TD1-Les-Listes

02 DECEMBRE 202

1 Cours et activités sur les listes

1.1 Définition

Une **liste** en python permet de stocker des données. On parle de *structure de données* ou bien d'objet *conteneur*, c'est à dire d'un objet qui peut en contenir un autre. Une liste est un objet modifiable (**mutable**) contrairement à un tuple (anglicisme informatique signifiant "Table UPLEt").

On notes les listes avec des crochets, les valeurs séparées par des virgules : [valeur1 , valeur2 ,valeur3]

Par exemple :

[ ]:
```
liste1 = [1,2,3] # une liste d'entiers
liste2 = [1.2,0.2,3.7] # une liste de flotants
liste3 = [[1,2,3],[4,5],[10,20,30]] # une liste de listes
liste4 = ['a','b','c'] # une liste de str
liste5 = [(1,2),(4,5)] # une liste de tuples, les tuples ne sont pas
↪modifiables, essayez !

liste6 = ["ab",[2,(5>4,False)],56.5,None] #Une liste de n'importe quoi .
print(liste1)
print(liste2)
print(liste3)
print(liste4)
print(liste5)
print(liste6)
```

1.2 Comment créer une liste en python ?

Pour **initialiser une liste**, on utile la syntaxe suivante qui va créer une variable de type list, vide.
- A l'aide de la fonction "list()" - A l'aide des crochets "[ ]"

[ ]:
```
a=list() # A l'aide de la fonction "list()"
b=[]     # A l'aide des crochets "[]"
```

On peut vérifier le type de la variable créée.

[ ]:
```
print('La liste a est de type : ' ,type(a) )
print('La liste b est de type : ' ,type(b) )
```

1.3 Comment ajouter des valeurs dans une liste python ?

1. Ajout à la création : Vous pouvez **ajouter les valeurs** que vous voulez lors de la création de la liste python.

```
[ ]: a=[10,20,30] # on créé une liste avec 3 éléments
     print(a)
```

2. Ajout avec la méthode **.append()** Ou les ajouter **après la création de la liste** avec la méthode **.append()** (qui signifie “ajouter” en anglais).

La syntaxe est alors : liste.append(élément)

```
[ ]: print('avant .append() :' , a)
     a.append(27) # on ajoute 27 à la fin de la liste
     print('après .append() :' , a)
```

1.4 Comment insérer une valeur dans une liste python ?¶

On peut insérer l’élément *elemt* dans la liste à l’indice *indice* et décaler les éléments suivant avec : **liste.insert(indice,elemt)**

```
[ ]: print('avant .insert(2,-150) :' , a)
     a.insert(2,-150) # on ajoute -150 en 3e position de la liste (ou à
     l'indice 2) print('après .insert(2,-150) :' , a)

     # si l'indice est suppérieur à la longueur de la liste, on l'ajoute à
     la fin a.insert(20,-70) # l'indice 20 est supérieur à la longueur de
     la liste, on␣
      ↪ajoute à la fin print('après
     .insert(20,-70) :' , a)
```

1.5 La longueur d’une liste python.

- La **longueur d’une liste** est le nombre des éléments qui la compose. On l’obtient avec la fonction *len(ma_liste)*.

```
[ ]: # Une liste de 4 tuples : (1,2) , (3,4) , (5,6) , (7,8)
     autre_liste=[(1,2),(3,4),(5,6),(7,8)]

     print(autre_liste)
     print('La longueur de la liste a est : ' , len(autre_liste))
     len(autre_liste)
```

1.6 Un premier exercice

```
[ ]: ## Exercice 1 : Question 1
     # 1.a Créer une liste vide nommé *maliste1*.
     # 1.b Cette liste comprendra les valeurs :
     2,3,5,7,11,13,17 # 1.c Afficher sa longueur.
     # 1.d Insérer 51 à l'indice 2
```

1.7 Les liste définies en compréhension

Les listes en compréhension sont une syntaxe présente dans le langage Python (entre autres) permettant de filtrer un itérable (comme une liste). En gros, cela permet l'écriture d'une boucle for dont la finalité est de créer une liste. Un exemple sera plus parlant.

