Projets de développement informatique : Labo 2

Expression régulière

1 Contexte

Le but de ce labo consiste à vous familiariser avec les expressions régulières, outil puissant qui vous permet notamment de vérifier si des chaines de caractères satisfont des motifs ou non, mais aussi d'extraire des parties de chaines de caractères selon un motif donné.

2 Exercices

Maitriser les expressions régulières est très important pour tout développeur. Elles sont en effet largement utilisée pour valider et traiter des données textuelles. Afin de vous y habituer, commençons avec une série de simples exercices :

1. Écrivez un programme qui demande à l'utilisateur d'encoder une chaine de caractères et qui vérifie si cette dernière satisfait exactement le motif suivant :
   (a) Un numéro de téléphone de la forme `+xx xxx xx xx xx` où les `x` sont tous des chiffres.
   (b) Un nombre entier, qui commence éventuellement avec un signe + ou − devant, et sans espaces aucun.
   (c) Une plaque d'immatriculation qui peut prendre l'un des deux formes `XLLLDDD` ou `XDDDLLL`, où `L` est une lettre et `D` est un chiffre, et `X` est un chiffre optionnel (entre 1 et 9).
   (d) Le nom d'un volume sous windows qui est, pour rappel, de la forme `C:\` (une lettre majuscule suivie de deux-points et backslash).

2. Écrivez un programme qui ouvre un fichier texte et qui retrouve tous les nombres qui y apparaissent. Le programme produit en sortie un inventaire de ces nombres, séparés par des virgules et avec le numéro de ligne où ils apparaissent. Voici un exemple d'exécution :

```
& python3 labo2-q2.py monfichier.txt
Line 3: 829, -12, 88
Line 12: 99
Line 28: +15
```

   Astuce : Pour lire le paramètre de la ligne de commande, importez le module `sys` et utilisez la liste `sys.args` qui va contenir tous ces paramètres.

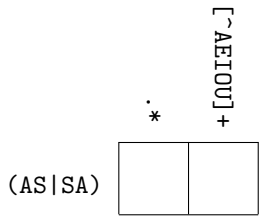
3. Écrivez un programme qui décomposer une URL en ses trois parties : le protocole, le nom de domaine de la machine et le chemin d'accès de la ressource. Faites deux versions de programme : la première n'utilise que les méthodes de la classe `str` et la seconde exploite les groupes capturants des expressions régulières. Voici un exemple d'exécution :

```
& python3 labo2-q3.py http://www.this.is/big/shit
Protocol: http
Domain: www.this.is
Path: big/shit
```

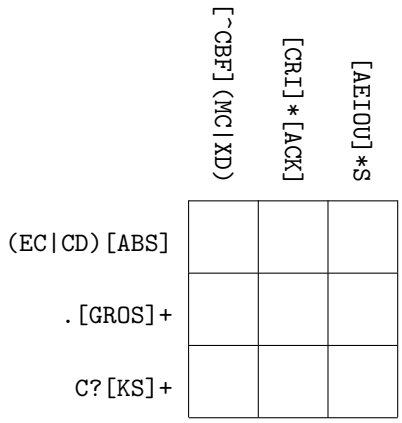
   Astuce : N'oubliez pas de penser au cas où le chemin d'accès de la ressource est vide.

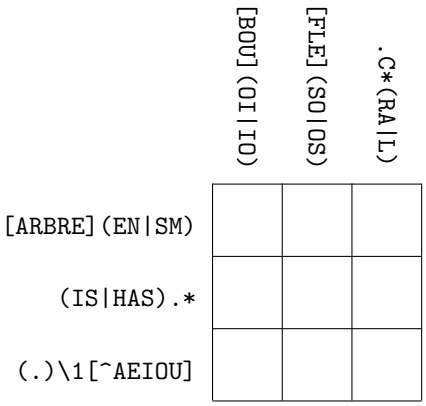
3 Mots-croisés

Plusieurs jeux utilisant des expressions régulières ont été mis au point et peuvent être utilisés pour renforcer ses connaissances. Par exemple, le site https://regexcrossword.com propose des mots-croisés dont les définitions sont des expressions régulières et dont les mots à trouver sont des chaines de caractères qui satisfont les expressions régulières données. Pour mieux comprendre ce concept, prenons un exemple simple :



La définition du mot horizontal indique que la chaine est soit `AS`, soit `SA`. La définition du premier mot vertical indique qu'on prend une chaine composée de n'importe quels caractères. Enfin, la définition du deuxième mot vertical indique qu'on prend une chaine composée d'au moins une fois n'importe quel caractère sauf les lettres majuscules `A`, `E`, `I`, `O` et `U`. On trouve très facilement que la réponse ne peut être que `AS`. À vous de jouer, voici deux mots-croisés à résoudre :





Afin de vérifier si votre solution est correcte, on va écrire une fonction de vérification en Python. On va se limiter aux grilles rectangulaires. La fonction prend trois paramètres : la liste des expressions régulières des lignes (de haut en bas), la liste des expressions régulières des colonnes (de gauche à droite) et la solution proposée sous la forme d'une liste de chaines de caractères (une chaine par ligne). La fonction renvoie un booléen comme résultat, selon que la solution proposée est valide ou non.

Le squelette de la fonction à écrire est :

```
1 def checkregexcrossword(linesregex, columnsregex, answer):
2     pass
```