Leçon d'informatique : chaînes de caractère

S. Benlhajlahsen - PCSI$_1$

Sommaire

Introduction On a déjà présenté les chaînes de caractères. On rappelle qu'une chaîne de caractère est un objet de type `str`.

I Rappels

Synthaxe de base L'accès aux caractères d'une chaîne se fait avec la même syntaxe qu'une liste. Pour une chaîne `a` :

- `a[i]` donne le i-ème caractère[a].
- `a[i:j]` est la sous-chaîne allant du i-ème caractère jusqu'au $(j-1)$-ème inclus.

Mais contrairement aux listes, les chaînes de caractères **ne sont pas mutables**. Donc :

- on ne peut pas changer un caractère avec une syntaxe du genre `a[i]="e"`.
- la fonction `append` plante avec un message d'erreur si on essaie de l'appliquer à une chaîne de caractères (voir figure 1).

Enfin, la chaîne vide s'écrit `""` ou `''`.

a. la numérotation démarrant à 0 pour le premier caractère

```
>>> a = "bonjour"
>>> b = 'salut'
>>> a[2]
'n'
>>> a[2:6]
'njou'
>>> b[0]="c"
TypeError: 'str' object does not support item assignment
>>> a.append("e")>
AttributeError: 'str' object has no attribute 'append'
```

FIGURE 1

Opérations sur les chaînes de caractères On rappelle que `+` est un opérateur de concaténation et `*` est opérateur de répétition (voir figure 2).

Méthode `str` La fonction/méthode `str` permet de transformer l'argument en la chaîne correspondante (voir figure 3).

```
>>> "bon"+'jour'
'bonjour'
>>> 2*'bon'
'bonbon'
```

FIGURE 2

```
>>> str(123)
'123'
>>> str(3.14)
'3.14'
>>> str([1,2,3])
'[1, 2, 3]'
```

FIGURE 3

II Caractères spéciaux

II.A Échapper guillemets et apostrophes

Idée Comme les guillemets (`"..."`) ou les apostrophes (`'...'`) sont utilisées pour délimiter une chaîne de caractères, comment fait-on pour inclure ces caractères dans une chaîne, par exemple pour écrire le mot aujourd'hui ? Il y a une astuce : le caractère d'échappement. Il s'agit de l'anti-slash `\`.

Synthaxe Dans une chaîne de caractères, `\"` et `\'` donnent, respectivement, un guillemet et une apostrophe lors de l'affichage.
Ce guillemet ou cet apostrophe n'est pas considéré comme un délimiteur de chaîne de caractères. On dit qu'il "a été échappé par l'anti-slash" (voir figure 4).

Remarque Sur la plupart des claviers Windows / Linux, `\` s'obtient par la combinaison de touches `Alt Gr+8`. Sur un clavier Mac, il s'obtient par `Alt+MAJ+/` (même si ça n'est pas écrit sur la touche).

```
>>> print('aujourd'hui')
SyntaxError: invalid syntax
>>> print('aujourd\'hui')
aujourd'hui
```

FIGURE 4

II.B Échapper…le caractère d'échappement ?

Idée Comment faire pour qu'un anti-slash contenu dans une chaîne ne soit pas traité comme le caractère d'échappement ? Il suffit de l'échapper ! (voir figure 5).

II.C Caractères invisibles

Un caractère invisible est un élément d'une chaîne de caractère qui ne correspond à aucun symbole visible (lettre, chiffe, ponctuation...) mais qui a tout de même un effet sur son apparence.

Synthaxe - saut de ligne Un passage à la ligne dans une chaîne de caractères est représenté par le caractère invisible `\n` (pour "new line"). On pourra se reporter à l'exemple de la figure 6.

```
print('Voici un \\ tout à fait ordinaire')
Voici un \ tout à fait ordinaire
```

FIGURE 5

```
>>> print("Hello\nWorld !")
Hello
World !
>>> len("a\nb")
3
```

FIGURE 6

Remarque Attention, `\n` est un seul caractère dans la mémoire de l'ordinateur, même si vous devez en taper deux pour l'écrire.

