Chapitre 28

Algorithmique et listes

I. Notion de liste dans le langage Python

Définition 1: *Liste dans un langage Python*

Dans le langage Python, une liste `L` est un tableau de valeurs dont les éléments sont indexés à partir de 0. Ces éléments peuvent être de différents types.

Exemple 1:

Pour définir la liste `L` constituée des éléments `5`, `"herve"`, `8.1` et `"A"`, on écrit `L=[5,"herve",8.1,"A"]`. Attention, comme les éléments sont indexés a partir de 0, on a `L[0]=5`, `L[1]="herve"`, `L[2]=8.1` et `L[3]="A"`.

Remarque :

- On peut afficher le contenu d'une liste `L` à l'aide de l'instruction `print(L)`.
- Lorsqu'on donne une liste sous cette forme, avec tous ses éléments entre crochets (nous verrons d'autres façons de procéder dans le suite), on dit qu'elle est définie en extension.
- L'ordre d'apparition dans la liste est très importante. Ainsi les listes `L=[5,"herve",8.1,"A"]` et `M=[5,"A",8.1,"herve"]` sont différentes

Définition 2: *Indices des éléments d'une liste*

Lorsqu'une liste `L` est définie, on peut manipuler (c'est à dire modifier, utiliser pour un calcul, ...) tous ses éléments en utilisant `L[i]` ou `i` est l'indice de l'élément dans la liste.

Exemple 2:

On considère le programme ci-contre dans lequel apparait une liste nommée pairs.

```
1 pairs=[2,4,6,7,9]
2 pairs[3]=8
3 pairs[4]=pairs[4]+1
4 print(pairs)
```

- A la ligne 1, on définit le contenu de cette liste. On a donc la configuration `pairs[0]=2`, `pairs[1]="4"` ...
- A la ligne 2, on modifie la valeur en `pairs[3]` qui devient `8`. On a alors `pairs=[2,4,6,8,9]`.
- A la ligne 2, on modifie la valeur en `pairs[4]` qui devient `pairs[4]+1` c'est à dire `9+1=10`. On a alors `pairs=[2,4,6,8,10]`.
- A la ligne 4, on affiche la liste `[2,4,6,8,10]`

Définition 3: *Ajout d'un élément à la fin d'une liste*

On peut ajouter un élément `x` à la fin d'une liste `L` avec l'instruction `L.append(x)`.

Exemple 3:

Soit la liste `pairs=[2,4,6,8,10]`. L'instruction `pairs.append(12)` ajoute l'élément `12` en fin de liste, on a donc `pairs=[2,4,6,8,10,12]`

Remarque :

Lorsque l'on veut définir une liste `L` par ajout successifs, on peut commencer par définir une liste vide `L=[]` puis lui ajouter des éléments avec `append`.

Exercice 1 Écrire un programme respectant les étapes suivantes.

- Créer une liste `L` contenant dans l'ordre les éléments 5, 3, 4 et 8.
- Modifier la liste `L` de sort qu'elle devienne `[5, 3, 4, 9]`.
- Ajouter l'élément 10 à la fin de cette liste.
- Afficher la liste `L`.

Exercice 2 Écrire un programme respectant les étapes suivantes.

- Créer une liste `L` contenant 4 réels de votre choix.
- Affecter à une variable `a` la somme des deux derniers éléments de la liste `L`
- Ajouter l'élément 16 à la fin de la liste `L`.
- Afficher la liste `L`.

II. Quelques instructions utiles

Dans ce paragraphe, `L`, `M` et `N` désignent trois listes.

Instructions	Explication
`len(L)`	renvoie la taille de la liste `L`, c'est à dire son nombre d'éléments.
`del(L[i])`	supprime l'élément d'indice `i` de `L` (et décale les suivants vers la gauche).
`L.insert(i,x)`	insère `x` en `L[i]` (et décale les éléments d'indice `k>i` vers la droite).
`L.count(x)`	renvoie le nombre d'occurrence de la valeur `x` dans `L`.
`L=M+N`	concaténation de `M` et `N` : `L` contient les éléments de `M` suivi de ceux de `N`.
`L.sort()`	`L` est modifiée de sort que ses éléments soient dans l'ordre croissant.
`if x in L :`	test si `x` est un élément de `L`.
`if x not in L:`	test si `x` n'est pas un élément de `L`.

