

**TP2B - FONCTIONS, INSTRUCTIONS CONDITIONNELLES – EXERCICES
SUPPLÉMENTAIRES**

Dans les exercices suivants, en plus de ce qui a été présenté aux TP 1 et 2, vous pourrez avoir besoin des outils suivants :

- Opérateur % entre deux entiers positifs. Le **reste** de la division euclidienne de **a** par **b** est donné par **a%b**.
- Opérateur // entre deux entiers positifs. Le **quotient** de la division euclidienne de **a** par **b** est donné par **a//b**.

Exercice 1. Définir une fonction **multiples(k)** qui construit la « table de multiplication par k » : la liste $[0, k, 2*k, \dots, 10*k]$.

Exercice 2. Construire la table de multiplication sous la forme d'une liste **T** qui contient onze listes : les multiples de 0, les multiples de 1, ceux de 2, ceux de 3, etc jusqu'à la liste des multiples de 10.

$T = [[0, \dots, 0] , [0, 1, 2, \dots, 10] , [0, 2, 4, \dots, 18, 20] , \dots , [0, 10, 20, \dots, 90, 100]]$

Exercice 3. Définir une fonction **factorielle(n)** qui renvoie la factorielle de l'entier n donné en argument.

Exercice 4. Définir une fonction **produit(L,G)** qui reçoit en argument deux listes de nombres de même longueur. En sortie, cette fonction devra renvoyer le produit terme-à-terme de ces deux listes.

Par exemple si $A = [3, 0, 6, 2, -7]$ et $B = [2, 10, -2, 8, 12]$, alors **produit(A,B)** doit renvoyer $[6, 0, -12, 16, -84]$

Exercice 5. Définir une fonction **accroissements(L)** qui, étant donné une liste **L**, construit la liste des variations entre un élément de **L** et le suivant. Ainsi si $L = [3, 1, 9, 2]$ ce programme construit la liste $[-2, 8, -7]$ (Remarquer que la liste à construire doit avoir un élément de moins que la liste de départ).

Exercice 6. Définir une fonction **jonction(L,G)** qui reçoit en argument deux listes **A** et **B** de même longueur. En sortie, cette fonction devra renvoyer la liste de listes suivante :

$[[A[0], B[0]] , [A[1], B[1]] , \dots , [A[n-1], B[n-1]]]$

Exercice 7. Définir une fonction **partial_reverse(L,a,b)** qui prend en argument une liste **L**, et deux entiers $a < b < \text{len}(L)$.

Cette fonction renvoie en sortie une liste dont les éléments sont ceux de **L**, mais avec les éléments entre les indices **a** et **b** compris dans l'ordre inverse.

Exemple : en entrée $[78, 10, 69, 43, 6, 32, 24, 83, 71, 94]$, et les entiers 4 et 8.

En sortie $[78, 10, 69, 43, 71, 83, 24, 32, 6, 94]$.

Exercice 8. On définit la suite u par $u_0 = 5$ et $\forall n \geq 0, u_{n+1} = \frac{3u_n - 1}{4}$. Construire la liste des 100 premiers termes de cette suite et conjecturer sa limite.

Exercice 9. Définir une fonction `termes_carres(L)` qui prend en argument une liste de nombres L et renvoie la liste des carrés des éléments de L . Exemple : `carres( [3,8,1,4] )` doit renvoyer `[9,64,1,16]`

Exercice 10. Définir une fonction `nb_positions_egalite(L,G)` qui prend en entrée deux listes de nombres de même longueur, par exemple `[ 6, 0, 2, 8, -7]` et `[ 5, 0, 8, 6, -7]`. Elle renvoie en sortie le nombre de positions où les deux listes ont le même élément. Dans cet exemple, il y en a deux : les positions 1 et 4. La fonction renvoie donc l'entier 2.

Exercice 11. Définir une fonction `est_arithmetique(L)` qui prend en entrée une liste d'entiers et qui renvoie un booléen disant si cette liste est en progression arithmétique. Par exemple, `[10,14,18,22]` donnera `True`, alors que `[6,3,9,0]` donnera `False`.

Exercice 12. Définir une fonction `diviseurs(n)` qui prend en entrée un nombre entier n , et qui renvoie la liste des *diviseurs* de n (les nombres d tel que n est un multiple de d). Exemple : si $n = 18$, la liste de ses diviseurs est `[1,2,3,6,9,18]`. Indication : pour savoir si n est un multiple de d , on se sert du reste de la division euclidienne de n par d .

Exercice 13. Définir une fonction `gd_nb_div(L)` qui prend en argument une liste d'entiers L , et renvoie en sortie l'élément de L qui a le plus grand nombre de diviseurs. Exemple : si $L=[67, 69, 31, 34, 77, 36, 8]$, on renvoie 36 car c'est celui qui a le plus grand nombre de diviseurs (il a 9 diviseurs).

Exercice 14. Un nombre naturel est *premier* s'il a exactement deux diviseurs : 1 et lui-même. Définir une fonction `est_premier(n)` qui teste la primalité de son argument, et renvoie un résultat booléen.

Exercice 15. En utilisant la fonction ci-dessus, calculer la somme des nombres premiers inférieurs à 100.

Exercice 16. Trouver trois nombres entiers strictement positifs a, b, c tels que $a+b+c = 1000$ et $a^2 + b^2 = c^2$.