

# Les Méthodes en Python : Comprendre les Objets et leurs Actions

Une introduction structurelle aux fonctions intégrées aux objets



Guide Technique - Niveau Débutant

## Le Concept Fondamental : Tout est un Objet

En Python, les données ne sont pas passives. Ce sont des objets qui combinent une valeur, un type et des méthodes.

Type: String (str)

"liz"

Valeur textuelle

Type: Float (float)

1.73

Valeur numérique

Type: List (list)

[ "liz",  
 1.73,  
 "emma" ]

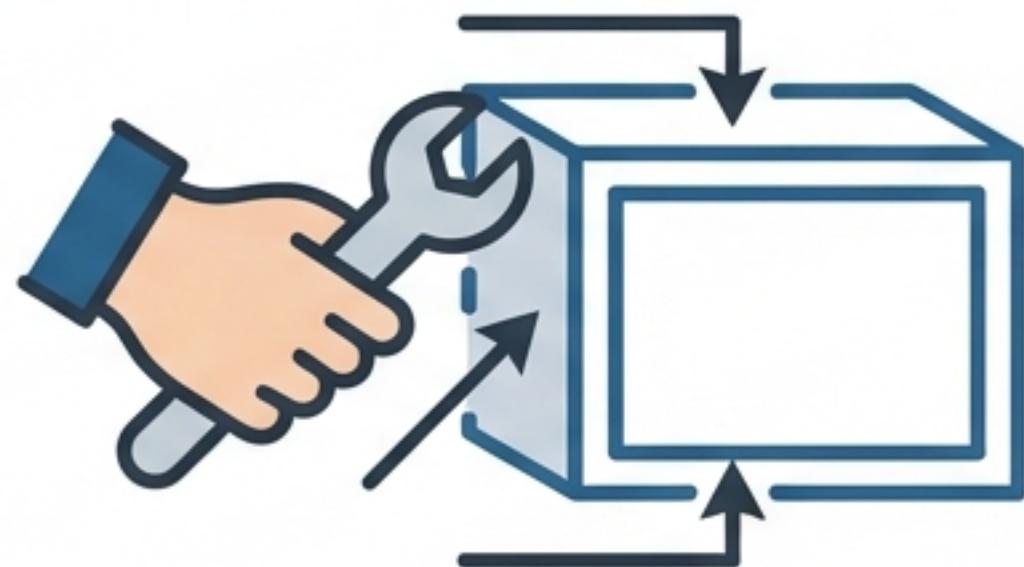
Collection ordonnée

Vos variables 'fam' et 'sister' sont des instances de ces objets.

# Distinction : Fonctions vs Méthodes

## Les Fonctions

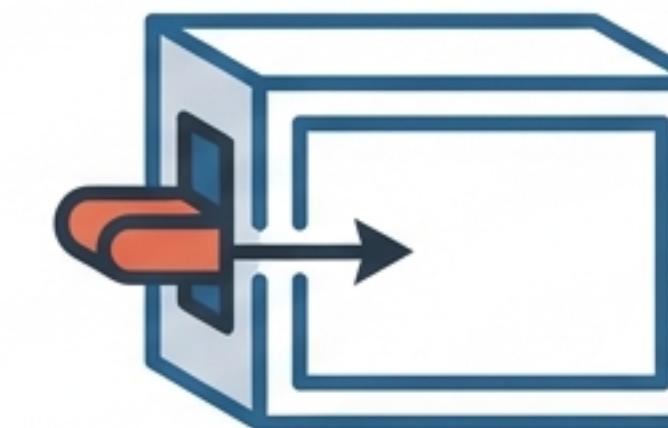
Des outils globaux disponibles pour tout.