[ ]:
```
# On va créer la liste des carrés de 0² à 10²
liste_boucle = []
for i in range(11):
    liste_boucle.append(i*i)
print(liste_boucle)
```

On aurait pu définir cette liste directement par compréhension :

[ ]:
```
liste_comprehension=[i*i for i in range(11)]
print(liste_comprehension)
```

[ ]:
```
## Exercice 1 : Question 2
# Ecrire par compréhension, la liste des nombres impairs de 1 à 31. On
 nommera␣ ↪cette liste maliste2
# Ecrire avec une boucle, la liste des nombres pairs de 2 à 32. On
 nommera␣ ↪cette liste maliste3
```

1.8 Comment récupérer une valeur dans une liste python?

Pour lire une liste, on peut demander à voir l'index de la valeur qui nous intéresse.

[ ]:
```
# Une liste de 4 tuples : (1,2) , (3,4) , (5,6) , (7,8)
autre_liste=[(1,2),(3,4),(5,6),(7,8)]
autre_liste[0] # désigne le premier élément de la liste, d'index 0
```

[ ]:
```
autre_liste[1] # désigne le 2e élément de la liste, d'index 1
```

Le premier item commence toujours avec l'index 0. Pour lire la premier item on utilise la valeur 0, le deuxième on utilise la valeur 1, etc.

Il est d'ailleurs possible de modifier une valeur avec son **index**.

[ ]:
```
print('Liste avant la modification du 3e élément' ,
autre_liste) autre_liste[2]='bonjour' # on modifie le 3e
élément, d'index 2 print('Liste après la modification du 3e
élément' , autre_liste)
```

Faites de même avec votre liste **maliste**. Changer des éléments de votre liste.

[ ]:
```
## Exercice 1 : Question 3a
 # Accéder aux valeurs de maliste2 et essayer un accès avec un index 20
```

[ ]:
```
## Exercice 1 : Question 3b
```

```
```

```
# On considère la liste3 qui est une liste de listes
Liste3 = [[100,200,300],[11,22],[4,5,6]] # une liste de listes
# Le premier item de cette liste est : liste3[0] soit [100,200,300]
 (qui est␣ ↪une liste)
# Le DEUXIEME item de cette liste est : liste3[1] soit
[11,22] # Afficher les 3 items de cette liste sans
afficher la liste.
```

[ ]:
```
## Exercice 1 : Question 3c
# Comment accéder directement à la valeur 300, le 3e élément de l'item
1 :␣
 ↪[100,200,300]?
# Comment accéder à la valeur 22, le 2e élément de l'item 2 : [11,22]
?
# Comment accéder à la valeur 4, le 1er élément de l'item 3 : [4,5,6]?
```

1.9 Supprimer une entrée : `del`, `.remove(x)` ou `.pop(i)`

Il y a trois méthodes pour supprimer une entrée : avec son index (del ou pop), avec sa valeur (remove).

1.9.1 Avec l'index : la fonction `del liste[ i ]` ou `pop(i)`

- La fonction `del liste[i]` va supprimer l'item à l'index i spécifié (pas besoin de connaître l'item).
- La méthode `liste.pop(i)` va supprimer l'item à l'index i spécifié et renvoyer sa valeur. Si on ne donne pas i, c'est le dernier élément qui est supprimé

[ ]:
```
liste_c=[(1,2),(3,4),(5,6),(70,80),(70,80)] print('Liste
avant la suppression du 2e élément' , liste_c) del liste_c[1]
# on supprime le 2e élément print('Liste après la suppression
du 2e élément avec del' , liste_c)
# on supprime le 3e élément et on le renvoie
print('Suppression du 3e élément avec pop qui le donne' ,liste_c.pop(2))
print('Liste après la suppression du 3e élément avec pop' , liste_c)
```

1.9.2 Avec sa valeur : la méthode .remove(x).

Il est possible de supprimer une entrée d'une liste avec sa valeur avec la méthode `.remove()` . `liste.remove(x)` : supprime de la liste le premier élément dont la valeur est égale à x

La syntaxe est : `maliste.remove(élément)`

[ ]:
```
print('Liste2 avant la suppression du 2e élément' , liste2)
liste2.remove(3.7) # on supprime 3.7
print('Liste2 après la suppression du 2e élément' , liste2)
```

On pourrait par exemple dans votre liste `maliste` précédent supprimer le 4e élément et la valeur 17.

