



# Les Listes en Python : Crédation et Structure

Guide technique pour la gestion de collections de données

---

Ce document détaille la syntaxe exacte et le comportement des listes (type `list`) à travers des exemples vérifiés.

# Le Regroupement de Valeurs

Traiter chaque point de données comme une variable indépendante est inefficace. Python permet de regrouper plusieurs valeurs sous un nom unique en utilisant des crochets [].

## ENTRÉE (Code Python)

```
# Création d'une liste simple contenant noms et tailles
liste_simple = ["Liz", 1.73, "Emma", 1.68, "mom", 1.71, "dad", 1.89]
print(liste_simple)
```

## SORTIE (Console)

```
['Liz', 1.73, 'Emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

La variable liste\_simple contient désormais 8 éléments distincts.

# Anatomie Syntaxique

La définition d'une liste respecte deux règles de syntaxe strictes :

1. Délimiteurs : La liste commence par [ et se termine par ].
2. Séparateurs : Chaque élément est séparé du suivant par une virgule ,.

SYNTAXE : [ élément1 , élément2 , élément3 ]  
          ^       ^       ^       ^       ^       ^  
          ↓       |       ↓       |       ↓       |       ↓  
EXEMPLE : ["Liz" , 1.73 , "Emma" ... ]

Note : Les espaces après les virgules sont optionnels mais recommandés pour la lisibilité (PEP 8).

# Hétérogénéité des Types

Une liste Python peut contenir des éléments de types différents simultanément (str, float, int, bool). Dans notre exemple, des chaînes de caractères (noms) cohabitent avec des nombres décimaux (tailles).

VALEUR (Dans la liste)	TYPE DÉDUIT
"Liz"	str (Texte)
1.73	float (Décimal)
"Emma"	str (Texte)
1.68	float (Décimal)
"mom" 1.71	str (Texte) float (Décimal)
"dad"	str (Texte)
1.89	float (Décimal)

Conclusion : Python ne force pas l'uniformité des types au sein d'une même liste.

# Représentation Séquentielle (Mémoire)

Les éléments sont stockés dans un ordre précis. C'est une séquence ordonnée.

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

- Longueur totale : 8 éléments
- Structure : Plate (1 dimension)
- Logique : Alternance Nom / Taille

# Structuration par Imbrication

Pour associer plus fortement chaque nom à sa taille correspondante, nous pouvons utiliser des listes imbriquées. Chaque paire (Nom + Taille) devient une sous-liste autonome.

## Code Python:

```
# Création d'une liste de listes
liste_imbriquee = [["Liz", 1.73], ["Emma", 1.68], ["mom", 1.71], ["dad", 1.89]]
print(liste_imbriquee)
```

## Sortie Console:

```
[['Liz', 1.73], ['Emma', 1.68], ['mom', 1.71], ['dad', 1.89]]
```

**Observation :** La structure change radicalement, passant d'une séquence plate à une structure hiérarchique.

# Syntaxe des Listes de Listes

Une liste est un type de données valide à l'intérieur d'une autre liste.

**LISTE PRINCIPALE (Conteneur)**

```
[  
  ['Liz', 1.73],      <-- Sous-liste 1 (Élément)  
  ['Emma', 1.68],    <-- Sous-liste 2 (Élément)  
  ['mom', 1.71],     <-- Sous-liste 3 (Élément)  
  ['dad', 1.89]       <-- Sous-liste 4 (Élément)  
]
```

Les virgules séparent désormais les sous-listes, et non plus les valeurs individuelles.

# Analyse de la Hiérarchie

Du point de vue de la liste principale, les éléments sont les sous-listes entières.

## 1. liste\_simple

```
"[Elem1, Elem2, Elem3, Elem4, Elem5,  
Elem6, Elem7, Elem8]"
```

→ Contient 8 éléments (str et float).

## 2. liste\_imbriquee

```
"[  
    Elem1,  
    Elem2,  
    Elem3,  
    Elem4  
]"
```

-> Contient 4 éléments (tous de type list).

L'imbrication réduit le nombre d'éléments de premier niveau mais augmente la complexité de la structure.

# Vérification Technique du Type

La fonction **type()** confirme que Python traite les deux structures comme des objets de même nature, indépendamment de leur contenu interne.

## CODE

```
type_simple = type(liste_simple)
type_imbriquee = type(liste_imbriquee)

print(type_simple)
print(type_imbriquee)
```

## RÉSULTAT

```
<class 'list'>
<class 'list'>
```

# Synthèse : Règles Fondamentales

- **Définition** : Une liste est une collection ordonnée de valeurs modifiables.
- **Syntaxe** : Utilisation des crochets [] et séparation par virgules ,.
- **Types** : Une liste peut contenir des types hétérogènes (str, float, etc.).
- **Imbrication** : Une liste peut contenir d'autres listes comme éléments.
- **Usage** : Idéal pour stocker et manipuler des ensembles de données liés (ex: données familiales).

# Vérification des Exemples

Confirmation de l'intégrité des données utilisées dans ce support.

VARIABLE	ÉLÉMENTS	VALEURS CLÉS
-----		
liste_simple	8	"Liz", 1.73 ... "dad", 1.89
liste_imbriquée	4	["Liz", 1.73] ... ["dad", 1.89]

**Statut : Code valide.** Syntaxe vérifiée. Sorties conformes à l'interpréteur Python standard.