IngeSUP - Cours 05 - Listes et Chaines II

Sommaire

Objectifs

- Utilisation avancée des listes ;
- Utilisation avancée des chaines de caractères ;
- Transformations entre chaines de caractères et listes.

Prérequis

Avant de réaliser ce notebook de cours, vous devez avoir préalablement visualisé le mimo suivant :

- Module 7 : Les tableaux et séquences en Python (https://courses.ionisx.com/courses/ref/m123/x/courseware/54c7a679a9354a2996ece5f4f11b02b9/4ca7a398
- Première partie du cours sur les listes et chaines : Cours 04 - Listes et Chaines I (./Cours%2004%20-%20Listes%20et%20Chaines%20I.ipynb)

Utilisation avancée des listes en Python

Rappel sur les listes

- Une liste est une séquence de données.
- Les listes sont des tableaux qui peuvent contenir tout type de variable.
- Elles sont de type `list` .
- Les listes sont modifiables.
- Les éléments d'une liste doivent être entourés par crochets `[…]` et séparés par des virgules `,` .

Listes en compréhension

0n va voir ce qu'on appelle **les compréhensions de listes**, que vous retrouverez également sous le nom de *liste en compréhension_, ou même _list comprehension* en anglais.

Les compréhensions de listes nous permettent **d'itérer sur une liste et de filtrer les éléments**, grâce à des structures conditionnelles, **tout ça en une seule ligne**.

Actuellement, si on veut filtrer les éléments d'une liste, on doit écrire pas mal de lignes de code.

Entrée [ ]:

```
liste = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
nombres_positifs = []
for i in liste:
    if i > 0:
        nombres_positifs.append(i)
```

Dans cet, on veut récupérer uniquement les nombres positifs dans la liste.

👉 On commence donc par créer une liste vide qui va contenir les nombres positifs.

👉 On boucle ensuite sur chaque élément de notre liste d'origine.

👉 On vérifie si l'élément sur lequel on itère est plus grand que 0.

👉 et si c'est le cas, on l'ajoute à la liste que l'on a définie au début du script.

Tout ça, on va pouvoir l'écrire beaucoup plus simplement sur une seule ligne, grâce aux **compréhensions de listes**.

Voici comment elles s'écrivent `[<expression> for <element> in <iterable>])` .

Supposons qu'on veut créer une liste composée des carrés des nombres entiers allant de 1 à 9.

On peut créer cette liste de la manière suivante :

Entrée [ ]:

```
liste = [] # On crée une liste vide.

for element in range(1, 10): # On parcours les nombres de 1 à 9. Donc <range(1,10)> est no
    element = element**2 # Pour chaque nombre on calcule son carré. Donc <element**2> est
    liste.append(element) # On ajoute l'élément à la fin de la liste

print(liste)
```

On peut aussi compresser ces instructions de la manière suivante :

Entrée [ ]:

```
liste = [element**2 for element in range(1,10)] # Crée une liste contenant les carrés des

print(liste)
```

Pour reprendre le tout premier exemple, on peut ajouter à la fin de notre compréhension de liste, une structure conditionnelle pour filtrer les éléments.

Entrée [ ]:

```
liste = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
nombres_positifs = [i for i in liste if i > 0]
```

On peut même modifier les éléments de la liste finale :

Entrée [ ]:

```
liste = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
nombres_positifs_fois_deux = [i * 2 for i in liste if i > 0]
```

Supprimer une variable liste

Pour supprimer une variable liste du programme, il suffit d'utiliser le mot-clé **del** en mentionnant le nom de la variable liste à supprimer.

Entrée [ ]:

```
del(liste)
print(ma_liste)
```

Récupérer une sous-liste d'une liste

Python offre un mécanisme pour obtenir le découpage d'une sous-liste à partir d'une liste donnée. C'est ce qu'on appelle le *slice* de liste.

