PROGRAMMATION PYTHON

Chapitre 8: Les chaînes de caractères

Sommaire

Définition

Les chaines de caractères vont nous permettre de stocker et traiter des valeurs alphanumériques des chiffres et des lettres comme « Chat, Chien, Toto1,… ». C'est un type d'objet incontournable en programmation.

Contrairement aux autres types vus jusqu'à présent, les chaines de caractères sont des types spéciaux appelées des objets.

Un objet est une structure de données, comme les variables qui peut contenir elle-même d'autres variables et fonctions.

Les chaines de caractères sont considérées comme des objets car elles contiennent des fonctions qui permettent de faire des opérations comme mettre en minuscule, majuscule…

Python traite les chaines de caractères comme de type **str** pour **String**.

```
msg = "dfd"
type(msg)
<class 'str'>
```

Définition

Certains langages disposent du type chaines de caractères et type caractère. Dans ces langages une variable qui contient un caractère est considérée comme un type caractère et pour plus de 1 caractère une chaine de caractères.

Python ne fait pas cette distinction, tout contenu d'une variable entourée par les guillemets simples ou double est considérée comme chaine de caractères.

```
msg = "toto"
type(msg)
<class 'str'>
msg1="t"
type(msg1)
<class 'str'>
msg = ""
type(msg)
<class 'str'>
msg1=''
type(msg1)
<class 'str'>
```

Un objet comme str est issu d'une classe. La classe est une forme de type de donnée qui permet de définir des fonctions et variables propres au type. Dans le cas des chaines de caractères on a les fonctions comme mettre en minuscule, majuscule… fournies par les développeurs de Python.

Les méthodes de bases de la classe str

La classe **str** de Python contient une multitude de fonctions disponible ici Opérations usuelles sur des chaînes ou Méthodes de chaînes de caractères.

Les plus utilisées sont décrites ci-dessous:

- Pour mettre une chaine de caractères en majuscule on utilise la fonction **upper** avec le format suivant: $chaine.upper()$. Elle retourne le résultat de la chaine de caractère en majuscule.

```
msg = "hello"
msg.upper()
'HELLO'
msg
'hello'
```

- Pour mettre une chaine de caractères en minuscule on utilise la fonction **lower** avec le format suivant: $chaine.lower()$. Elle retourne le résultat de la chaine de caractères en minuscule.

```
msg = "HELLO"
msg.lower()
'hello'
msg
'HELLO'
```

Les méthodes de bases de la classe str

- Pour mettre la première lettre en majuscule on utilise la fonction **capitalize**.

```
msg = "bonjour tout le monde!"
msg.capitalize()
'Bonjour tout le monde!'
```

- Pour mettre chaque mot de la chaine en majuscule on utilise la fonction **title.**

```
msg = "bonjour tout le monde!"
msg.title()
'Bonjour Tout Le Monde!'
```

- Pour supprimer les espaces au début et à la fin de la chaine, on utilise la fonction **strip.**

```
msg = "    bonjour tout le monde!    "
msg
'    bonjour tout le monde!    '
msg.strip()
'bonjour tout le monde!'
```

- Pour avoir le nombre de caractères d'une chaine on utilise la fonction **len.**

```
msg = "hello"
len(msg)
5
msg = ""
len(msg)
0
```

Formater et afficher une chaîne

Pour afficher le contenu d'une variable ainsi qu'une chaine de caractères on utilise la fonction **print.**

Par contre pour afficher une chaine avec le contenu d'une ou plusieurs variables, on utilisera les méthodes de formatage de chaînes de caractères.

Python propose une méthode **format** qui permet de formatter une chaine.

```
nom = "John"
prenom = "Doe"
age = 14
print("Je suis {0} {1} j'ai {2} ans.".format(nom, prenom, age))
Je suis John Doe j'ai 14 ans.
```

La méthode **format** permet de formater une chaîne de gauche à droite on a:

- Une chaîne de caractères qui ne présente rien de particulier que des nombres qui indiquent l'ordre d'insertion des valeurs des variables 0, puis 1..
- La méthode **format** qui va passer les paramètres des variables dont les valeurs seront insérées dans la chaîne.
- Quand Python exécute cette méthode, il remplace la chaine {0} par la première variable passée à la méthode $format$ ensuite la deuxième variable ainsi de suite.

Formater et afficher une chaîne

La méthode **format** peut être utilisée pour créer une nouvelle chaîne de caractères.

```
nom = "John"
prenom = "Doe"
age = 14
personne = "{0} {1} {2}".format(nom, prenom, age)
personne
'John Doe 14'
type(personne)
<class 'str'>
print(personne)
John Doe 14
```

L'ordre des entre les accolades {} est importante le premier paramètre de format correspond à l'élément {0} dans la chaine.

```
personne = "{1} {0} {2}".format(nom, prenom, age)
personne
'Doe John 14'
```

On peut ne pas spécifier des valeurs dans entre les accolades dans ce cas c'est le format naturel qui sera pris en compte.

```
personne = "{} {} {}".format(nom, prenom, age)
print(personne)
John Doe 14
```

Formater et afficher une chaîne

On peut également nommer les variables que l'on va afficher qui est plus intuitif que leur indice.

```
print("Je suis {nom} {prenom} j'ai {age} ans.".format(nom="John", prenom="Doe", age=14))
Je suis John Doe j'ai 14 ans.
```

Python propose une autre méthode la **f-string** qui permet de faire les mêmes opérations que **format.** En plus avec la f-string on peut afficher le nom de variable.

```
# Formater avec f-string
nom = "John"
prenom = "Doe"
age = 14
print(f"{nom} {prenom} {age}")
John Doe 14
# Formater en affichant le nom des variables
print(f"{nom=} {prenom=} {age=}")
nom='John' prenom='Doe' age=14
# Création de variable
nomComplet = f"{nom} {prenom}"
print(nomComplet)
John Doe
```

Contrairement à **format** avec la f-string on n'a pas besoin de spécifier ici l'ordre d'insertion des valeurs des variables lors du formatage de la chaine de caractères.

Concaténation de chaînes

La concaténation de chaînes est une opération qui consiste à regrouper deux chaines en une en mettant la seconde à la suite de la première. Pour faire de la concaténation on utilisera le signe plus +.

```
# Concaténation simple
nom = "John"
prenom = "Doe"
nom + prenom
'JohnDoe'
# Concaténation en mettant un espace
nom + " " + prenom
'John Doe'
# Concaténation de plusieurs chaînes
"Persone " + nom + " " + prenom
'Persone John Doe'
```

La concaténation ne marche que sur les types chaine de caractères si on essaie de faire une concaténation entre un type chaine de caractères et un nombre Python lève une exception car ambiguë Python ne sait pas s'il doit faire une opération de nombre ou une concaténation de chaines. On utilisera la fonction **str** pour convertir un nombre en chaine si on veut éviter une erreur.

```
# Essaie concaténation entre chaine et nombre
nom = "John"
age = 14
nom + age
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    nom + age
TypeError: can only concatenate str (not "int") to str
# Concaténation avec convertion
nom = "John"
age = 14
nom + str(age)
'John14'
```

Parcours de chaîne par indice

1. XXX

Parcours de chaîne avec for

1. XXX

Sélection de chaînes

1. XXX

Exercices

1. XXX

FIN CHAPITRE 8