afficher un entier | print_int |

## Expressions flottantes

| | |
|---|---|
| opérations arithmétiques | +., -., *., /. |
| puissance | ** |
| minimum, maximum | min a b, max a b |
| fonctions mathématiques | abs_float, exp, log, sqrt, sin, cos, tan, sinh, cosh, tanh, asin, acos, atan, atan2 |
| flottant ⟼ entier | int_of_float |
| entier ⟼ flottant | float_of_int |
| flottant ⟼ chaîne | string_of_float |
| chaîne ⟼ flottant | float_of_string |
| flottant aléatoire entre 0 et a | Random.float a |
| afficher un flottant | print_float |

## Listes

| | |
|---|---|
| liste | [x ; y ; z ; ...] |
| liste vide | [ ] |
| tête et queue | List.hd, List.tl, h : :t |
| longueur d'une liste | List.length |
| concaténation | @ |
| image miroir | List.rev |
| appliquer une fonction | List.map fonction liste |
| appliquer un traitement | List.iter traitement liste |
| itérer une opération | List.fold_left f e liste |
| test d'appartenance | List.mem élément liste |
| test de présence | List.exists prédicat liste,<br>List.for_all prédicat liste |
| indice d'un élément | List.nth liste élément |
| tri | Sort.list ordre liste |

## Tableaux

| | |
|---|---|
| tableau | [\|x ; y ; z ; ...\|] |
| tableau vide | [\| \|] |
| i-ème élement | v.(i) |
| modification | v.(i) <- qqch |
| longueur d'un tableau | Array.length |
| création d'un tableau | Array.make longueur valeur |
| d'une matrice | Array.make_matrix n p valeur |
| extraction | Array.sub tableau début longueur |
| concaténation | Array.append $tableau_1$ $tableau_2$ |
| copie | Array.copy tableau |
| appliquer une fonction | Array.map fonction tableau |
| appliquer un traitement | Array.iter traitement tableau |
| tableau ⟼ liste | Array.to_list |
| liste ⟼ tableau | Array.of_list |

## Chaînes de caractères

| | |
|---|---|
| caractère | 'x' |
| chaîne de caractères | "xyz..." |
| $i^{ième}$ caractère | chaîne.[i] |
| modification | chaîne.[i] <- qqch |
| longueur d'une chaîne | String.length |
| création d'une chaîne | String.make longueur caractère |
| caractère ⟼ chaîne | String.make 1 caractère |
| extraction | String.sub chaîne début longueur |
| concaténation | $chaîne_1$^$chaîne_2$, String.concat |
| afficher un caractère | print_char |
| une chaîne | print_string |

## Commande de l'interpréteur

| | |
|---|---|
| tracer une fonction | #trace fonction |
| ne plus tracer | #untrace fonction |
| charger une bibliothèque | open nom, #load "nom" |
| charger un fichier source | #use "nom" |

## Entrées-sorties

impression formatée Printf.fprintf

### Sur le terminal

| | |
|---|---|
| impression de valeurs | print_int, print_float, print_char, print_string |
| changer de ligne | print_newline () |
| lecture de valeurs | read_int, read_float, read_line |

### Dans un fichier

| | |
|---|---|
| ouverture en lecture | let canal = open_in "nom" |
| en écriture | let canal = open_out "nom" |
| lecture | input_char, input_line, input_byte, input_value |
| écriture | output_char, output_string, output_byte, output_value, flush |
| fermeture | close_in, closeout |