TP 4 : Listes (II), chaînes de caractères

▶ Manipulation de chaînes de caractères

Les chaînes de caractères (de type `string`) partagent de nombreuses propriétés avec les listes, et on peut voir une chaîne comme la liste de ses caractères.
Ainsi, si `c` est une chaîne de caractères, alors :

- `c[0]` est le premier caractère de `c`, le suivant est `c[1]`, etc. Ces caractères sont encore de type `string`.
- `len(c)` est le nombre de caractères de la chaîne, le dernier étant donc `c[len(c)-1]`.
- on peut accéder directement au dernier caractère d'une chaîne à l'aide de `c[-1]` (puis `c[-2]`, etc).
- le *slicing* fonctionne encore pour accéder à une partie seulement d'une liste. Par exemple, si `c = 'Maitre corbeau, sur un arbre perche'`, alors `c[7 :14]` est la chaîne `'corbeau'`.
  De même, je vous laisse essayer (et comprendre) le résultat des commandes suivantes :

```
c = 'élu par cette crapule'
print(c[ : :-1])
```

- il est possible d'itérer sur les caractères d'une chaîne `c` avec `for l in c :`
- si `c1` et `c2` sont deux chaînes de caractères, alors `c1+c2` est la chaîne obtenue par concaténation de `c1` et `c2`. Et si `n` est un entier, `c1 * n` est la concaténation de n copies de `c1`.

En revanche, la principale différence entre les objets de type `list` et `string` réside dans le fait que ces derniers ne sont pas *mutables*, ce qui signifie *grosso modo* qu'on ne peut pas en changer partiellement la valeur.
S'il était possible de changer la valeur du second élément d'une liste `L` avec `L[1] = ...`, il est impossible de changer de la même manière un seul caractère d'une chaîne `c`.
Et donc pour modifier partiellement une chaîne de caractères, le plus simple est de créer une autre chaîne de caractère.
Essayer par exemple la fonction suivante, et essayez de comprendre son fonctionnement :

```
def sans_espaces(c) :
        r = ''
        for l in c :
                if l !=' ' :
                        r = r+l
        return(r)
```

Enfin, notons que l'immuabilité des chaînes de caractère n'empêche pas de les redéfinir :
`message = 'Bonjour' ; message = 'Au revoir'`

Exercice 1 Écrire une fonction `nb_occurences` qui prend comme paramètres une chaîne `c` et une lettre `l` (de type `string` également) et renvoie le nombre de fois où `l` apparaît dans `c`. PD

Exercice 2 Écrire une fonction qui prend comme paramètre une chaîne de caractère `c` et renvoie une chaîne dans laquelle on a échangé les a et les o. L'essayer sur la chaîne "C'est l'heure de l'opéra !" PD

Exercice 3 La fonction `int` permet de convertir une chaîne formée de chiffres en un entier (de type `int`). Par exemple `int('564')` renvoie l'entier 564. PD
La fonction `str` réalise l'opération inverse en convertissant un entier (ou à peu près n'importe quoi d'autre) en chaîne de caractères.

1. Écrire une fonction qui prend comme paramètres deux entiers n et k et renvoie le $k^{\text{ème}}$ chiffre de l'écriture décimale de n. ⚠ *En français, le chiffre que l'on appelle premier chiffre est celui des unités, le second celui des dizaines, etc.*

2. Écrire une fonction `somme_des_chiffres` qui fait ce que son nom indique.
3. Utiliser la fonction précédente pour écrire une fonction testant si un nombre est divisible par 3 sans utiliser de division (ni le reste d'une division).

▶ Encore des manipulations de listes et de chaînes

Exercice 4 Le crible d'Ératosthène AD

On rappelle qu'un nombre premier est un nombre dont les seuls diviseurs positifs sont 1 et lui-même. Par exemple, 2 et 5 sont premiers, alors que $12 = 4 \times 3$ ne l'est pas.
La méthode proposée par Ératosthène pour obtenir tous les nombres premiers inférieurs à n est la suivante :

1. Écrire la liste des entiers de 1 à n.
2. Barrer tous les multiples de 2, qui ne peuvent donc pas être premiers.
3. Chercher le plus petit nombre non barré (en l'occurrence, ici c'est 3). Barrer alors tous ses multiples (qui ne sont pas premiers).
4. On continue ainsi jusqu'à arriver au dernier entier non barré inférieur ou égal à $\sqrt{n}$. Les entiers qui n'ont jamais été barrés sont les nombres premiers.

Si besoin, vous trouverez sur la page Wikipédia «Crible d'Ératosthène» un exemple animé.

Implémenter cet algorithme pour créer une liste de tous les nombres premiers inférieurs à 10000. Comme on ne peut pas «barrer» les entiers, il est conseillé d'utiliser une liste de booléens, qui valent initialement tous `True`, et pour «barrer» un entier, on remplacera cette valeur par `False`.