III Découper une chaîne de caractères

Exemple On souhaite découper les chaînes de caractères `a = "2.54,3.14,6.28"` et `b = "2.54\t3.14\t6.28"` pour produire des listes en découpant sur la virgule `','` ou le caractère de tabulation `"\t"`. On peut le faire avec une boucle qui parcourt caractère par caractère (voir figure 7).

```
a="2.54,3.14,6.28"
b="2.54\t3.14\t6.28"
L1=[""]
indice=0
for x in a:
    if x != ",":
        L1[indice] += x
    else:
        indice+=1
        L1.append("")
print(L1)
####################
L2=[""]
indice=0
for x in b:
    if x != "\t":
        L2[indice] += x
    else:
        indice+=1
        L2.append("")
print(L2)
```

FIGURE 7

Méthode `split` La fonction `split` découpe une chaîne de caractères suivant une syntaxe similaire à append, avec une différence : elle n'agit pas in place mais renvoie quelque chose.

```
liste = chaine.split(separateur)
```

- `liste` est une variable de type `list`, chaque élément étant un morceau de chaîne.

- `chaine` est la chaîne de caractères à découper.
- `separateur` est de type chaîne de caractères. C'est le caractère séparateur, qui sera supprimé des morceaux ainsi obtenus (voir figure 8).

```
a="2.54,3.14,6.28"
b="2.54\t3.14\t6.28"
print(a.split(","))
print(b.split("\t"))
```

```
# renvoie
['2.54', '3.14', '6.28']
['2.54', '3.14', '6.28']
```

FIGURE 8

IV Autre fonctions (hors-programme)

Fonctions fournies par Python En exécutant `help(str)`, on obtient la liste de fonctions fournies par Python. A titre d'exemple :

- `"bonjour".count("o")` compte le nombre d'occurence de la lettre `"o"` dans la chaîne `"bonjour"`. Cela renvoie `2`.
- `"bonjour".index('o')` renvoie l'indice de la première occurence du caractère `"o"` dans la chaîne `"bonjour"`. Cela renvoie `1`.
- `"bonjour".upper()` convertit tous les caractères alphabétiques de la chaîne "bonjour" en majuscules. Cela renvoie `"BONJOUR"`.

V Exercices

Exercice 1 : Ecrire une fonction `compte(a, ch)` qui renvoie le nombre d'apparitions d'une lettre `a` dans une chaîne de caractères. On n'utilisera pas `count`.
Réponse en figure 9.

Exercice 2 : Ecrire une fonction `premier_indice(a, ch)` qui renvoie le premier indice d'apparition d'une lettre dans une chaîne de caractères ; la fonction renverra -1 si le caractère n'est pas dans la chaîne. On utilisera pas la fonction `index`.
Réponse en figure 10.

Exercice 3 : Ecrire une fonction `majuscules(ch)` qui renvoie une chaîne de caractères obtenue à partir de `ch` en remplaçant toutes les minuscules non accentuées par des majuscules ; les majuscules et les minuscules se suivent dans le même ordre dans la liste des caractères donnés par la fonction `ord`.
Réponse en figure 11.

Exercice 4 : Ecrire une fonction `separe(ch, a)` qui renvoie la liste des chaînes extraites de `ch` en découpant de part et d'autre du caractère `a`.
Réponse en figure 12.

```python
def compte(a,ch) :
    compteur = 0
    for x in ch:
        if x == a :
            compteur += 1
    return compteur
```

FIGURE 9 – Réponse pour l'exercice 1.

```python
def premier_indice(a,ch) :
    resultat = -1
    n = len(ch)
    position = 0
    while position < n and resultat = -1:
        if ch[position] == a :
            resultat = position
        position += 1
    return resultat
```

FIGURE 10 – Réponse pour l'exercice 2.

```python
def majuscules(ch) :
    ecart = ord("A") - ord("a")
    resultat = ""
    for x in ch:
        k = ord(x)
        if ord("a") <= k and k <= ord("z") :
            y = chr(ord(x)+ecart)
        else :
            y = x # On ne change que a .. z
        resultat = resultat + y
    return resultat
```

FIGURE 11 – Réponse pour l'exercice 3.

```python
def separe(ch,a):
    reponse = []
    mot = ""
    for x in ch:
        if x != a :
            mot = mot + x
        else :
            reponse.append(mot)
            mot = ""
    return reponse
```

FIGURE 12 – Réponse pour l'exercice 4.