IA01-Compte_rendu_TD3

Objectifs TD

Ce TD comporte deux objectifs principaux. Le premier étant de se familiariser avec les boucles en langage Lisp, le second étant de représenter et manipuler une page web HTML.

Exercice_1

Avec ce premier exercice, nous découvrons les différentes manières de parcourir une liste en lisp. Pour illustrer cette notion, nous devons afficher chaque élément d'une liste.

Nous utilisons comme exemple de liste la liste suivante :

```
(setq ll '(A 1 BB 2 CCC 3 DDD 4))
```

Nous voulons donc avoir le résultat suivant pour l'appel d'une fonction d'impression Fx ayant comme paramètre la liste ll :

```
> (Fx ll)
A
1
BB
2
CCC
3
DDD
4
```

- 1) Pour la première méthode, nous utilisons un mappage appliqué à la fonction print.

```
(defun F1 (L) ; Par mappage
   (mapcar #'print L)
   )
```

- 2) Pour la deuxième méthode, nous parcourons la liste par récurrence.

```
(defun F2 (L) ; Par récurrence
  (if L       ; condition d arret
      (progn (print (car L))(F2 (cdr L))) ; progn afin de créer un bloc contenant
          plusieurs s-expressions
    "done"
    )
  )
```

- 3) ensuite nous pouvons, de la manière la plus simple, le faire avec un dolist

```
(defun F3 (L) ; Avec un dolist
   (dolist (x L 'done)(print x))
   )
```

- 4) Pour la quatrième méthode, nous faisons une boucle sur tous les éléments x de la liste L

```
(defun F4 (L)    ; avec chaque élément x de la liste L
   (loop for x in L
       do (print x)
         )
   'done
   )
```

- 5) enfin, on utilise un compteur x de 0 à length de la liste (exclu) afin d'afficher chaque élément L(x) de la liste

```
(defun F5 (L)   ; avec x dans [0 ; length L [ (x entier)
  (dotimes (x (length L) 'done)(print (nth x L)))
  )
```

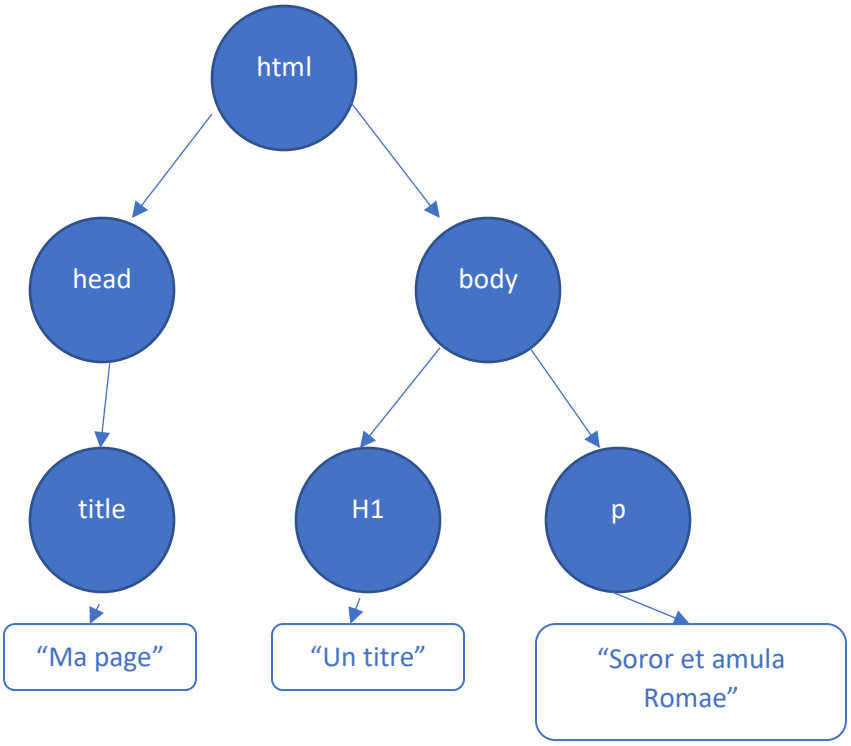
Exercice_2

Le but de cet exercice est de comprendre comment l'on peut représenter sous forme d'une liste une page HTML qui fonctionne avec un système de balisage <a>content.