⚠ Remarque :

Comme `range(a)` désigne les entiers de `0` à `n-1` inclus, le bloc ci-contre permet d'afficher tous les éléments de `L`. Si `len[L]=4`, il affiche les 4 éléments `L[0]`, `L[1]`, `L[2]` et `L[3]`

```
for i in range(len[L]):
        print(L[i])
```

Exercice 3 :

On considère le programme ci-contre. Le compléter pour qu'il affiche la taille de la liste `L`, puis qu'il ajoute le nombre 3 en première position dans `L`, puis affiche `5 est dedans` si le nombre 5 est présent dans la liste `L`.

```
import random
L=[]
for i in range(1,10):
        if random.random()<0.5:
                L.append(random.randint(1,10))
```

Exercice 4 Écrire un programme dans lequel deux listes `L=[0,1,1,0,0]` et `M=[0,1,0]` sont définies, et qui :

1. Crée une liste `N` consitutée des valeurs de `L` suivies de celles de `M`.
2. Trie la liste `N` dans l'ordre croissant.

Exercice 5 Écrire un programme dans lequel une liste `L=[1,4,29,256]` est définie, et qui :

1. Affecte un élément aléatoire entre 0 et 3 à une variable `i`.
2. Affiche l'élément d'indice `i` de la liste `L` puis le supprime.

III. Itération sur les éléments et listes en compréhension

Définition 4: *Itération sur les éléments d'une liste*

Soit `L` une liste, `for i in L` veut dire « pour `i` prenant successivement pour valeur les éléments de `L` ».

Exemple 4:

Le tableau donnant l'évolution de l'exécution du programme ci-contre est :

```
taille_cm=[173,145,156,169]
for i in taille_cm:
        print(i/100)
```

i	173	145	156	169
Affichage	1.73	1.45	1.56	1.69

La variable `i` a pris pour valeurs tous les éléments de la liste `taille_cm` (dans l'ordre).

Exercice 6 :

Décrire le fonctionnement du programme ci-contre.

```
L=[5,4,3,2,1,0]
for k in L:
        print(2*k)
```

Exercice 7 :

Décrire le fonctionnement des programmes ci-contre.

```
L=[1,2,3,4,5]
p=1
for j in L:
        p=p+j
```

```
L=["chou","hibou","sou"]
for k in L:
        print(k+"a")
```

Exercice 8 Écrire un programme dans lequel la liste `L=[59,41,62,99,74]` est définie, et qui calcule, en itérant sur les éléments de `L`, la somme des termes de la liste `L` dans une variable `a` initialisé à 0.

Exercice 9

1. Définir une liste de 5 entiers puis faire afficher le double de ses éléments.
2. Définir une liste de 8 entiers relatifs de votre choix puis faire calculer la somme de leurs carrés.

L'ensemble des entiers pairs entre 0 et 1000 peut s'écrire $[0;2;4;\ldots;998;1000]$ mais ce n'est par totalement satisfaisant car on n'a pas écrit tous les éléments il y en a trop !).
Ainsi, on préférera l'écrire $[2i|i \in \mathbb{N} et\, i \leqslant 500]$ qui se lit « l'ensemble des nombres de la forme $2i$ (expression) tels que i (variable) est un entier naturel inférieur ou égale à 500 (ensemble) ». Il en est de même pour les listes.

Définition 5: *Liste définie en compréhension*

On définit une liste `L` en compréhension par `L=[` *expression* `for` *variable* `in` *ensemble* `]` ou
`L=[` *expression* `for` *variable* `in` *ensemble* `if` *condition* `]`

Exemple 5:

On considère trois listes définies par `P=[2*i for i in range(0,501)]`, `IMP1=[i+1 for i in P]` et `IMP2=[i+1 for i in P if i<100]`.

- `P` contient des nombres de la forme `2*i` avec `i` entier entre `0` et `500` soit `P=[0,2,4,...,1000]`.
- `IMP1` contient des nombres de la forme `i+1` avec `i` dans `P` soit `IMP1=[1,3,5,...,1001]`.
- `IMP2` contient des nombres de la forme `i+1` avec `i` dans `P` qui sont supérieur à `100` c'est à dire avec `i` pair entre `0` et `98` soit `IMP2=[1,3,5,...,99]`.

Exercice 10 Définir en compréhension la liste de tous les carrés des entiers :

1. de 0 à 30
2. de 0 à 10 puis de 20 à 30