08-listes

April 4, 2019

1 Les listes

Comme son nom l'indique, une liste permet de lister différents éléments.

Les éléments d'une liste : - s'écrivent entre crochets, - sont séparés par une virgule - peuvent être de nature différente (chaine de caractères, nombre entier, nombre réel,...) - sont repérés par leur position dans la chaîne, appelée indice.

ATTENTION : le premier indice d'une liste a pour valeur 0 !

Remarque sur l'organisation de ce fichier noteboook : très souvent, les cellules fonctionnent par paire; elles portent alors le même titre. La première cellule explique le cours et la suivante est une cellule de codes illustrant le cours.

2 Plusieurs manières de récupérer les éléments d'une liste : cours

Nous allons travailler sur un exemple :

```
L = [5,2,8,17,6,14]   crée une liste contenant les éléments 5, 2, 8, 17, 6, 14
L[0]                  renvoie le premier élément, ici 5
L[2]                  renvoie l'élément d'indice 2 (en 3ème position donc), ici 8
L[-1]                 renvoie le dernier élément, ici 14
L[-2]                 renvoie l'avant-dernier élément, ici 6
L[1:3]                renvoie les éléments d'indice 1 et 2  (ATTENTION : indice 3 non inclus)
L[3:]                 renvoie les éléments à partir de l'indice 3.
len(L)                renvoie la longueur de la liste, ici 6
L1 = []            crée une liste vide
```

In [ ]:
```
# Plusieurs manières de récupérer les éléments d'une liste : applications

L=[5,2,8,17,6,14]
print(L)
print(L[0])
print(L[2])
print(L[-1])
print(L[-2])
print(L[1:3])
print(L[3:])
print(len(L))
```

3 Plusieurs méthodes pour assigner une valeur ou supprimer une valeur d'une liste (cours et applications)

Les méthodes modifient les listes et leur syntaxe est toujours silmilaire : L.méthode()

```
In [ ]: # PLusieurs méthodes pour assigner une valeur ou supprimer des valeurs d'une liste (co

        L=[5,2,8,17,6,14]
        print(L)
        L.append(20)                # Ajoute lélément 20 à la fin de la liste L.
        print(L)
        L.insert(2,20)              # Insère lélément 20 à la position d'indice 2 (donc en tr
        print(L)
        L.remove(20)                # Supprime la première occurrence de lélément 20 dans la
        print(L)
        L.pop(-1)                   # Supprime lélément dindice -1 (donc le dernier élément)
        print(L)
```

4 Quelques fonctions (et une méthode) permettant d'analyser le contenu d'une liste : cours

Les fonctions ne modifient pas les listes et leur syntaxe est toujours similaire : fonction(L)
Les méthodes modifient les listes et leur syntaxe est toujours silmilaire : L.méthode()
min(L) : Renvoie le plus petit élément de la liste L.
max(L) : Renvoie le plus grand élément de la liste L.
sorted(L) : Renvoie une copie triée de la liste contenant les éléments de la liste L rangés par ordre croissant ou alphabétique. MAIS, la liste L n'est pas modifiée !
sorted(L,reverse=True) : Renvoie une copie triée de la liste contenant les éléments de la liste L rangés par ordre décroissant ou inverse du sens alphabétique. MAIS, la liste L n'est pas modifiée !
L.sort() : Modifie la liste L qui dorénavant contiendra les éléments triés (mais ne la renvoie pas).
Remarque : sort() est une méthode et non une fonction... D'où sa syntaxe différente.
choice(L) : Renvoie en choisissant au hasard un élément de la liste L. Cette fonction appartient au module random.
sample(L,3) : Retourne une liste de 3 éléments choisis aléatoirement et sans remise dans la liste L. Cette fonction appartient au module random.

```
In [ ]: # Quelques fonctions permettant d'analyser le contenu d'une liste : Applications
        from random import choice, sample          # Les fonctions sample et choice appartienn

        L=[5,2,8,17,6,14]
        print(L)
        print(min(L))
        print(max(L))
        print(sorted(L))
        print("Comme vous pouvez le constater, la liste L n'est pas modifiée",L)
        print(sorted(L,reverse=True))
```

```
print("Comme vous pouvez le constater, la liste L n'est pas modifiée",L)
L.sort()
print("Comme vous pouvez le constater, la liste L est modifiée",L)
print(choice(L))
print(sample(L,3))
```

5 Comment parcourir le contenu d'une liste : cours

La boucle ń for ż est particulièrement bien adaptée aux listes de valeurs.

