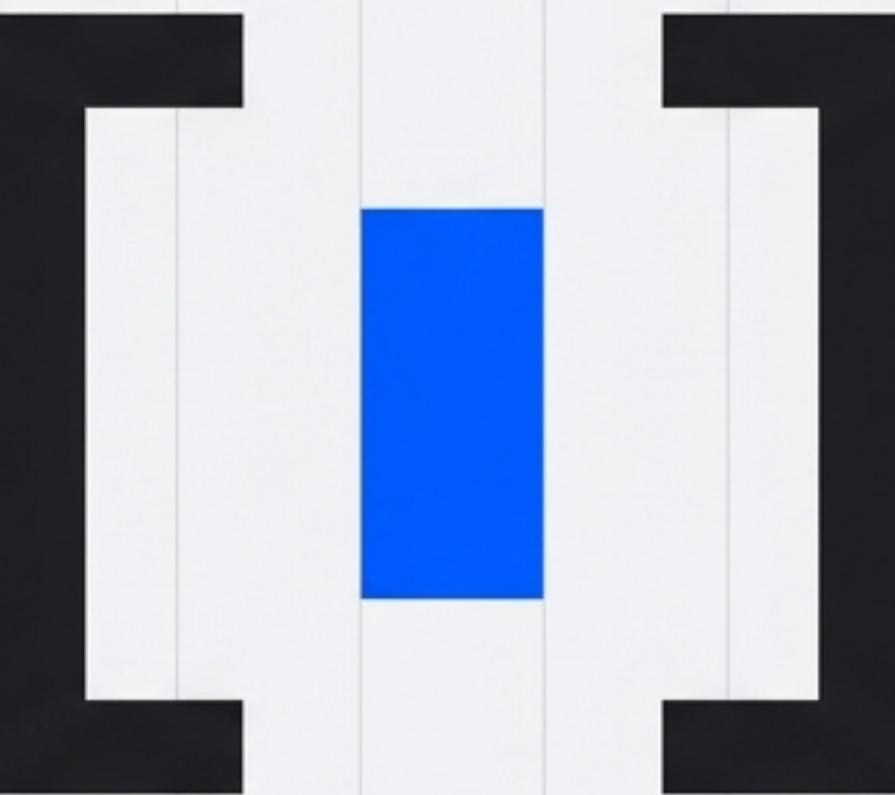


L'Indexation des Listes en Python

Accéder et Extraire les Données

Guide technique sur l'indexation positive, négative et le découpage (slicing)



La liste de référence

Pour tous les exemples de ce document, nous utiliserons la liste suivante. Elle contient des chaînes de caractères (strings) et des nombres décimaux (floats).

```
liste = ["Liz", 1.73, "Emma", 1.68, "mom", 1.71, "dad", 1.89]
```

```
+-----+-----+-----+-----+-----+-----+-----+
| "Liz" | 1.73 | "Emma" | 1.68 | "mom" | 1.71 | "dad" | 1.89 |
+-----+-----+-----+-----+-----+-----+-----+
```

Concept 1 : L'indexation positive

En Python, le comptage commence à 0. Le premier élément se trouve à l'index 0, le deuxième à l'index 1, et ainsi de suite.

INDEX :	0	1	2	3	4	5	6	7
VALEUR:	"Liz" 1.73 "Emma" 1.68 "mom" 1.71 "dad" 1.89							

Accès par position

Pour accéder à un élément, placez son index entre crochets `[]`.
Exemple : récupérer la taille d'Emma (4ème élément) à l'index 3.

INDEX :



VALEUR: ... | 1.68 | ...

```
>>> liste[3]  
1.68
```

Concept 2 : L'indexation négative

Python permet de compter à rebours depuis la fin. Le dernier élément est toujours -1.

		0		1		2		3		4		5		6		7	
INDEX :	+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+																
VALEUR:	"Liz" 1.73 "Emma" 1.68 "mom" 1.71 "dad" 1.89																
INDEX NEG:	+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+																
	-8	-7	-6	-5	-4	-3	-2	-1									

Accès inversé

L'index -1 cible immédiatement le dernier élément.
Utile quand la longueur de la liste est inconnue.

VALEUR: ... | 1.89 |

	-1
+-----+	
1.89	
+-----+	
#0055FF	

```
>>> liste[-1]
```

1.89

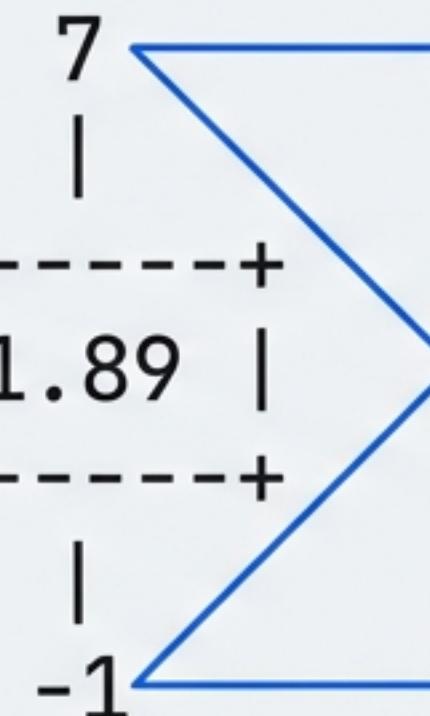
Équivalence des index

Deux adresses pour la même donnée. Le résultat est identique.

