PYTHON ET CHAINES DE CARACTERES

September 15, 2019

1 Les chaînes de caractères avec Python.

1.1 Valeurs saisies ou définies.

Les chaînes de caractères (*string*) en Python sont un type de variables au même titre que les entiers (*int*) ou les nombres décimaux (*float*). Par défaut, lors de l'utilisation d'une saisie avec *input()*, on récupère une chaîne de caractères.

```
In [1]: ch=input()

345


In [2]: type(ch)

Out[2]: str

In [3]: ch

Out[3]: '345'
```

La valeur saisie est donc reconnue comme une chaîne de caractères et lorsqu'on l'utilise, on identifie son type par la présence des ' '. Inversement, si on veut créer une chaîne de caratères, on la délimite par ces ' '.

```
In [8]: ch=345
        type(ch)

Out[8]: int

In [7]: ch='345'
        type(ch)

Out[7]: str
```

1.2 Concaténation.

La concaténation est l'assemblage de plusieurs chaînes de caractères. On utilise l'opérateur **+** ou si doit reproduire la même chaîne, l'opérateur *.

```
In [9]: ch1='Trop '
        ch2='fort !'
        ch=ch1+ch2
        ch

Out[9]: 'Trop fort !'
```

```
In [10]: ch1='ah'
         ch=6*ch1
         ch

Out[10]: 'ahahahahahah'
```

1.3 Manipulation des caractères.

On retrouvera les méthodes employées avec les listes dans une autre partie. Une chaîne de caractères est une suite de caractères ordonnées, le premier à la position 0. Pour récupérer un caractère, on utlise les crochets [ ].

Dans la chaîne *ch*, si on veut le caractère à la position i, on saisira *ch[i]*.

```
In [11]: ch='la NSI à Couteaux'
         ch[0]

Out[11]: 'l'
```

```
In [12]: ch='la NSI à Couteaux'
         ch[3]

Out[12]: 'N'
```

On peut récupérer une partie entre le iéme (inclus) et le jiéme (exclus) avec l'instruction *ch[i:j]*.

```
In [13]: ch='la NSI à Couteaux'
         ch[3:5]

Out[13]: 'NS'
```

ch[i:j] sera ainsi composé de j-i caractères.
ch[-1] donne le dernier caractère :

```
In [14]: ch='la NSI à Couteaux'
         ch[-1]

Out[14]: 'x'
```

ch[i:] permet de récupérer les caractères à partir du iéme :

```
In [15]: ch='la NSI à Couteaux'
         ch[3:]

Out[15]: 'NSI à Couteaux'
```

ch[:i] permet de récupérer les i premiers caractères :

```
In [16]: ch='la NSI à Couteaux'
         ch[:3]

Out[16]: 'la '
```

On notera qu'un espace est un caractère comme les autres.

1.4 Manipulation des chaînes.

On est souvent amené à vouloir changer une chaîne de caractères (mettre en majuscule le premier caractère, ajouter une ponctuation, remplacer un mot par un autre, crypter un message ...).

Remplaçons par exemple le 4iéme caractère par un 'o':

```
In [17]: ch='la NSI à Couteaux'
         ch[3]='o'
         ch


         ---------------------------------------------------------------------------

         TypeError                                 Traceback (most recent call last)

         <ipython-input-17-83b31f253d71> in <module>()
           1 ch='la NSI à Couteaux'
     ----> 2 ch[3]='o'
           3 ch


         TypeError: 'str' object does not support item assignment
```

On obtient un message d'erreur : la chaîne de caractère ne supporte pas les affectations. On dit que la chaîne de caractère est un objet **non mutable** (contrairement à une variables entière ou décimale). On doit donc si on veut obtenir une modification créer une nouvelle variable et procéder, par exemple, par concaténation :

```
In [20]: ch='la NSI à Couteaux'
         ch1=ch[:3]+'o'+ch[4:]
         ch1

Out[20]: 'la oSI à Couteaux'
```

Dans les exemples des bases sur Python, on avait procédé de la même manière avec les méthodes *upper()* et *lower()*.

Méthode très utile, **len(ch)** qui donne le nombre de caractères d'une chaîne :

```
In [21]: ch='la NSI à Couteaux'
         len(ch)

Out[21]: 17
```

1.5 Boucle et chaîne de caractères

Une méthode pratique pour manipuler tous les caractères et de faire une boucle sur cette chaîne avec **for . in** :

```
In [23]: ch='la NSI à Couteaux'
         i=0
         for c in ch:
             print(c,' est à la position ',i)
             i=i+1
```

```
l  est à la position  0
a  est à la position  1
   est à la position  2
N  est à la position  3
S  est à la position  4
I  est à la position  5
   est à la position  6
à  est à la position  7
   est à la position  8
C  est à la position  9
o  est à la position  10
u  est à la position  11
t  est à la position  12
e  est à la position  13
a  est à la position  14
u  est à la position  15
x  est à la position  16
```

On verra que ces méthodes se retrouve sur les listes (et tableaux).