Exercice 5 Écrire une fonction qui prend comme paramètre une liste `L` de nombres entiers, ordonnée dans l'ordre croissant, et qui teste si ces nombres sont en progression arithmétique (et renvoie un booléen). PD
(⋆⋆) Même question, mais cette fois :

- on ne suppose plus la liste triée
- on souhaite savoir si la liste contient `len(L)` termes distincts et consécutifs d'une suite arithmétique. Autrement dit, on veut savoir si l'ensemble des éléments de la liste soit de la forme $\{a+kr, 0 \leqslant k \leqslant n-1\}$ où n =`len(L)`.
  En particulier, il faut que les n éléments de la liste soient 2 à 2 distincts et qu'il n'y ait pas de «trous». Par exemple `[5,15,10,25,30]` ne convient pas : ses termes sont ceux d'une suite arithmétique de raison 5, mais il manque 20.
- on souhaite que le nombre d'opérations élémentaires (somme/produit/comparaison de deux éléments) soit de l'ordre de grandeur de n =`len(L)`.
  Plus précisément, on veut que ce nombre d'opérations soit inférieur à une constante fois n. C'est-à-dire que $2n$ ou même $50n$ sont des nombres d'opérations acceptables, mais n^2 ne l'est pas, de même que $n\ln(n)$.
  Nous dirons prochainement que la complexité de notre algorithme est linéaire.
  Remarque : commencer par utiliser une fonction qui trie la liste n'est pas acceptable, car cela demande trop d'opérations, en moyenne de l'ordre de $n\ln(n)$ (ce que nous expliquerons plus tard).

Exercice 6 $n^{\text{ème}}$ nombre premier D

Proposer une fonction qui prend comme paramètre un entier n et renvoie le $n^{\text{ème}}$ nombre premier (le 1^{er} étant 2, le $2^{\text{ème}}$ étant 3, etc).
Vérification : le $10\,000^{\text{ème}}$ nombre premier est 104 729.

Exercice 7 Le javanais AD

Le javanais (ou langue de feu) est une langue inventée, qui consiste à ajouter «av» devant chaque voyelle (ou plutôt devant chaque groupe de voyelles d'un mot). Par exemple, «python» devient en javanais «pavythavon».
Plus exactement, les règles pour traduire un mot en javanais sont les suivantes :

- si le mot commence par une voyelle, alors «av» est ajouté devant cette voyelle
- «av» est ajouté après chaque groupe de consonnes, autrement dit, avant chaque groupe de voyelles
- on n'ajoute rien après une consonne finale.

Ainsi , «physique» devient «phavysaviqavue», «maths» devient «mavaths» et «avion» devient «avavavion».
Écrire une fonction qui traduit en javanais un mot (en minuscule et sans accents) passé en paramètre.

Exercice 8 *Dans cet exercice, on ne considère que des chaînes qui ne contiennent pas de symboles de ponctuation.* AD
Écrire une fonction `retourne(c)` qui, si `c` est une chaîne contenant une phrase, doit retourner la phrase formée des mêmes mots, mais en sens inverse.
Par exemple `retourne("J'en suis tout retourné")` devra renvoyer `"retourné tout suis J'en"`.

Exercice 9 Lapins de Fibonacci mortels D
Dans un précédent TP, nous avions rencontré la suite de Fibonacci, définie par $F_0 = F_1 = 1$ et pour tout $n \in \mathbf{N}$, $F_{n+2} = F_{n+1} + F_n$.
Cette suite fut initialement introduite par Leonardo Fibonacci ($\simeq$ 1175–1250) pour l'étude de la croissance d'une population (fictive) de lapins : le premier mois (numéroté 0), on dispose un couple de lapereaux dans un pré.
Chaque mois, à partir de son deuxième mois d'existence, ce couple donne naissance à un nouveau couple de lapins, qui eux-mêmes commenceront à se reproduire deux mois après leur naissance, etc. On suppose que les lapins ne meurent jamais. Combien de (couples de) lapins a-t-on à la fin du mois n ?
À la fin du mois numéro 0, notre couple initial ne s'est toujours pas reproduit, donc $F_1 = 1$.
Puis, à partir du deuxième mois, le nombre de lapins est égal au nombre de lapins présents le mois précédent, plus le nombre de couples de lapins fraîchement nés. Mais n'ont donné naissance à un couple que les couples pubères, c'est-à-dire ceux qui étaient déjà présents deux mois auparavant. Donc $F_{n+2} = F_{n+1} + F_n$.

Modifions un peu la règle du jeu en supposant que nos lapins sont mortels et finissent en civet au bout de m mois. Ils se seront donc reproduit exactement $m - 2$ fois.
Écrire une fonction qui prend comme paramètres deux entiers n et $m \geqslant 3$ et retourne le nombre de couples de lapins à la fin du mois numéro n.

Exercice 10 Nombres premiers circulaires D
Un nombre premier est dit circulaire si tout nombre obtenu par rotation de ses chiffres est encore premier.
Par exemple, 197 est un nombre premier circulaire car 197, 971 et 719 sont tous premiers.
Combien existe-t-il de nombre premiers circulaires inférieurs à un million ?