Les chaînes

I. Définitions

Une *chaîne* est un ensemble d'éléments, des *caractères*, encadrés soit par des apostrophes (*simple quotes*), soit par des guillemets (*double quotes*), c="Bonjour". Le nombre d'éléments d'une chaîne est la longueur de la chaîne. Cette longueur est donnée par len(c).

Pour une chaîne de longueur *N*, les indices **valides** varient entre $-N$ et $(N-1)$ au sens large : $-N \leqslant k \leqslant N-1$.
L'utilisation d'un indice non valide provoque une erreur :

> *IndexError: string index out of range*

La chaîne vide est "". Elle ne contient aucun élément et sa longueur est nulle.

II. Indexation

Accès à un élément

On considère une chaîne c de longueur N.

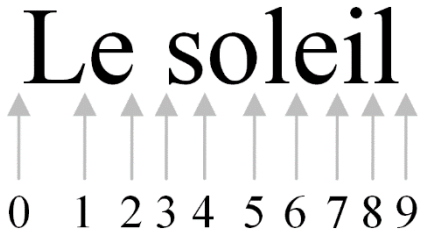
- L'indice du premier élément est nul.
- L'indice du dernier élément est égal à (N − 1).
- L'élément d'indice k est c[k].
- Si l'entier k est strictement négatif,
  alors $0 \leqslant N + k < N$ et la valeur de c[k] est égale à c[N+k].

Une chaîne est une variable *non mutable*. Contrairement aux listes, on ne peut modifier la valeur de chaque élément en lui affectant une nouvelle valeur : l'instruction c="Bonjour" suivie de c[0] = "b" provoquera l'erreur :

> *TypeError: 'str' object does not support item assignment*

Le slicing

Comme pour les chaînes, les indices pointent les emplacements **entre** les caractères comme le montre le schéma ci-dessous. Cette opération est connue sous le nom de tranchage ou saucissonnage (slicing en anglais).



On considère une chaîne c de longueur N et deux indices valides $i \leqslant j$.
On suppose que i et j sont positifs.

- La tranche c[i:j] est la sous-chaîne $(c_k)_{i \leqslant k < j}$
- La tranche c[i:] est la sous-chaîne $(c_k)_{i \leqslant k} = (c_k)_{i \leqslant k < N}$
- La tranche c[:j] est la sous-chaîne $(c_k)_{k < j} = (c_k)_{0 \leqslant k < j}$

III. Les caractères

Définition d'un caractère

Les chaînes de caractères sont des chaînes Unicode, ce qui signifie que les identifiants numériques de leurs caractères sont uniques (il ne peut exister qu'un seul caractère typographique pour chaque code) et universels (les identifiants choisis couvrent la gamme complète de tous les caractères utilisés dans les différentes langues du monde entier). La norme d'encodage Utf-8 est désormais la norme préférentielle pour la plupart des textes courants, parce que :

• d'une part, elle assure une parfaite compatibilité avec les textes encodés en « pur » ASCII (ce qui est le cas de nombreux codes sources de logiciels), ainsi qu'une compatibilité partielle avec les textes encodés à l'aide de ses variantes « étendues », telles que Latin-1 ;

• d'autre part, cette nouvelle norme est celle qui est la plus économe en ressources, tout au moins pour les textes écrits dans une langue occidentale.

Suivant cette norme, les caractères du jeu ASCII standard sont encodés sur un seul octet. Les autres seront encodés en général sur deux octets, parfois trois ou même quatre octets pour les caractères les plus rares.

Accès à un caractère

La fonction *ord*() accepte n'importe quel caractère comme argument. En retour, elle fournit la valeur de l'identifiant numérique correspondant à ce caractère. Ainsi ord('A') renvoie la valeur 65, et ord('Ĩ') renvoie la valeur 296.

