TP3 - Chaînes de caractères

Jusqu'ici on a manipulé différents types d'objets dans nos programmes
— des nombres entiers
— des nombres à virgule, qu'on appelle en informatique des *flottants*
— des booléens
— des listes
— des fonctions

Ajoutons le type *chaîne de caractères*, pour que nos programmes puissent manipuler des textes.

Exercice 1. Voici quelques opérations, à exécuter dans le shell pour faire connaissance.
— Création d'une chaîne de caractères : écrire des caractères entre des guillemets ou apostrophes. Exemple : `a="bon"`, ou `b='jour'`.
— Concaténation (mise bout-à-bout) de chaînes de caractères avec l'opérateur «+» pour en construire une nouvelle. Exemple : `c=a+b`, ou `d=a+" "+b`
— Longueur d'une chaîne de caractères, avec la fonction `len` : `n=len(d)`
— Obtention du caractère à une position donnée dans une chaîne de caractères, comme pour une liste : `a[2]`. Le résultat de cette opération est une chaîne de longueur 1.
— Le caractère «retour à la ligne» s'écrit «`\n`» dans une chaîne de caractères. Exemple : exécuter `print("a\nb")`

Construction de chaîne de caractères. Un objet de type chaîne de caractères ne peut pas être rallongé, ni raccourci, ni modifié. Pour construire une chaîne de caractères petit à petit, on ne peut pas utiliser `append` comme pour une liste, on doit procéder par concaténations successives, comme dans l'exercice ci-dessous

Exercice 2. Voici la définition d'une fonction qui reçoit en argument une chaîne de caractères `c`.

```
def mystere(c):
    n=len(c)
    resultat=""
    for i in range(n):
        resultat=resultat+c[n-1-i]
    return resultat
```

Que fait cette fonction ? .....

Exercice 3. Définir une fonction `souschaine(c,deb,fin)` qui renvoie la sous-chaîne de `c` composée de ses caractères entre les position `debut` et `fin` inclus.
Exemple : `souschaine("Préparationnaire",1,10)` renvoie `"réparation"`.

Exercice 4. Définir une fonction `indices_espaces(c)` qui renvoie les indices des caractères de `c` qui sont égaux à `" "`. Exemple : `indices_espaces("Le chat est gris")` renvoie la liste `[2,7,11]`

Exercice 5. Définir une fonction `retire_espaces(s)` qui reçoit en argument une chaîne de caractères et renvoie la chaîne obtenue en lui retirant ses espaces.

Exercice 6. Définir une fonction qui reçoit en argument une chaîne de caractères et détermine si cette chaîne est un palindrome (un texte qui se lit dans les deux sens, comme `"elle a mon nom a elle"`. On renverra un booléen (c'est-à-dire `True` ou `False`).

Exercice 7. Définir une fonction qui reçoit en argument une chaîne de caractères et détermine si cette chaîne est un palindrome si l'on ignore les espaces (comme `"un roc si biscornu"`.).

Pour les plus rapides. Exercice 8. Écrire une fonction qui prend en entrée une chaîne de caractères `s`, comme : `"Bonjour Python"`, et qui renvoie une chaîne de caractères de trois lignes (utiliser le caractère `"\n"` pour aller à la ligne) où la chaîne `s` est écrite en zig-zag. Exemple (ici on a utilisé des points pour rendre visibles les espaces, mais vous utiliserez des espaces) :

```
B.......o.......P.......o
..o...j...u... ...y...h...n
....n.......r.......t
```

La chaîne à renvoyer sera donc :
`"B.......o.......P.......o\n..o...j...u... ...y...h...n\n....n.......r.......t"`
(Lors de vos tests, pour afficher une chaîne de caractères «joliment», notamment avec des retours à la ligne effectifs, il faut utiliser la fonction `print`)

Exercice 9. Écrire une fonction analogue à la précédente, qui renvoie cette fois la chaîne formée des caractères de `s` dans l'ordre où ils apparaissent dans l'écriture en zig-zag de `s`. Dans l'exemple ci-dessus on retournera `"BoPooju yhnnrt"`
On peut imaginer utiliser cette technique pour fabriquer un message codé

Exercice 10. Écrire la fonction de décodage, qui prend en argument une chaîne de caractères codée avec la technique ci-dessus (exemple `"v zmsosae opiuvcr"`) et renvoie le message d'origine.