Langages de script (Python)
CMTP n° 5 : Expressions rationnelles

Dans ce sujet nous aborderons les expressions rationnelles (ou expressions régulières, voire (*regular expressions* en anglais, ou *regex*, ou *RE* en abrégé) en Python, notamment avec le module `re`, fournissant des outils très puissants pour l'analyse et la manipulation des chaînes de caractères.
Voir https://docs.python.org/fr/3/howto/regex.html#regex-howto pour une introduction aux expressions régulières en Python et https://docs.python.org/fr/3/library/re.html#module-re pour la documentation du module `re`.

Exercice 1 : Échauffement

Pour chaque motif donné par une expression régulière ci-dessous, dire quelles chaînes correspondent *exactement* au motif, et, si ce n'est pas le cas, dire si elles contient une sous-chaîne correspondant au motif et quelle sous-chaîne sera détectée par `re.search`.

Motif	Chaîne
`A.C`	`"abc"`
	`"AZZC"`
	`"AC"`
	`"ABC"`

Motif	Chaîne
`[ABC]`	`"A"`
	`"B"`
	`"AC"`
	`"ABC"`

Motif	Chaîne
`[a-k].*[1-5]`	`"a2"`
	`"A05"`
	`"ab12"`
	`"k1k"`

Motif	Chaîne
`[^a-k].*[1-5]`	`"a2"`
	`"A05"`
	`"ab12"`
	`"k1k"`

Motif	Chaîne
`[a-k].[1-5]{,3}`	`"ak1234"`
	`"a123"`
	`"ad12bd123"`
	`"z1k"`

Motif	Chaîne
`(a?b)+`	`"abababab"`
	`"aaaab"`
	`"bbbbba"`
	`"bbbab"`

Motif	Chaîne
`(a|[AB]*)[0-9]`	`"aAB0"`
	`"AAB0"`
	`"0"`
	`"A-Z9"`

Motif	Chaîne
`.\^[^^]+.`	`a^0^0`
	`"aa^0^"`
	`"0^^^0"`
	`"A^Z9"`

Utiliser `re.fullmatch` et `re.search` pour vérifier vos réponses. Attention : les expressions régulières utilisent la barre inverse \ comme caractère spécial, comme par exemple dans le motif `.\^[^^]+.` ci-dessus. Ceci implique que dans la représentation d'un motif en tant que chaîne de caractères chaque \ doit être doublé. Pour éviter la prolifération de barres obliques inverses vous pouvez utiliser les *chaînes brutes* de Python, en préfixant une chaîne avec `r` (pour *raw*, brut en anglais). Par exemple le motif `.\^[^^]+.` est représenté dans Python par la chaîne `".\\^[^^]+."`, comme par la chaîne brute `r".\^[^^]+."`.

Exercice 2 : Les prénoms en France 1

Pour cet exercice et le prochain, nous utiliserons le fichier `prenoms.txt` que vous trouverez sur Moodle. Ce fichier provient des données *open data* sur les prénoms en France du 1905 au 2015, disponible également sur le site http://www.data.gouv.fr/. Comme le fichier est assez gros, veillez à optimiser le temps de réponse de vos programmes pour que celui-ci soit raisonnable.
Chaque ligne de `prenoms.txt` contient une chaîne de la forme : `year,number,idn,name` (sans espace après les virgules), où :

- `year` est un nombre de 4 chiffres donnant une année,
- `number` est le nombre des nouveaux nés en `year` ayant eu le nom `name`,
- `idn` est un entier donnant un identificateur numérique à `name`,
- `name` est une chaîne quelconque non vide.

1. Écrire une fonction `parse_line(s)` qui prend en entrée une chaîne `s` de la forme `year,number,idn,name` et retourne un 4-uplet de la forme `(year, number, idn, name)`. Vous devez utiliser uniquement la fonction `re.match` ainsi que la fonction `str.strip` pour supprimer de `name` les blancs (`"\t"`., `"\n"`., ...) non désirés. (*Astuce : n'hésitez pas à utiliser les références aux groupes délimités pas des parenthèses dans une expression rationnelle.*)
2. Écrire une fonction `parse_file(path)` qui prend en entrée un nom de fichier texte et qui, pour l'ensemble des lignes du fichier qui suivent le format de la question précédente, retourne un dictionnaire associant à chaque couple `(name,year)` le nombre de fois où le prénom a été choisi pendant cette année.

En plus, la fonction doit afficher les lignes du fichier qui ne correspondent pas au format décrit ci-dessus (il devrait y en avoir 6 dans le fichier `prenoms.txt`).

Attention : le fichier `prenoms.txt` peut contenir plusieurs lignes avec exactement le même couple `(name,year)` !