POS :

	+-----+ ... +-----+
VALEUR:	"Liz" ... 1.89
	+-----+ ... +-----+

NEG :



`liste[7] == liste[-1]`

Concept 3 : Le découpage (Slicing)

Le slicing permet de créer une nouvelle liste contenant un sous-ensemble d'éléments.

liste[début : fin]

La règle d'exclusion

1. L'index de début est INCLUS.
2. L'index de fin est EXCLU.



Le découpage s'arrête strictement AVANT l'index de fin.

Application du découpage

Sélection des index 3 et 4.

Nous devons utiliser l'index de fin 5 (exclu).

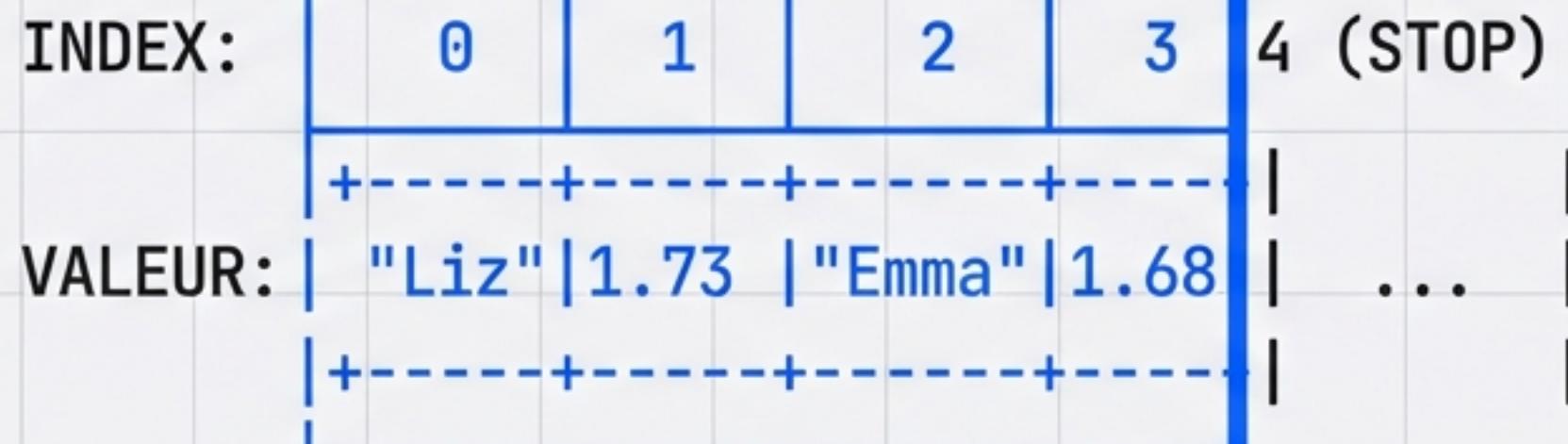
```
>>> liste[3:5]  
[1.68, 'mom']
```



Découpage implicite : Le début

Omettre l'index avant les deux-points implique "depuis le début" (0).

```
>>> liste[:4]  
['Liz', 1.73, 'Emma', 1.68]
```



Découpage implicite : La fin

Omettre l'index après les deux-points implique 'jusqu'à la fin'.

```
>>> liste[5:]  
[1.71, 'dad', 1.89]
```

INDEX:

VALEUR: ...

5	6	7
-----+-----+-----+		
1.71 "dad" 1.89		
-----+-----+-----+		

Vue d'ensemble : Carte des index

Référence complète.

POS	VALEUR								NEG
POS	0	1	2	3	4	5	6	7	NEG
	+-----+-----+-----+-----+-----+-----+-----+-----+								
VALEUR	"Liz" 1.73 "Emma" 1.68 "mom" 1.71 "dad" 1.89								
	+-----+-----+-----+-----+-----+-----+-----+-----+								
NEG	-8	-7	-6	-5	-4	-3	-2	-1	

Vérification des exemples

Confirmation technique des résultats présentés.

Syntaxe	Description	Résultat exact
liste[3]	Index positif	1.68
liste[-1]	Index négatif	1.89
liste[7]	Équivalence	1.89
liste[3:5]	Slicing (fin exclue)	[1.68, 'mom']
liste[:4]	Début implicite	['Liz', 1.73, 'Emma', 1.68]
liste[5:]	Fin implicite	[1.71, 'dad', 1.89]

Synthèse

- **Indexation `[]`**

Récupère un seul élément à une position précise.

- **Slicing `[:]`**

Crée une nouvelle liste contenant une séquence d'éléments.

- **Rappel clé**

Le slicing INCLUT l'index de début mais EXCLUT toujours l'index de fin.