

Les Listes en Python : Gérez vos données efficacement

Une introduction essentielle pour les futurs Data Scientists

Ce guide autonome est conçu pour vous faire passer de la gestion de variables simples à la structuration de données complexes.

Rappel : Les Blocs de Construction

Vous maîtrisez déjà les types de données atomiques qui stockent une valeur unique.

- float : Nombres réels (ex: `1.73`)

1.73

- int : Nombres entiers (ex: `100`)

100

- str : Texte (ex: "'Python'")

'Python'

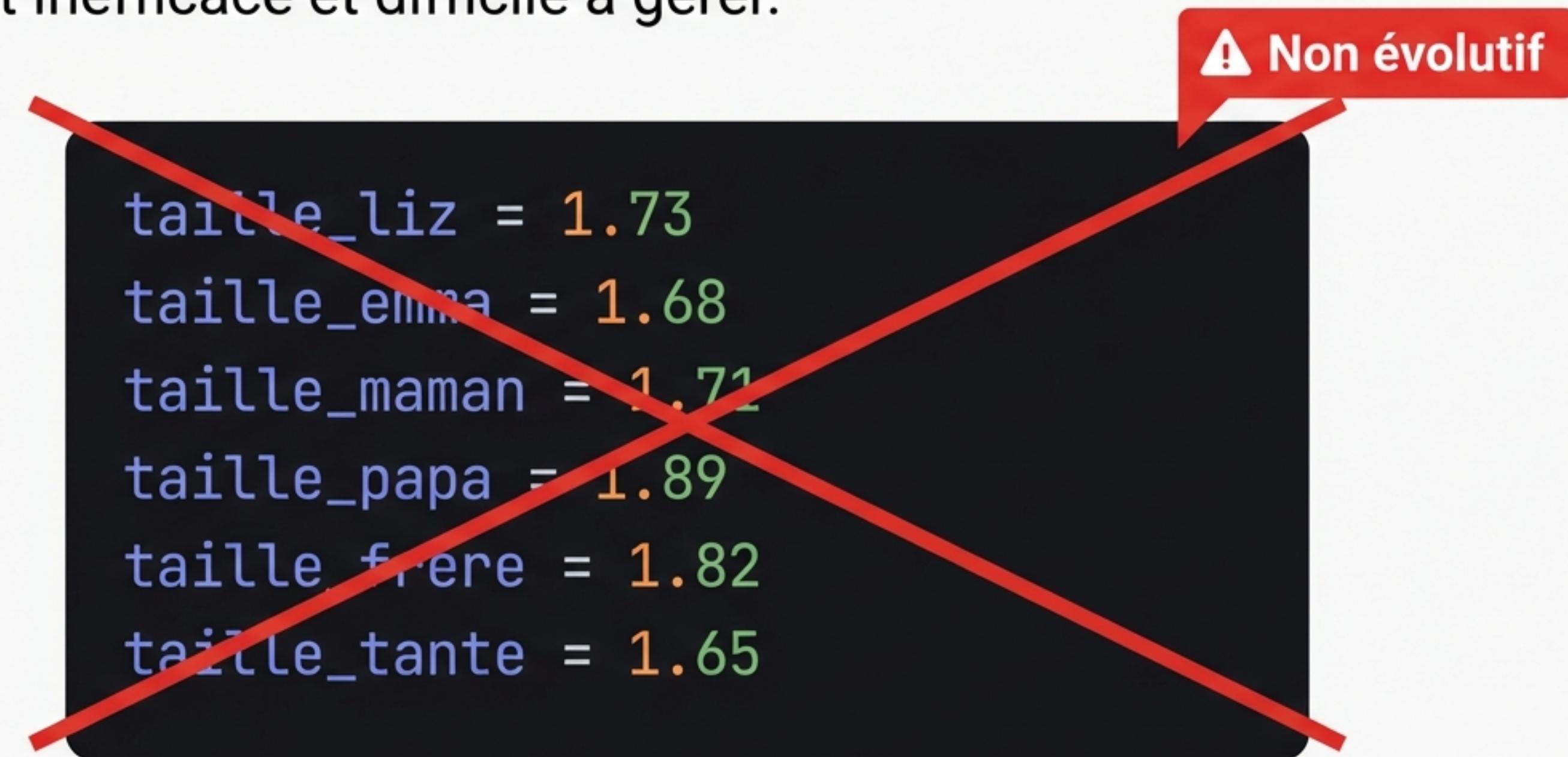
- bool : Logique (Vrai/Faux)

