6 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 : 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
1.5
>>> print(mesures[1])     # Le deuxième élément
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(5 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.

◇ Découpage d'une liste (*slicing*).

⚠ Dans une liste de n éléments, le premier élément a pour indice 0 et le dernier $n-1$. Dans une construction du type **`L[a:b]`**, on extrait de la liste L tous les éléments d'indice i tel que $a \leqslant i < b$ (a est le premier indice qui est pris et b le premier indice exclu).

```
>>> L = [0,1,2,3,4,5,6,7,8,9]
>>> print(L[1:5])
[1, 2, 3, 4]
>>> print(L[:4])
[0, 1, 2, 3]
>>> print(L[3:])
[3, 4, 5, 6, 7, 8, 9]
```

```
>>> L = [2,1,3,1]
>>> L.remove(1); print(L)
[2, 3, 1]
```

```
>>> L.append(10); print(L)
[2, 3, 1, 10]
```

◇ Sommes et produits avec des listes :

```
>>> L = [1,2]
>>> L+[3,4]
[1, 2, 3, 4]
>>> L*2
[1, 2, 1, 2]
```

◇ On peut avoir des listes de listes :

```
>>> L = [[1,2],[3,4,5]]
>>> L[1]
[3, 4, 5]
>>> L[0][1]
2
```

◇ On peut très simplement parcourir tous les éléments d'une liste à l'aide d'une boucle *for*. Par exemple pour calculer la somme des termes *positifs* d'une liste :

```
L = [0,-1.1,2,3.6,-0.5,6]
s = 0
for x in L:
    if x>=0:
        s = s+x
print("Somme des termes >=0 :",s)
```

```
Somme des termes >=0 : 11.6
```

Exemple Python. Déterminer le maximum d'une liste (de nombres entiers ou de flottants) contenant au moins un élément. On donne deux versions : la première est celle que l'on peut retrouver dans tout langage de programmation, la seconde utilise les spécificités de PYTHON (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]))
```

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


8