Informatique

7

Listes

Résumé

Créer une liste vide	L=[]	
Créer une liste d'objets	L=[objet1,objet2,…,objetn]	Liste dont les indices existent mais avec des valeurs n'existant pas : L = [None,None…] L[0] ne retourne pas d'erreur, ni de valeur
Taille	len(L)	Renvoie le nombre de termes de L, ou dernier séparateur - Le dernier élément de L s'obtient en écrivant $L[len(L)-1]$
Texte	L=['a','b',…] L='ab…'	Même si le type est différents, on les utilise de la même manière
Ajouter un objet	L.append(objet) L+=[objet]	Ne pas écrire L=L+[objet] en O(n)
Retirer le dernier objet	L.pop()	Pour récupérer l'objet supprimé : x = L.pop()
Retirer le 1° objet	L.pop(0)	Pour récupérer l'objet supprimé : x = L.pop(0)
Retirer un objet quelconque	L.pop(i)	I étant l'indice de l'objet à retirer (indice 0 pour le premier terme !)
Affecter des valeurs à plusieurs variables	d1,d2,p1,d3,p2,p3,p4 = mot	Si mot = [1,0,0,0,0,1,0] Ne pas aller chercher chaque case de mot mot[0] mot[1] etc
Copier une liste en créant une nouvelle liste !	LL = L.copy() LL=L[:]	Si on écrit LL=L, le pointeur L et le pointeur LL pointent vers la même mémoire – Toute modification de L ou LL modifie aussi l'autre !
Listes de listes	L=[[],[],[]] Accès à la seconde valeur de la seconde liste par exemple : L[1][1]	On peut faire autant de listes de listes de listes… que l'on veut
	L[:][0] L[0][1:3]	Récupération d'une liste contenant toutes les premières valeurs de chaque sous liste de L Récupération d'un morceau de liste de liste

Erreur à éviter	~~L = L.append(Objet)~~	Efface L ! Exécuter L.append(Objet) change L en mémoire et ne renvoie rien dans la console, soit renvoie NONE, donc c'est équivalent à écrire L = NONE Essayer : print(L)
Méthodes des objets listes	dir(list)	Affiche toutes les « magic methods » et méthodes des objets de type liste

ATTENTION

La modification d'un objet (par exemple une liste) stocké dans une liste, modifie l'objet… dans la liste !!! (Pas vrai pour entiers et flottants, quels autres ?)

```
L = []
A = [1,1]
L.append(A)
```

```
A += [1]
print(L)
```
`[[1, 1, 1]]`

Cependant, écrire A= crée un nouveau A :

```
A = A + [1]
print(L)
```
`[[1, 1]]`

Ne pas écrire `L = [[]]*5` mais `L = [[] for _ in range(5)]`
Valable quelle que soit la liste intérieure `[]`, par exemple pour ajouter des lignes à une table

Indices	<table><tr><td>Obj1</td><td>Obj2</td><td>Obj3</td><td>Obj4</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td></tr><tr><td>-4</td><td>-3</td><td>-2</td><td>-1</td></tr></table>	Appeler un terme par son indice L[i]
Séparateurs	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>Objet</td><td>"a"</td><td>"b"</td><td>"c"</td><td>"d"</td><td></td></tr><tr><td></td><td>-4</td><td>-3</td><td>-2</td><td>-1</td><td>rien</td></tr></table>	Appeler une partie de la liste à l'aide des commandes : L[1 :3]=L[-4 :-2] Un mauvais ordre donne une liste vide Ou appeler tous les termes à partir d'un séparateur ou jusqu'à ce séparateur (résultat toujours croissant) : L[2 :] ; L[:2]=L[:-3] L[-1 :0] ne fonctionne pas, il faut obligatoirement L[-1 :] ou mieux L[-1] L[:i] est de complexité O(i)

Range	Liste par « compréhension » L=[f(i) for i in range(n)] L=list(range(n)) n de type int	On crée une liste contenant les valeurs de f(i) pour i allant du séparateur 0 au séparateur n Elle contient n termes pour i allant de 0 à n-1
	L=[i for i in range(x,y)] X,y de type int	On obtient tous les termes du séparateur x au séparateur y
	L=[i for i in range(x,y,p)] x,y,p de type int	On obtient tous les termes du séparateur x au séparateur y et séparés d'un pas p à partir de x - Possibilité d'aller en arrière : range(10,-1,-1) va de 10 à 0
	L=range(x,y,p) x,y,p de type int	Fonctionne comme pour i for i in range… Mais pour afficher la liste, L retourne range(…), il faut écrire : List(L)
	range(0,n)= range(n) donc ne pas mettre 0 !	

Parcourir une liste	$for\ t\ in\ L$	On parcourt les éléments de L
		`L = [i for i in range(0,11,2)]`
		`LL = [2*t for t in L]`
		Utile aussi avec les boucles for par exemple mais attention à ne pas modifier la liste L dans la boucle !!!

Dernier terme	$L[len(L)-1]$ ou $L[-1]$	
Opérations	$L.remove(objet)$	Retire la première occurrence de l'objet en question de la liste en partant de la gauche
	$del\ L[i]$	Suppression d'un objet à l'indice i
	$L.reverse()$	Inverse le sens des termes de la liste
	$L.count(objet)$	Retourne le nombre de fois où l'objet apparaît
	$L.index(objet)$	Retourne l'index de la première occurrence de l'objet
	$L.extend(LL)$ $L+LL$	Ajoute la liste LL à la liste L
	$L.insert(i,x)$	i : séparateur - x : objet à ajouter à L
Echange de termes	`L[i],L[j]=L[j],L[i]`	Evite de stocker une valeur et 3 lignes
Test ==	Le test d'égalité entre deux listes peut être utilisé (contrairement aux array)	