Algorithmique et Programmation 1 – TD - TP 10
COMPRÉHENSIONS DE LISTES

Exercice 1 - Compréhensions de listes (1)
Quelles sont les listes construites par les expressions suivantes ?

```
1 [2 * k for k in range(-2,3)]
2 [k * (k+1) / 2 for k in range(2,5)]
3 [1 for c in 'abc']
4 [c+c for c in 'abc']
5 [(1,2,k) for k in range(1,3)]
6 [[i, j] for i in range(0, 3) for j in range(3, 5)]
7 [k for k in range(10) if k % 3 == 0]
8 [k + 2 if k <= 5 else k * 2 for k in range(10)]
9 [(i, j) for i in range(0, 4) for j in range(2, 6) if (i + j)%2 == 0 ]
```

Exercice 2 - Compréhensions de listes (2)
A l'aide d'une compréhension :

1. Construire la liste des 5 premiers entiers pairs
2. Construire la liste de tous les couples (i, i + 1) pour i compris entre 1 et 10
3. Donner une définition de la fonction qui, étant donnée une liste d'entiers `l`, renvoie la liste des entiers pairs de `l`
4. Donner une définition de la fonction qui, étant donnée une chaîne de caractères `s`, renvoie la liste des voyelles contenues dans `s`
5. Donner une définition de la fonction qui, étant donnée une liste de liste `l`, renvoie la liste contenant les nombres entiers multipliés par 2 et les chaînes de caractères multipliées par 3. Par exemple :

```
>>> l = [[0, 'a'], [2, 'b'], [3, 'c']]
>>> double_int_triple_string(l)
[0, 'aaa', 4, 'bbb', 6, 'ccc']
```

Exercice 3 - Crible d'Eratosthène
Cet exercice consiste à implémenter le crible d'Eratosthène qui décrit un moyen de calculer la listes des nombres premiers inférieurs à un entier fixé. On rappelle qu'un nombre est premier s'il n'est divisible que par 1 et lui-même. On rappelle également que, par convention, 1 n'est pas un nombre premier.

La méthode du crible d'Eratosthène consiste à créer dans un premier temps la liste des entiers compris entre 2 et n, puis itérer le processus suivant :

1. Récupérer le premier élément de la liste (ce nombre est un nombre premier)
2. Retirer de la liste restante tous les multiples de ce nombre
3. Recommencer à l'étape 1 tant qu'il reste des éléments dans la liste

Par exemple, pour $n = 10$, on commence par créer la liste `entiers = [2, 3, 4, 5, 6, 7, 8, 9, 10]`, la liste contenant les nombres premiers est initialement vide `premiers = []`

- `entiers[1] = 2` est un nombre premier. On retire de la liste `entiers` tous les multiples de 2. On obtient `entiers = [3, 5, 7, 9]`, `premiers = [2]`

- `entiers[1] = 3` est un nombre premier. On retire de la liste `entiers` tous les multiples de 3. On obtient `entiers = [5, 7]`, `premiers = [2, 3]`
- `entiers[1] = 5` est un nombre premier. On retire de la liste `entiers` tous les multiples de 5. On obtient `entiers = [7]`, `premiers = [2, 3, 5]`
- `entiers[1] = 7` est un nombre premier. On retire de la liste `entiers` tous les multiples de 7. On obtient `entiers = []`, `premiers = [2, 3, 5, 7]`
- La liste est vide. La liste des nombres premiers inférieurs ou égaux à 10 est donc `premiers = [2, 3, 5, 7]`

1. A l'aide d'une compréhension, définir une fonction `liste_non_multiple` qui, étant donné un entier n non nul et une liste d'entiers `l`, renvoie la liste des éléments de `l` qui ne sont pas multiples de n. Par exemple :

```
>>> liste_non_multiple(2,[2,3,4,5,6,7,8,9,10])
[3, 5, 7, 9]
```

2. Donner la définition et la spécification de la fonction `eratosthene` qui calcule la liste des nombres premiers inférieurs ou égaux à un entier n en utilisant le crible d'Eratosthène décrit ci-dessus.
3. Un facteur premier d'un entier n est un nombre premier qui divise n. A l'aide de la fonction `eratosthene`, et en utilisant une compréhension, donner la définition et la spécification de la fonction `liste_facteurs_premiers` qui, étant donné un entier n supérieur ou égal à 2, calcule la liste des facteurs premiers de n, c'est à dire la liste des nombres premiers inférieurs ou égaux à n et qui divisent n. Par exemple :

```
>>> liste_facteurs_premiers(2)
[2]
>>> liste_facteurs_premiers(10)
[2, 5]
>>> liste_facteurs_premiers(2*3*4*7*9)
[2, 3, 7]
```

Exercice 4 - Triplets

1. Donner , en utilisant une compréhension, une définition de la fonction `triplets` qui retourne la liste des triplets (i,j,k) sur l'intervalle $[1,n]$ (avec n un entier naturel). Par exemple :

```
>>> triplets(2)
[(1, 1, 1), (1, 1, 2), (1, 2, 1), (1, 2, 2), (2, 1, 1), (2, 1, 2), (2, 2, 1), (2, 2, 2)]
```

2. Donner, en utilisant une compréhension, une définition de la fonction `décomposition` qui retourne la liste des triplets (i,j,k) sur l'intervalle $[1,n]$ (avec n un entier naturel) tels que $i+j=k$. Par exemple :

```
>>> decomposition(2)
[(1, 1, 2)]
>>> decomposition(3)
[(1, 1, 2), (1, 2, 3), (2, 1, 3)]
```

3. Donner, en utilisant une compréhension, une définition de la fonction `differents` qui retourne la liste des triplets (i,j,k) sur l'intervalle $[1,n]$ (avec n un entier naturel) tels que i, j et k sont tous différents. Par exemple :

```
>>> differents(2)
[]
>>> differents(3)
[(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```