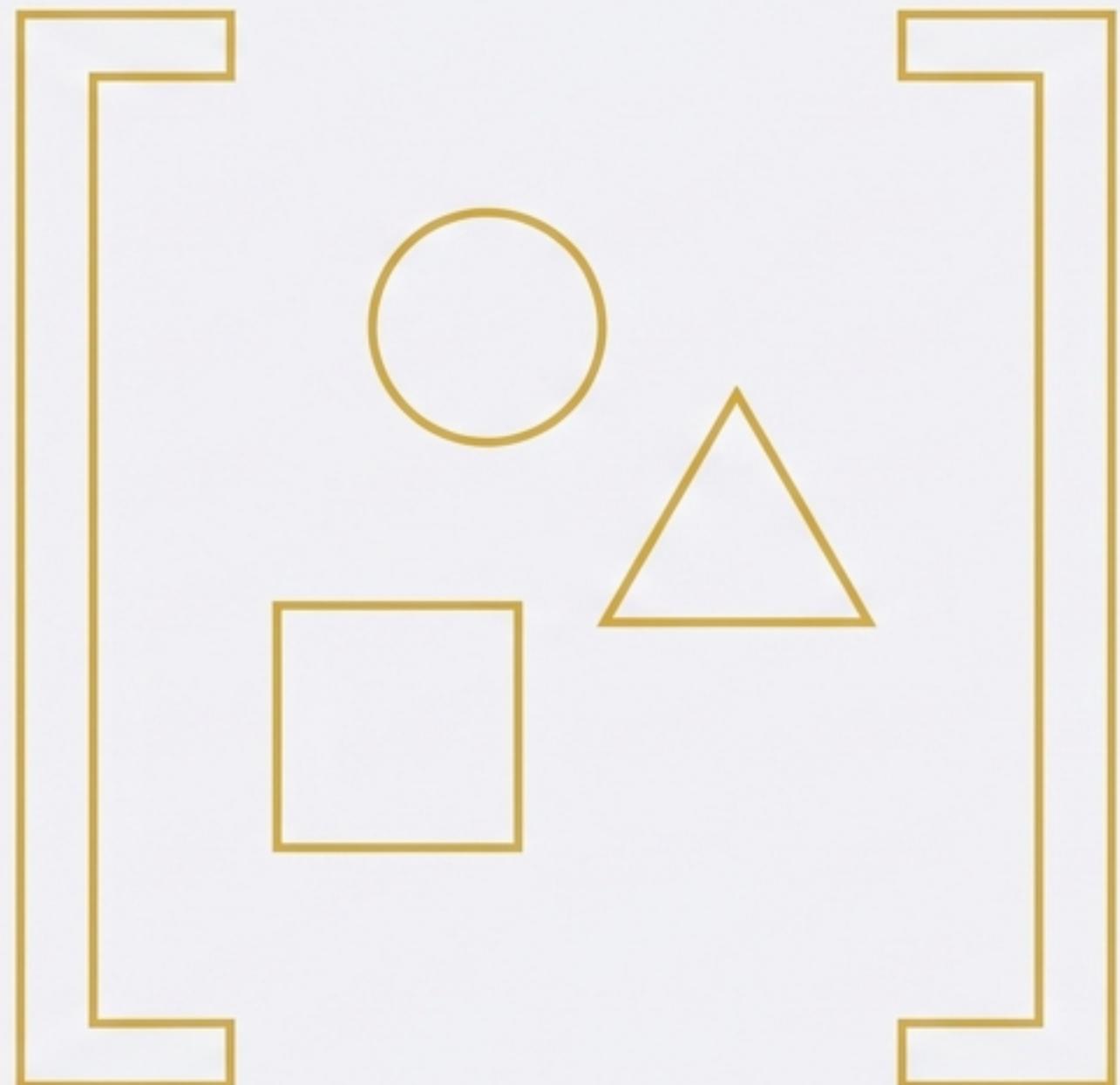


# Les Listes en Python : Structure et Création

Gérer des collections de données  
hétérogènes

Ce document détaille la syntaxe, la création et la structure des listes en Python, une compétence fondamentale pour la science des données.



# Rappel : Les types de données scalaires

Avant de manipuler des collections, rappelons les types de base qui stockent une valeur unique. Une variable standard représente un point de donnée unique.

Type	Description	Exemple
float	Nombre réel	1.73
int	Nombre entier	42
str	Chaîne de caractères	"Liz"
bool	Booléen (Vrai/Faux)	True

# La limite des variables individuelles

Mise en situation : Vous souhaitez enregistrer la taille de chaque membre de votre famille. Créer une nouvelle variable pour chaque point de donnée est fastidieux et difficilement gérable pour de grands ensembles de données.