Entrée [ ]:

```
liste1 = [0, 1, 2, 3, 4, 5, 6]
i, j, k = 1, 3, 2

liste2 = liste1[:]      # Revoie tous les éléments de la liste.
print(liste2)           # Affiche : [0, 1, 2, 3, 4, 5, 6]

liste2 = liste1[i:j]    # Revoie les éléments entre l'indice « i » inclus et l'indice « j »
                        # L'indice de l'élément récupéré doit vérifier la condition i <= in
print(liste2)           # Affiche : [1, 2]

liste2 = liste1[i:]     # Revoie les éléments à partir de l'indice « i » inclus jusqu'à la
print(liste2)           # Affiche : [1, 2, 3, 4, 5, 6]

liste2 = liste1[:j]     # Revoie les éléments à partir du début jusqu'à l'indice « j » excl
print(liste2)           # Affiche : [0, 1, 2]
```

Entrée [ ]:

```
liste2 = liste1[::k]    # Revoie tous les éléments de la liste en avançant de « k » pas.
                        # Si k est positif alors on récupére les éléments dont l'indice cor
                        # Si k est négatif alors on récupére les éléments dont l'indice cor
print(liste2)           # Affiche : [0, 2, 4, 6]

liste2 = liste1[i:j:k]  # Revoie les éléments entre l'indice « i » inclus et l'indice « j »
                        # Si k est positif alors on récupére les éléments dont l'indice cor
                        # Si k est négatif alors on récupére les éléments dont l'indice cor
print(liste2)           # Affiche : [1]

liste2 = liste1[i::k]   # Revoie les éléments à partir de l'indice « i » inclus jusqu'à la
                        # Si k est positif alors on récupére les éléments dont l'indice cor
                        # Si k est négatif alors on récupére les éléments dont l'indice cor
print(liste2)           # Affiche : [1, 3, 5]

liste2 = liste1[:j:k]   # Revoie les éléments à partir du début jusqu'à l'indice « j » excl
                        # Si k est positif alors on récupére les éléments dont l'indice cor
                        # Si k est négatif alors on récupére les éléments dont l'indice cor
print(liste2)           # Affiche : [0, 2]
```

Entrée [ ]:

```
liste2 = liste1[::-1]   # Revoie tous les éléments de la liste en partant de la fin.
print(liste2)           # Affiche : [6, 5, 4, 3, 2, 1, 0]
```

Modifier une sous-liste d'une liste

Pour modifier une sous liste à l'intérieur d'une liste, il suffit de sélectionner la sous liste concernée et la remplacer par la nouvelle liste.

Entrée [ ]:

```
liste1 = [0, 1, 2, 3, 4, 5, 6]
liste1[1:4] = [11, 22, 33] # Remplace les éléments entre l'indice « 1 » inclus à l'indice «
print(liste1)              # Affiche : [0, 11, 22, 33, 4, 5, 6]
```

Insérer plusieurs éléments dans une liste à un indice donnée

Supposant qu'on veut insérer plusieurs éléments à la fois dans une liste, à un rang donnée `i` .

Dans ce cas il suffit de sélectionner la sous-liste `liste[i:i]` et de la remplacer par la liste d'éléments qu'on veut insérer dans la liste à la position `i` :

Entrée [ ]:

```
liste = [0, 1, 2, 3, 4, 5, 6]
liste[1:1] = [11, 22, 33] # Insère les éléments de la liste [11, 22, 33] à l'indice 1.
print(liste)              # Affiche : [0, 11, 22, 33, 1, 2, 3, 4, 5, 6]
```

Supprimer une sous-liste d'une liste

Pour supprimer une sous-liste d'une liste, il suffit de sélectionner la sous-liste à supprimer et la donner à l'opérateur `del` :

Entrée [ ]:

```
liste = [0, 1, 2, 3, 4, 5, 6]
del liste[1:4]    # Supprime les éléments entre l'indice « 1 » inclus à l'indice « 4 » excl
print(liste)      # Affiche : [0, 4, 5, 6]
```

Copier une liste

L’instruction suivante :

```
liste = [1, 2, 3, 4, 5]
listeCopie = liste
```

Implique que les deux listes « listeCopie » et « liste » pointent sur la même adresse mémoire, ce qui signifie que la modification de l'une entrainera la modification de l'autre.

Entrée [ ]:

```
liste = [1, 2, 3, 4, 5]

listeCopie = liste

listeCopie[1] = 99 # On modifie l'élément au rang 1 dans la liste "listeCopie"

# On trouve que les 2 listes sont modifiées :
print(liste)
print(listeCopie)
```

Afin d'éviter cela, on utilise la copie disjointe.

