Les chaînes de caractères : `string`

Séquences finies de caractères.

Longueur :	$[0, 2^{57} - 9]$ (32-bit : $2^{24} - 5$
Valeurs :	`"a string"` `"a"` `""` (chaîne vide)
Opérateurs :	`^` (concaténation) `=` `<>` `<` `>` `<=` `>=`
Accès à un caractère :	`s.[i]` est le caractère (de type `char`) au rang `i` ($0 \leq i <$longueur(s)) de `s`
Quelques fonctions :	`String.length` `String.sub`

Quelques fonctions de conversion

`float_of_int : int -> float`	entier → flottant
`int_of_float : float -> int`	flottant → entier (tronqué)
`int_of_char : char -> int = Char.code`	caractère → code ASCII
`char_of_int : int -> char = Char.chr`	code ASCII → caractère
`Char.escaped : char -> string`	caractère → chaîne
`string_of_int : int -> string`	entier → chaîne
`int_of_string : string -> int`	chaîne → entier
`string_of_float : float -> string`	flottant → chaîne
`float_of_string : string -> float`	chaîne → flottant

2 Les phrases : expressions et définitions

Pour l'instant nous connaissons deux types de phrases : les **expressions** et les **définitions**. Dans le système interactif, une phrase est toujours terminée par `;;`

2.1 Les expressions

Une expression peut être :

une valeur simple	`15`	`x`
une opération	`a + 15`	`"Hello " ^ "world"`
parenthésée	`(3*a)`	`(true || false)`
une application de fonction	`cos x`	`float_of_int 15`

```
# (666 * 42 = 27972) && ("Hello" < "World");;
 - : bool = true
```

Une expression peut aussi contenir une(des) définition(s) locale(s) (voir ci-dessous).

2.2 Les définitions

Définitions globales simples → Définition d'une variable utilisable à postérieri

`let` *ident* = *expression*

Une définition est une liaison de nom à une valeur (celle de l'expression).

```
# let a = 1 + 2 ;;
 val a : int = 3
```

Une fois défini, un nom a toujours la même valeur,
mais il peut être "masqué" par une autre définition utilisant le même nom...

Définitions globales multiples

`let` $ident_1$ = $expression_1$
`and` $ident_2$ = $expression_2$
...
`and` $ident_n$ = $expression_n$

```
# let one = 1 and two = 2. and three = '3' ;;
 val one : int = 1
 val two : float = 2.
 val three : char = '3'
```