[ ]:
```
## Exercice 1 : Question 4
# Supprimer le 4e élément et la valeur 17 de maliste2.
```

1.10 Inverser les valeurs d'une liste

Vous pouvez inverser les items d'une liste avec la méthode `.reverse()` .

[ ]:
```
print('Liste3 avant .reverse()' , liste3)
liste3.reverse() # on inverse les éléments de la liste
print('Liste3 après .reverse()' , liste3)
```

[ ]:
```
## Exercice 1 : Question 5
#  Inverser les valeurs de maliste2
```

1.11 Compter le nombre d'occurences d'une valeur

Pour connaitre le nombre d'occurences d'une valeur dans une liste, vous pouvez utiliser la méthode `.count()`

[ ]:
```
liste5=[1,1,1,2,2,2,4,9,7,5,3,69,4,1,2,6,2,3,4,1,3,2,8]
liste5.count(1)
```

1.12 Trouver l'index d'une valeur

La méthode `.index()` vous permet de connaitre la position de l'item cherché. la syntaxe est : `liste.index(v1)`

[ ]:
```
liste6 = ["a","b","c","d","e","e"]
liste6.index("a") # on cherche l'index de l'élément 'a'
```

[ ]:
```
liste6.index("e") # on cherche l'index de l'élément 'e', il y en a 2, il renvoit le 1er indice
```

[ ]:
```
liste6.index("f") # on cherche l'index de l'élément 'f', il n'y en a pas
```

[ ]:
```
## Exercice 1 : Question 6
#  Trouver l'index de 21 dans maliste2
```

1.13 Astuces - Manipuler une liste : les slices

Voici quelques astuces pour manipuler des listes:

[ ]:
```
liste = [1, 10, 100, 250, 500]
liste[0] # le 1er élément
```

```
[ ]: liste = [1, 10, 100, 250, 500]
     liste[-1] # Cherche le dernier élément
```

```
[ ]: liste = [1, 10, 100, 250, 500]
     liste[-4:] # Affiche les 4 dernièrs éléments (occurrences)
```

```
[ ]: liste = [1, 10, 100, 250, 500]
     liste[:] # Affiche touts les occurences (éléments)
```

```
[ ]: liste = [1, 10, 100, 250, 500,600,700]
     liste[2:5] # les éléments d'indice 2 à 4
```

```
[ ]: # On peut créer facilement une liste par "multiplication"
     mult=['r']*10
     print(mult)
```

1.14 Comment parcourir une liste python?

1.14.1 Une boucle directe sur la liste

```
[ ]: liste9=[10,20,30,40,50,60]
     for element in liste9:
         print (element)
```

1.14.2 Une boucle avec les indices, il faut alors connaître la longueur de la liste

```
[ ]: liste9=[10,20,30,40,50,60] longueur=len(liste9) # pour obtenir la
     longueur for index in range(longueur): print("La valeur d'index ",
     index, "de la liste9 est :", liste9[index])
```

1.14.3 On peut aussi utiliser la fonction `enumerate()` qui renvoie l'indice et la valeur

```
[ ]: for index,valeur in enumerate(liste9):
         print("La valeur d'index ", index, "de la liste9 est :",
         liste9[index])
```

```
[ ]: ## Exercice 1 : Question 7
     # 5.1 Afficher les valeurs de votre liste maliste2 avec les
      différentes␣ ,→méthodes proposées
     # 5.2 Afficher indices et les valeurs de votre liste
```

1.15 Comment copier une liste : méthode copy() ou fonction list()

Supposons que l'on souhaite effectuer une copie de la liste9.

```
[ ]: print(liste9)
une_copie=liste9
print(une_copie)
```

```
[ ]: # On va modifier la liste copiée, on remarque qu'alors la liste9 aussi est
     ↪ modifiée.
une_copie[0]='OK'  # on modifie le 1er élément
print('liste9= ',liste9)
print('une_copie= ',une_copie)
```

On remarque que les deux listes ont été modifiées. Cela montre qu'il n'y a en fait qu'un seul objet et que liste9 et une_copie sont en fait des références vers le même objet. Ces instructions ne créent pas une copie de la liste. Elles se contentent de créer ce que l'on appelle un alias, c'est-à-dire un autre nom pour désigner le même objet.