Entrée [ ]:

```
liste = [1, 2, 3, 4, 5]

listeCopie = liste[:]

listeCopie[1] = 99 # On modifie l'élément au rang 1 dans la liste "listeCopie"

# On trouve que les 2 listes ne sont plus modifiées :
print(liste)
print(listeCopie)
```

Utilisation avancée des chaines de caractères en Python

Rappel sur les chaines de caractères

- Les chaines de caractères sont des variables capables de contenir une suite de caractères. Ils sont de type <class 'str'>.
- Les chaines de caractères ne sont pas modifiables en Python.
- Les caractères doivent être entourés par une simple quote ' ou double qotes " .

> ⚠ **Attention**
>
> Attention **les chaines de caractères** ne sont pas modifiables en python. On ne peut donc pas modifier ou supprimer des caractères à l'interieur d'une chaine de caractère !

Les caractères spéciaux qu'on peut utiliser dans les chaines

- `\n` : représente le retour à la ligne.
- `\t` : représente la tabulation.
- `\r` : représente le retour chariot, parfois utilisé pour le retour en début de ligne.
- `\caractère spécial` : si on met `\` devant un caractère spécial comme le caractère spéciale `"` , qui permet de délimietr une chaine de caractère, alors ce caractère sera inscrit dans le texte et non considéré comme caractère spécial.

Entrée [ ]:

```
chaine = "\"\\"
print(chaine) # Affiche : "\
```

Formater une chaine pour intégrer des variables dedans

Pour insérer les valeurs des variables à l'interieur d'une chaine sans avoir à la décomposer, on utilise la méthode `format` .

Entrée [ ]:

```
chaine = "Bonjour {}, vous avez obtenu un excellent score {}."
nom = "Oliver"
score = 90
chaineFormatee = chaine.format(nom, score) # La première {} sera remplacée par "A", et la s
print(chaineFormatee)                      # Affiche: « Bonjour Oliver, vous avez obtenu u
```

On peut placer les éléments à formater de différentes manières

Entrée [ ]:

```
txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)
txt2 = "My name is {0}, I'm {1}".format("Paul",37)
txt3 = "My name is {}, I'm {}".format("Phil",38)

print(txt1)
print(txt2)
print(txt3)
```

Récupérer l'indice d'une suite de caractères d'une chaine de caractères

Entrée [ ]:

```
s = "Bonjour"
chaineATrouver = "o"
i = s.find(chaineATrouver)  # Renvoie l'indice de la première occurrence de la chaîne cher
                            # Renvoie « -1 » s'il trouve pas.
print(i)                    # Affiche : 1

i = s.index(chaineATrouver) # Renvoie l'indice de la première occurrence de la chaîne cher
                            # Lève une exception de type « ValueError » s'il trouve pas.
print(i)                    # Affiche : 1
```

Récupérer une suite de caractères d'une chaine de caractères

> **Note**
>
> On peut utiliser le **slice sur les chaînes de la même manière que sur les listes**

Entrée [ ]:

```
chaine1 = "Bonjour"
i, j, k = 1, 3, 2

chaine2 = chaine1[:]    # Revoie tous les caractères de la liste.
print(chaine2)          # Affiche : Bonjour

chaine2 = chaine1[i:j]  # Revoie les caractères entre l'indice « i » inclus et l'indice « j
                        # L'indice du caractère récupéré doit vérifier la condition i <= in
print(chaine2)          # Affiche : on

chaine2 = chaine1[i:]   # Revoie les caractères à partir de l'indice « i » inclus jusqu'à l
print(chaine2)          # Affiche : onjour

chaine2 = chaine1[:j]   # Revoie les caractères à partir du début jusqu'à l'indice « j » ex
print(chaine2)          # Affiche : Bon

#etc... Voir exemple sur les listes
```

Remplacer des caractères d'une chaine de caractères

Pour remplacer des sous-chaines dans une chaine de caractère, on utilise la méthode `replace(chaineARemplacer, chaineDeRemplacement)`

Cette méthode renvoie une chaîne en remplaçant toutes les occurrences de la chaîne passée en 1er argument `chaineARemplacer` , par la chaîne passée en 2ème argument `chaineDeRemplacement` .

