TP 7 : Slicing

2 BCPST 2 - TP Informatique

CONSEIL : à chaque nouvelle question faites une nouvelle cellule en tapant "#%%" .
Vous pourrez ainsi traiter toutes les questions dans un même fichier.

Objectif du TP :
- Savoir faire du slicing : découper une chaine de caractères afin d'obtenir une liste.

Dans ce TP on propose plusieurs exercices basés sur le slicing, c'est à dire la séparation d'une chaîne de caractères en morceaux différents, basés sur des caractères de séparation.

On rappelle qu'on peut parcourir une chaîne de caractère exactement comme on parcoure une liste.

I Exercice 1 : Une liste de mots

Un premier exemple de slicing va consister à partir d'une chaîne de caractères de la forme `s = "arbre,maison,jardin,crabe"` et de construire la liste `["arbre","maison","jardin",crabe]`.

Pour obtenir cela, le principe est le suivant :
- On initialise une variable `m` qui va représenter les mots de `s`.
- On initialise la liste de mots `L`.
- On parcourt caractère par caractère la chaîne `s`.
- Si on lit sur une lettre normale, on l'ajoute dans le mot `m`.
- Si on lit le caractère séparateur (`","` dans l'exemple), alors le mot `m` est terminé, on l'ajoute dans la liste `L` et on le réinitialise.
- A la fin, attention souvent le dernier mot n'est pas ajouté à la liste car il n'y a pas de caractère de séparation final, n'oublions pas de l'ajouter à `L`.
- On retourne `L`

Q.1. En suivant la méthode expliquée ci dessus, coder la fonction `slicing(s)` qui prend en entrée une chaine de caractères `s` qui contient des mots séparés par des virgules et qui retourne la liste des mots de `s`.

Q.2. On propose une petite variation, cette fois on considère que la liste contient des nombres séparés par des virgules : `s = "10,456,789,12099"` et on veut créer la liste `L = [10,456,789,12099]`.
Coder la fonction python `liste_nombres(s)` qui prend en entrée à chaîne de caractères `s` et qui renvoie la liste de nombres associée.

Remarque : Attention, la liste `L` doit vraiment contenir des nombres (`int`) et pas des chaînes de caractères qui représentent des nombres.

II Exercice 2 : coder une calculatrice

Ces questions sont inspirées de sujets oraux de concours.

Q.3. Niveau 1 : On considère des chaînes de caractère de la forme `s = "12+485+790+1"`, elle ne contient que des additions..
Ecrire une fonction `calcul1(s)` qui prend en entrée une chaîne de caractère `s` sous cette forme et qui renvoie le résultat.

Exemple :

```
print( calcul1("15+125+1") )
-----------------------------------
-> 141
```

Q.4. Niveau 2 : On ajoute les soustractions : cette fois `s = "12-14+7+58-1"`.
Ecrire une fonction `calcul2(s)` qui prend en entrée une chaîne de caractère `s` sous cette forme et qui renvoie le résultat.

Q.5. Niveau 3 : On ajoute les multiplications : cette fois `s = "12*2+5-7+58-1"`.
Ecrire une fonction `calcul3(s)` qui prend en entrée une chaîne de caractère `s` sous cette forme et qui renvoie le résultat.

Astuce : Attention, les multiplications sont prioritaires ! On doit les calculer en premier. On peut procéder en deux étapes : calculer d'abord les multiplications et reconstituer la chaîne obtenue pour la traiter ensuite comme précédemment.
Par exemple : `"2*8+5*2-9"` est transformée en `16+10-9` qu'on peut calculer avec `calcul2`.

Q.6. Niveau 4 : On passe au niveau au-dessus en ajoutant les puissances : cette fois on peut avoir `s = "12*3+5-7**2-1"`.

Ecrire une fonction `calcul4(s)` qui prend en entrée une chaîne de caractère `s` sous cette forme et qui renvoie le résultat.

Remarque : Attention les puissances ne sont pas associatives, et donc $2 * *3 * *2$ n'est pas forcément clair. Dans les calculatrices, on considère que l'ordre qui prime est par la droite à savoir $2 * *(3 * *2)$.

Astuce : L'ordre de priorité est : d'abord les puissances, puis les multiplications. On peut donc raisonner en plusieurs étapes comme pour la question précédente. Ceci étant dit, les puissances sont composées de deux caractères `"**"` ce qui les rend particulières à gérer !