Laurent Signac – CC-BY-SA – 11-01-21 1245 dcaba3746a4e7d81d7f2

- les chaînes ne sont pas mutables
- les chaînes sont itérables
- les chaînes sont des séquences
- les chaînes ont une longueur (`len`)
- les chaînes (python 3) sont des chaînes unicode

Un **litéral** chaîne de caractères est entre `" "` ou entre `' '` (ou triples si multilignes). Une chaîne de caractère est séquence, donc on peut utiliser la notation avec `[.]` comme pour les listes.

```
s = "Chapeau"
print("Chapeau")
print(s)
print(s[3])
print(s[1:5])
s2 = """Une chaîne de caractères
multi-lignes. Il suffit de l'encadrer
entre triple guillements (ou triple quotes)"""
print(s2[4:10])
```

Dans ce qui précède, `s` est une variables de type chaîne de caractère (on dira parfois juste « `s` est une chaîne de caractères »), alors que `"Chapeau"` est un **litéral** chaîne de caractères (mais on dira aussi parfois juste « `"Chaines"` est une chaîne de caractères). Vous devrez comprendre pourquoi il y a des guillemets autour de `"Chapeau"`, mais pas autour de `s`. Les littéraux chaînes de caractères, s'ils occupent plusieurs lignes doivent être délimités par `"""` ou `'''`.

1 Chaines ↔ listes

Les chaînes sont non mutables. On les transformer donc parfois en liste pour pouvoir les modifier, avant de les retransformer à nouveau en chaînes.

```
s1 = "Chapeau"
lst = list(s1)
print(lst)

--> ['C', 'h', 'a', 'p', 'e', 'a', 'u']
```

Puis on modifie la liste :

```
lst[3] = 't'
s2 = "".join(lst) # Recolle les bouts de lst...
print(s2)

--> "Chateau"
```

2 Code Unicode

Python utilise des chaînes unicode. On peut donc accéder au numéro d'un caractère, et inversement, produire le caractère qui correspond à un numéro.

```
val = ord('A') # Le code de A est 65
val = val + 1  # val contient 66
print(val)

--> 66
```

```
car = chr(val) # Caravtère de code 66 ?
print(car)

--> 'B'
```

Unicode contient beaucoup de choses :

```
lst = [chr(c) for c in range(128512, 128520)]
print(" ".join(lst))

--> 😀 😁 😂 😃 😄 😅 😆 😇
```

3 Itérer sur les chaînes

Puisqu'une chaîne est une séquence, on peut itérer sur une chaîne, comme on le faisait sur les listes :

```
for ch in "allohomora collaporta":
    if ch not in 'aeiou':
        print(ch, end='')

--> llhmr cllprt
```

Rem : l'écriture `ch not in 'aeiou'` est préférable à `not ch in 'aeiou'` (PEP 8 [713])

4 Conversion Chaînes ↔ nombres

Lorsqu'on lit des fichiers de données, ou qu'on en écrit, on doit savoir convertir des nombres en chaîne et inversement.

Pour convertir une chaîne `ch` en `int` ou `float`, on écrit simplement : `int(ch)` ou `float(ch)` (il faut bien sûr que la conversion soit possible).

Dans l'autre sens, on peut simplement écrire `str(val)` où `val` est un `int` ou un `float`. Mais bien souvent, on veut pouvoir contrôler différentes choses : combien de chiffres après la virgule, notation scientifique ou non etc… Ceci se fait par le biais de la méthode `format`, qui possède trop d'options pour qu'on puisse la détaille complètement ici. Voici juste quelques exemples :

```
pi = 3.1415926535
print("{}".format(pi))

--> 3.1415926535

print("{:.2f}".format(pi))

--> 3.14

print("{:07.3f}".format(pi))

--> 003.142

print("{:e}".format(pi * 100))

--> 3.141593e+02
```

5 Autres fonctions/méthodes

- `len(str0)` : longueur de la chaîne `str0`

- `str1 in str2` : booléen qui vaut `True` si `str1` est une sous-chaîne de `str2`
- `str0.replace(str1, str2)` : remplace les occurrences de `str1` par `str2` dans `str0`
- `str0.count(str1)` : compte le nombre d'apparitions (non recouvrantes) de `str1` dans `str0`
- `str0.upper()` : renvoie une version de `str0` en majuscules
- `str0.lower()` : renvoie une version de `str0` en minuscules
- `str0.index(str1)` : cherche la position de `str1` dans `str0`
- `str0.split(",")` : découpe `str0` en liste de chaînes en découpant sur les virgules

Exemples :

```
st = "Wingardium leviosa, Prior incanta"
print("gard" in st)

--> True

print(st.replace("io", "yo"))

--> Wingardium levyosa, Pryor incanta

print(st.count("i"))

--> 5

print(st.upper())

--> WINGARDIUM LEVIOSA, PRIOR INCANTA

print(st.lower())

--> wingardium leviosa, prior incanta

print(st.index("io"))

--> 14

print(st.split(", "))

['Wingardium leviosa', 'Prior incanta']
```