L'exemple de page HTML est le suivant :

```
<html>
	<head>
		<title>Ma page</title>
	</head>
	<body>
		<h1>Un titre</h1>
		<p>Soror et aemula Romae</p>
	</body>
</html>
```

1) Nous pouvons représenter la page HTML ci-dessus par l'arbre suivant :



Nous pouvons donc représenter cet arbre sous forme de liste :

```
(setq *html* '(html (head (title "Ma page")) (body (h1 "Mon titre")(p "Soror et amula
    Romae"))))
```

2) Afin d'afficher correctement notre page HTML que nous avons représentée en tant que liste, nous parcourons chaque nœud. Pour chaque nœud n'étant pas une feuille, nous affichons les balises HTML, s'il s'agit d'une feuille, nous affichons son contenu. Nous avons alors :

```lisp
(defun make-html (parent file)
  (if (listp parent)   ; Si parent est une liste
      (progn (format file "~%<~S>" (car parent))   ; balise HTML ouvrante <balise>
        (loop for node in (cdr parent)             ; boucle pour chaque node de cette
            balise
            do (make-html node file)
              )
        (format file "~%</~S>" (car parent))       ; balise HTML fermante </balise>
        )
    (format file "~A" parent)                      ; affichage du contenu
    )
  )

```

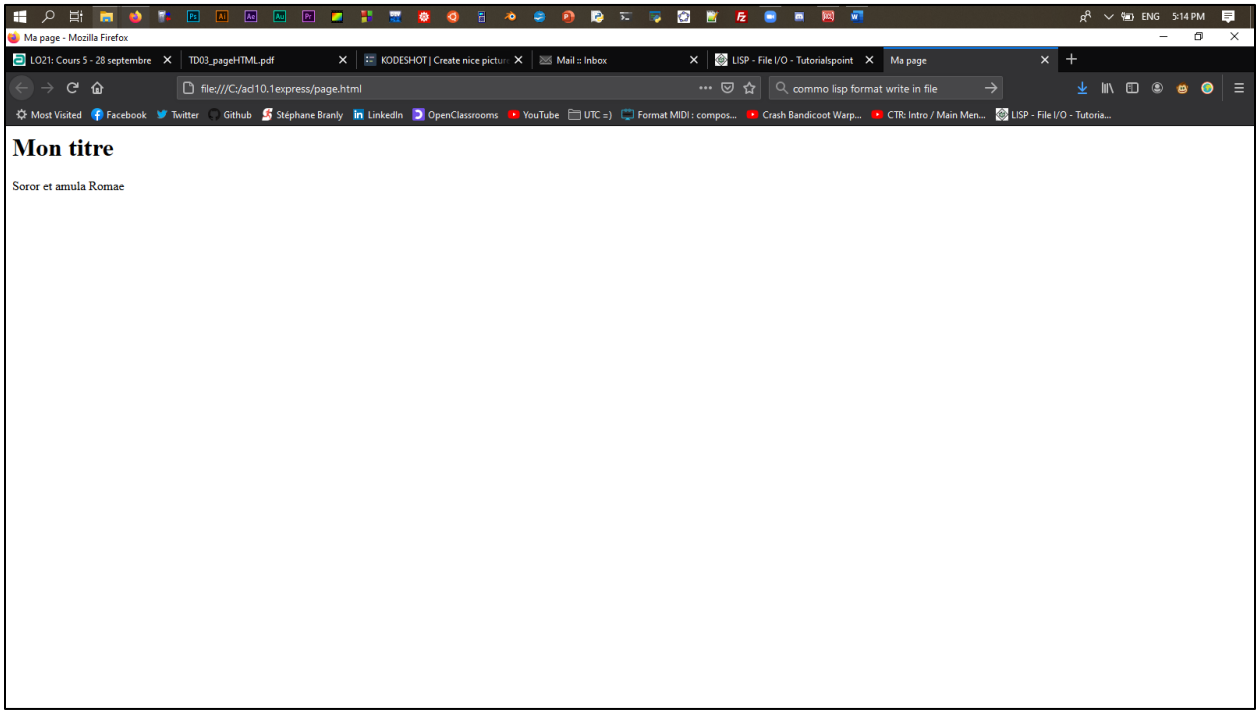
Ici, il est important de noter que l'enregistrer se fait dans un fichier file transmis en paramètre.

3) Pour enregistrer dans un fichier, il suffit alors de créer un fichier et d'appeler la fonction make-html en lui donnant en paramètres la page ainsi que le fichier.

```lisp
(with-open-file (file "page.html" :if-does-not-exist :create :if-exists :overwrite
    :direction :output)
    (make-html *html* file)
    )

```

Nous pouvons alors ouvrir le fichier page.html dans un navigateur tel que Mozilla Firefox



A noter : cette structuration de pages html ne permet pas de prendre en compte toutes les propriétés telles que les classes, id, balises autofermantes, propriétés, etc. Il s'agit ici d'un modèle simplifié.