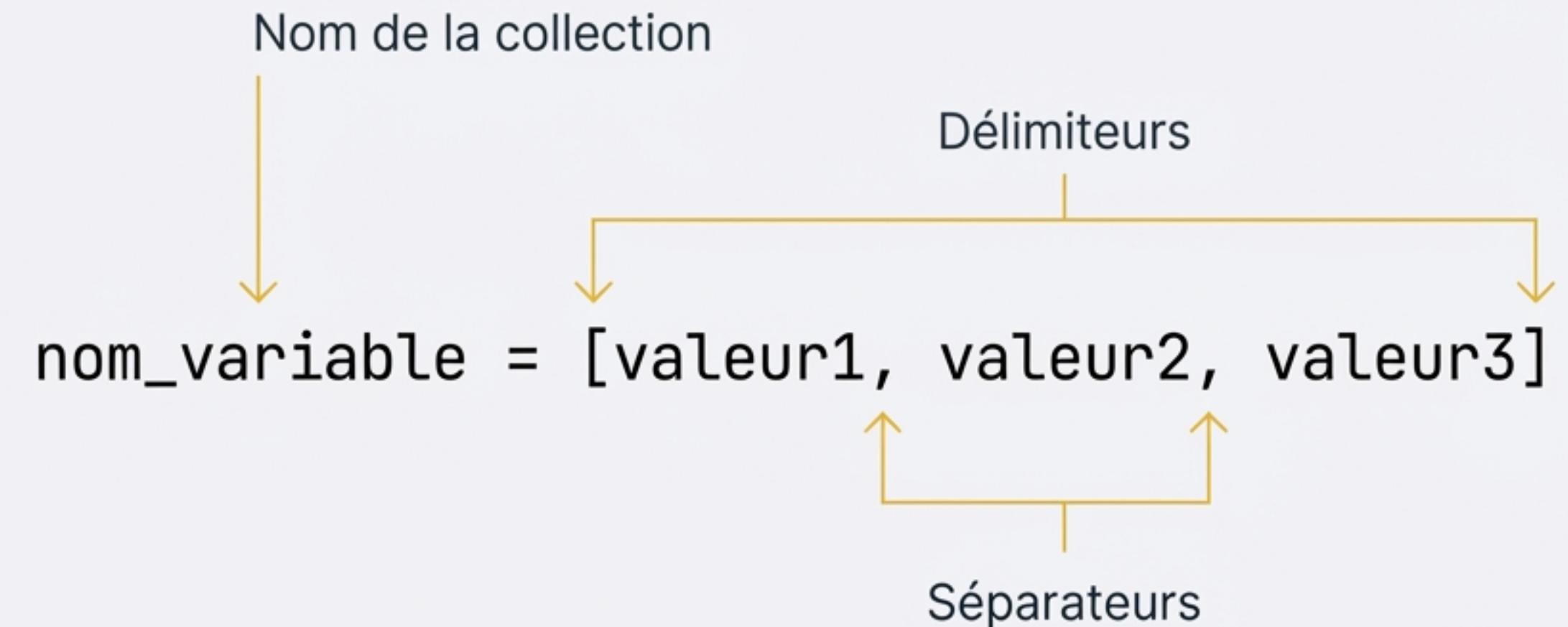
```
hauteur_liz = 1.73  
hauteur_emma = 1.68  
hauteur_mom = 1.71  
hauteur_dad = 1.89
```

Approche inefficace

# La solution : La Liste Python

Une liste permet de donner un nom **unique** à une collection de valeurs.

- Délimitée par des crochets []
- Éléments séparés par des virgules ,
- Assignation via le signe égal =



# Création d'une liste simple

Objectif : Stocker les noms et les tailles dans une séquence linéaire.

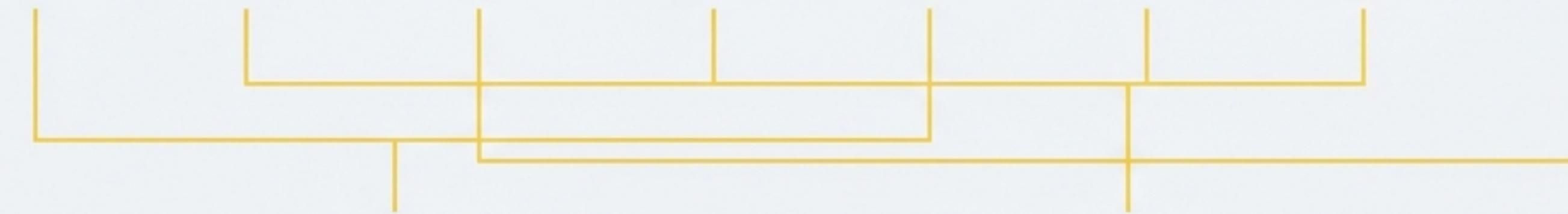
```
fam = ["Liz", 1.73, "Emma", 1.68, "mom", 1.71, "dad", 1.89]
```

Cette structure regroupe 8 éléments distincts dans une seule variable nommée fam.

# Flexibilité des types (Mixed Types)

Les listes Python peuvent contenir des éléments de types différents simultanément (str, float, int, bool). Il n'est pas nécessaire d'homogénéiser les types de données.

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



**str** (chaînes)

**float** (nombres)

# Visualisation de la structure linéaire

Les chaînes de caractères permettent d'identifier à qui appartient la taille qui suit, mais la structure reste plate.

