Chaines de caractères

◇ Le type chaine de caractères (*string*) permet de représenter des suites de caractères, c'est à dire du texte. Manipuler du texte est très fréquent en informatique : affichage des réponses à l'écran, traitement des informations entrées au clavier et, souvent, les données textuelles constituent un format d'échange commun entre deux programmes.

```
>>> chaine1 = "Langage"
>>> chaine2 = "Python"
>>> chaine1[1]
'a'
>>> chaine1[-1]
'e'
>>> chaine1[1:5]
'anga'
>>> chaine1*2
'LangageLangage'
>>> chaine1+chaine2
'LangagePython'
>>> len(chaine1)
7
```

⚠ On retrouve des constructions semblables aux listes, cependant les chaines ne sont pas *modifiables* : on ne peut par exemple pas écrire `chaine[3:5]=...` par contre on peut écrire :

```
>>> ch = "Un texte"
>>> ch[3:]
'texte'
>>> ch = ch[0:3]+"essai"
>>> print(ch)
Un essai
```

⚠ Il ne faut pas confondre la chaine de caractère `"1"` avec le nombre entier 1. Il est par contre possible de réaliser des conversions : on retiendra que la fonction **`str`** permet de convertir un objet en chaine de caractères et les fonctions **`float`** et **`int`** permettent de convertir une chaine de caractères en nombre flottant ou entier (il existe d'autres types de conversions).

```
>>> a = 1e-3
>>> type(a)
<class 'float'>
>>> s = str(a)
>>> type(s)
<class 'str'>
>>> s
'0.001'
```

```
>>> s = "12.5"
>>> type(s)
<class 'str'>
>>> x = float(s)
>>> type(x)
<class 'float'>
>>> print(x)
12.5
```

Exemple Python. Les chaines de caractères interviennent typiquement lorsqu'il faut demander des informations à l'utilisateur par l'intermédaire du clavier. Le programme suivant demande un nombre (entier) et affiche le carré de ce nombre :

```
x = int(input("Entrez un nombre : "))
print("Le carré de ce nombre est",x**2)
```

Noter que le programme produira une erreur si on entre, par exemple, un nombre flottant tel que 1.2.

Exemple Python. Un algorithme (naïf) de recherche d'un texte donné à l'intérieur d'une chaine. On considère une chaine de caractères *txt* ainsi qu'un motif *mot* (c'est également une chaine de caractères). On veut déterminer l'ensemble des indices i pour lesquels on retrouve *mot* dans *txt* à la position i.

```
def positions(txt,mot):
    """
    Construit la liste des indices i t.q.
    mot se retrouve dans txt à la position i
    """
    Pos = []
    for i in range(0,len(txt)-len(mot)+1):
        nb_egaux = 0
        for j in range(0,len(mot)):
            if mot[j]==txt[i+j]:
                nb_egaux = nb_egaux+1
        if nb_egaux==len(mot):
            Pos.append(i)
    return Pos
txt = "1101001011101"
mot = "101"
Pos = positions(txt,mot)
print(mot,"apparait dans",txt,"aux positions",Pos)
```

```
101 apparait dans 1101001011101 aux positions [1, 6, 10]
```

Complexité : si n est la longueur de *txt* et ℓ celle de *mot*, alors le temps d'exécution de **`positions(txt,mot)`** est $O(n\ell)$.

Exercices pour la semaine prochaine

Question 1. On dispose d'une variable **x** de type *float*. Écrire les instructions permettant de définir la variable **s** de type *str* qui contient le texte **`"la valeur de x est ..."`** où `...` est remplacé par la valeur exacte de **x**. Par exemple si **x** vaut 2, alors **s** doit être égale à **`"la valeur de x est 2"`**.

Question 2. On dispose de deux variables **x** et **y** de type *str* contenant des nombres. Écrire le code permettant d'afficher la somme de ces deux nombres. Par exemple si **`x="23"`** et **`y="0.5"`**, le code doit afficher 23.5.

Question 3. Écrire une fonction **`compte_espaces(s)`** qui compte combien de fois le caractère `' '` (espace) apparait dans la chaine de caractères **s**. Par exemple si **`s='a b c'`** alors **`compte_espaces(s)`** doit renvoyer 2.

Corrections