`type(fam)`  
`max(fam)`  
`len(fam)`

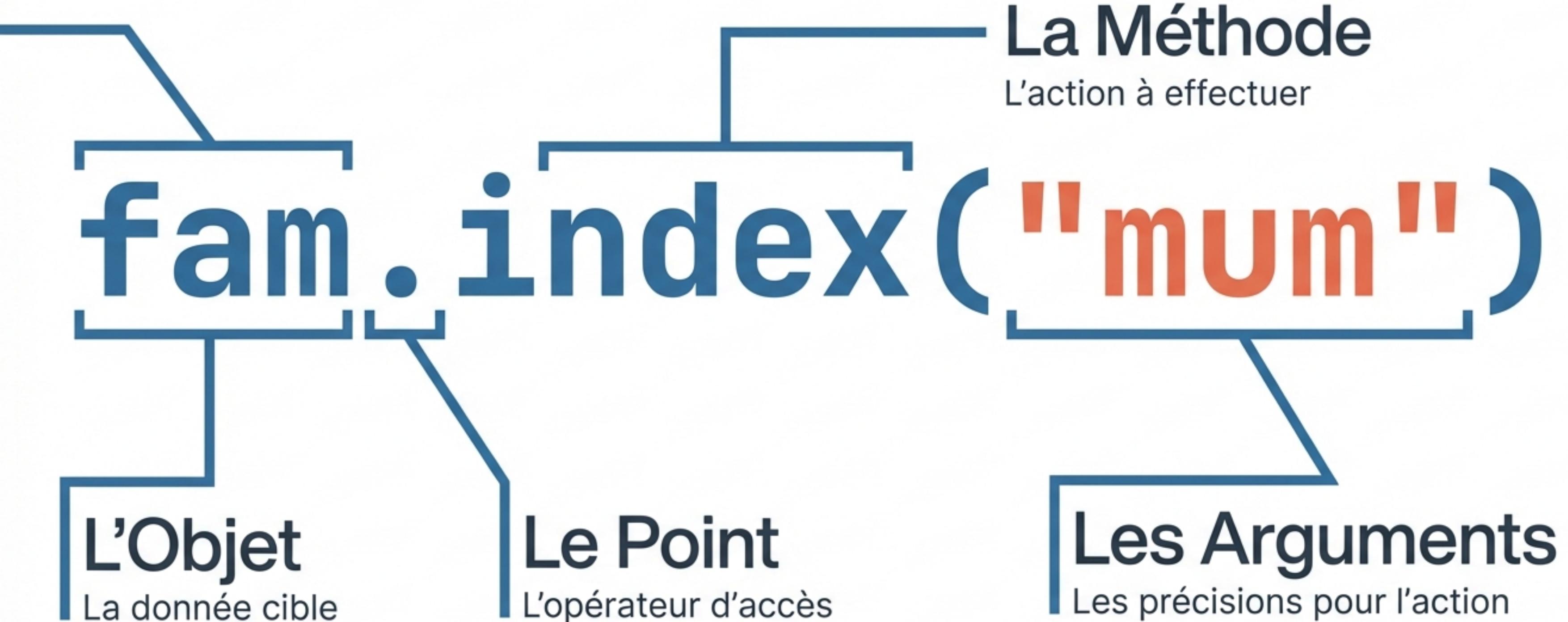
## Les Méthodes

Des fonctions spécifiques qui appartiennent à l'objet.



`fam.index("mum")`  
`fam.count(1.73)`

# La Syntaxe : La Notation Point (Dot Notation)



# Méthodes de Liste : Localiser un élément

```
fam.index("mum")
```

Index: 0 ————— 1 ————— 2 ————— 3 ————— 4 ————— 5  
Liste: ["liz", 1.73, "emma", 1.68, "mum", 1.71]

Résultat : 4

Note : Retourne l'index de la première occurrence trouvée.

# Méthodes de Liste : Compter les occurrences

```
fam.count(1.73)
```

1

Liste: ["liz", 1.73, "emma", 1.68, "mum", 1.71]

Résultat : 1

La valeur 1.73 apparaît exactement une fois dans la liste.

# Méthodes de Chaîne (String) : Formatage

Variable : sister = 'liz'