Soit L une liste de longueur n :

→ Si on a besoin de parcourir une liste élément par élément grâce à leur indice pour ensuite utiliser une instruction faisant intervenir cet indice, on utilise l'instruction : **for i in range(0, len(L)) :**

la variable i (qui représente l'indice d'un élément) prendra les valeurs de 0 à len(L)-1 soit de 0 à n-1

→ Si on a besoin de parcourir une liste, élément par élément, en nous intéressant uniquement à leur valeur pour ensuite utiliser une instruction permettant de travailler sur ces valeurs, on peut utiliser l'instruction : **for x in L :**

la variable x prendra l'une après l'autre toutes les valeurs de la liste L.

Remarque 1: On peut toujours utiliser la première instruction à la place de la deuxième, mais pas l'inverse !

Remarque 2: les lettres i, j et k sont traditionnellement utilisées pour désigner les indices alors que les autres lettres désignent des variables. Par exemple, ici, la lettre x parcourt les valeurs de L.

```
In [ ]: #Comment parcourir le contenu d'une liste : applications de base

        L=[5,2,8,17,6,14]
        for i in range(0,len(L)):
            if i%2==0 :      # test permettant de sélectionner les indices i pairs (reste de la
                print(L[i])

        for x in L :
            if x > 7:
                print(x)

In [ ]: #Comment parcourir le contenu d'une liste : applications plus poussées

        L=[5,2,8,17,6,14]
        for i in range (0, len (L)):
            L[i] = L[i] + 1
        print(L)

        L=[5,2,8,17,6,14]
        s = 0
        for x in L :
            s = s + x
        print (" Somme : ", s)
```

```
if 8 in L : print("le nombre 8 est présent dans la liste")
else : print("le nombre 8 n'est pas présent dans la liste")
```

6 Opérations mathématiques sur les listes : cours

On peut effectuer diverses opérations mathématiques (addition, multiplication...) entre des listes.

En voici quelques exemples : (L1, L2 et L sont des listes)

L = 3 * L1 : L sera une liste contenant les éléments de L1, répétés 3 fois. Elle sera donc trois fois plus longue.

L = L1 + L2 : L sera la concaténation de L1 et L2. elle contiendra d'abord les éléments de L1, puis ceux de L2

```
In [ ]: # Opérations mathématiques sur les listes : applications
        L1=[1,2,3]
        L2=[4,5,6]
        L=3*L1
        print(L1)
        print(L)
        L=L1+L2
        print(L)
```

7 Conversion d'une chaîne de caractères vers une liste de caractères ou l'inverse : cours

→ D'une chaîne de caractères vers une liste de caractères

```
ch='ISN'   :     Création d'une chaîne de caractères nommée ch  et assignation de cette chaîne a

list(ch)   :     Convertit la chaîne de caractères ch en liste de caractères
```

→ D'une liste de caractères vers une chaîne de caractères

Attention : la liste doit être forcément constituée de caractères

```
L = ['I','S','N']  :   crée une liste contenant 3 éléments de type caractère
'sep'.join(L)  :  renvoie une chaîne de caractères obtenue en concaténant les éléments de la li
```

```
In [ ]: ch='ISN'
        chbis=list(ch)
        print(ch)
        print(chbis)

        L=['I','S','N']
        chter='-'.join(L)      # ici le séparateur est un tiret -
        print(L)
        print(chter)
```

8 Création de listes de listes (donc de tableau !) : cours

Nous allons travailler sur un nouvel exemple

```
tableau = [['Anne','Tom','Léo','Eva'], [6,7,8,9],[10,20,30,40]]  :  crée un tableau contenant 3
Tableau[0]                          # renvoie la première liste
Tableau[i][j]                       # renvoie le jème élément de la ième liste
```

In [ ]:
```
# On peut créer des listes de listes (donc un tableau !)

tableau = [['Anne','Tom','Léo','Eva'], [6,7,8,9],[10,20,30,40]]
print(tableau)
print(tableau[0])
print(tableau[0][0])
print(tableau[1][2])
print(tableau[-1][-1])
print(tableau[-1][0])
```