

Les listes

1 Définition en extension

Une liste, de type `list`, est une suite finie ordonnée d'éléments de types quelconques (qui n'ont pas besoin d'être tous les mêmes: on peut mélanger les nombres, les booléens, les chaînes de caractères, les listes, ...)

SYNTAXE: pour définir une liste, on place les éléments entre crochets en les séparant par des virgules:

$[x_1, x_2, \dots, x_n]$

On dit alors que la liste est définie **en extension**.

Remarque 1 Dans une liste, l'ordre des éléments compte.

Exemple 1 :

```
[1, 2, 3] == [2, 1, 3]
```

Que renvoie l'ordinateur?

Exemple 2 :

(1) Liste vide: `a=[]`

La liste vide est plus utile qu'il n'y paraît. Elle est souvent le point de départ d'une liste que l'on veut construire (pensez à l'initialisation d'une boucle, par exemple).

(2) On donne:

```
[1.5, "bonjour"]; [1, 5, "bonjour"]  
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Combien de termes y a-t-il dans chaque liste?

2 Opérations

Opérations sur les listes	instruction PYTHON
égalité	<code>==</code>
fusion	<code>+</code>
répétition n fois	<code>n*</code>
longueur	<code>len()</code>
appartenance	<code>in</code>
somme	<code>sum()</code>

Exemple 3 :

```
a=[1,3]+[2,4,5]; 4 in a ; "4" in a  
3 * [1,2]  
sum([1,2,3,4,5,6,7])
```

Que renvoie l'ordinateur?

3 Accès aux éléments d'une liste

L'affectation d'une liste à une variable permet d'accéder individuellement aux caractères de la liste.

Si `L` désigne une liste et `i, j` deux variables de type entier:

Opérations sur les listes	instruction PYTHON
Nombre d'éléments de <code>L</code>	<code>len(L)</code>
Accès à l'élément de <code>L</code> situé en position <code>i</code>	<code>L[i]</code>
Extraction des éléments de <code>L</code> compris entre les positions <code>i</code> et <code>j-1</code>	<code>L[i:j]</code>
Extraction des éléments de <code>L</code> compris entre les positions <code>i</code> et la dernière	<code>L[i:]</code>
Ajout de la valeur de <code>x</code> à la fin de la liste <code>L</code>	<code>L.append(x)</code>
Suppression de la valeur de <code>x</code> de la liste <code>L</code>	<code>L.remove(x)</code>

Exemple 4 :

```
L1=["vive","les","maths"]; L1.append("en BCP"); L2=L1+["en BCP"]
```

Que renvoie l'ordinateur?

Remarque 2 La commande `remove` renvoie un message d'erreur si la valeur de `x` ne figure pas dans la liste `L`, et supprime uniquement la première occurrence de la valeur de `x` si celle-ci y figure plusieurs fois.

Remarque 3 :

(1) La numérotation des éléments d'une liste commence toujours au numéro 0, donc l'élément `L[k]` et le $(k+1)$ ième élément de la liste `L`.

(2) Le dernier élément d'une liste de longueur `n` est numéroté `n-1` (et non `n`). Si l'on dépasse l'indice maximal `n-1`, Python renvoie un message d'erreur.

(3) L'utilisation d'un négatif permet d'accéder aux éléments à partir de la fin de la liste. Ainsi `L[-1]` désigne le dernier élément d'une liste, `L[-2]` l'avant-dernier, etc ...

L'accès au dernier élément en utilisant l'indice `-1` est surtout utile quand on ne connaît pas la longueur de la liste.

Remarque 4 (objets mutables):

Il est possible de modifier (ou d'effacer) un élément d'une liste au sein même de cette liste. On dit que les listes sont **mutables**.

La mutabilité des listes oblige à la prudence quand on souhaite dupliquer une liste. Par exemple, si on tape dans la console:

```
x=[5,2,9]; y=x
```

on dispose en fait d'une seule et même liste qui possède deux noms. Si l'on écrit alors dans la console :

```
x[2]=10; x; y
```

la liste devient `[5,2,10]` mais son adresse ne change pas en mémoire. Les variables `x` et `y` pointent donc vers la même liste, et ont donc la même valeur

Tout se passe donc comme si la mutation de `x` s'était répercutée sur `y`.

La syntaxe pour dupliquer la liste, c'est-à-dire créer une autre variable de même valeur que `x`, est :

```
y=x[:]
```

Exemple 5 :

```
K=[10,15,12]
L=K
M=L
N=M[:]
M[1]=17
N[0]=19
```

Donner les valeurs des listes `K,L,M` et `N`.

4 Définition en compréhension

SYNTAXE:

```
expression for indice in liste donnée
```

Il est même possible d'utiliser plusieurs indices en écrivant plusieurs instructions du type **for** indice **in** liste donnée.

Exemple 6 : Donner les valeurs des listes `L1`, `L2` et `L3`:

```
L=[i**2 for i in range(-3,6)]
L1=[L[2*i+1] for i in range(3)]
L2=[x**2 for x in L]
L3=[(x,y) for x in [1,2,3] for y in [3,1,4] if x != y]
```