Fonctions XPath

1. `concat()`

- `concat(string1, string2, ...)` : retourne la concaténation de `string1` et `string2` (et plus).

2. `upper-case()` et `lower-case()`

- `upper-case(string)` : retourne `string` en haut-de-casse ;
- `lower-case(string)` : retourne `string` en bas-de-casse.

3 `translate()` et `replace()`

- `translate(string, 'abc', 'ABC')` : retourne `string` avec :
  - `a` remplacé par `A` ;
  - `b` remplacé par `B` ;
  - `c` remplacé par `c`.
  - *NB : le troisième paramètre n'a pas besoin d'être de la même taille que le deuxième. Avec `translate(string, 'abc', '')`, tous les `a`, les `b` et les `c` seront enlevés de `string`.*
  - Ce n'est pas l'équivalent d'un "rechercher/remplacer".
- `replace(string, abc, ABC)` : retourne `string` avec `abc` replacé par `ABC`.
  - Ce peut être considéré comme l'équivalent d'un "rechercher/remplacer".
  - Attention : ne fonctionne qu'à partir de XSLT version 2.

4. `contains()`

- `contains(string1, string2)` : retourne `true` ou `false` selon que `string1` contienne (`true`) ou ne contienne pas `string2` (`false`).

5. `starts-with()` et `ends-with()`

- `starts-with(string1, string2)` : retourne `true` ou `false` selon que `string1` commence (`true`) ou ne commence pas par `string2` (`false`) ;
- `ends-with()` : retourne `true` ou `false` selon que `string1` se termine (`true`) ou ne se termine pas par `string2` (`false`) ;

6. `not()`

- `not(arg)` : évalue `arg` et retourne la valeur opposée.
  - `\\title[not(@level = 'm')]` renvoie tous les `<title>` dont le `@level` n'a pas pour valeur `m` ainsi que ceux qui n'ont pas de `@n`.
  - *Attention : ce n'est pas l'équivalent de l'opérateur booléen `!=`. `\\title[@level != 'm')]` renvoie tous les `<title>` qui ont un `@level` dont la valeur n'est pas `m`.*

- Peut être combiné avec `contains()` sous la forme de `not(contains(..., ...))`.

7. `last()`, `position()` et `count()`

- *NB : dans XPath, la numérotation commence à `1`.*
- `last( )` : évalue la taille d'un ensemble de balises.
  - Exemple d'utilisation : appliquer une règle à la dernière balise d'un ensemble. Ainsi, `//div/p[last()]` renvoie tous les `<p>` en dernière position dans les `<div>`.
- `position( )` : évalue la position d'une balise au sein de l'ensemble de balises auquel s'applique la règle courante.
  - Exemple d'utilisation : atteindre une balise en fonction de son emplacement. Ainsi, `//div/p[position() = 5]` renvoie tous les `<p>` en cinquième position dans les `<div>`.
  - `//div/p[position() = 5]` et `//div/p[5]` sont deux expressions identiques.
  - *Attention : `position( )` ne permet pas de déterminer la position de la balise par rapport à son parent.*
- `count(arg)` : évalue le nombre de `node` et retourne ce nombre.
  - Le résultat est identique à `last()`, la différence étant que `count()` prend une expression XPath en argument.
  - Exemple d'utilisation : connaître le nombre total d'élément au sein d'un parent (comme le nombre de `<p>` au sein d'une `<div>`).