Les chaînes de caractères en Python

Site Internet : **www.gecif.net**	Type de document : **Fiche pratique**	Intercalaire :	Date :

I – Écriture d'une chaîne de caractères en Python

Une chaîne de caractères est une suite de caractères représentant un message et encadrée par le caractère "double quote". Exemple : **ch="Ceci est une chaîne de caractères"**

La chaîne de caractères peut être également encadrée par des 'simples quotes'. Exemple : **ch='bonjour'**

Si la chaîne de caractères est encadrée par des ***doubles quotes***, elle peut contenir des simples quotes en interne (servant par exemple d'apostrophe). Exemple : **ch="Ici la simple quote sert d'apostrophe"**

Si la chaîne de caractères est encadrée par des ***simples quotes***, elle peut contenir des doubles quotes en interne (servant par exemple de guillemets). Exemple : **ch='Ici la "double quote" sert de guillemets'**

Si on veut à la fois des simples quotes et des doubles quotes dans la chaîne de caractères il y a plusieurs solutions :

- Soit ***échapper*** (faire précéder d'un caractère ***anti-slash***) les caractères encadrant :
  - **ch='pour l\'anti-slash faire "Alt Gr 8" au clavier'**
  - **ch="pour l'anti-slash faire \"Alt Gr 8\" au clavier"**
- Soit ***tripler*** les caractères encadrant (triple simple quote ou triple double quote pour délimiter la chaîne) :
  - **ch='''pour l'anti-slash faire "Alt Gr 8" au clavier'''**
  - **ch="""pour l'anti-slash faire "Alt Gr 8" au clavier"""**

Pour ces 4 cas, un **print(ch)** affichera dans la console la chaîne contenant des simples et des doubles quotes :

pour l'anti-slash faire "Alt Gr 8" au clavier

II – Les instructions Python

Les opérateurs applicables sur une chaîne de caractères	
opérateur	**opération effectuée**
+	concaténation de deux chaînes
*****	multiplication d'une chaîne
in	test d'appartenance à une chaîne
%	formatage d'une chaîne de caractères contenant %d %f ou %s

Exemples :

```
>>> "bon"+"jour"
'bonjour'
>>> "ha "*7
'ha ha ha ha ha ha ha '
>>> "j" in "bonjour"
True
>>> "m" in "bonjour"
False
>>> print("il y a %d livres" % 6)
il y a 6 livres
>>> article="livres"
>>> quantite=17
>>> prix=23.45
>>> print("Il reste %d %s coûtant chacun %f €" % (quantite,article,prix))
Il reste 17 livres coûtant chacun 23.450000 €
>>> print("Il reste %d %s coûtant chacun %.2f €" % (quantite,article,prix))
Il reste 17 livres coûtant chacun 23.45 €
```

Le motif %f affiche par défaut 6 chiffres après la virgule. Dans une chaîne formatée par l'opérateur % :

- le motif **%d** permet d'afficher un nombre **entier**
- le motif **%f** permet d'afficher un nombre **décimal** (%.2f pour afficher 2 chiffres après la virgule)
- le motif **%s** permet d'afficher une **chaîne de caractères**

Les fonctions externes applicables à une chaîne de caractères	
fonction	**opération effectuée**
len	renvoie la longueur de la chaîne (nombre de caractères)
str	convertit un objet en chaîne de caractères
list	convertit une chaîne de caractères en liste
sorted	renvoie une liste avec les caractères rangés dans l'ordre croissant

Exemples :

```
>>> len("bonjour")
7
>>> str(123)
'123'
>>> list("bonjour")
['b', 'o', 'n', 'j', 'o', 'u', 'r']
>>> sorted("bonjour")
['b', 'j', 'n', 'o', 'o', 'r', 'u']
```

Les méthodes propres à une chaîne de caractères	
méthode	**opération effectuée**
upper	convertit toute la chaîne en lettres majuscules
capitalize	met en majuscule la première lettre de la chaîne
title	met en majuscule la première lettre de chaque mot de la chaîne
swapcase	bascule la casse (minuscule/majuscule) de chaque caractère de la chaîne
count	compte le nombre de sous-chaîne présents dans la chaîne
find	renvoie la position d'une sous-chaîne (renvoie -1 si non trouvé)
index	renvoie la position d'un caractère (erreur si non trouvé)
replace	remplace une sous-chaîne par une autre
split	découpe la chaîne selon un séparateur (renvoie une liste)
join	utilise la chaîne comme séparateur pour joindre les éléments d'une liste (renvoie une chaîne)
isxxxxx	permet de tester si la chaîne est dans un format précis (numérique, alphabétique, etc.)

Pour obtenir l'ensemble des méthodes applicables à une chaîne de caractères on pourra taper **dir("chaine")** dans une console Python : toutes les méthodes utilisables sont alors renvoyées dans une liste.

s[n] permet d'atteindre le caractère de rang n dans la chaîne s : **s[0]** pour le premier et **s[-1]** pour le dernier.
s[m:n] permet d'extraire de la chaîne s la sous-chaîne allant de **s[m]** (compris) à **s[n]** (non compris).

Exemples :

```
>>> "le soleil brille".upper()
'LE SOLEIL BRILLE'
>>> "le soleil brille".capitalize()
'Le soleil brille'
>>> "le soleil brille".title()
'Le Soleil Brille'
>>> "Python".swapcase()
'pYTHON'
>>> "le soleil brille".count("il")
2
>>> "le soleil brille".find("bri")
10
>>> "le soleil brille".index("s")
3
>>> "le soleil brille".replace("il","O")
'le soleO brOle'
>>> "le soleil brille".split(" ")
['le', 'soleil', 'brille']
>>> '#'.join(['A','B','C'])
'A#B#C'
>>> '+'.join('123456')
'1+2+3+4+5+6'
>>> " ".join(["il","fait","beau"])
'il fait beau'
```

```
>>> "123".isnumeric()
True
>>> "bonjour".isalpha()
True
>>> "56euros".isalpha()
False
>>> "56euros".isnumeric()
False
>>> "56euros".isalnum()
True
>>> "56 euros".isalnum()
False
>>> s="bonjour"
>>> s[0]
'b'
>>> s[1]
'o'
>>> s[-1]
'r'
>>> s[-2]
'u'
>>> s[2:5]
'njo'
>>> s[-6:-2]
'onjo'
```