TD 04 Structure des données : les chaînes

Lycée Louis Thuillier - Informatique - PCSIB -

Pour enregistrer et stocker des données, on les "*range*" dans des structures Python spécifiques. Cette année, on va s'intéresser à 4 types de structure :
- ▷ les chaines de caractères (*pour du texte*)
- ▷ les listes (*pour tout et n'importe quoi*)
- ▷ tableaux *numpy* (*pour des nombres*)
- ▷ les dictionnaires (*pour des paires*)

> ***Exemple 1 :*** *Une image en informatique est un tableau de nombres, où chaque nombre représente les niveaux de vert, rouge et bleu d'un pixel.*

Une chaîne de caractères est une séquence de caractères qui est :
- ▷ **de longueur fixée** (on ne peut pas lui ajouter un élément sans recréer la chaîne)
- ▷ **non mutable** (on ne peut pas modifier un élément)

C'est comme un PDF ....

1 Définition et structure

1.1 Créer une chaîne

♡ **Commande ♡. Créer une chaîne**

A partir d'une séquence d'éléments on peut créer une chaîne de deux façons :

- ▷ avec la fonction *str* qui permet de convertir certains objets en chaînes de caractères
- ▷ on peut les entourer d'apostrophes (') ou de guillemets (").

```
>>> "Ceci est une chaîne"
'Ceci est une chaîne'

>>> x="Ceci est une chaine"

>>> print(x)
Ceci est une chaine
```

> ***Application 1 :***
> - ▷ *Créer une variable "nom" qui pointe vers une chaîne de caractère contenant les lettres de votre nom.*
> - ▷ *Créer une variable "mot_de_passe" qui est la réponse à "Donner votre mot de passe".*

> ***Remarque :*** *On peut passer à la ligne à l'aide de la commande \n.*
>
> ```
> >>> x="Maitre Corbeau, sur un arbre perché \nTenait dans son bec un fromage"
>
> >>> print(x)
> Maitre Corbeau, sur un arbre perché
> Tenait dans son bec un fromage
> ```

1.2 Accéder à des éléments d'une chaine

Les éléments d'une chaîne sont indexés, c'est-à-dire qu'à chaque élément est associé un nombre qui représente sa place dans la chaîne.

♡ **Commande ♡. Accéder à un élément d'un chaine**

On accède au $n^{\text{ième}}$ caractère d'une chaîne à l'aide de la commande $[n]$ apposé juste après le nome de la chaîne.

nom_de_la_chaine[*n*]

```
>>> mdp="Saucisse"
>>> mdp[0]
'S'
>>> mdp[7]
'e'
```

Attention ! On commence à compter à partir du 0 ! ! Du zéro ! ZERO ! ! 0000000000000

Attention ! les espaces sont des caractères à part entière, tout comme le $\backslash n$.

♡ **Commande ♡. Nombre d'éléments**
La commande *len*(), avec entre parenthèse le nom de la chaîne, permet de renvoyer la longueur de la chaine.

Application 2 : La variable mot_de_passe est une chaine de caractère inconnue. Donner les commandes pour afficher :
- ▷ *le nombre de caractères de mot_de_passe*
- ▷ *le premier caractère de mot_de_passe*
- ▷ *le dernier caractère de mot_de_passe*
- ▷ *l'avant dernier caractère de mot_de_passe*

Remarque : *La commande $[i : j : k]$ permet de sélectionner tous les caractères entre le $i^{\text{ème}}$ inclus et le $j^{\text{ème}}$ exclu avec un pas de k. Si on ne précise rien $k = 1$.*

```
>>> x[0:10]
'Maitre Cor'

>>> x[0:len(x)]
'Maitre Corbeau, sur un arbre perché \nTenait dans son bec un fromage'

>>> x[0:len(x):2]
'Mir oba,sru rr ecé\neatdn o e nfoae'
```

Application 3 : Définir une variable pointant vers votre nom et afficher les 3 première lettres de votre prénom. Les 3 dernière. Toutes les lettres situés à un emplacement impair.

