

## Rappels sur le langage OCaml

### Commentaires

`(* Les commentaires commencent par (* et finissent par *). *)`

### Variables locales

```
let x = 10 in
let y = 2 in
  x + y
```

### Fonctions

Déclaration :

```
let somme x y = x + y  (* définit une fonction à deux arguments *)
```

Appel :

```
somme 3 4  (* retourne 7 *)
```

### Format.fprintf

Syntaxe similaire à la fonction `fprintf` en C :

```
let s = "chaine" in
let i = 42 in
Format.fprintf Format.std_formatter "s = %s, i = %d\n" s i
```

### Expression conditionnelle

Valeur absolue :

```
if x >= 0 then x else -x
```

Tester si x est pair :

```
if x mod 2 = 0 then
  "pair"
else
  "impair"
```

## Types somme

Déclaration :

```
type t =  
  | A  (* constructeur sans argument *)  
  | B of int * int  (* constructeur avec deux entiers *)  
  | C of string  (* constructeur avec une chaine *)  
  | D  (* autre constructeur sans argument *)
```

Création de valeurs de ce type :

```
let x1 = A  
let x2 = B (10, 32)  
let x3 = B (12, 64)  
let x4 = C "chaine"  
let x5 = D
```

Pattern matching :

```
match x with  
  | A -> 0  
  | B (12, _) -> 1  
  | B _ -> 2  
  | _ -> 3
```

Les cas (motif entre | et ->) du pattern matching sont testés successivement de haut en bas et le code à droite de la flèche -> du premier cas correspondant est exécuté (ou une erreur est levée si aucun cas ne correspond). Le soulignement \_ peut être utilisé pour signifier « dans tous les cas ». Par exemple, le code précédent retourne :

- 0 si x vaut A (par exemple la variable x1 ci dessus) ;
- 1 si x vaut B (n1, n2) avec n1 = 12 et n2 quelconque (par exemple x3) ;
- 2 si x vaut B (n1, n2) avec n1 ≠ 12 et n2 quelconque (par exemple x2) ;
- 3 si x vaut C s avec s quelconque ou D (par exemple x4 ou x5).