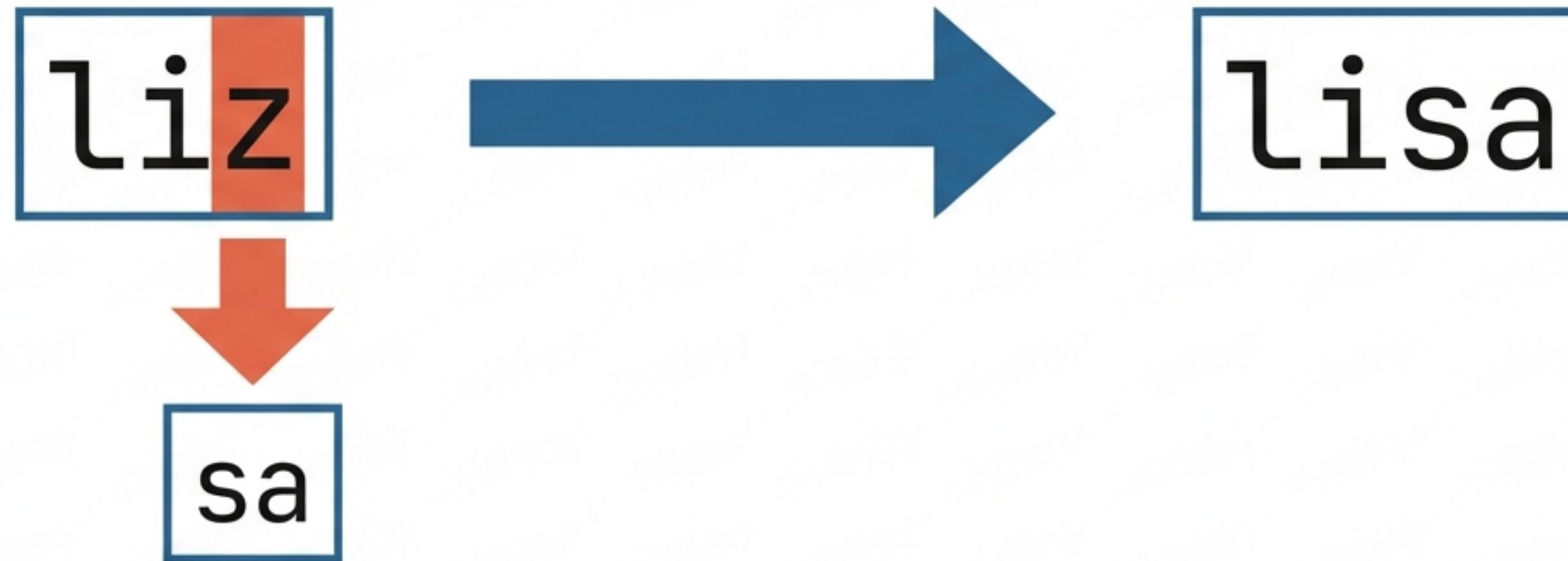


**Observation:** Cette méthode ne nécessite aucun argument (parenthèses vides).

# Méthodes de Chaîne (String) : Remplacement

Variable : sister = 'liz'

```
sister.replace("z", "sa")
```



Prend deux arguments : (ancien, nouveau).

# La Règle de Compatibilité

Les méthodes sont strictement liées au type de l'objet.

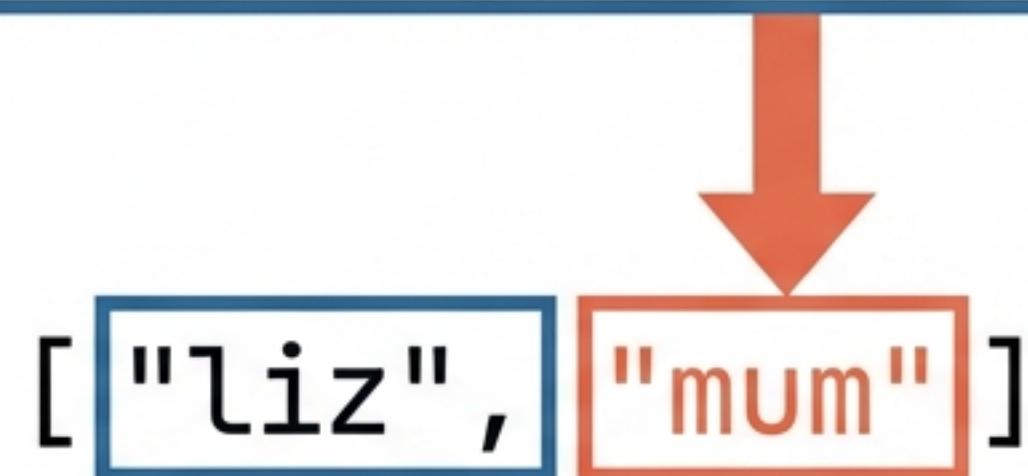
```
>>> fam.replace("mum", "mom")
AttributeError: 'list' object has no attribute 'replace'
```

Un objet de type list ne possède pas la méthode replace.  
C'est exclusif aux strings.

