Partie II : Algorithmique et Programmation

SOMMAIRE

Chapitre 4 : Les Listes

Les listes

I. Tableaux

Les variables, telles que nous les avons vues, ne permettent de stocker qu'une seule donnée à la fois. Or, pour de nombreuses données, comme cela est souvent le cas, des variables distinctes seraient beaucoup trop lourdes à gérer. Heureusement, tous les langages proposent des structures de données permettant de stocker l'ensemble de ces données dans une «variable commune». Ainsi, pour accéder à ces valeurs il suffit de parcourir la variable en utilisant les indices.

Un tableau est une structure référencée par un seul nom et qui est composée de plusieurs données stockées de manière contiguë en mémoire (les unes à la suite des autres).

Au niveau de la mémoire, un tableau est donc une suite de cases (espace mémoire). La taille de chacune des cases est conditionnée par le type de donnée que contient cette case.

Voici donc une manière de représenter un tableau :

Donnée	Donnée	Donnée	Donnée	Donnée	Donnée	….
0	1	2	3	4	5	

II. Listes (Tableaux en PYTHON)

En python, les tableaux sont appelés autrement : « les listes ».

A la différence des tableaux où on ne peut stocker que des données de même type, une liste peut contenir des données de types différents.

La création d'une liste se fait simplement par insertion des différentes valeurs entre crochets [square brackets] en les séparées par virgules.

Exemples :
L = [ ] # Pour déclarer une liste vide
L = [10, 28 , 3, 9, -5] # Pour déclarer une liste de 5 entiers
L = [2.5 , False, "Casa" , 50] # Pour déclarer une liste de quatre données de types différents

III. Accès aux valeurs d'une liste

On peut accéder aux valeurs de la liste en indiquant l'indice ou les indices entre crochets qui peuvent être positifs ou bien négatifs.

Ex : Liste des matières que vous étudiez.
matieres = ["MATH", "PHY", "SI", "INFO", "FR", "EN", "TRA", "SPO"]

MATH	PHY	SI	INFO	FR	EN	TRA	SPO
0	1	2	3	4	5	6	7
-8	-7	-6	-5	-4	-3	-2	-1

Indices des déférentes valeurs de la liste

N.B : Les indices négatifs sont utilisés pour accéder aux valeurs d'une liste à partir de la fin. L'indexage dans ce cas commence à -1.

IV. Listes sont modifiables (mutable)

On peut modifier les éléments d'une liste par la simple affectation, en indiquant le nom de la liste et l'indice de l'élément à modifier.

Exemple : Modification de PHY à l'indice (1 ou -7) en PHYSIQUE
matieres[1] = "PHYSIQUE" # ou : matieres[-7] = "PHYSIQUE"

V. Opérations sur les listes

L1= [0, 1, 2, 3, 4]
L2 = [5, 6, 7, 8, 9]

EXPRESSION	**RESULTAT**	**DESCRIPTION**
L1 + L2	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	Concaténation de deux listes
L1*2	[0, 1, 2, 3, 4, 0, 1, 2, 3, 4]	Répétition d'une liste
3 in L1	True	Teste d'appartenance
5 not in L2	False	Teste de non appartenance.
for x in L1 : print(x, end='')	0 1 2 3 4	Itération d'une liste première méthode (par valeur)
for i in range(0, len(L1)) : print(L1[i], , end='')	0 1 2 3 4	Itération d'une liste deuxième méthode (par indice)

VI. Manipulation des listes (Fonctions)

Python inclut la liste ci-dessous des fonctions de manipulation des listes :

FONCTION	**DESCRIPTION**
len(liste)	Retourne la longueur d'une liste (le nombre de ces éléments)
max(liste)	Retourne la plus grande valeur dans la liste.
min(liste)	Retourne la plus petite valeur dans la liste.

VII. Manipulation des listes (Méthodes)

Python inclut la liste ci-dessous des méthodes de manipulation des listes :

METHODE	DESCRIPTION
List.append(elem)	Ajoute l'élément **elem** à la fin de la liste. A = [4, 6 , 9] A.append(0) A = [4, 6, 9, 0]
List.count(elem)	Compte combien de fois l'élément **elem** est présent dans la liste. A=[2, 3, 2, 2 , "AB"] A.count(2) donne 3 A.count(9) donne 0
List.extends(seq)	Ajoute le contenu de la séquence **seq** à la fin de la liste. A = [1 , 2, 3 , 4] B = [5, 6] C = A.extends(B) donne C =[1, 2, 3, 4, 5, 6]
List.index(elem)	Retourne le plus petit indice dans lequel l'élément **elem** apparait dans a liste. A = [3,4,5,9,4] A.index(4) retourne 1
List.insert(index, elem)	Insert **elem** à la position **index** dans la liste. A = [0, 8, 9] A.insert(5,1) A= [0, 5, 8, 9]
List.pop(i)	Retourne et supprime l'élément d'indice i, l'indice par défaut est -1. A = [2 , 5 , -3, 9] L'appel à x = A.pop() donne : A = [2 , 5, -3] et x=9
List.remove(elem)	Supprime **elem** de la liste. A = [2 , 5 , -3, 9] A.remove(5) => A =[2 , -3 , 9]
List.reverse()	Inverse l'ordre des éléments dans la liste
List.sort()	Trie la liste dans un ordre croissant par défaut List.sort(reverse=True) pour l'ordre décroissant.

VIII. Les sous-liste ou tranches

Les tranches sont comme son nom l'indique, des parties découpées. Pour créer une tranche, tout se passe entre les crochets quand on appelle la liste.

```
liste = [1, 2, 3, 4]
tranche = liste[0:2]
print(tranche)
```

Si vous essayez ce code, vous verrez apparaître "[1,2]" sur votre écran.
En fait on crée une tranche en mettant "[index du début (inclus) **:** index de la fin (exclus)]" à la place de l'index seul. Ici on a demandé de copier dans tranche que liste[0] et liste[1].

```
liste = [1, 2, 3, 4]
tranche = liste[:3]
print(tranche)
```

Ceci permet de copier du début jusqu'à l'index demandé (exclus). Ici on obtient [1, 2, 3]

```
liste = [1, 2, 3, 4]
tranche = liste[1:]
print(tranche)
```

Ceci permet de copier de l'index demandé (inclus) jusqu'à la fin. Ici on obtient [2, 3, 4]

```
liste = [1, 2, 3, 4]
tranche = liste[:]
print tranche
```

Ceci crée une copie complète de "liste". Ici on obtient [1, 2, 3, 4]