# Séance 7b: Exercices sur les listes (Part 2)

Université Paris-Diderot

Objectifs:

- Manipuler les listes
- Concevoir et programmer des algorithmes sur les listes

Dans cette séance, vous résoudrez des exercices et des problèmes sur des listes. Vous écrirez des boucles et définirez des fonctions intermédiaires pour rendre votre code plus lisible et concis.

**Exercice 1 (Listes de mots, ⋆)**

1. *Écrire une fonction* `letters2word` *qui prend en argument une liste de caractères* `lis` *et qui affiche le mot obtenu en concatenant ces lettres.*

   ***Contrat:***
   `lis` = *["p","l","a","c","a","r","d"]*
   `letters2word(lis)` *affichera "placard"*

2. *Écrire une fonction* `stutterword` *qui prend en argument une liste de caractères* `lis1` *et une liste d'entiers* `lis2` *et qui affiche le mot obtenu en concatenant les lettres de la liste* `lis1`*, comme suit : la lettre sur la position* `i` *dans* `lis1` *est répétée autant de fois que l'indique l'entier en position* `i` *dans* `lis2`*. La fonction doit afficher "Erreur" si les deux listes n'ont pas la meme longueur.*

   ***Contrat:***
   `lis1` = *["a","b","c","d"]*
   `lis2 = [2,2,3,4]`

   `stutterword(lis1, lis2)` *affichera "aabbcccdddd"*

3. *Écrire une fonction* `word2letters` *qui prend en argument un mot et renvoie la liste de ses lettres.*

   ***Contrat:***
   `word2letters`*("placard") renvoie ["p","l","a","c","a","r","d"]*

4. *Écrire une fonction* `letters` *qui prend en argument un mot (chaine de caractères)* `word` *et qui renvoie la liste de ses lettres, cette fois ci sans doublons.*

   ***Contrat:***

   `letters`*("electroacoustique") renvoie ["e","l","c","t","r","o","a","u","s","i","q"]*

□

**Exercice 2 (Des ensembles, ⋆- ⋆⋆)**

1. *Écrire une fonction* `search` *qui prend en argument une liste d'entiers* `lis` *et un entier* `x` *et qui renvoie* `True` *si* `lis` *contient la valeur* `x`*, et* `False` *sinon.*

***Contrat:***

```
lis = [6,20,12,1000,8]
   print(search(lis, 12)) affiche True
   print(search(lis, 50)) affiche False
```

2. *Écrire une fonction* `union` *qui prend en argument deux listes d'entiers* `lis1`, *et* `lis2` *(considérées sans doublons) et qui renvoie l'union de* `lis1` *et* `lis2`. *L'union ne devra pas comporter de doublons non plus.*

   ***Contrat:***

   ```
   lis1 = [6,20,12,1000,8], lis2 = [2,8,6,7,12]
   union(lis1, lis2) = [6,20,12,1000,8,2,7]
   ```

3. *Écrire une fonction* `différence` *qui prend en argument deux listes d'entiers* `lis1`, *et* `lis2` *(considérées sans doublons) et qui renvoie la liste représentant la différence symétrique de* `lis1` *et* `lis2`. *NB : La 'différence symétrique' de A et B est l'ensemble des éléments appartenant soit à A, soit à B, mais pas aux deux à la fois.*

   ***Contrat:***

   ```
   lis1 = [6,20,12,1000,8], lis2 = [2,8,6,7,12]
   difference(lis1, lis2) renvoie [20,1000,2,7]
   ```

□

**Exercice 3 (Tri, ⋆⋆- ⋆⋆⋆)**

1. *Écrire une fonction* `position` *qui prend en argument une liste d'entiers* `lis`, *qu'on considéra déjà triée, et un entier* `x` *et qui renvoie la position dans* `lis` *dans laquelle on devrait insérer* `x`, *pour que la liste obtenue reste triée.*

   ***Contrat:***

   ```
   lis = [0,2,4,6,7,8]
   position(lis,1) renvoie 1
   position(lis,-5) renvoie 0
   position(lis, 10) renvoie 6
   ```

2. *Écrire une fonction* `insert` *qui prend en argument une liste d'entiers* `lis`, *et deux entiers,* `pos` *et* `x`, *et qui renvoie la liste obtenue en insérant l'element* `x` *dans la liste* `lis` *sur la position* `pos`. *Si* `pos` *est plus grand que la taille de* `lis` , *la fonction renvoie la liste* `lis` *sans modifications. Indice : pour obtenir une liste composée des elements compris entre les positions* `i` *et* `j` *(positions* `i` *et* `j` *y comprises) d'une liste* `lis`, *on pourra écrire :* `lis[i:(j+1)]`.

   ***Contrat:***

   ```
   lis = [2,5,4,3]
   insert(lis, 0, 1) renvoie [1,2,5,4,3]
   insert(lis, 2, 100) renvoie [2,5,100,4,3]
   ```

3. *Écrire une fonction* `sort` *qui prend en argument une liste d'entiers*`lis` *et qui renvoie la liste triée. (Pour cela, pensez a utiliser les fonctions* `position` *et* `insert`*)*

   ***Contrat:***

   ```
   lis = [40,1,20,3,8,6]
   sort(lis) renvoie [1,3,6,8,20,40]
   ```

□

**Exercice 4 (Circulaire, $\star\star\star$)**

*Écrivez une fonction* `circulaire` *qui prend en argument deux listes d'entiers* `lis1` *et* `lis2` *et qui renvoie* `True` *si* `lis2` *est une permutation circulaire de* `lis1`*, et* `False` *sinon (ou si les listes n'ont pas la meme longueur). Conseil : utilisez la fonction* `shift` *que vous avez écrit pour l'exercice 4 de TP6b.*

***Contrat:***
`lis1 = [1,2,3,4,5]`
`lis2 = [3,4,5,1,2]`
`lis3 = [3,5,4,1,2]`
`circulaire(lis1,lis2)` *renvoie* `True`
`circulaire(lis1, lis3)` *renvoie* `False`

□