Chapitre 11 : Chaînes de caractères

Une *chaîne de caractères* (string) est une suite ordonnée de caractères. Cette suite par des guillemets "" ou, plus rarement, par deux apostrophes ’’ (en Python les deux sont possibles). Une chaîne de caractères n’est pas modifiable.

Exemples de chaînes littérales

`'bonjour'` est une chaînes de caractères contenant 7 caractères. `''` est la chaîne de caractères vide contenant 0 caractère.

On peut initialiser une variable avec une chaîne de caractères littérale, cette variable sera de type chaîne de caractères.

Concaténation de chaînes de caractères

L’opérateur + permet de concaténer des chaînes de caractères.

```
In [ ]: message='L\'hiver approche.'
        messageBis=' Couvrons nous ...'
        messageTierce='les oreilles ! '
        superMessage = message + messageBis + '\n'+ messageTierce
        print(superMessage)
L'hiver approche. Couvrons nous ...les oreilles !
```

Les chaînes de caractères vues comme des tableaux

Même si les chaînes de caractères peuvent être soumises à une partie des opérations prévues pour les tableaux, elle ne sont pas de **"type"** tableau, c’est-à-dire que `'abc'` est différent de `['a, 'b', 'c']` : le second est modifiable, pas le premier.

```
In [ ]: message='L\'hiver approche, vite vite, il faut se couvrir des morsures du froid.'
        print(message[4]) # affichage : v
        print(message[3] + message[6]) # affichage : ir
        message[1] = 'c'# erreur
```

On peut déterminer la longueur (c’est-à-dire le nombre de caractères) d’une chaîne, en faisant appel à la fonction intégrée **len()**.

Les chaînes sont comparables

La fonction **ord(c)** permet de connaître la valeur du code ASCII représentant le caractère `c`. Les chaînes peuvent ensuite être comparées alphabétiquement selon l’ordre du code ASCII. Tous les opérateurs de comparaison `(>, <..)` fonctionnent alors avec les chaînes de caractères. Cela est très utile pour trier des mots par ordre alphabétique (usuel) quand ils sont écrits en majuscules/minuscles :

```
In [ ]: mot = input("Entrez un mot quelconque : ")
        if mot < "chat":
            place = "est avant"
        elif mot > "chat":
            place = "est après"
        else:
            place = "se confond avec"
        print ("Le mot", mot, place, "le mot 'chat' dans l'ordre alphabétique")
```

Quelques fonctions sur les chaines

- **index(c)** : retourne l'index de la première occurrence du caractère `c` dans la chaîne ou déclenche une erreur si `c` est absent de la chaîne.

```
In [6]: ch = "Bonjour à tous"
        print (ch.index("t"))
10
```

- **find(mot)**: renvoie la position du début de la sous-chaîne `mot` dans la chaîne, en partant du début de la chaîne et `-1` si `mot`est absent de la chaîne.

```
In [5]: ch= "je suis étudiant à Paris 13"
        mot = "suis"
        print (ch.find(mot))
3
```

- **count(mot)** : renvoie le nombre d'occurrencs de la sous-chaîne `mot` qui apparaissent sans chevauchement dans la chaîne.

```
In [4]: ch = "Le héron au long bec et au long cou"
        mot = 'long'
        print (ch.count(mot))
2
```

- **lower()** : convertit une chaîne en minuscules. **upper()**: convertit une chaîne en majuscules.

```
In [3]: phrase1 = "Merci beaucoup"
        print (phrase1.upper())

        phrase2 ="ATTENTION : Danger !"
        print (phrase2.lower())
MERCI BEAUCOUP
attention : danger !
```

- **strip()**: enlève les espaces éventuels au début et à la fin de la chaîne.

```
In [2]: phrase = "   Monty Python   "
        print(phrase.strip())
Monty Python
```

- **replace(old, new)**: remplace tous les caractères `old` par des caractères `new` dans la chaîne.

```
In [1]: phrase = "Je suis étudiant en Informatique"
        print(phrase.replace(" ","_"))
Je_suis_étudiant_en_Informatique
```