True



Le Problème des Variables Uniques

En tant que Data Scientist, créer une variable pour chaque point de données est inefficace et difficile à gérer.



La Solution : La Liste Python

Au lieu de multiplier les variables, vous pouvez stocker toutes ces informations dans une seule structure : la Liste.

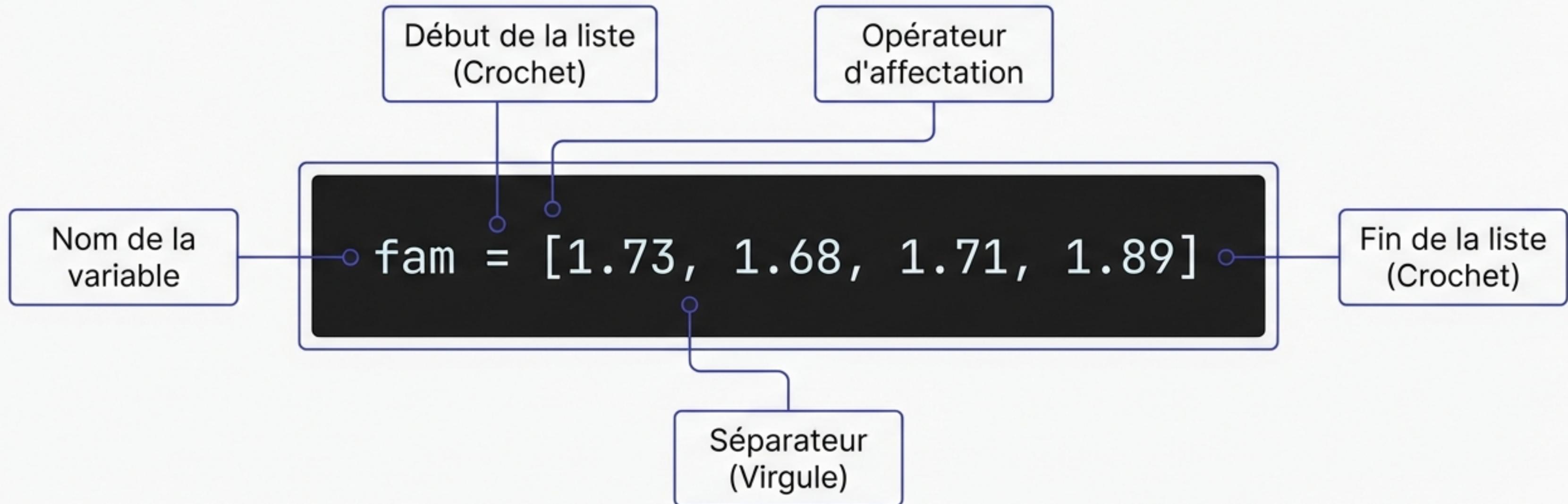
Définition : Une liste est un moyen de donner un nom unique à une collection de valeurs.



```
fam = [1.73, 1.68, 1.71, 1.89]
```

