

# La structure des chaînes de caractères

## 1 Le type

Le type des chaînes de caractères est appelé `str` en Python (et `string` dans presque tous les autres langages de programmation).

## 2 L'ensemble des valeurs

Il existe 256 « caractères » qui couvrent à peu près tout ce que l'on peut taper au clavier en une fois, Une *chaîne de caractères*, comme son nom l'indique, est une suite de caractères (de longueur aussi grande que nécessaire, avec les limitations habituelles d'occupation mémoire qui ne seront pas atteintes dans le cadre de ce cours).

Une chaîne de caractères doit être délimitée par des guillemets comme dans `"toto"`.

## 3 Les opérations

Les opérations les plus simples sont :

Opération	Entrée	Sortie
<code>+</code> <code>%</code>	<code>str×str</code>	<code>str</code>
<code>len</code>	<code>str</code>	<code>int</code>
<code>==</code> <code>&lt;</code> <code>&gt;</code> <code>&lt;=</code> <code>&gt;=</code> <code>!=</code>	<code>str×str</code>	<code>bool</code>

Plus généralement,

l'opération `%` peut avoir plusieurs types d'entrées; voir ci-dessous.

## 4 La sémantique des opérations

Le `+` dénote la concaténation (mise bout à bout) des chaînes de caractères; `len` est la longueur (c'est-à-dire le nombre de caractères) d'une chaîne; les comparaisons sont les comparaisons alphabétiques inspirées de celles d'un dictionnaire français ou anglais, sachant que les chiffres et ponctuations viennent avant les lettres, et que les majuscules viennent avant les minuscules.

« `%` » est l'opération de *substitution*. Si la chaîne de caractères  $s_0$  contient « `%s` » et si  $s_1$  est une autre chaîne de caractères, alors  $s_0 \% s_1$  est la chaîne obtenue en remplaçant dans  $s_0$  les deux caractères `%s` par la chaîne  $s_1$ .

S'il y a plusieurs « `%s` », il faut mettre entre parenthèses autant de chaînes ( $s_1, \dots, s_n$ ) après l'opération `%`.

S'il y a un « `%d` » au lieu de « `%s` » il faut mettre un entier relatif à la place d'une chaîne (`d` comme décimal et `s` comme string).

S'il y a un « `%f` » (`f` comme float) il faut mettre un nombre réel. On peut imposer le nombre de chiffres après la virgule : « `%.4f` » par exemple limite la précision à 4 chiffres après la virgule.

L'opération de substitution `%` admet d'autres types de substitutions; celles-ci sont les plus utiles.

## 5 Exemples

```
>>> "toto" + "tutu"
'tototutu'
>>> len("toto")
4
>>> "toto" < "tutu"
True
>>> "ab" < "aab"
False
>>> "a b" == "ab"
False
```

Noter l'absence d'espace entre `"toto"` et `"tutu"` après concaténation.

Et pour la substitution :

```
>>> "bonjour %s ; il faut beau aujourd'hui." % "Pierre Dupond"
"bonjour Pierre Dupond ; il faut beau aujourd'hui."
>>> nom = "Dupond"
>>> prenom="Pierre"
>>> genre = "homme"
>>> "Ce %s s'appelle %s et c'est un %s" % (prenom,nom,genre)
"Ce Pierre s'appelle Dupond et c'est un homme"
>>> age = 41
>>> "%s %s a %d ans." % (prenom,nom,age)
'Pierre Dupond a 41 ans.'
>>> "%s %s mesure %fm." % (prenom,nom,1.85)
'Pierre Dupond mesure 1.85000m.'
>>> "%s %s mesure %.2fm." % (prenom,nom,1.85)
'Pierre Dupond mesure 1.85m.'
```