Manipulation des chaînes de caractères en Python

Table des matières

Introduction

Les chaînes de caractères sont une partie fondamentale de la programmation, en particulier en Python. Elles permettent de stocker et manipuler du texte. Ce cours couvre les différentes opérations possibles sur les chaînes de caractères, les fonctions intégrées utiles et des exemples pour bien comprendre leur utilisation, y compris la manière dont les chaînes sont saisies et stockées dans une variable.

Objectifs

- Comprendre ce qu'est une chaîne de caractères en Python.
- Manipuler des chaînes de caractères en utilisant des méthodes et des opérations courantes.
- Saisir et stocker une chaîne de caractères dans une variable.
- Formater, extraire, et modifier des chaînes de caractères.

1 - Introduction aux chaînes de caractères

Une chaîne de caractères (ou "string") est une séquence de caractères. En Python, elle peut être créée en utilisant des guillemets simples (') ou doubles (").

Exemple :

```
chaine1 = "Bonjour tout le monde"
chaine2 = 'Python est génial'
print(chaine1)
print(chaine2)
```

Résultat :

```
Bonjour tout le monde
Python est génial
```

2 - Saisie d'une chaîne de caractères et stockage

2.1 - Saisie d'une chaîne

Pour permettre à un utilisateur de **saisir une chaîne de caractères**, on utilise la fonction `input()`. La chaîne saisie par l'utilisateur est ensuite **stockée dans une variable**.

Exemple :

```
nom = input("Veuillez entrer votre prénom : ")
print("Bonjour, " + nom + " !")
```

- **`input()`** permet de capturer une chaîne de caractères saisie par l'utilisateur.
- Cette chaîne est ensuite **stockée** dans une **variable** (`nom` dans cet exemple).
- Le contenu de cette variable est ensuite utilisé dans une phrase de bienvenue.

Résultat (exemple avec une entrée utilisateur) :

```
Veuillez entrer votre prénom : Alice
Bonjour, Alice !
```

2.2 - Comment une chaîne est-elle stockée dans une variable ?

Une chaîne de caractères saisie par l'utilisateur est stockée dans une variable comme une **séquence de caractères**. Chaque caractère a une position (ou index) et est accessible en utilisant cette position. La chaîne de caractères est traitée comme un **tableau de caractères**.

Par exemple, si l'utilisateur saisit "Alice", Python stocke cela comme suit :

```
Index :      0   1   2   3   4
Caractères : A   l   i   c   e
```

2.3 - La fonction `len()` et les caractères invisibles

La fonction `len()` permet de **compter le nombre de caractères** présents dans une chaîne. Cela inclut tous les caractères visibles et **les espaces**.

Exemple :

```
nom = "Alice"
print(len(nom))
```

Résultat :

```
5
```

La fonction `len()` compte tous les **caractères visibles et les espaces**, mais elle **ne compte pas le caractère de fin de chaîne**, qui est utilisé en interne par Python pour marquer la fin de la chaîne.

Exemple avec espaces :

```
chaine = "   Alice   "
print(len(chaine))
```

Résultat :

```
11
```

Python compte aussi les espaces au début et à la fin de la chaîne.

3 - Les opérations de base

3.1 - Accéder aux caractères d'une chaîne

Les chaînes de caractères sont indexées. Chaque caractère dans une chaîne a une position qui commence à 0.

Exemple :

```
chaine = "Python"
print(chaine[0])  # Premier caractère
print(chaine[-1]) # Dernier caractère
```

Résultat :

```
P
n
```

3.2 - Slices (Tranches)

Les slices permettent de découper une chaîne de caractères en spécifiant un intervalle d'indices.

Exemple :

```
chaine = "Python"
print(chaine[1:4])  # Caractères de l'indice 1 à 3
print(chaine[:3])   # Du début à l'indice 2
print(chaine[3:])   # De l'indice 3 à la fin
```

Résultat :

```
yth
Pyt
hon
```

4 - Les méthodes de manipulation

Python propose de nombreuses méthodes intégrées pour manipuler les chaînes.

4.1 - `len()` : Longueur d'une chaîne

Cette fonction renvoie le nombre de caractères dans une chaîne.

Exemple :

```
chaine = "Bonjour"
print(len(chaine))
```

Résultat :

```
7
```

4.2 - `lower()` et `upper()` : Conversion en minuscules/majuscules

Ces méthodes permettent de convertir une chaîne en minuscules (`lower()`) ou en majuscules (`upper()`).

Exemple :

```
chaine = "Python"
print(chaine.lower())
print(chaine.upper())
```

Résultat :

```
python
PYTHON
```

4.3 - `strip()` : Suppression des espaces

La méthode `strip()` supprime les espaces en début et en fin de chaîne.

Exemple :

```
chaine = "  Python  "
print(chaine.strip())
```

Résultat :

```
Python
```

4.4 - `replace()` : Remplacement de sous-chaînes

La méthode `replace()` permet de remplacer une sous-chaîne par une autre.

Exemple :

```
chaine = "Bonjour tout le monde"
print(chaine.replace("tout le monde", "à tous"))
```

Résultat :

```
Bonjour à tous
```

5 - La concaténation et la répétition

5.1 - Concaténation

La concaténation permet d'assembler plusieurs chaînes ensemble en utilisant l'opérateur +.

Exemple :

```
prenom = "Alice"
salutation = "Bonjour, " + prenom + "!"
print(salutation)
```

Résultat :

```
Bonjour, Alice!
```

5.2 - Répétition

L'opérateur * permet de répéter une chaîne plusieurs fois.

Exemple :

```
mot = "Python! "
print(mot * 3)
```

Résultat :

```
Python! Python! Python!
```

6 - Formatage des chaînes de caractères

6.1 - Utilisation de `format()`

La méthode `format()` permet d'insérer des valeurs dans une chaîne.

Exemple :

```
nom = "Alice"
age = 25
phrase = "Je m'appelle {} et j'ai {} ans".format(nom, age)
print(phrase)
```

Résultat :

```
Je m'appelle Alice et j'ai 25 ans
```

6.2 - F-strings

Introduites dans Python 3.6, les f-strings permettent un formatage encore plus lisible.

Exemple :

```
nom = "Alice"
age = 25
phrase = f"Je m'appelle {nom} et j'ai {age} ans"
print(phrase)
```

Résultat :

```
Je m'appelle Alice et j'ai 25 ans
```

7 - Recherche dans une chaîne de caractères

7.1 - `in` : Vérifier la présence d'une sous-chaîne

L'opérateur `in` permet de vérifier si une sous-chaîne est présente dans une chaîne.

Exemple :

```
chaine = "Bonjour tout le monde"
print("Bonjour" in chaine)
print("Bonsoir" in chaine)
```

Résultat :

```
True
False
```

7.2 - `find()` : Trouver la position d'une sous-chaîne

La méthode `find()` retourne l'indice de la première occurrence d'une sous-chaîne. Si elle n'est pas trouvée, elle renvoie `-1`.

Exemple :

```
chaine = "Bonjour tout le monde"
print(chaine.find("tout"))
print(chaine.find("soir"))
```

Résultat :

```
8
-1
```

8 - Conclusion

Les chaînes de caractères en Python sont riches en fonctionnalités. Leur manipulation est essentielle dans la plupart des programmes. En connaissant les méthodes et opérations sur les chaînes, vous serez en mesure d'extraire, modifier et formater du texte efficacement.