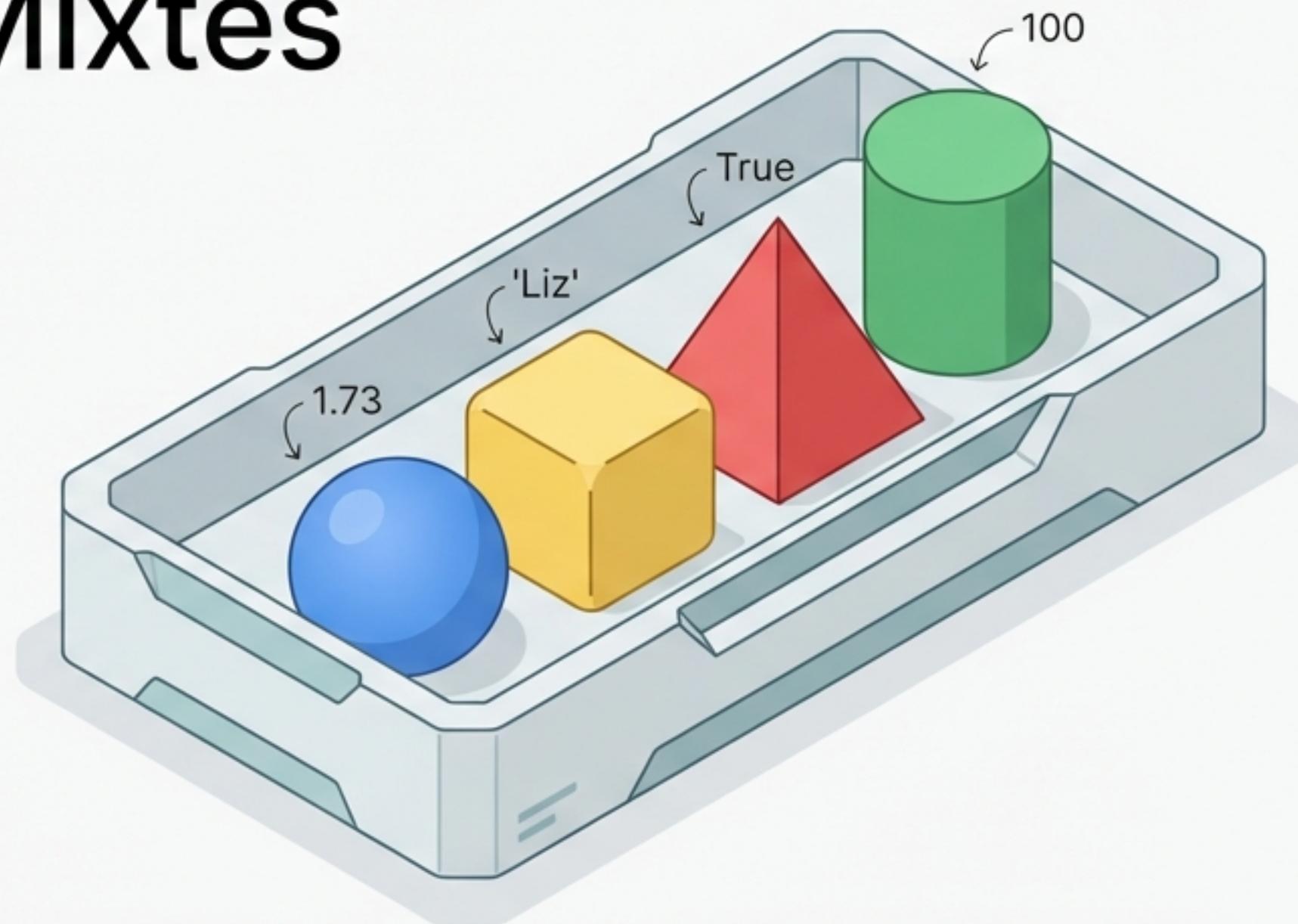
Anatomie d'une Liste

La syntaxe est stricte mais simple



Flexibilité : Types Mixtes

Les listes Python ne discriminent pas. Elles peuvent contenir n'importe quel type de données : entiers, booléens, chaînes de caractères et nombres flottants.



```
mixte = [1.73, "Liz", True, 100]
```

Contexte Pratique : Associer Noms et Valeurs

Pour savoir à qui correspond chaque taille, nous pouvons ajouter des chaînes de caractères (noms) directement dans la liste.