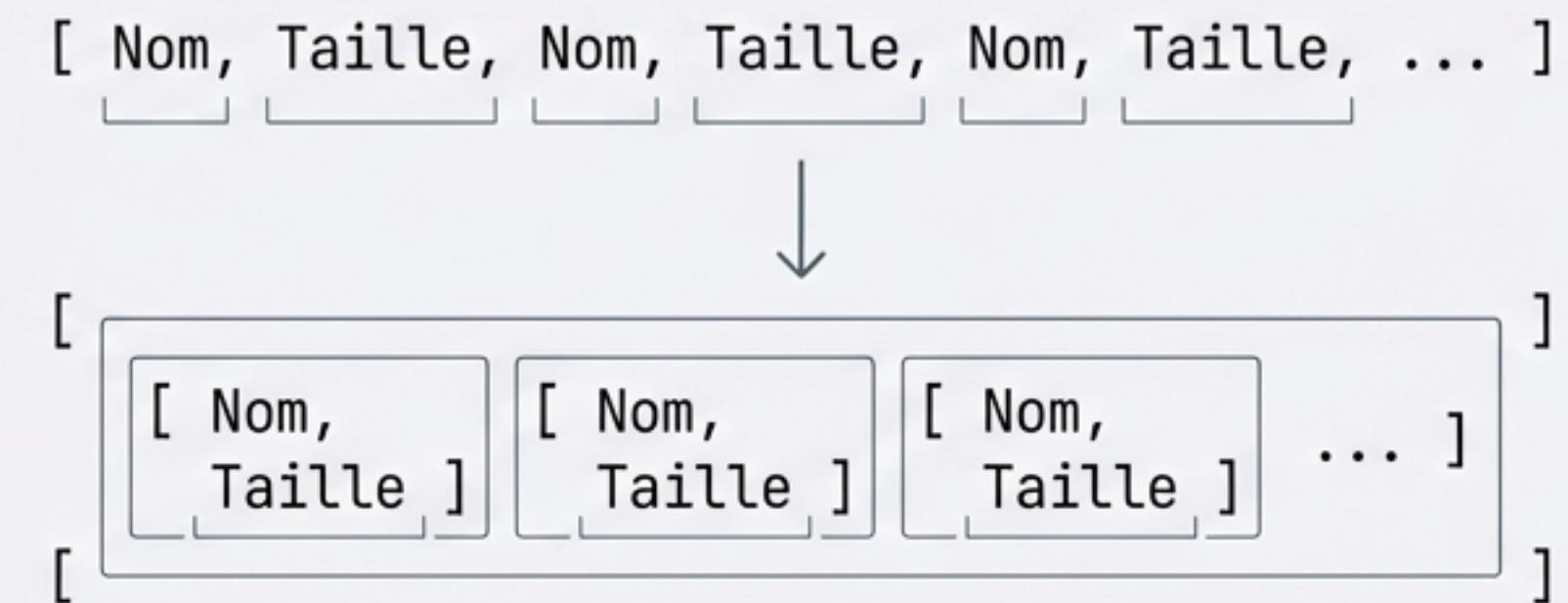
Variable : fam

```
[ "Liz", 1.73, "Emma", 1.68, "mom", 1.71, "dad", 1.89 ]  
      ^      ^      ^      ^      ^      ^      ^  
    str    float   str    float   str    float  str    float
```

# Listes de Listes (Structure Imbriquée)

Une liste peut contenir d'autres listes comme éléments. Cela permet de grouper logiquement les données.

Nouvelle approche : Créer une sous-liste pour chaque personne contenant son nom et sa taille.



# Création d'une liste imbriquée

Les paires sont entre crochets et séparées par des virgules à l'intérieur de la liste principale.

```
fam2 = [[{"Liz": 1.73},  
         {"Emma": 1.68},  
         {"mom": 1.71},  
         {"dad": 1.89}]
```

# Visualisation de la hiérarchie

Les données sont désormais structurées par entité.

Variable : fam2

V

```
[ ←———— Liste Principale
  ["Liz", 1.73], ← Élément 1 (Liste)
  ["Emma", 1.68], ← Élément 2 (Liste)
  ["mom", 1.71], ← Élément 3 (Liste)
  ["dad", 1.89]   ← Élément 4 (Liste)
]
```

# Validation du type de données

La fonction `type()` confirme la nature de la structure, quel que soit son contenu (mixte ou imbriqué). Python traite `fam` et `fam2` comme le même type d'objet fondamental.

```
>>> type(fam)
<class 'list'>
```

```
>>> type(fam2)
<class 'list'>
```

# Synthèse : Les propriétés des Listes

## 1. Conteneur

Une liste regroupe plusieurs valeurs sous un seul nom de variable.

## 2. Syntaxe

Utilisation des crochets [...] et séparation par virgules.

## 3. Hétérogénéité

Une liste peut contenir des types différents (str, float, bool).

## 4. Imbrication

Une liste peut contenir d'autres listes (Listes de listes).