Les chaînes de caractères en Python

Planche de TD pour un cours dispensé à l'université de Rennes 2

Romain Tavenard

Le but de cette séance est de se pencher sur l'éventail des possibilités offertes par le type `str` (*String*) en Python.

1 Travail à préparer chez vous avant la séance

1. Écrivez, en pseudo-code, un algorithme permettant de rechercher une sous-chaîne de caractères dans une autre chaîne et de retourner, si elle est présente, la première position à laquelle elle apparaît. Ce pseudo-code ne devra pas faire appel à des fonctions avancées sur les chaînes de caractères.

2 Manipulations de base

Pour les exercices de cette section, vous n'utiliserez pas les fonctions de base sur les chaînes de caractères. Pour créer une chaîne de caractères en Python, il existe deux manières :

```
ma_chaine = 'abcdef'  # Ici, on aurait pu utiliser des guillemets doubles
ma_chaine = """voici un texte
sur plusieurs lignes"""
```

Dans le deuxième cas, la chaîne de caractères pourra contenir des retours à la ligne, des guillemets (simples ou doubles). En Python, la concaténation de chaînes de caractères se fait, comme pour les listes, à l'aide de l'opérateur `+` :

```
prenom, nom = 'Romain', 'Tavenard'
nom_complet = prenom + ' ' + nom
```

De manière générale, la manipulation de chaînes de caractères est très proche de ce que l'on peut faire avec les listes. On peut par exemple accéder au i-ème caractère d'une chaîne en écrivant `ma_chaine[i]` ou parcourir les caractères d'une chaîne avec une boucle `for` tel qu'on le ferait pour une liste.

De même, on peut utiliser l'opérateur $*$ sur les chaînes de caractères. Essayez par exemple d'afficher le contenu de la variable `nom3fois` affectée comme suit :

```
nom3fois = 'Tavenard' * 3
```

2. Écrivez une fonction qui prenne un entier n en argument et affiche à l'écran la figure suivante (pour $n = 4$) :

```
*
**
***
****
```

3. Écrivez une fonction qui prenne en argument une chaîne de caractères et un caractère et retourne le nombre d'occurrences du caractère dans la chaîne.
4. Écrivez une fonction qui retourne la version "miroir" d'une chaîne passée en argument (si l'argument est `"bonjour"`, la fonction retournera `"ruojnob"`). Il est également possible d'extraire des sous-chaînes en utilisant la syntaxe :

```
chaine[indice_debut:indice_fin]
```

Attention, dans ce cas, le caractère d'indice `indice_debut` est inclus alors que celui d'indice `indice_fin` ne l'est pas.

5. Écrivez une fonction correspondant au pseudo-code de la première question qui ne contiendra qu'une boucle `for` (la comparaison de sous-chaînes de caractères pouvant être réalisée par une extraction de sous-chaîne).

3 Interface de la classe `str`

La classe `str` fournit un ensemble d'outils très utiles pour la manipulation de chaînes de caractères. Vous utiliserez ici quelques unes des méthodes de cette classe.

6. Écrivez une fonction qui compte le nombre de mots d'une chaîne de caractères.
7. Écrivez une variante de la méthode `find` qui ne soit pas sensible à la casse.

4 Exercice de synthèse

8. Écrivez une fonction qui remplace, dans une chaîne de caractères, une sous-chaîne par une autre.

Remarque finale : la fonction que vous venez d'implémenter existe en fait déjà dans la classe `str` : il s'agit de la méthode `replace`.