Cours3_Listes

November 4, 2019

1 Les listes

2 Définitions et opérations sur les listes

2.0.1 Prise en main

Pour une présentation des listes et de leurs méthodes : voir aussi Python_seconde

```
In [1]: L = [-3, 'ISN', 2.016]
```

Accès en lecture à l'élément d'index `k` par `L[k]`. Les index repèrent les éléments parcourus de gauche à droite en commençant par 0.

```
In [2]: L[0], L[1], L[2]
```

```
Out[2]: (-3, 'ISN', 2.016)
```

L'accès par index est aussi possible en écriture

```
In [4]: L[1] = 'NSI'
```

```
In [5]: L
```

```
Out[5]: [-3, 'NSI', 2.016]
```

Longueur d'une liste

```
In [9]: len(L)
```

```
Out[9]: 3
```

Accès hors de la plage d'index => Erreur !

```
In [10]: L[len(L)-1]
```

```
Out[10]: 2.016
```

```
In [11]: L[len(L)]
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-11-6889782ef819> in <module>
----> 1 L[len(L)]


IndexError: list index out of range
```

Pourtant des index négatifs sont possibles (index de -1 à -len(L), les éléments étant parcourus de droite à gauche

```
In [12]: L[-1], L[-2], L[-3]

Out[12]: (2.016, 'NSI', -3)

In [13]: L[-4]
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-13-ac8eb235dc48> in <module>
----> 1 L[-4]


IndexError: list index out of range
```

Vérification de type

```
In [3]: type(L)

Out[3]: list

In [6]: type(L[0]), type(L[1]), type(L[2])

Out[6]: (int, str, float)
```

Les listes sont donc des séquences ordonnées d'éléments éventuellement hétérogènes en type.

```
In [7]: listeVide = []
```

Liste de listes => pour représenter un tableau à 2 (n) dimensions, une grille, une matrice ...

```
In [8]: grille = [[0,1,0], [1,0,0], [0,0,1]]
```

Accès aux éléments d'une liste de listes (en dimension 2)
Premier niveau d'index => `grille[k]` est une liste

```
In [14]: grille[0]

Out[14]: [0, 1, 0]
```

Second niveau d'index : `grille[0][1]` est un entier celui situé en ligne d'index 0 (la première) et colonne d'index 1 (la seconde)

```
In [15]: grille[0][1]

Out[15]: 1
```

2.0.2 Méthodes sur les listes

Ajout d'un élément

```
In [16]: L = []

In [17]: L.append(734)

In [18]: L

Out[18]: [734]

In [19]: L.append(733)

In [20]: L

Out[20]: [734, 733]
```

Suppression d'un élément
Première méthode à partir de son index avec la méthode `pop`

```
In [21]: L.pop(1)

Out[21]: 733

In [22]: L

Out[22]: [734]
```

Deuxième méthode à partir de sa valeur avec Première méthode à partir de sa valeur avec la méthode `remove`

```
In [26]: L = [734,732,734,733]

In [27]: L.remove(734)

In [28]: L
```

```
Out[28]: [732, 734, 733]
```

Explication ?

```
In [29]: help(L.remove)

Help on built-in function remove:

remove(...) method of builtins.list instance
    L.remove(value) -> None -- remove first occurrence of value.
    Raises ValueError if the value is not present.
```

Extension par une autre liste

```
In [ ]: L = [734,732,734,733]

In [30]: L.extend([735, 736])

In [31]: L

Out[31]: [732, 734, 733, 735, 736]
```

Autre méthode pas tout à fait équivalente si on suit les objets à la trace dans la mémoire

```
In [33]: L = [734,732,734,733]
         L = L + [735, 736]
         L

Out[33]: [734, 732, 734, 733, 735, 736]
```

Trions !
Sur place

```
In [ ]: L = [734,732,734,733]

In [34]: L.sort()

In [35]: L

Out[35]: [732, 733, 734, 734, 735, 736]
```

Ou en créant une nouvelle liste

```
In [36]: L1 = [734,732,734,733]
         L2 = sorted(L1)

In [37]: L1, L2

Out[37]: ([734, 732, 734, 733], [732, 733, 734, 734])
```

2.0.3 Parcours de liste

On peut parcourir une liste de deux façons, les deux codes suivants sont équivalents.

- par index :

```
L = [731,734,732]
for k in range(len(L)):
    print(L[k])
```

- par valeurs :

```
L = [731,734,732]
for v in L:
    print(L)
```

2.0.4 Exercice 0

```
In [38]: #parcours par index
         def somme(t):
             """Retourne la somme
             des éléments de la liste de nombres
             t"""
             s = 0
             for k in range(len(t)):
                 s = s + t[k]
             return s

         #parcours par valeur
         def somme2(t):
             """Retourne la somme
             des éléments de la liste de nombres
             t"""
             s = 0
             for x in t:
                 s = s + x
             return s

         def moyenne(t):
             # s = somme2(t)
             # m = s / len(t)
             # return m
             return somme2(t) / len(t)
```

2.0.5 Exercice 1

```
In [2]: from random import randint
        L = [randint(0, 20) for _ in range(35)]  #liste de 35 entiers aléatoires entre 0 et 20
```

```
        x = 10                                          #le seuil
        print("Liste : ", L)
        print("Seuil : ", x)
        n = 0
        for e in L:
            if e > x:
                n = n + 1
        print("Nombre d'éléments de la liste L supérieurs au seuil : ", n)
```

```
Liste :  [18, 10, 2, 20, 3, 15, 20, 12, 0, 8, 9, 11, 12, 4, 13, 8, 16, 17, 7, 1, 15, 4, 4, 19,
Seuil :  10
Nombre d'éléments de la liste L supérieurs au seuil :  18
```

