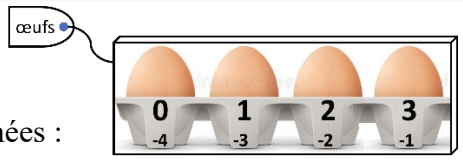
Cours 20.21 : les listes

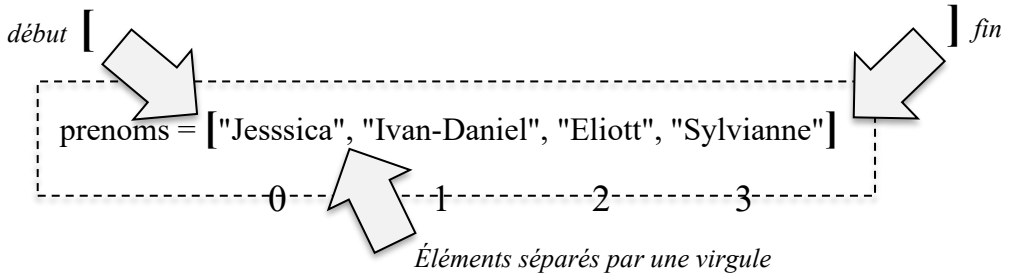
Introduction aux listes

Une liste est un **type de données** permettant
de **stocker** plusieurs éléments dans **une seule variable**.



Ces éléments peuvent être de **n'importe quel type** de données :
(int, float, string, boolean, autres) et même des listes.

Il y a plusieurs manières de définir une liste, nous aborderons dans ce chapitre :
la **définition en extension** (énumération de tous ses éléments) :



Les éléments sont indexés par un **nombre entier**. ! Le premier élément est : **0**.

Remarque : dans une liste, l'ordre des éléments est important :

Soient liste1 = **[1, 2 ,3]** et liste2 = **[3, 1, 2]**

liste1 ≠ liste2

Les **actions** possibles sur une **liste** (non exhaustives) :

- *Définir et afficher une liste* `lst=[el1, el2] / print(lst);`
- *Accès à un élément* `lst[pos-1];`
- *Modification d'un élément* `lst[pos-1]=nouvelEl;`
- *Ajout d'un nouvel élément* `lst.append(el)ou lst.insert(pos,el);`
- *Suppression d'un élément* `lst.remove(el) ou lst.pop(pos-1);`
- *Connaitre la taille de la liste (le nombre d'éléments)* `len(lst);`
- *Tester si un élément existe, ou pas, dans une liste* `in ou not in;`
- *Trier la liste* `lst.sort();`
- *Parcours d'une liste ;*
- *Liste de listes* `[[], [], []] ;`
- *Choisir aléatoirement un élément dans une liste* `import.choice(lst)`

Exercices de consolidations complets.

La série d'exercices ci-dessous présente chaque action par un exercice à réaliser en autonomie.

Définir et afficher une liste

Exercice 1a *(Nom fichier : C17_S01_E01)*

1. Définir une liste avec 4 noms de villes. (Le nom de la variable est : villes.)
2. Afficher la liste.

Exercice 1b

1. Définir une liste avec les 5 premiers nombres premiers. (Le nom de la variable est : premiers)
2. Afficher la liste.

Accès à un élément

Il est possible de récupérer une valeur dans une liste :
Rappel : le premier élément possède l'indice 0.

- Par son indice : `v = ville[2] #Récupérer le nom de la 3ème ville (premier indice 0)`
- Dernière valeur :
  - `v = ville[3]  #Il y a 4 éléments → dernier indice 3`
  - `v = ville[-1]  #Indice négatif, départ depuis la fin`
- Tranches : permet de copier une partie de la liste depuis un début (m **inclus**) jusqu'à une fin (n+1 **exclu**)
  - [:] # copie superficielle de toute la liste ;
  - [m :] # copie superficielle de la liste à partir du m^{ème} élément, jusqu'à la fin ;
  - [:n] # copie superficielle de la liste du début jusqu'au n^{ème} élément exclu ;
  - [m :n] # copie superficielle de partir du m^{ème} élément, jusqu'au n^{ème} élément exclu.

