Listes (tableaux)

◇ Si on doit manipuler les résultats de 10 mesures, on ne définit pas 10 variables **mesure1**, **mesure2**, ..., **mesure10**. On définit une seule variable qui va contenir l'ensemble des 10 valeurs. En PYTHON, une telle variable sera du type liste (*list*).

Remarque. Certains langages possèdent également une notion de tableau. La différence entre un tableau et une liste est ténue, disons simplement que les tableaux sont gérés plus efficacement en mémoire mais les listes se manipulent plus simplement. □

```
>>> mesures = [1.5,2,3.5,5.9,-0.1,10.2]
>>> print(type(mesures))
<class 'list'>
>>> print(len(mesures))   # Nombre d'éléments de la liste
6
>>> print(mesures[0])     # Le premier élément (attention !)
1.5
>>> print(mesures[1])     # Le deuxième élément (attention !)
2
>>> print(mesures[-1])    # Le dernier élément
10.2
>>> mesures[3] = 4        # Changer la valeur d'un élément
>>> print(mesures)
[1.5, 2, 3.5, 4, -0.1, 10.2]
>>> print(4 in mesures)   # Tester si un élément est présent
True
>>> print(7 not in mesures)
True
```

⚠ Les différents éléments d'une liste ne doivent pas nécessairement être du même type. Cependant, en pratique, une liste regroupe un ensemble de valeurs qui ont un certain rapport entre elles et elles ont en général le même type.

⚠ Dans une liste de n éléments, le premier élément a pour indice 0 et le dernier $n-1$.

◇ La commance **append** permet de modifier une liste en ajoutant un nouvel élément à la fin.

```
>>> L = [1,2,3]
>>> L.append(7)
>>> print(L)
[1, 2, 3, 7]
```

◇ La commande **pop** permet de retirer l'élément qui se trouve à une certaine position.

```
>>> L.pop(2)
3
>>> print(L)
[1, 2, 7]
```

◇ L'opération **+** appliquée sur deux listes réalise leur *concaténation* (ce qui consiste à créer une nouvelle liste réunissant les éléments des deux listes).

```
>>> L = [1,2]
>>> T = [3,4]
>>> U = L+T
>>> print(U)
[1, 2, 3, 4]
```

Exemple. Création d'une liste élément par élément. Le code ci-contre construit la liste des carrés des nombres compris entre 2 et 100 (au sens large).

```
L = []
for i in range(2,101):
    L.append(i**2)
```

Exemple. **(À retenir.)** On peut très simplement parcourir tous les éléments d'une liste à l'aide d'une boucle *for*. On donne ci-contre une fonction **somme(L)** qui calcule et renvoie la somme des éléments de la liste **L**.

```
def somme(L):
    s = 0
    for k in range(0,len(L)):
        s = s+L[k]
    return s
```

Exemple. **(À retenir.)** Déterminer le maximum d'une liste (de nombres entiers ou de flottants) contenant au moins un élément. La fonction de recherche du maximum existe déjà dans PYTHON (**max**, on a également **min**). Il faut retenir le raisonnement *algorithmique* de cet exemple.

```
def maximum(L):
    m = L[0]
    for i in range(1,len(L)):
        if L[i]>m:
            m = L[i]
    return m
print(maximum([1,3,8,2,7,4]))
```


◇ On dispose d'une autre opération sur les listes : le découpage (*slicing*). Si **L** est une liste, la notation **L[a:b]** désigne une nouvelle liste constituée des éléments de **L** dont l'indice i est tel que $a \leqslant i < b$ (attention à la deuxième inégalité qui est stricte). La liste **L** de départ est inchangée.

◇ On peut avoir des listes de listes. Par exemple si on définit **L = [[1,2],[3,4,5]]**, alors **L[1]** désigne la liste **[3,4,5]** et **L[0][1]** donne la valeur 2.

Exercices pour la semaine prochaine

Question 1. Écrire un programme PYTHON construisant la liste `L` de taille 100 dont les éléments sont, alternativement, 0 et 1.

Question 2. Écrire une fonction `sommep(L)` qui calcule et renvoie la somme des éléments *positifs* de la liste `L`.

Question 3. Écrire une fonction `minimum(L)` qui détermine et renvoie le minimum des éléments de la liste `L`.

Question 4. Quels résultats donnent les instructions `sommep([])` et `minimum([])` ?

Corrections

Q 1. On propose plusieurs possibilités. **Méthode 1.**

```
L = []
for k in range(0,100):
    if k%2==0:
        # k est pair, on ajoute 0 à L
        L.append(0)
    else:
        L.append(1)
```

Méthode 2.

```
L = []
for k in range(0,100):
    L.append(k%2)
    # Puisque k%2 vaut alternativement 0 ou 1
```

Méthode 3.

```
L = []
for k in range(0,50):
    L.append(0)
    L.append(1)
```

Q 2.

```
def sommep(L):
    s = 0
    for k in range(0,len(L)):
        if L[k]>=0:
            s = s+L[k]
    return s
```

Q 3.

```
def minimum(L):
    m = L[0]
    for k in range(1,len(L)):
        if L[k]<m:
            m = L[k]
    return m
```

Q 4. L'instruction `sommep([])` renvoie la valeur 0 (dans la fonction, la variable *s* reste égale à 0). L'instruction `minimum([])` déclenche une erreur (lorsque l'on effectue l'instruction `m=L[0]`).