2.1 Exercice 2

```
In [7]: from random import randint
        L = [randint(0, 20) for _ in range(35)]  #liste de 35 entiers aléatoires entre 0 et 20
        x = 10                                          #le seuil
        print("Liste : ", L)
        print("Cible : ", x)
        occurence = []
        for k in range(len(L)):
            if L[k] == x:
                occurence.append(k)
        print("Liste des indices des éléments de la liste L égaux à la cible : ", occurence)
```

```
Liste :  [4, 9, 4, 11, 10, 20, 15, 11, 8, 14, 1, 4, 4, 20, 11, 7, 9, 19, 8, 11, 1, 5, 7, 9, 0,
Cible :  10
Liste des indices des éléments de la liste L égaux à la cible :  [4]
```

3 Listes par compréhension

3.0.1 Exercice 3

Quelle liste produit ce code ?

```
L = []
for k in range(10):
    L.append(k ** 2)
```

```
In [13]: #Corrigé  de l'exercice 3
         L = []
         for k in range(10):
             L.append(k ** 2)
         print(L)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
In [15]: #La même liste mais décrite par compréhension
         L2 = [ k ** 2 for k in range(10)]
         print(L2 == L)
```

```
True
```

3.1 Exercice 4 Liste de tous les carrés des entiers pairs entre 0 et 100

```
In [1]: [x ** 2 for x in range(0, 101, 2)]
```

```
Out[1]: [0,
         4,
         16,
         36,
         64,
         100,
         144,
         196,
         256,
         324,
         400,
         484,
         576,
         676,
         784,
         900,
         1024,
         1156,
         1296,
         1444,
         1600,
         1764,
         1936,
         2116,
         2304,
         2500,
         2704,
         2916,
         3136,
         3364,
         3600,
         3844,
         4096,
         4356,
         4624,
         4900,
         5184,
```

```
5476,
5776,
6084,
6400,
6724,
7056,
7396,
7744,
8100,
8464,
8836,
9216,
9604,
10000]
```

3.2 Exercice 4 Images positives des entiers entre 0 et 100 par la fonction cosinus

```
In [2]: from math import cos

In [3]: [cos(n) for n in range(0, 101) if cos(n) >= 0]

Out[3]: [1.0,
         0.5403023058681398,
         0.28366218546322625,
         0.960170286650366,
         0.7539022543433046,
         0.004425697988050785,
         0.8438539587324921,
         0.9074467814501962,
         0.1367372182078336,
         0.6603167082440802,
         0.9887046181866692,
         0.40808206181339196,
         0.424179007336997,
         0.9912028118634736,
         0.6469193223286404,
         0.15425144988758405,
         0.9147423578045313,
         0.8342233605065102,
         0.7654140519453434,
         0.9550736440472949,
         0.26664293235993725,
         0.5551133015206257,
         0.9998433086476912,
         0.5253219888177297,
         0.3005925437436371,
         0.9649660284921133,
         0.7421541968137826,
```

```
0.022126756261955736,
0.8532201077225842,
0.8998668269691937,
0.11918013544881928,
0.6735071623235862,
0.9858965815825497,
0.39185723042955,
0.4401430224960407,
0.9933903797222716,
0.6333192030862999,
0.17171734183077755,
0.9217512697247493,
0.8243313311075577,
0.7766859820216312,
0.9496776978825432,
0.24954011797333814,
0.569750334265312,
0.9993732836951247,
0.5101770449416689,
0.31742870151970165,
0.9694593666699876,
0.7301735609948197,
0.0398208803931389,
0.8623188722876839]
```

3.3 Exercice 4 Tables de multiplications

```
In [20]: # Peuplement par boucles imbriquées
         tables = []
         for i in range(1, 11):
             nouvelle = []
             for j in range(1, 11):
                 nouvelle.append(i * j)
             tables.append(nouvelle)
         print(tables)
```

```
[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10], [2, 4, 6, 8, 10, 12, 14, 16, 18, 20], [3, 6, 9, 12, 15, 18, 2
```

```
In [19]: # Peuplement par listes en compréhension
         tables2 = [ [i * j for j in range(1, 11)]  for i in range(1, 11) ]
         print(tables2 == tables)
```

```
True
```

3.4 Exercice 5 Permuter les éléments d'une liste

Écrire un code pour permuter les éléments d'indices a et b de la liste L.

```
L = [2 , 3 , 5 , 7 , 11 , 13 , 17 ]
a = 2
b = 5
```

```
In [4]: #Corrigé
        L = [2 , 3 , 5 , 7 , 11 , 13 , 17 ]
        a = 2
        b = 5
        L[a], L[b] = L[b], L[a]
```

3.5 Exercice 6

Compléter l'instruction évaluée dans la console ci-dessous pour que la liste `L3` contienne alternativement les éléments des listes `L1` et `L2` de même taille.

```
In [4]: L1 =[2, 3, 5, 7]

In [5]: L2 = [17, 24, 35, 81]

In [6]: L3 = [L1[i // 2] if i % 2 == 0 else L2[i//2] for i in range(2 * len(L1))]

In [7]: L3

Out[7]: [2, 17, 3, 24, 5, 35, 7, 81]
```

3.6 Exercice 7

```
In [8]: #Pour l'utilisation de Python Tutor dans le notebook
        from metakernel import register_ipython_magics #nécessite d'installer metakernel avec
        register_ipython_magics()

In [10]: %%tutor

         M = [ [0, 0, 0] for i in range(3) ]
         N = M
         P = [e for e in M ]
         Q = [ e[:] for e in M ]
         M[2][1] = 3
```

```
<IPython.lib.display.IFrame at 0x7fbcd0be6ba8>
```