Exercice 2 *(Nom fichier : C17_S01_E02)*

Soit la liste :

```
jours = ["lu", "ma", "me", "je", "ve", "sa", "di"]
```

Quels sont les résultats que vous obtenez avec les instructions suivantes ?

a) print(jours[3])

b) print(jours)

c) print(type(jours[3]))

d) print(type(jours))

e) print(jours[2 :5])

f) print(jours[4 :])

g) print(jours[:3])

h) print(jours[:])

Cours 20.21 : les listes

Modification d'un élément

Pour modifier un élément :

- Accéder à l'élément de la liste ;
- Assigner une nouvelle valeur.

```
villes = ["Genève", "Morges", "Lausanne"]
villes[1] = "Nyon"    #Modification du nom de la deuxième ville ([1])
print(villes)         #["Genève", "Nyon", "Lausanne"]
```

Exercice 3 *(Nom fichier : C17_S01_E03)*

Soit la liste :

```
jours = ["lu", "ma", "me", "je", "ve", "sa", "di"]
```

Modifier :

a) "lu" par "lundi"

b) "je" par "jeudi"

c) "di" par "dimanche"

Ajouter un élément

Pour ajouter un élément :

```
liste = [2, 4, 6, 8, 10, 12]
```

- A la fin d'une liste : `liste.append(14)`
  - Résultat : `print(liste)` `#[2, 4, 6, 8, 10, 12, 14]`
- A une position donnée (les autres éléments sont décalés) : `liste.insert(2, 999)`
  - Résultat : `print(liste)` `#[2, 4, 999, 6, 8, 10, 12, 14]`

Exercice 4 *(Nom fichier : C17_S01_E04)*

Soit la liste :

```
villes = ["Genève", "Morges"]
```

Ajouter :

a) A la fin de la liste : `"Lausanne"`

b) Entre Genève et Morges : `"Nyon"`

Supprimer un élément

Pour supprimer un élément :

```
jours = ["lu", "ma", "me", "je", "ve", "sa", "di"]
```

- Un élément spécifique : `jours.remove("ve")`
  - Résultat : `print(jours)` `#["lu", "ma", "me", "je", "sa", "di"]`
- A une position donnée (les autres éléments sont décalés) : `jours.pop(1)`
  - Résultat : `print(jours)` `#["lu", "me", "je", "sa", "di"]`

Exercice 5 *(Nom fichier : C17_S01_E05)*

Soit la liste :

```
fruits = ["banane", "abricot", "cerise", "poire"]
```

Ajouter :

a) Supprimer le fruit : `"abricot"`

b) Supprimer le premier fruit

Taille d'une liste

La taille d'une liste correspond au nombre de ses éléments (de premier niveau) `len(liste)` :

```
jours = ["lu", "ma", "me", "je", "ve", "sa", "di"]
print(len(jours)) #Réponse → 7 éléments

vide = []
print(len(vide)) #Réponse → 0 élément

multi = [["x","y","z"], [[1,2], [3,4]], [1,2,3], "a", 10]
print(len(multi)) #Réponse → 5 éléments (uniquement les premiers niveaux)
```

Exercice 6 *(Nom fichier : C17_S01_E6)*

Quelle est la taille de chacune des listes ci-dessous :

```
fruits = ["banane", "abricot", "cerise", "poire"]

nombres = [1,2,3,4,5]

etrange = [[], [[[], []], [] ,[]]]
```

Tester si un élément existe, ou pas, dans une liste

```
fruits = ["banane", "abricot", "cerise", "poire"]
```

Pour tester l'existence d'un élément dans la liste ci- dessus:

```
if "abricot" in fruits :
    print("L'abricot est dans la liste.")
```

Pour tester la non-existence d'un élément dans la liste ci-dessus:

```
if "pomme" not in fruits :
    print("Pas de pomme dans cette liste.")
```

Exercice 7 *(Nom fichier : C17_S01_E07)*

Soit la liste : `villes = ["Genève", "Nyon", "Morges", "Lausanne"]`

a) Tester si : `"Lausanne" fait partie de la liste`

b) Tester si : `"Vevey" ne fait pas partie de la liste`

Cours 20.21 : les listes

Trier une liste

```
fruits = ["poire", "banane", "abricot", "cerise"]
```

Pour trier une liste en ordre ascendant : `fruits.sort()`:

```
print(fruits) #["abricot", "banane", "cerise", "poire"]
```

Le tri se fait par ordre ASCII :

Si strings : d'abord les lettres majuscules puis minuscules par ordre alphabétique ;
Si nombres : par ordre numérique.

Remarque les éléments doivent être d'un même type.

Il est possible de trier en ordre descendant :

```
fruits.sort(reverse=True)
print(fruits) #["poire", "cerise", "banane", "abricot"]
```

Exercice 8 *(Nom fichier : C17_S01_E08)*

Trier les listes ci-dessous, que remarquez-vous ?

a) `cantons = ["Zurich" , "Berne" , "Lucerne" , "Genève" ]`

b) `mots = ["catalogues", "Zoo", "arpège", "Nomenclature"]`

c) `nombres = [99, 7, 1, -8, 0]`

d) Trier la liste ci-dessus, en ordre descendant.