Pour obtenir une vraie copie d'une liste, il faut utiliser, la méthode `copy()` ou la fonction `list()` :

```
[ ]: a3=liste9.copy()
a3[2]='modif' # on modifie le 3e élément
print('liste9= ',liste9)
print('a3= ',a3)

a4=list(liste9)
a4[1]='COUCOUmodif' # on modifie le 2e élément
print('liste9= ',liste9)
print('a4= ',a4)
```

```
[ ]: ## Exercice 1 : Question 8
# Copier votre liste maliste2 dans une autre liste et vérifier les résultats du
     ↪ cours.
# Vérifier alors les remarques du cours.
```

1.16 Concaténer deux liste : +

Il est possible de concaténer deux listes en les ajoutant :

```
[ ]: listeA = ['apples', 'oranges', 'pears']
listeB= ['pears' , 'grapes' , 'lemons']
concatenation = listeA+listeB
print(concatenation)
```

1.17 Trier une liste : méthode .sort()

On peut facilement trier une liste avec la méthode `.sort()`

```
[ ]: liste10=[20,54,2,97,3,111,6,7]
liste10.sort()
print(liste10)
```

1.18 Des chaînes aux listes : split

Pour convertir une chaîne en liste on peut utilisee la méthode `split()`. La syntaxe est : `maChaine.split(elementSeparateur)` ou `list(chaîne)`

```
[ ]: p="Bonjour tout le monde, comment ça va ?"
     p.split(" ")
```

1.19 Des chaînes aux listes : list()

Pour convertir une chaîne en liste on peut aussi utiliser la fonction `list()`. La syntaxe est : `list(chaîne)`

```
[ ]:
print(s)
s=list('bonjour')
```

2 Résumé des notions abordées

- Une liste L est un objet conteneur qui est mutable.
- Pour créer une liste, on utilise la syntaxe :
  - `liste = [valeur1,valeur2,valeur3]`
- Pour récupérer une valeur dans une liste python. -
  - la méthode directe : `nom_liste[index]`
  - Attention le 1er élément de la liste L est L[0], le 2e est L[1] …
- On peut ajouter ou remplacer un élément dans un dictionnaire:
  - `liste[index] = valeur` # pour modifier la valeur
  - `liste.append(valeur)` # pour ajouter valeur à la fin de la liste
- On peut supprimer un valeur d'une liste
  - Avec l'index : del ou pop
    - La fonction `del liste[i]` va supprimer l'item à l'index i spécifié (pas besoin de connaître l'item).
    - La méthode `liste.pop(i)` va supprimer l'item à l'index i spécifié et renvoyer sa valeur.
  - Avec sa valeur : la méthode `.remove()`.
    - `liste.remove(x)` : supprime de la liste le premier élément dont la valeur est égale à x
- On peut parcourir une liste grâce aux méthodes:
  - par les index : `for index in range(len(liste))`
  - directement : `for élément in liste`
- Trouver l'index d'une valeur v1
  - La méthode : `liste.index(v1)`
- Trier une liste
  - méthode `.sort()` : `liste.sort()`

- `liste.count(x)` : renvoie le nombre d'apparitions de x dans la liste
- `liste[i:j]` : renvoie une sous liste composée des éléments d'indice i à (j-1)
  - Par exemple `liste[2:4]` renvoie la liste composée des élément `liste[2]` et `liste[3]`

```
[ ]: # Pour accéder à toutes les fonctions et méthodes liées à la classe␣
     ↪dictionnaire (la notion de classe sera vue plus tard)
     # il suffit de taper la commande dir(nom_liste) :
     a=[]
     dir(a)
```