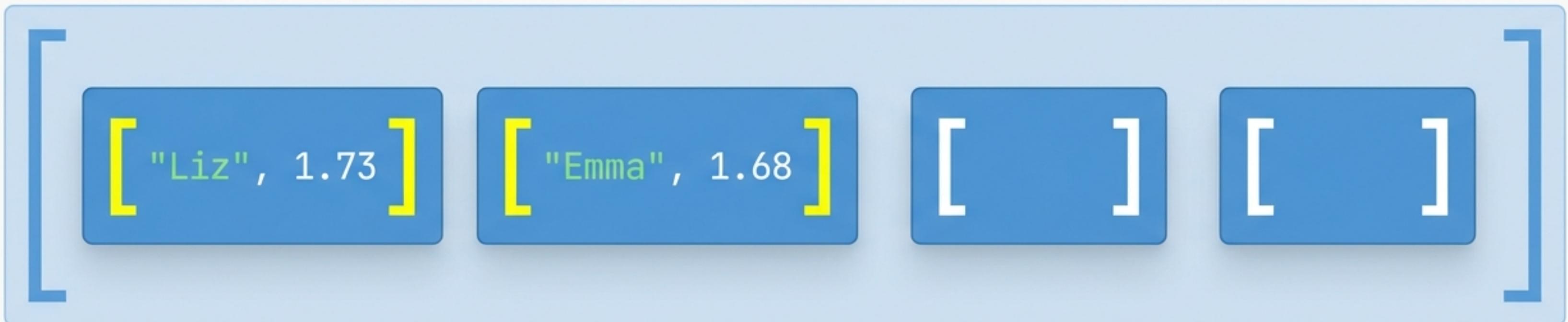
```
fam = ["Liz", 1.73, "Emma", 1.68, "Maman", 1.71, "Papa", 1.89]
```



Structure valide, mais plate et difficile à lire.

Le Niveau Supérieur : Listes dans Listes

Une liste peut contenir d'autres listes. Nous créons des sous-listes pour chaque membre de la famille.



```
fam2 = [["Liz", 1.73], ["Emma", 1.68], ["Maman", 1.71], ["Papa", 1.89]]
```

Visualiser la Structure des Données

Le code peut être formaté pour révéler la structure multidimensionnelle. Voici fam2 formatée pour la lisibilité.

4 Sous-listes

Indentation pour clarté →

```
fam2 = [  
    ["Liz", 1.73],  
    ["Emma", 1.68],  
    ["Maman", 1.71],  
    ["Papa", 1.89]]
```

Vérification du Type

La fonction `type()` confirme que nous manipulons un objet 'list', quelle que soit sa complexité interne.

```
type(fam)
```

Output: <class 'list'>

```
type(fam2)
```

Output: <class 'list'>



En Résumé : Le Pouvoir des Listes



Création

Utilisez des crochets.



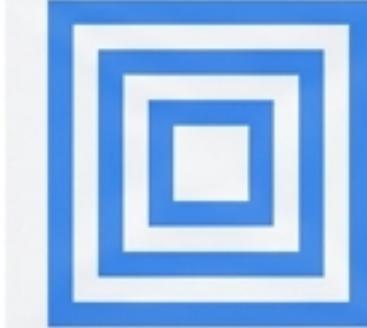
Séparation

Séparez les éléments avec des virgules.



Flexibilité

Types mixtes acceptés.



Structure

Imbriquez des listes pour créer des dimensions.

Prochaine étape : Les listes possèdent leurs propres méthodes pour filtrer et adapter les données.

Aide-Mémoire Technique

Vocabulaire

- **Liste** : Collection ordonnée d'éléments.
- **Élément** : Une valeur individuelle.
- **Sous-liste** : Une liste dans une liste.
- **Crochet []** : Délimiteur de liste.

Syntaxe

```
# Liste simple
fam = ["Liz", 1.73, "Emma", 1.68]

# Liste de listes
fam2 = [
    ["Liz", 1.73],
    ["Emma", 1.68]
]
```