Parcourir une liste

Il est possible d'utiliser la boucle bornée **for**

Par exemple :
Afficher le contenu du panier des courses :

```
panier = ["poire", "abricot", "cerise"]
```

Méthode 1:

```
for f in panier:
    print("Article acheté : ", f)
```

Résultat :

```
Article acheté :  poire
Article acheté :  abricot
Article acheté :  cerise
```

Méthode 2:

```
for i in range(len(panier)):
    print("Article", i,"acheté : ", panier[i])
```

Résultat :

```
Article 0 acheté :  poire
Article 1 acheté :  abricot
Article 2 acheté :  cerise
```

Cours 20.21 : les listes

Exercice 9 *(Nom fichier : C17_S01_E09)*

Afficher la liste ci-dessous comme l'exemple :

```
fruits = ["Banane", "Abricot", "Cerise", "Poire"]

 Mes achats de fruits :
1) Banane
2) Abricot
3) Cerise
4) Poire
```

Liste de listes

Une liste peut contenir des éléments de tout type, et donc également de type liste !
Lorsqu'une liste contient des listes, on parle de liste de listes.

Le principe pour atteindre les éléments est de référencer chaque niveau de tableau avec un indexage standard [i]

Par exemple :
Soit la liste :

```
etats = [["chaud", "tiède", ["froid", "glacé"]], ["lumineux", "éclairé", "sombre"]]
```

Pour accéder à "lumineux" → 2ème tableau (indice 1) → premier élément (indice 0)
e1 = etats[1][0]

Pour accéder à "`glacé`" → 1er tableau (ind 0) → troisième élément (ind 2) → deuxième choix (ind 1)
e2 = etats[0][2][1]

Exercice 10 *(Nom fichier : C17_S01_E10)*

Soit la liste :

```
etats = [["chaud", "tiède", ["froid", "glacé"]], ["lumineux", "éclairé", "sombre"]]
```

a) Accéder à l'élément `"froid" ;`

b) Modifier l'élément "`sombre`" par "`foncé`"

Choisir un élément aléatoire d'une liste

Choice() du module random renvoie un seul élément aléatoire à partir d'une liste

```
import random
couleurs = ["rouge", "vert", "bleu"]
print(random.choice(couleurs)) → retourne une des couleurs de la liste
```

Exercice 11 *(Nom fichier : C17_S01_E11)*

Soit la liste :

```
fruits = ["Banane", "Abricot", "Cerise", "Poire"]
```

a) Tirer un fruit au hasard ;

b) Tirer 20 fruits au hasard (utiliser une boucle) → est-ce équilibré ?

Cours 20.21 : les listes

A tester *(Nom fichier : C17_S02_E00)*

Déterminer l'action des opérateurs + et * sur les listes en exécutant le script ci-dessous :

```
lst1=[1,2]
lst2=[3,4]

print(lst1)
print(lst2)

lst3=lst1 + lst2
print(lst3)

lst4=2 * lst1
print(lst4)

lst5=10 * []
print(lst5)

lst6=10 * [0]
print(lst6)

lst7=lst1 * lst2 # !!!
print(lst7 )
```

Exercices de consolidation

Exercice 01b *(Nom fichier : C17_S02_E01b)*

Écrire un programme qui demande le nombre de joueurs.
Ensuite le programme demande le prénom de chaque joueur et les mémorise.
A la fin des saisies des prénoms, le programme affiche tous les prénoms des joueurs.

```
Quel est le nombre de joueurs ? 4
Prénom du joueur 0 : Jessica
Prénom du joueur 1 : Ivan-Daniel
Prénom du joueur 2 : Sylvianne
Prénom du joueur 3 : Eliott
---------------------------------
Liste des joueurs :
joueur 0  :  Jessica
joueur 1  :  Ivan-Daniel
joueur 2  :  Sylvianne
joueur 3  :  Eliott
```

Cours 20.21 : les listes

Exercice 02b *(Nom fichier : C17_S02_E02b)*

Écrire un programme qui demande à l'utilisateur de saisir sa liste des achats qu'il souhaite réaliser dans le magasin.
La saisie se termine avec le mot "fin".

Une fois les achats terminés, le client peut vérifier les articles achetés : l'utilisateur entre les achats réalisés et le programme répond par "présent" ou "non présent".
La validation se termine avec le mot "fin".

Le programme affiche en conclusion les achats "non présents".
Si tous les achats sont présents, le programme affiche "Bravo !"

```
Saisir votre article à acheter (fin pour terminer) : fruits
Saisir votre article à acheter : pain
Saisir votre article à acheter : lait
Saisir votre article à acheter : beurre
Saisir votre article à acheter : fin
-----------------------------------------------
Saisir votre article à valider (fin pour terminer) : lait
présent
Saisir votre article à valider (fin pour terminer) : fruits
présent
Saisir votre article à valider (fin pour terminer) : chocolat
non présent
Saisir votre article à valider (fin pour terminer) : fin
-----------------------------------------------
Les achats manquants sont :
Achat  0  :  pain
Achat  1  :  beurre
Achat  2  :  chocolat
```