# Même nom, Comportement différent

## Sur une Liste

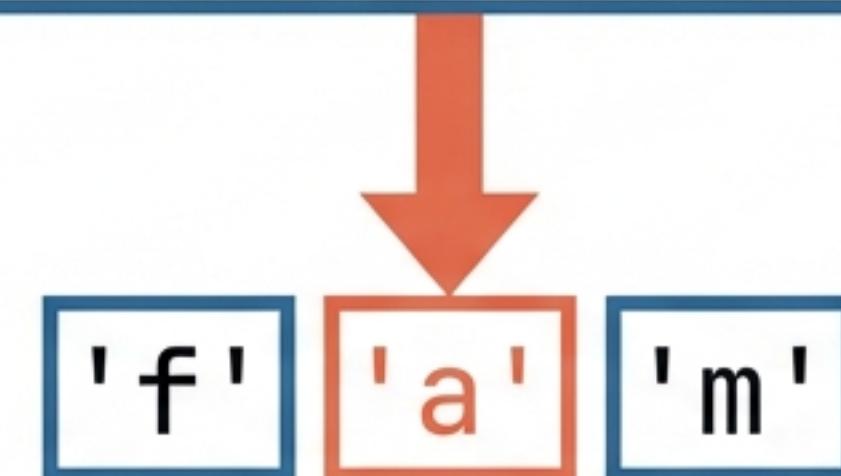
```
fam.index("mum")
```



Cherche un élément complet.

## Sur une Chaîne (String)

```
"fam".index("a")
```

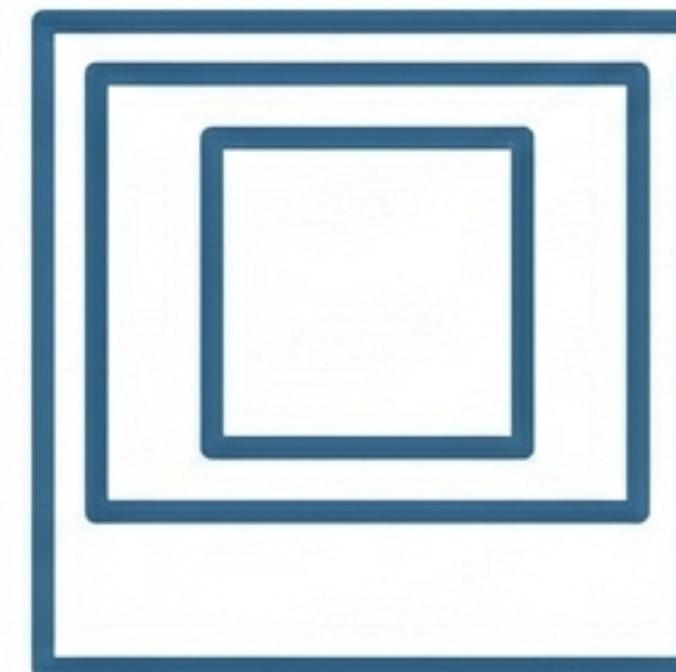


Cherche une lettre dans le texte.

**Le type d'objet dicte le comportement de la méthode.**

# Concept Avancé : La Modification 'In-Place'

**Attention :** Certaines méthodes ne renvoient pas de nouvelle valeur, mais modifient directement l'objet d'origine.



Return Value



Modification In-Place

# Modification de Liste : append()

## État 1 (Avant)

```
["liz", 1.73, "emma", 1.68, "mum", 1.71]
```

Aucun résultat visible n'est retourné, mais la liste a changé.

```
fam.append("me")
```

## État 2 (Après)

```
["liz", 1.73, "emma", 1.68, "mum", 1.71, "me"]
```

# Modification Continue

fam.append(1.79)

La liste s'agrandit dynamiquement. Le code est concis, mais il faut **surveiller les changements d'état** de vos données.

Index: ... 5 6 7  
-----  
Liste: ... 1.71, "me", 1.79 ]

# Résumé Technique

Méthode	Type	Action	Modifie l'objet ?
index()	List/Str	Trouve la position	Non
count()	List	Compte les éléments	Non
capitalize()	Str	Formate le texte	Non (renvoie une copie)
replace()	Str	Remplace une partie	Non (renvoie une copie)
append()	List	Ajoute un élément	OUI

# Ce qu'il faut retenir

## 1. Tout est objet

Listes, Strings et Floats ont leurs propres outils intégrés.

## 2. Identifiez le type

Un objet list ne se comporte pas comme un str.

## 3. Surveillez le retour

La méthode renvoie-t-elle une valeur ou modifie-t-elle l'objet (comme append) ?

## 4. Vérifiez

Utilisez print() pour visualiser l'état de vos données après une modification.