TD2-TP2 : Les Liste et les Tuples en Python

Exercice 1

1. Créer les listes suivantes :
   - $L1$: Une liste croissante des entiers naturels pairs strictement inférieurs à 20.
   - $L2$: La tranche du 3ème au 6ème élément inclus de la liste $L1$ précédente.
   - $L3$: La liste $L1$ sans les premier et dernier éléments.
   - $L4$: La liste décroissante des entiers naturels impairs strictement inférieurs à 10.
   - $L5$: La liste croissante des racines carrées des entiers naturels impairs inférieurs à 10.
2. Définir les listes suivantes **par compréhension** :
   (a) La liste des couples (x, y) d'entiers positifs tels que $x + y = 20$.
   (b) La liste des couples (x, y) d'entiers positifs tels que $x < y$ croissante selon y avec $x, y \in [0, 10]$.
   (c) La liste des entiers positifs inférieurs à 50 qui ne sont ni multiples de 3 ni de 5.

Exercice 2

Écrire une fonction **variance(L)** qui prend en argument une liste L et qui renvoie la variance de cette liste.

$$\text{var}(x_1, \ldots, x_n) = \frac{1}{n} \sum_{k=1}^{n} |x_k - \text{moy}| \quad \text{avec} \quad \text{moy} = \frac{1}{n} \sum_{k=1}^{n} x_k.$$

Exercice 3

On donne la fonction Python suivante où le paramètre L est une liste :

```
def Tri(L):
n = len(L)
for p in range(n-1):
pmin = p
for j in range(p,n):
if L[j] < L[pmin]:
pmin = j
L[pmin], L[p] = L[p], L[pmin]
return L
```

1. On saisit l'instruction $L = Tri([7, 4, 3, 2, 5])$. Remplir le tableau de variables suivant permettant d'illustrer le fonctionnement de ce tri :

p	0	1	2	3
pmin	3			
L	$[2, 4, 3, 7, 5]$			

2. Comment peut-on appeler ce tri ?

3. Réécrire la fonction Tri pour trier dans l'ordre décroissant et non dans l'ordre croissant.

Exercice 4

Écrire une fonction **decalage(L,k)** qui prend en argument une liste L et qui renvoie une copie de la liste mais les valeurs sans décaler vers la droite par k-positions.

Exemple : Si $L = [1, 2, 3, 4]$, alors `decalage(L,3)` renvoie la liste $[2, 3, 4, 1]$.

Exercice 5

Médian d'une liste de nombres

1. Écrire la fonction **grands(L,x)** qui reçoit en paramètres une liste de nombres L, et un élément x de L. La fonction renvoie le nombre d'éléments de L qui sont supérieurs strictement à x.

2. Écrire la fonction **petits(L,x)** qui reçoit en paramètres une liste de nombres L, et un élément x de L. La fonction renvoie le nombre d'éléments de L qui sont inférieurs strictement à x.

   L est une liste de taille n qui contient des nombres, et m un élément de L. L'élément m est un médian de L, si les deux conditions suivantes sont vérifiées :

   - Le nombre d'éléments de L, qui sont supérieurs strictement à m, est inférieur ou égale à $n/2$.

- Le nombre d'éléments de L, qui sont inférieurs strictement à m, est inférieur ou égale à $n/2$.

3. Écrire la fonction **median(L)** qui reçoit en paramètre une liste de nombres L non vide, et qui renvoie un élément médian de la liste L.

Exercice 6

La suite de Fibonacci est définie par :

$$F_0 = 1, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2} \quad \forall n \geq 2.$$

1. Écrire une fonction récursive **Fibo_rec(n)** qui prend en argument un entier positif n et qui renvoie la valeur du n-ème terme de la suite (F_n).

```
>>> print(Fibo_rec(5))
8
```

2. Écrire une fonction itérative **Fibo_iter(n)** qui prend en argument un entier positif n et qui **renvoie une liste** contenant les n-premiers termes de la suite.

```
>>> print(Fibo_iter(5))
[1, 1, 2, 3, 5, 8]
```

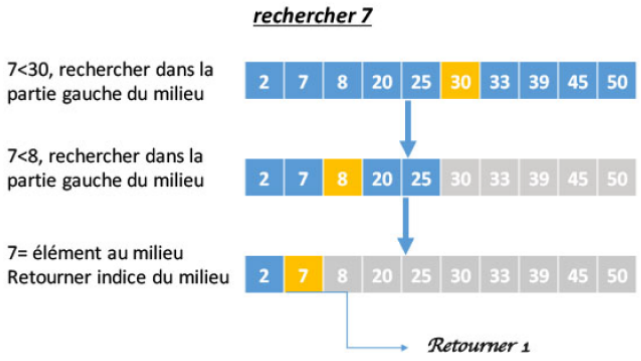
3. Calculer F_{50} par les deux fonctions ? Que remarquez-vous ?

Exercice 7

Soit L une liste triée. On cherche à vérifier si un élément x appartient à L ou non. Pour cela, on adopte le principe de la recherche dichotomique suivant :

- Comparez x avec l'élément du milieu.
- Si x correspond à l'élément du milieu, nous retournons l'indice du milieu.
- Sinon, si x est supérieur à l'élément milieu, alors x ne peut se situer que dans la moitié droite du sous-tableau après l'élément milieu. Alors x est maintenant à chercher dans la partie de liste se trouvant à gauche de l'élément milieu.
- Sinon, recherchez dans la moitié gauche.

Écrire une fonction **rech_dico(L,x)** qui effectue une recherche dichotomique de x dans L et retourne la position si $x \in L$.