Attention ! *Nous rencontrons ici notre première structure. Les structures sont des lignes de code permettant de réaliser une tâche précise. Elles sont presque plus importantes que les commandes car ce sont elles qui vont constituer l'ossature des programmes qui vous allez écrire par la suite.*

Structure : Structure : accéder à tous les éléments d'une chaine
on accède à tous les éléments d'une chaine via la structure :

```
for k in range(len(chaine)):
    ... chaine[k] ...
```

```
for elt in chaine :
    ... elt ...
```

> *Application 4 : Écrire une fonction occurence qui prend en entrée une lettre et une chaine de caractère et renvoie le nombre de fois que la lettre apparaît dans la chaine de caractère.*

1.3 Opération sur les chaînes

Si on souhaite modifier un élément d'une chaîne, on serait tenter de taper la commande suivante :

```
>>> nom="Soulart"

>>> nom[len(nom)-1]="d"
Traceback (most recent call last):
  File "<console>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

C'est impossible : on ne peut pas modifier l'élément d'une chaîne de caractère !!!

Il y a relativement peu d'opérations sur les chaîne car ces dernières ne sont pas modifiables.

Opérateurs :

▷ l'opérateur "+" permet de concaténer ($\sim$ accoler) deux chaînes
Comme il n'est pas possible d'ajouter un élément à une chaine, on utilise l'opérateur "+" pour "ajouter" un élément.

♡ **Commande ♡. "Ajouter" un élément à une chaine**

On incrémente la chaine de l'élément souhaité :

$$chaine = chaine + élement$$

🔥🔥🔥 **Attention !** l'élément doit être également une chaine !!

```
>>> md="Sauciss"

>>> mdp=mdp+"e"

>>> print(mdp)
Saucissee
```

▷ (*peu utile*) l'opérateur "*" permets de concaténer plusieurs fois à la suite la copie d'une même chaîne

♡ **Commande ♡. Test d'appartenance**
Pour tester si un élément x est dans une chaine de caractères nommée *chaine*, on utilise l'opérateur *in*

$$x \text{ in } chaine$$

Ce test d'appartenance est un Booléen. Il vaut :
▷ *True* si l'élément x est dans la chaîne
▷ *False* sinon

> *Exemple 2 : Les Booléens c'est pour les structures du type if/else ou while :*

```
>>> if "Z" in nom :
...     print("Il y a de S dans votre nom'")
... else :
...     print("Il n'y a pas de S dans votre nom'")
...
...
Il n'y a pas de S dans votre nom'
```

Application 5 : Sans taper la fonction ou l'exécuter, dire ce que fait la fonction "mystere" suivante :

```
def mystere(ch):
    resultat=""
    n=len(ch)
    for i in range(n):
        resultat=resultat+ch[n-i-1]
    return resultat
```

2 Exercices

Petits exercices

1. Ecrire une fonction "*voyelle*" qui prend en argument une lettre sous la forme d'une chaine de 1 caractère et renvoie $True$ si la lettre est une voyelle, $False$ sinon.
   Attention ! Python prend en compte les majuscules $a \neq A$.
2. A l'aide de la fonction précédente, écrire une fonction *nombre_de_voyelles* qui, recevant une chaîne de caractères, renvoie le nombre de voyelles que contient cette chaîne.
3. Ecrire une fonction *test_palindrome* qui, recevant une chaîne de caractères, renvoie $True$ si la chaîne est un palindrome et $False$ sinon.
   Un palindrome est un mot qui se lit à l'identique dans les deux sens : kayak, elle, ...

Petit projet

Soit n un entier plus petit que 26. On souhaite écrire un programme qui code un mot en décalant chaque lettre de l'alphabet de n lettres. On appelle n la clef de chiffrage.
Par exemple,pour $n = 3$ le décalage sera le suivant :

avant décalage	a	b	c	d	...	...	x	y	z
après décalage	d	e	f	g	...	...	a	b	c

Le mot "saucisse" devient "vdxflvvh".

Codage :

1. Ecire une fonction *numero_lettre* qui prend en entrée une lettre (sous la forme d'une chaine) et renvoie sa position dans l'alphabet.
   Attention ! a est en position 0 ! !
2. Ecrire une fonction "*alphabet_decalage*" qui prend en argument un entier $n < 26$ et qui renvoie une chaîne de caractère contenant toutes les lettres dans l'ordre alphabétiques décalées de n, comme dans l'exemple précédent avec $n = 3$.
3. Écrire une fonction "*codage*" qui prend en argument un entier $n < 26$ et un mot sous la forme d'une chaine de caractère et qui renvoie le mot codé, décalé de n.

Décodage :
On souhaite décoder un texte codé avec le principe décrit précédemment. Problème on ne connaît pas le valeur de n qu'a utilisé la personne : on va chercher à la trouver à l'aide d'une analyse fréquentielle.
Le principe de l'analyse fréquentielle repose sur le fait que le "e" est la lettre la plus présente en français. La lettre la plus présente dans un texte codée sera alors, selon une bonne probabilité, celle qui code le e.

Proposer un programme qui permet de décoder un mot codé. On pourra le découper en deux fonction : une qui trouve la clef de chiffrage n et une qui décode le mot sachant n.