

## Généralités sur les listes

Une liste contient des éléments, on accède à ces éléments grâce à leur indice dans la liste en commençant à compter à partir de 0. Tapez le code suivant :

```
L = [-10,5,-10,8]#Crée une liste L
print(L[0])
print(L[1])
print(L[2])
print(L[3])
print(L[4])#Provoque une erreur ! Pourquoi ?
L[2] = 23#Modifie la valeur du troisième élément de L
print(L)
print(len(L))#Affiche le nombre d'éléments
print(L[len(L)-1])#Affiche le dernier élément de la liste
L = []#Crée une liste vide
L.append(3)#Rajoute 3 à la liste L
L.append(5)
print(L)
M = [2,3]
N = L + M#Concaténation de listes
print(N)
P = M*5
print(P)
```

### Attention : la numérotation commence à 0

 L[1] n'est pas la valeur du premier élément de la liste mais du deuxième !

**Exercice 1** (\*). Écrire une fonction `AfficherListe`, qui dépend d'un paramètre - une liste - et qui, à l'aide d'une boucle `for`, affiche tous les éléments de la liste<sup>1</sup>.

**Exercice 2** (♫). Écrire une fonction `SommeListe`, qui dépend d'un paramètre - une liste - qui renvoie la somme de tous les éléments de la liste. On

1. C'est un rare exemple de fonction qui ne comporte pas de `return`.

pourra s'aider d'une variable `S` à qui on ajoute à l'étape `k` d'une boucle `for` le `k`-ième élément de la liste. Ainsi, `SommeListe([1,2,3,-4])` vaudra 2.

**Exercice 3** (♫). 1. Écrire une fonction `Maxi`, qui dépend d'un paramètre - une liste de nombres - et qui renvoie la plus grande valeur de la liste. Pour cela, on créera une variable `M`, et on parcourra les éléments de la liste. Si on rencontre un élément plus grand que `M` alors on change la valeur de la variable `M` qui prendra cet élément.

2. Modifier la fonction précédente, pour qu'elle renvoie également, la/une position du maximum<sup>2</sup>. Ainsi, `Maxi([1,4,4,3])` renverra (4,1) ou bien (4,2).

**Exercice 4** (♫). Écrire une fonction `EstDedans` qui dépend de deux paramètres : une liste `L` et un élément `x`. Cette fonction renverra `True` si `x` est un élément de la liste `L` et `False` sinon. Pour ce faire, pour chaque élément de la liste, vérifier si `x` est égal à cet élément. Par exemple, `EstDedans([1,4,3],4)` renvoie `True` tandis que `EstDedans([1,4,3],2)` renvoie `False`.

### Attention à True et False

 `True` et `False` sont deux variables particulières, appelées variables booléennes, en particulier ce ne sont ni des nombres, ni des chaînes de caractères, on n'écrira donc pas "True" mais juste `True` et on oubliera pas la majuscule en début de mot.

**Exercice 5** (♫). Si `L=[1,5,7,8,9]` alors `L[2]=7` est au milieu de la liste. Si `M=[1,5,7,8,9,11]`, alors 7 ou 8 sont à peu près au milieu de la liste. Soit une liste que l'on suppose **triée par ordre croissant** (comme `L` et `M`).

1. Quelle commande utiliseriez-vous pour obtenir la plus grande/petite valeur de la liste ?
2. Quelle commande utiliseriez-vous pour obtenir un élément qui est à peu près au milieu de la liste ?

2. Le maximum pouvant être atteint plusieurs fois, dans ce cas-là on renverra une des positions possibles, on pourrait complexifier l'exercice en demandant de renvoyer la liste des indices du maximum.

On veut savoir si un élément  $x$  est dans cette liste sans tester tous les éléments de la liste contrairement à l'exercice précédent. Pour cela, si  $x$  est strictement inférieur au plus petit élément de la liste ou strictement supérieur au plus grand élément de la liste, alors  $x$  n'est pas dans la liste. Dans le cas contraire, on compare  $x$  à l'élément au milieu, si  $x$  est plus grand on va regarder la partie droite de la liste, sinon la partie gauche. Puis on continue en divisant à chaque fois la taille de l'intervalle par deux.

- Appliquer l'algorithme à la main, avec  $x=19$  et  $L=[1,2,3,5,12,14,16,18,22,24,25]$ . Compter le nombre de comparaisons faites à la main, et compter le nombre de comparaison que vous auriez du faire si vous aviez appliqué l'exercice 4 à la main.

Cet algorithme utilise la dichotomie.

- Implémenter cet algorithme avec une fonction `Dichotomie` à deux paramètres :  $L$  et  $x$ .
- Tester cet algorithme sur des exemples variées (liste ayant un nombre pair ou impair d'éléments, élément en début de liste, fin de liste, milieu de liste etc.).
- Comparer le nombre de comparaisons nécessaires dans le pire des cas dans cet exercice ainsi que dans le précédent en fonction de la taille de la liste.

**Exercice 6.** Imaginons une collection de vignettes Panini :

- On achète un album vierge qui contient  $N$  emplacements dans lesquels on peut coller une vignette destinée à cet emplacement.
- On achète ensuite une vignette, prise au hasard parmi les  $N$  possibles, vignette qu'on colle à l'emplacement prévu.

La collection est finie lorsque l'on a toutes les vignettes.

- Créer une fonction `CreerAlbum` qui dépend d'un paramètre  $N$  et qui crée un album vide à  $N$  emplacements.
- Créer une fonction `AlbumPlein` qui vérifie si l'album est rempli.
- Créer une fonction `Achat` qui tire au hasard une vignette et l'ajoute à l'album si l'album n'avait pas cette vignette.
- Créer une fonction `Remplissage` qui achète des vignettes jusqu'à ce que l'album soit rempli.
- Créer une fonction qui évalue le nombre moyen de pochettes à acheter pour remplir son album.

- Tracer, sur un même graphique, la courbe du nombre moyen de pochettes à acheter en fonction de  $N$  obtenue par simulation et la courbe de  $N \mapsto N \ln(N)$  qui est une approximation théorique de cette valeur quand  $N$  grand.

On pourra s'aider des codes suivants :

```
#Chargement des bibliothèques (à mettre en début de script)
import numpy.random as rd#Appel d'une bibliothèque pour
#les nombres aléatoires
import matplotlib.pyplot as plt#Appel d'une bibliothèque
#pour les images
import numpy as np

#Zone de script principal
L = [0]*5
print(L)
n = rd.randint(5)#Tire un nombre entier uniformément entre 0 et 4
a = np.log(2.7)#logarithme népérien de 2.7

A = [1,2,3,4]#Liste des abscisses
O = [1,4,9,16]#Liste des ordonnées
plt.plot(A,O)#Trace la liste de points (1,1), (2,4), (3,9) et (4,16)
```

## Quand il y en a plus, y en a encore !

**Exercice 7.** Créer un compte sur [le site du projet Euler](#) et résoudre un maximum de problèmes, vous devez écrire un code qui mette une minute ou moins à s'exécuter. Ou continuez les problèmes si vous en avez déjà fait lors du TP précédent.