3 TD - Les listes sous Python

3.1 Exercice 2

1. Créer par compréhension une liste L1 de 6 entiers aléatoires compris entre 1 et 100. Aide : utiliser `randint(a,b)` du module `random`
2. Créer par compréhension une liste L2 de 7 entiers aléatoires compris entre 1 et 100. Aide : utiliser `randint(a,b)` du module `random`
3. Créer les liste `L3=[100,25,15,40]` et `L4=[20,9,4,7]` et `L5=[40,29,14,7,15]` puis calculer la moyenne et la médiane des valeurs de chacune des listes (à la main).
4. Ecrire une fonction moyenne qui calcule la moyenne d'une liste de flottants et la tester sur vos deux listes L3 et L4 puis sur L1 et L2. Vous n'avez pas le droit d'utiliser de module externe comme statistics.
5. Ecrire une fonction médiane qui calcule la médiane d'une liste de flottants et la tester sur vos deux listes L3 et L4 puis sur L1 et L2. Vous n'avez pas le droit d'utiliser de module externe comme statistics.

```
[ ]: ## Exercice 2
     import random
     L3=[100,25,15,40] # L3 de moyenne = ... et de médiane = ...
     L4=[20,9,4,7,1]  # L4 de moyenne = ... et de médiane = ...
     L5=[40,29,14,7,15] # L4 de moyenne = ... et de médiane = ...
```

3.2 Exercice 3

Ecrire une fonction qui renvoie la liste des caractères ASCII **affichables**.

Aide :

```
>>> ord('a') # renvoie la valeur entiere de la table ASCII correspondante au
caractère 'a' 97
>>> chr(97) # renvoie le caractère carrespondant à la valeur 97 dans la table
ASCII
'a'
```

[ ]:
```
## Exercice 3
def f():
    '''IN  : pas d'entrée
       OUT : renvoie la liste des caractère ASCII'''
```

3.3 Exercice 4 : question 1

`uneListeAlea=[[liste1],[liste2],[liste3],[liste4],[liste5]]` et `liste1=[e1,e2,e3]`

1. Ecrire une liste L de 5 listes aléatoires de longueur 3 contenant des entiers compris entre 1 et 100, en utilisant le module `random` et une boucle.

Par exemple : `uneListeAlea = [ [1,2,3] , [4,5,6] , [7,8,9] , [10,20,30] , [12,22,32] ]`

[ ]:
```
## Exercice 4 : question 1
import random
```

Exercice 4 : question 2 2. Ecrire une fonction `f(L,n,p)` qui renvoie le n-ieme (n entre 1 et 3) élément de l'item p (p entre 1 et 5) de la liste L

Par exemple : `f(uneListeAlea,2,4)` doit renvoyer `20` et `f(uneListeAlea,3,5)` doit renvoyer `32` avec `uneListeAlea = [ [1,2,3] , [4,5,6] , [7,8,9] , [10,20,30] , [12,22,32] ]`

[ ]:
```
## Exercice 4 : question 2
#2. Ecrire une fonction f(L,n,p) qui renvoie le n-ieme (n de 1 à 3) élément de␣
↪la l'item p (p de 1 à 5) de la liste L
```

Exercice 4 : question 3

3. Ecrire une fonction `g(L,n)` qui renvoie le n-ieme élément (n de 1 à 3) de chacune des 5 listes sous dorme de liste.

Par exemple : `g(uneListeAlea,2)` doit renvoyer `[2,5,8,20,22]` avec `uneListeAlea = [ [1,2,3] , [4,5,6] , [7,8,9] , [10,20,30] , [12,22,32] ]`

[ ]:
```
# Exercice 4 : question 3
def g(L,n):
    Liste_n=[]
    for i in L:
        Liste_n.append(i[n-1])
    return Liste_n
uneListeAlea = [  1,2,3] , [4,5,6] , [7,8,9] , [10,20,30] , [12,22,32]
print(g(uneListeAlea,2))
```