Entrée [ ]:

```
chaine1 = "Bonjour"
chaineARemplacer = "o"
chaineDeRemplacement = "ESME"
chaine2 = chaine1.replace(chaineARemplacer, chaineDeRemplacement)
print(chaine2)  # Affiche :  BESMEnjESMEur
```

Compter le nombre d'occurrences d'une suite de caractères dans une chaine

Entrée [ ]:

```
chaine = "Bonjour"
s = "o"

n = chaine.count(s) # Renvoie le nombre d'occurrences de la chaîne cherchée "s" dans la cha
print(n)            # Affiche : 2
```

Transformer les caractères d'une chaine de caractères

Entrée [ ]:

```
chaine = " Bonjour tout le monde."

s = chaine.lower()      # Renvoie la chaîne en minuscule.
print(s)

s = chaine.upper()      # Renvoie la chaîne en majuscule.
print(s)

s = chaine.capitalize() # Renvoie la chaine avec une majuscule pour la première lettre, pou
print(s)

s = chaine.title()      # Renvoie la chaine avec une majuscule pour la première lettre de c
print(s)

s = chaine.strip()      # Renvoie une chaine après avoir enlevé les espaces au début et à l
print(s)
```

> ⚠ **Attention**
>
> L'ensemble des méthodes qui permettent de transformer les caractères d ’une chaine de caractères, **travaillent sur une copie de la chaine** et ne modifie en aucun cas la chaine.

Transformations entre chaines de caractères et listes

Transformer une liste en une chaine de caractères

Pour pouvoir transformer une liste en une chaine de caractères, les éléments de la liste doivent être des chaines de caractères. Si ce n'est pas le cas, il faut parcourir la liste et convertir tous ses éléments en chaines de caractères avec « str(élément) ».

Entrée [ ]:

```
liste = ['A', str(2), str(True), str(8.2)]
chaine = "".join(liste) # Construit une  chaine à partir des éléments de la liste « liste »
print(chaine)           # Affiche : A2True8.2
```

Si on veut que les éléments de la liste soient séparés par un séparateur dans la chaine, alors on le précise dans la chaine sur laquelle on applique `join(…)` :

Entrée [ ]:

```
liste = ['A', str(2), str(True), str(8.2)]

chaine = " ".join(liste) # Construit une chaine à partir des éléments de la liste en les se
print(chaine)            # Affiche : A 2 True 8.2.

chaine = "|".join(liste) # Construit une chaine à partir des éléments de la liste en les se
print(chaine)            # Affiche : A|2|True|8.2.
```

Transformer une chaine de caractères en une liste

- **Transformer une chaine de caractères en une liste de caractères :**

Entrée [ ]:

```
chaine = "Hello"
liste = list(chaine)   # Renvoie la liste des caractères de la chaine « chaine ». Ici: « li
print(liste)           # Affiche : ['H', 'e', 'l', 'l', 'o']
```

- **Transformer une chaine de caractères en une liste de mots :**

Entrée [ ]:

```
chaine = "Bonjour tout le monde. Vous êtes les meilleurs"
liste = chaine.split(".")  # Renvoie une liste de chaînes de caractères correspondant aux c
                           # de la chaîne « chaine » découpées par le séparateur « "." ».
print(liste)               # Affiche : ['Bonjour tout le monde', ' Vous êtes les meilleurs

liste = chaine.split()     # Par défaut, c'est l'espace blanc « ' ', '\t', '\n', ou '\r' »
print(liste)               # Affiche : ['Bonjour', 'tout', 'le', 'monde.', 'Vous', 'êtes',
```

Exercices de TD

Vous pouvez maintenant vous exercer à partir du notebook [Cours 05 - Listes et Chaines II (../TD/TD%2005%20-%20Listes%20et%20cha%C3%AEnes%202.ipynb#IngeSUP---TD-05---Listes-et-cha%C3%AEnes-2)](../TD/TD%2005%20-%20Listes%20et%20cha%C3%AEnes%202.ipynb#IngeSUP---TD-05---Listes-et-cha%C3%AEnes-2).