

Chapitre 3 : Chaînes de caractères et parcours de fichiers

I Chaîne de caractères

Nous avons déjà rencontré le type `string` (chaîne de caractères). Nous allons ici préciser quelques opération que l'on peut faire sur ce type d'objet. C'est une structure de données homogènes : chaque élément est une lettre, et non modifiable : on ne peut pas, par exemple, changer une lettre dans le mot.

I. 1 Opérations sur les chaînes de caractères

Instruction Python	Opération effectuée	Exemples
<code>+</code>		"truc" + "muche" ↓
<code>*</code>		3*"bla" ↓
<code>==</code>		"Python"=="python" ↓
<code>len</code>		<code>len("bcpst")</code> ↓
<code>in</code>		"mi" in "minuscule" ↓

I. 2 Accès aux éléments d'une chaîne de caractère

Les lettres d'une chaîne de caractères `s` sont indexées de 0 à `len(s)-1`. Par exemple, dans la chaîne `s="bonjour"`, les éléments sont indexés de 0 à 6 :

caractère	b	o	n	j	o	u	r
indice	0	1	2	3	4	5	6

Si l'on veut accéder à certains éléments, on utilise les syntaxes suivantes :

Instruction Python	Opération effectuée	Exemples
<code>s[i]</code>		<code>s[2]</code> ↓
<code>s[i:j]</code>		<code>s[2:5]</code> ↓
<code>s[i:]</code>		<code>s[3:]</code> ↓
<code>s[:j]</code>		<code>s[:3]</code> ↓

Si l'on écrit simplement `s[:]` on récupère simplement la chaîne de caractère.

Mise en garde !



Les chaînes de caractères ne sont pas des types mutables. Si l'on veut changer une lettre on est obligé de redéfinir notre chaîne de caractères. Par exemple, si je me trompe en définissant : `s="ronjour"`, pour changer la première lettre sans retaper toute la chaîne de caractères, je dois redéfinir `s` de la façon suivante : `s =`.

I. 3 Boucles sur les chaînes de caractères

De même que sur les listes, on peut faire des boucles `for` sur les chaînes de caractères.

Il y a de nouveau deux possibilités pour faire des boucles `for` sur une chaîne de caractères :

PARCOURS PAR ELEMENTS :

```
for e in ch :  
    instructions_dépendant_de_e
```

PARCOURS PAR INDICES :

```
for i in range(len(ch)) :  
    instructions_dépendant_de_i
```

Les deux types de parcours sont utiles voici quelques exemples pour s'en convaincre.

Les deux exemples suivants affichent 'b', puis 'o', puis 'n'...

PARCOURS PAR ELEMENTS :

```
1 ch = 'bonjour'  
2 for c in ch :  
3     print(c)
```

PARCOURS PAR INDICES :

```
1 ch = 'bonjour'  
2 for i in range(len(ch)) :  
3     print(ch[i])
```

I. 4 Exercices

Exercice 1. Un élève un peu étourdi a confondu les "b" et les "p" dans toute la phrase `s` qu'il vient d'écrire. Écrire une fonction `cor(ch)` qui prend en argument une chaîne de caractère écrit par cet élève et qui renvoie la chaîne corrigée.

Réponse :

Exercice 2. Écrire une fonction `voycons(ch)` qui prend en argument une chaîne de caractères ne contenant ni espace ni caractères spéciaux ni accent et qui renvoie le nombre de voyelles et de consonnes contenues dans cette chaîne.

Réponse :

Exercice 3. Écrire une fonction qui cherche si un mot est présent dans une chaîne de caractères sans utiliser `in`. Combien d'opérations au maximum a-t-on effectuées ?

Réponse :

Exercice 4. Chiffre de César.

Le chiffre de César est une méthode de chiffrement utilisée par Jules César pour ses correspondances secrètes. Son principe est très simple : on remplace chaque lettre de l'alphabet par une lettre décalée d'un certain cran dans l'alphabet. Par exemple, si l'on choisit de décaler de 3 crans l'alphabet, le mot "python" devient "sbwkrq". Pour déchiffrer une phrase, il suffit alors de décaler les lettres dans l'autre sens.

Le principe mathématique est simple : si l'on numérote les lettres entre 0 et 25, et que l'on choisit de décaler l'alphabet de n lettres vers la droite, chaque lettre numérotée k est remplacée par $k + n$ modulo 26.

1. Écrire une fonction qui permet de chiffrer un message en décalant les lettres de n crans vers la droite.
2. Écrire une fonction qui permet de déchiffrer un message sans connaître le décalage à l'avance.
3. Déchiffrer le message : "kxkxtapqrehi" :

Remarquons tout de même que ce code est très peu efficace : il existe très peu de possibilités pour coder un message (26), et en plus le chiffrement peut être trouvé très facilement grâce à une analyse de fréquence des lettres contenues dans le message (certaines lettres apparaissent beaucoup plus souvent que les autres, en français le "e" par exemple).

