TP 3 : listes

1 Définir une liste par extension

On en donne les éléments, entre deux crochets, séparés par des virgules.

```
liste=[1,2,'a','b',5]
print(liste);type(liste);len(liste)
```

2 Accéder aux données d'une liste

```
liste[0],liste[1],liste[len(liste)-1]
liste[-1],liste[-2],liste[5]
liste[1:3],liste[:]
M=[['a','b'],['c','d']];M[0];M[0][1];
print(M[0][0],M[0][1],'\n',M[1][0],M[1][1])
```

3 Modifier une liste

```
liste[0]='toto';liste[3]=[9,8,7];print(liste)
```

4 Tester l'appartenance à une liste

Exercice 1 :
Écrire un programme qui prenne en argument une liste et une donnée, et vérifie si la donnée est un élément de la liste et renvoie `True` dans ce cas, et `False` dans le cas contraire. Comparer avec ce qui suit !

```
2 in liste;7 in liste;5.0 in liste
```

5 Parcourir une liste

```
for variable in liste:
    print(variable,end=';')
```

Proposer une autre façon d'obtenir cet affichage, en parcourant les indices des éléments.

```
L1=[1,2,3]
for variable in L1:
    variable=0
for _ in range(len(L1)):
    L1[k]=0
```

6 Exercices

Exercice 2 :
Écrire un programme qui calcule la somme des termes d'une liste numérique.

Exercice 3 :
Écrire un programme qui calcule la moyenne des termes d'une liste numérique.

Exercice 4 :
Écrire un programme qui cherche le maximum des termes d'une liste numérique.

Exercice 5 :
Écrire un programme qui cherche l'indice porté par le maximum des termes d'une liste numérique.

7 Les méthodes de la classe liste

- `liste.append(x)` : sert à ajouter `x` à la fin de la liste `liste` ;
- `liste.pop()` : sert à retirer et renvoyer le dernier élément de la liste `liste` ;
- `liste.extend(liste2)` : sert à ajouter les éléments de la liste `liste2` à la fin de la liste `liste` ;
- `liste.insert(i,x)` : sert à ajouter `x` à la position `i` de la liste `liste` ;
- `liste.pop(i)` : sert à retirer et renvoyer l'élément en position `i` de la liste `liste` ;
- `liste.remove(x)` : sert à retirer la première occurrence de `x` dans la liste `liste` ;
- `liste.index(x)` : sert à renvoyer la première position de `x` dans la liste `liste`. Renvoie un message d'erreur s'il n'y en a pas ;
- `liste.count(x)` : sert à compter le nombre d'occurrences de `x` dans la liste `liste` ;
- `liste.sort(x)` : sert à trier la liste `liste` par ordre croissant ;
- `liste.reverse(x)` : sert à renverser l'ordre des éléments de la liste `liste`

En général, il vaut mieux reprogrammer ces fonctions à la main lorsqu'on en a besoin (à l'exception des trois premières, qui servent si on veut conserver la même liste).

Exercice 6 :
Reprogrammer ces fonctions.
Créer une fonction qui cherche toutes les occurrences de `x` et la construise la liste des indices auxquels il apparaît.
Créer une fonction qui supprime teoues les occurrences de `x`

8 Définition en compréhension

```
Liste = [x**2 for x in range(10)];Liste2 = [x**2 for x in range(10) if x%2==0]
```

Exercice 7 :
Écrire, avec une boucle `for` puis en compréhension la liste des multiples de 7 entre 0 et 100.
Même question avec la liste des diviseurs d'un entier `n`.

Exercice 8 :
Trouver tous les couples d'entiers positifs dont la somme est 12.

9 Danger : copier une liste

```
Liste=[1,2,3];CopieListe=Liste;Copie;Liste[0]='a';Copie;Copie[1]='b';Liste
```

Pour pallier ce problème, on utilise parfois `L[:]` ou mieux : la commande `deepcopy` de la bibliothèque `copy`.