La fonction *chr*(num) fait exactement le contraire, en vous présentant le caractère typographique dont on a donné l'identifiant Unicode. Pour que cela fonctionne, il faut cependant que deux conditions soient réalisées:

• la valeur de num doit correspondre effectivement à un caractère existant (la répartition des identifiants unicode n'est pas continue : certains codes ne correspondent donc à aucun caractère) ;

• votre ordinateur doit disposer d'une description graphique du caractère, ou en d'autres termes connaître le dessin de ce caractère, que l'on appelle un *glyphe*. Les systèmes d'exploitation récents disposent cependant de bibliothèques de glyphes très étendues, ce qui devrait vous permettre d'en afficher des milliers à l'écran.

Ainsi, par exemple, chr(65) renvoie le caractère A, et chr(1046) renvoie le caractère cyrillique Ж.

Le caractère spécial « \ » (antislash) permet quelques subtilités complémentaires:
• En premier lieu, il permet d'écrire sur plusieurs lignes une commande qui serait trop longue pour tenir sur une seule (cela vaut pour n'importe quel type de commande).
• A l'intérieur d'une chaîne de caractères, l'antislash permet d'insérer un certain nombre de codes spéciaux (sauts à la ligne, apostrophes, guillemets, etc.).

```
c='N\'est-ce pas ?'
print(c)   #affiche :
        N'est-ce pas ?
d="Sur deux\n lignes"
print(d)   # affiche :
        Sur deux
         Lignes  # remarquez l'espace en début de ligne
```

IV. Opérations et méthodes sur les chaînes

Lien avec les nombres

La fonction *str()* convertit (ou représente) un objet en une chaîne de caractères, cet objet pouvant être une donnée d'à peu près n'importe quel type :

```
a,b=17,['SECONDE' , 7.65]
ch=str(a)+' est un entier et '+str(b)+' est une liste.'
print(ch)
```

permet l'affichage :

```
17 est un entier et ['SECONDE', 7.65] est une liste.
```

On peut convertir en nombre véritable une chaîne de caractères qui représente un nombre. Dans cet exemple, la fonction intégrée int() convertit la chaîne en nombre entier. Il serait également possible de convertir une chaîne de caractères en nombre flottant, à l'aide de la fonction intégrée float().

```
ch='8647'
print(ch+45)
```

provoque l'erreur :

```
TypeError: Can't convert 'int' object to str implicitly
```

Alors que

```
n=int(ch)
print(n+65)
```

donne l'affichage

```
8712
```

Concaténation

On concatène deux chaînes avec l'opérateur + :

```
c1= "Une chaîne de caractères "
c2="plus longue."
print(c1 + c2) # affiche Une chaîne de caractères plus longue.
```

L'opérateur * effectue des concaténations répétées.

```
c = '*'
print(10*c)       # affiche '**********'
```

• Recherche d'un élément : La sous-chaîne x est un élément de la chaîne c si, et seulement si, le booléen **x in c** est égal à True.
• Le nombre d'occurrences de l'objet x dans la chaîne c est donné par c.count(x). Ainsi, l'élément x appartient à la chaîne c si, et seulement si, l'entier c.count(x) est strictement positif.
• L'instruction c.index(x) retourne l'indice de la première occurrence de l'objet x dans la chaîne c ou une erreur ValueError: substring not found si x n'est pas dans la chaîne c.
• Insertion et suppression d'un élément

Nous pouvons retenir que chaîne est un objet *non mutable* c'est-à-dire que la modification n'est pas possible. Par contre, la construction de la chaîne attendue l'est. Un exemple :

```
c='Il doit i avoir une erreur !'
```

Les deux scripts suivants :

```
d=c[:8]+'y'+c[9:]
print(d)
```

et

```
d=''
for i in range(0,8):
   d=d+c[i]
d=d+'y'
for i in range(9,len(c)):
   d=d+c[i]
print(d)
```

donnent l'affichage

```
'Il doit y avoir une erreur !'
```

De nombreuses méthodes sur les chaînes existent. Il suffit de taper dans le shell help(str) pour s'en convaincre !