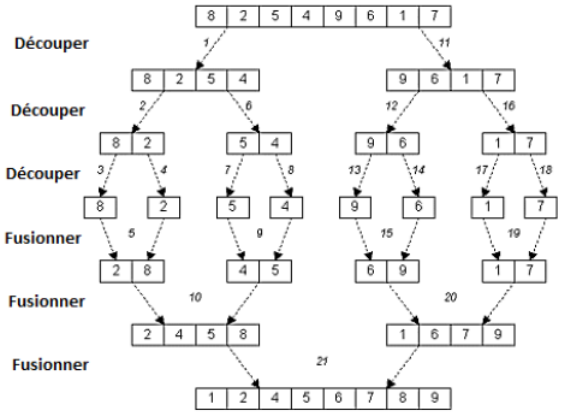
Exercice 8

1. Écrire une fonction **fusion(T1,T2)** qui prend en entrée deux listes $T1$ et $T2$ supposés triées en ordre croissant et renvoie une liste T contenant les mêmes éléments que $T1$ et $T2$ mais rangés par ordre croissant sans utiliser une fonction de tri.

   ```
   >>> print(Fusion([2,5,6], [1,4,5]))
   [1,2,4,5,5,6]
   ```

   Le principe du tri par fusion est le suivant :

   - On divise en deux moitiés la liste à trier.
   - On trie chacune d'entre elles récursivement.
   - On fusionne les deux moitiés obtenues pour reconstituer la liste triée.



2. Écrire une fonction **tri_fusion(L)** qui permet de trier une liste L en utilisant l'algorithme de tri par fusion.

```
>>> tri_fusion([8,2,5,4,9,6,1,7])
[1, 2, 4, 5, 6, 7, 8, 9]
```

Exercice 9

1. Écrire une fonction **matrice()** qui permet de retourner la matrice A suivante :

$$\begin{bmatrix} (0,0), & (0,1), & (0,2), & (0,3), & (0,4) \\ (1,0), & (1,1), & (1,2), & (1,3), & (1,4) \\ (2,0), & (2,1), & (2,2), & (2,3), & (2,4) \\ (3,0), & (3,1), & (3,2), & (3,3), & (3,4) \\ (4,0), & (4,1), & (4,2), & (4,3), & (4,4) \end{bmatrix}$$

2. Donner le résultat des instructions suivantes :

$$A[3][1], \quad A[1][3][-1], \quad \text{len}(A[0]), \quad \text{len}(A), \quad A[:], \quad A[:-1]$$

3. Donner l'instruction qui permet d'échanger la première ligne et la quatrième ligne de la matrice A.

   Un jeu de 52 cartes est constitué de quatre couleurs (Pique, carreau, Cœur et Trèfle) et pour chacune il y a le 2, 3, 4, 5, 6, 7, 8, 9, 10, valet, dame, roi et as.

4. Constituez deux listes : La liste `couleurs`, La liste `valeurs` constituée de tuples contenant la valeur en entier et la valeur en chaîne : exemple $(2, \text{"Deux"}), (1, \text{"Valet"})$.

5. Écrire une fonction **jeu_cartes()** qui renvoie un jeu de carte. Celui-ci sera une liste de tuples de la forme $[(\text{"Pique"}, 2, \text{"Deux"}), (\text{"Pique"}, 3, \text{"Trois"}), \ldots, (\text{"Trefle"}, 14, \text{"As"})]$.

   Pour mélanger aléatoirement les éléments d'une liste L, on utilise la bibliothèque `random` et l'instruction `random.shuffle(L)`.

6. Écrire un programme qui mélange les cartes puis les séparent en deux listes de 26 cartes (en simulant une distribution une par une).

Exercice 10

Toute suite finie d'entiers peut être décomposée de manière unique en une suite de séquences strictement croissantes maximales. En représentant une suite dans une liste l, la liste :

$$[1\,2\,5\,7\,2\,6\,0\,5\,2\,4\,6\,7\,8\,9\,3\,4\,6\,1\,2\,7\,8\,9\,4\,2\,3\,1\,5\,9\,7\,1\,6\,6\,3]$$

se décompose ainsi en la liste de 13 listes d'entiers suivante :

[1 2 5 7][2 6][0 5][2 4 6 7 8 9][3 4 6][1 2 7 8 9][4][2 3][1 5 9][7][1 6][6][3]

Les premiers éléments de ces séquences sont, en plus du premier élément de la liste, les éléments (soulignés sur l'exemple) qui sont inférieurs ou égaux à leur prédécesseur dans la liste ($l[i] \leq l[i-1]$).

1. Écrire une fonction **numRuptures(l)** qui renvoie le nombre d'indices i tels que $l[i] \leq l[i-1]$.

   ```
   >>> numRuptures([1 2 6 7 2 6 0 5 2 4 6 7 8 9 3 4 6 1 2 7 8 9 4 2 3 1 5 9 7 1 6 6 3])
   12
   ```

2. Écrire une fonction **rupture(l)** qui, étant donnée une liste l d'entiers, renvoie une liste contenant 0 en premier élément et les indices des éléments de l inférieurs ou égaux à leur prédécesseur en ordre croissant.

   ```
   >>> rupture([[1 2 6 7 2 6 0 5 2 4 6 7 8 9 3 4 6 1 2 7 8 9 4 2 3 1 5 9 7 1 6 6 3])
   [0, 4, 6, 8, 14, 17, 22, 23, 25, 28, 29, 31, 32]
   ```

3. Écrire une fonction **factorisation(l)** qui, étant donné une liste l d'entiers, renvoie une liste de listes d'entiers (résultat tel que celui donné dans l'exemple).