M. Quet

Python : Cours n° 7 : les listes (tableaux)

Les listes (ou tableaux) :

Une **liste** est une variable dans laquelle on peut mettre plusieurs variables.
Une liste est une variable qui peut stocker une suite (une collection) de nombres ou de caractères.

Une liste est caractérisée par le couple occurrence/index :

- **l'occurrence** est le rang de la donnée stockée, le premier rang étant par convention égal au rang 0 ;
- **le index** est la donnée stockée à un rang donné.

Ex : on saisit :

liste = [2,5,10]

On saisit liste dans la zone d'écriture ou print(liste) dans un programme :

→[2,5,10]

Python a créé un tableau comme suit :

occurrence	0	1	2
index	2	5	10

Cela signifie :

liste[0] est égal à 2
liste[1] est égal à 5
liste[2] est égal à 10

Ex : voiture = [''renault'', ''peugeot'', ''citroen'']

voiture →[renault,peugeot,citroen]

rang	0	1	2
contenu	renault	peugeot	citroen

voiture[0] →renault
voiture[1] →peugeot
voiture[2] →citroen

Voici quelques commandes essentielles que nous allons utiliser sur la liste suivante :

A = [2, 5, 10, 4]

Taille d'une liste : len(nom_de_la_liste)

Une liste possède une taille définie par le nombre de données qu'elle contient.

Ex : **len**(A)→4

Ajouter un élément dans une liste : append

La commande .append(valeur) ajoute une colonne à la liste à droite, elle agrandit la liste.

Ex **:** **A.append**(5) →ajoute 25 à la suite de la liste
A → [2, 5, 10, 4, 5]

Trier une liste dans l'ordre croissant : .sort()

La commande .sort() trie les données, numériques ou alphanumériques, dans l'ordre croissant.

Ex **:** **A.sort**() →effectue un tri
A → [2, 4, 5, 5, 10]

M. Quet

Modifier un élément/une valeur dans une liste : On peut redéfinir une case du tableau

Ex : **A[3] = 13**

A → [2, 4, 5, 13, 10]

Supprimer un élément dans une liste à partir de son index (de son rang) : del[index]

Ex : **del A[2]**

A → [2, 4, 13, 10]

Supprimer un élément dans une liste à partir de sa valeur : remove (valeur)

Ex : **A.remove(13)**

A → [2, 4, 10]

Inverser les valeurs d'une liste : reverse()

Ex : **A.reverse()**

A → [10, 4, 2]

Réinitialisation de la liste pour les fonctionnalités suivantes :

A = [1, 1, 2, 2, 3, 3, 3]

Compter le nombre d'occurrences d'une valeur : count(valeur)

Ex : **A.count(3)** →3

Trouver la première position d'un index : index

Ex : **A.index(3)** →4

Afficher les 3 premières occurrences :	liste[:3]	→ [1, 1, 2]
Afficher la dernière occurrence :	liste[-1]	→3
Afficher les 4 dernières occurrences :	liste[-4:]	→[2, 2, 3, 3]
Afficher la 4è occurrence en partant de la fin	liste[-4]	→2
Afficher toutes les occurrences :	liste[:]	→[1, 1, 2, 2, 3, 3, 3]
Afficher de la 2ème à la 5ème occurrence	liste[1:5]	→ [1, 2, 2, 3]

Vider une liste **liste[:]=[]**

On peut additionner deux listes :

x = [1, 2, 3]
y = [4, 5, 6]
x + y →[1, 2, 3, 4, 5, 6]

NB : x et y restent inchangés

z = *x* + y
print(z) →[1, 2, 3, 4, 5, 6]

On peut multiplier deux listes :

x = [1, 2]
x *5
[1, 2, 1, 2, 1, 2, 1, 2, 1, 2]

→cela peut être utile pour **initialiser une liste** :

[0] * 5
[0, 0, 0, 0, 0]

M. Quet

On peut boucler sur une liste pour récupérer chaque occurrence :

```
liste = ["a", "d", "m"]
for lettre in liste:
    print(lettre)          → a
                             d
                             m
```

On peut boucler sur une liste pour récupérer chaque occurrence et son index : enumerate

```
liste = ["a", "d", "m"]
for lettre in enumerate(liste):
   print(lettre)
                    → (0, 'a')
                      (1, 'd')
                      (2, 'm')
```

La fonction range génère une liste composée d'une simple suite arithmétique.

Ex : range(10) →[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
range(1,6) →[1, 2, 3, 4, 5]

On peut mettre à bout deux listes : extend(liste)

```
x = [1, 2, 3, 4]
y = [4, 5, 1, 0]
print(x + y)      →[1, 2, 3, 4, 4, 5, 1, 0]
x.extend(y)
print(x)          →[1, 2, 3, 4, 4, 5, 1, 0]
```

Pour savoir si un élément est dans une liste : in

```
liste = [1, 2, 3, 5, 10]
3 in liste        →True
11 in liste       →False
```

```
liste = ['a', 'e', 'i', 'o', 'u']
'o' in liste      →True
'y' in liste      →False
```

Pour copier une liste dans une autre liste qui soit indépendante de la première !!!

Méthode qui ne marche pas :

```
x = [1, 2, 3]
y = x
y[0] = 4
x                 →[4, 2, 3]
```

En fait cette syntaxe permet de travailler sur un même élément nommé différemment

Méthode qui marche :

```
x = [1, 2, 3]
y = x [:]
y[0] = 9
x                 →[1, 2, 3]
y                 →[9, 2, 3]
```

Exercice

Générer un tableau contenant la liste des noms de tous les élèves de la classe :

```
liste_classe = []
for i in range(36):
   nom = input("Saisir un nom :")
   liste_classe.append(nom)
print(liste_classe)
```

→pour cinq valeurs, on obtient par exemple :

```
['Alain', 'Pierre', 'Thierry', 'Fredo', 'Charles']
```

M. Quet

Exercice

Générer deux tableaux contenant la liste des noms de tous les élèves de la classe et dans l'autre les âges respectifs :

```
liste_classe = []
liste_âge = []
for i in range(3):
    nom = input("Saisir un nom :")
    age = int(input("Saisir son âge :"))
    liste_classe.append(nom)
    liste_âge.append(age)
print(liste_classe)
print(liste_âge)
```

→pour cinq valeurs, on obtient par exemple :

```
['Ombeline', 'Alyssa', 'Louison']
 [15, 16, 15]
```