Exercice 5. Suite audioactive de Conway

On considère la suite récurrente $(C_n)_{n \in \mathbb{N}}$ dont le premier terme est $C_0 = 1$ et chaque terme s'obtient en décrivant le terme précédent :

- Dans le terme C_0 il y a un 1 donc $C_1 = 11$;
- Dans le terme C_1 il y a deux 1 donc $C_2 = 21$;
- Dans le terme C_2 il y a un 2 puis un 1 donc $C_3 = 1211$,

et ainsi de suite. Cette suite s'appelle la suite audioactive (en raison de la façon dont elle est construite) elle a été étudiée par John Conway en 1986. Voici les premiers termes de la suite :

n	0	1	2	3	4	5	6	7
C_n	1	11	21	1211	111221	312211	13112221	1113213211

Écrire une fonction `Conway` qui à partir du terme C_n sous forme d'une chaîne de caractères renvoie le terme C_{n+1} sous la forme d'une chaîne de caractères.

On pourra utiliser la fonction `str()` qui convertie un nombre en chaîne de caractères.

II Lecture et écriture dans un fichier .txt

Jusqu'à présent, nous avons toujours stocké nos données et nos résultats dans des variables internes à **python**. En pratique, lorsque l'on s'intéresse à de grandes quantités de données (par exemple, utilisation d'une base de données contenant des informations médicales sur 10000 patients), ou lorsque l'on veut traiter les résultats de grosses simulations (par exemple, stocker les forces de pressions appliquées sur la surface d'un avion lors d'un vol), il devient indispensable de séparer les données et résultats du code **python**.

Pour cela, on conserve les informations dans un fichier à part, et il est nécessaire de pouvoir accéder à ce fichier, et de pouvoir le modifier. Nous nous intéresserons pour le moment à des données stockées dans un fichier .txt.

Important !

Pour créer, ouvrir, modifier un fichier, il est important de sauvegarder son script dans le dossier où se trouve ce fichier.

Pour indiquer à Python que c'est dans ce dossier qu'il faut aller regarder, vous devez effectuer la manipulation suivante à chaque fois que vous lancez Pyzo : faites un clique droit dans la console, puis sélectionnez (suivant l'installation que vous avez)

"change current directory to editor file path".

ou "Définir le répertoire courant en accord avec le fichier ouvert dans l'éditeur"

II. 1 Ouvrir un fichier

Pour ouvrir le fichier texte "monfichier.txt" (qui doit être situé dans le même répertoire que votre script **python**, sinon il convient d'indiquer le chemin d'accès du fichier), il existe 3 possibilités :

- **f=open("monfichier.txt", "r")** : permet d'accéder en lecture ("r" pour read) seulement aux données du fichier : on peut récupérer les informations contenues dans le fichier, mais il n'est pas possible de modifier le fichier d'origine. Si le fichier n'existe pas, **python** le crée.
- **f=open("monfichier.txt", "a")** : permet d'ajouter ("a" pour add) des données au fichier (sans modifier celles qui sont déjà présentes).
- **f=open("monfichier.txt", "w")** : permet d'écrire ("w" pour write) sur les données existantes du fichier. Ainsi, les informations contenues précédemment seront effacées : évitez d'utiliser ce mode dès que vous tenez un peu à votre fichier d'origine !

II. 2 Récupérer les données d'un fichier

Il faut pour cela ouvrir le fichier en mode lecture (**read**).

Ensuite, on peut copier le contenu du fichier de deux façons différentes :

- **s=f.read()** : permet de récupérer toutes les données du fichier sous la forme d'une chaîne de caractères.
- **L=f.readlines()** : permet de récupérer toutes les données du fichier sous la forme d'une liste dont chaque élément est une chaîne de caractères correspondant à une ligne du fichier.

II. 3 Écrire des données dans un fichier

Il faut pour cela ouvrir le fichier en mode ajout "a" ou écriture "w".

Il suffit ensuite d'utiliser la commande **f.write("texte")**, où **texte** est la chaîne de caractères que l'on souhaite écrire.

On peut passer à la ligne grâce au caractère \n et ajouter une tabulation grâce à \t.

Remarque. Il faut toujours penser à refermer le fichier après son utilisation à l'aide de la commande **f.close()**. En effet, si l'on veut modifier le fichier, l'écriture ne se fera que lors de la fermeture du fichier.

II. 4 Pour se familiariser

On veut tester les commandes précédentes. Pour cela :

1. Créer un fichier **test.txt** en mode écriture.
2. Y écrire une phrase de votre choix, et fermer le fichier.
3. Trouver le fichier dans le dossier, et l'ouvrir avec un éditeur de texte quelconque pour vérifier que le texte a bien été écrit.
4. Rouvrir votre fichier en mode ajout, y ajouter une phrase après être passé à la ligne.
5. Fermer le fichier, puis le rouvrir en mode lecture. Récupérer vos données avec les deux méthodes existantes et afficher la chaîne de caractères / la liste obtenue.

Nous allons créer deux fonctions qui seront utile tout au long du TP.

6. Écrire une fonction **TexteToCara(nom)** qui prend en argument le nom d'un fichier texte placé dans le répertoire pour qu'elle ouvre le fichier en lecture et qu'elle renvoie la chaîne de caractère correspondant à ce texte.

Onoubliera pas de fermer le fichier ouvert avant la fin de la fonction.

Réponse :

7. Écrire une fonction **CaraToTexte(ch,nom)** qui prend en argument le nom d'une chaîne de caractères et le nom d'un fichier pour qu'elle ouvre le fichier en écriture et qu'elle écrive le texte correspondant à la chaîne de caractères.

Onoubliera pas de fermer le fichier ouvert avant la fin de la fonction.

Réponse :

8. Utiliser les deux fonctions précédentes pour remplacer tous les "e" par des "a" dans le fichier **test.txt**.