3. Écrire une fonction `name_frequency(d)` qui prend en entrée un dictionnaire dont les clefs sont des paires `(name,year)` et les valeurs sont le nombre de fois où le prénom a été choisi cette année, et retourne un dictionnaire dont les clefs sont les prénoms et la valeur associée est le nombre total de fois où le prénom a été choisi, sur toutes les années.
4. Écrire un script `popular_name.py` qui prend en entrée un nom de fichier contenant une liste des prénoms sous le format décrit ci-dessus et un entier `n` et affiche les `n` noms les plus populaires dans le fichier, ensemble au nombre total de fois où le prénom a été choisi.

   Par exemple sur le fichier `prenoms.txt` et l'entier `5` le script devrait afficher :

```
Marie 2290210
Jean 1962089
Pierre 885734
Michel 818404
Andre 710477
```

Exercice 3 : Les prénoms en France 2

Écrire un script pour répondre aux questions suivantes qui portent sur les données du fichier `prenoms.txt` :

1. Combien de prénoms sont des carrés (c'est à dire. des chaînes de caractères de la forme `rr` pour une certaine sous-chaîne `r`) ? La réponse devrait être 18.

   (*Astuce : écrire une fonction auxiliaire* `is_square` *en utilisant les groupes sur les expressions régulières. N'oubliez pas que la première lettre est majuscule en* `prenoms.txt`.)
2. Combien de prénoms contiennent un carré de longueur au minimum 4 ? La réponse devrait être 63.
3. Combien de prénoms contiennent au moins 5 fois la même lettre ? La réponse devrait être 3.
4. Combien de prénoms contiennent au moins 4 voyelles consécutives ? La réponse devrait être 25.

   (*Astuce : rappelez vous du modificateur* `,nm,n` *qui fait valider par l'expression rationnelle résultante entre* `m` *et* `n` *répétitions de l'expression qui précède. En particulier on peut omettre une de deux bornes* `m, n`.)
5. Combien de prénoms contiennent 4 consonnes consécutives ? La réponse devrait être 7.

   (*Attention aux apostrophes, espaces et traits d'union dans le prénoms*).
6. Combien de prénoms finissent par 4 consonnes consécutives ? La réponse devrait être 1.

   (*Astuce : rappelez vous du modificateur* `$` *qui valide la fin d'une chaîne de caractères, ou juste avant le fin de ligne.*)
7. Quelle est la proportion de prénoms composés (contenant un espace ou un trait d'union) ? La réponse devrait être 0.077 (arrondie au millième).
8. Quelle proportion de la *population* née entre 1905 et 2015 a un prénom composé ? La réponse devrait être 0.030 (arrondi au millième).
9. Pour chaque lettre, quel est le prénom le plus populaire qui commence par cette lettre ?

Exercice 4 : Star Wars

Nous voulons récupérer le script de l'Episode IV de Star Wars : "*A new Hope*". Nous avons trouvé une page web `Star-Wars-A-New-Hope.html` contenant le texte du script et nous voulons l'extraire dans un format textuel plus approprié.
Nous traitons ici un fichier `html` en utilisant les fonctionnalités du module `re`, notamment les méthodes `re.search` et `re.sub`, pour un but pédagogique, mais en général il y a des modules dédiées au traitement du `html` en Python.

1. Télécharger depuis Moodle le fichier `Star-Wars-A-New-Hope.html`, ouvrir le code source avec un éditeur de texte. Quelles sont les balises qui délimitent le texte du script du film ?

2. Écrire un script qui crée un nouveau fichier texte contenant que le script du film et éventuellement les balises `<b>` et `</b>`.

(*Astuce : l'option* `DOTALL` *fait en sorte que* `.` *corresponde à n'importe quel caractère, caractère de retour à la ligne inclus. Par exemple* `re.search("0(.*)0", s, re.DOTALL)` *cherche une chaîne de caractères dans* `s` *délimitée par le chiffre* `0` *et contenant éventuellement des retours à la ligne.*)

3. Quels sont les rôles des chaînes de caractères délimitées par la balise `<b>` ? Comment reconnaître le nom d'un personnage qui commence un dialogue par rapport aux autres chaînes délimités par `<b>` ? Comment reconnaître la fin d'un dialogue d'un personnage ?

Modifier le script afin de générer un fichier texte tel que :

- un dialogue soit introduit par le nom du personnage parlant en majuscule suivit par deux-points et ensuite le dialogue délimité par les guillemets, comme par exemple :
  `BEN'S VOICE: "Luke, the Force will be with you."`
- aucune balise `<b>`, `</b>` n'y apparaît plus ;
- tout le texte doit être aligné à gauche, aucune nouvelle ligne peut commencer avec des espaces vides, ni il faut avoir deux ou plus espaces vides consécutives.

(*Astuce : Rappelez vous que les répétitions telles que* `*` *sont gloutonnes (ils rendent la chaîne la plus longue compatible avec le motif), pour y modifier le comportement afin qu'elles rendent la sous-chaîne la plus courte possible, ajoutez les modificateur* `?`*, comme par exemple dans* `*?`).