Les notions abordées

Au cours de ce TD vous allez revoir des éléments de base mais surtout, aborder une notion nouvelle :

- Les REGEX (expressions régulières).

Note *: Pas de Python pour cet « exo ». Vous pouvez aller sur « https://regex101.com/ ».*

Exercice 1 : divers motifs REGEX…

Donnez les motif REGEX qui permettent d'effectuer les recherches suivantes :

A. Prénoms ou noms composés. L'élément discriminateur est le « - » entre les prénoms.

Hypothèses :

- Le 1er prénom débute par une majuscule NON accentuée suivie de minuscules (éventuellement).
- Le ou les prénoms suivants peuvent commencés par une majuscule **ou pas**…
- Le prénom composé peut être constitué de 2 ou 3 prénoms. Pas plus !

échappement

Indication *: Le prénom composé doit former un « mot » et doit donc être placés entre deux* `\b`.

Le motif pour le 1er prénom est : `[A-Z][a-zéèêëàâäôöûüÿç]*`. *Les prénoms suivant sont similaire sauf que la 1ère majuscule est facultative :* « `?` »…

Motif REGEX : .....

Échantillons de test :

```
José-Luis, Anne-Sophie, Louis Alexandre, Elise-Marie-Cécile,
Marie-claude_Anne, pierre-albert, Gaëtan-Pierre
```

B. Des nombres dans un texte.

Hypothèses :

- Les nombres sont entiers ou à virgules. Ils peuvent être représentés par une fraction de 2 entiers.
- La « virgule » peut être : « , » ou « . » et pour la fraction, on accepte le « / ».

Indication *: Les nombres doivent former un « mot » et doivent donc être placés entre deux* `\b`.

Pour une fraction ou un nombre à virgule le séparateur peut être représenté par le motif : `(\.|,|\/)?` *ou alors* `[\.,\/]?`

Cela signifie : un « whitespace » OU un tiret OU un point et cela 0 ou 1 fois !!!

Motif REGEX : .....

Échantillon de test :

```
Mon numéro favori est 42.5, mais j'aime aussi le 7,3, le 128.60,
le 31415 et 1/3 sont aussi dans la liste...
```

C. Un numéro de téléphone français.

Hypothèses :

- Les numéros de téléphones (mobiles, fixes et spéciaux) commencent par « 01 » à « 09 ».
- On se limite au format NON international : « 0XXXXXXXXX ».
- Les chiffres peuvent êtres contigus ou séparés : « 0123456789 » ou « 01 23 45 67 89 ».
- Les séparateurs acceptés sont l'espace, le « . » ou le « - »…

Indication *: pour les séparateurs, on peut utiliser le motif :* `(\s|-|\.)?` *ou bien* `[\s\-\.]?`

Cela signifie : un « whitespace » OU un tiret OU un point et cela 0 ou 1 fois !!!

Motif REGEX : .....

Échantillon de test :

02-99-88-77-66	06.47.74.47.74
0536638558	0808080808
+33 9 23 45 67 89	03 89466482
+330432233223	0800 777 999
0810-556677	0820 00 11 22

Exercice 2 : nettoyage d'un fichier…

On souhaite avoir une liste de tous les « *verbes Powershell* ». Dans le terminal de Pycharm, on peut lancer « *pwsh* » et taper : « **Get-Verb** ». On obtient une longue liste illustrée par cet extrait.

Verb	AliasPrefix	Group	Description
Add	a	Common	Adds a resource to a container, or attaches an item to another item
Clear	cl	Common	Removes all the resources but does not delete the container
Close	cs	Common	Changes the state of a resource to make it inaccessible or unusable
Copy	cp	Common	Copies a resource to another name or to another container

On souhaite générer une liste qui ne contient que les verbes sans aucune autre information :

Add, Clear, Close, Copy, Enter, Exit, Find, Format, Get, Hide, Join, Lock, Move, New, Open, Optimize, Push, Pop, Redo, Remove, Rename, Reset, Resize, Search, Select, Set, Show, Skip, Split, Step, Switch, Undo, …

Pour cela, il est demandé de suivre les étapes suivantes :

1. Générez un fichier « `verbes.txt` » en utilisant la CLI « ***get-verb*** » combinée avec une redirection : « **`get-verb > verb.txt`** » dans l'interface « *pwsh* » du Terminal de « *Pycharm* ».
2. Éditez le fichier « `verb.txt` » dans « *Pycharm* » et supprimez à la main le lignes 1 à 3…
   La 1ère ligne qui reste doit commencer par « **Add** » !
3. Codez dans le script « `extract_verb.py` » ce qui est nécessaire pour :
   - Ouvrir le fichier « `verb.txt` ».
   - Lire l'intégralité de ce fichier.
   - Définir le motif REGEX en utilisant https://regex101.com/ pour le valider sur 4-5 lignes d'échantillons extraites du fichier « `verb.txt` ».
   - Remplacer avec ce motif tout ce qui suit chaque verbe (OEL inclus) par une virgule puis un espace.
   - Afficher le résultat.

Remarque : Vous pouvez directement effectuer ces remplacements sous Pycharm en faisant :

1. **Ctrl R**.
2. *Activer les Expressions Régulières en cliquant sur :*
3. *Taper le motif REGEX dans le ligne de recherche.*
4. *Mettre « , » dans la ligne de remplacement…*
5. *Cliquez « **Remplace All** »*

Un espace ICI…

Activité complémentaire

Suite de l'exercice 1 :

D. Un numéro de téléphone français : cas général.
On souhaite compléter le motif élaboré dans le paragraphe 'C.' de l'activité précédente avec la représentation internationale des numéros ainsi que les numéros mnémotechniques…
Hypothèses :
- Le format peut être international ou pas : « +33 » ou « 0 »
- Les numéros à mémorisation facile : « 0800 400 400 » ou « +33 800 520 520 ».

Motif REGEX : ……

……

……

……

Échantillon de test :

02-99-88-77-66	06.47.74.47.74
0536638558	0808080808
+331.2345.6789	03 89466482
+330432233223	+33 9 23 45 67 89
0800 777 999	0810-556677
0820 00 11 22	+33 820 710 710

Indication *: Codez trois nouveaux motifs REGEX indépendants pour respectivement : « **+33.n.nn.nn.nn.nn** » , « **0nnn.nnn.nnn** » et « **+33.nnn.nnn.nnn** » avec possibilité de mettre à la place des « . », des espaces, des « - » ou rien…*

Effectuez ensuite un OU logique entre eux en incluant aussi le motif trouvé dans le paragraphe 'C.'…

E. Un adresse mail.
Hypothèses :
- Format : *nom_utilisateur***@***domaine***.***extension*
- Le « *nom_utilisateur* » peut contenir des lettres minuscules & majuscules, des chiffres et les caractères : « `!#$%&'*+=?^_.{|}~-/` ».
- Le « *nom_utilisateur* » **ne peut pas** commencer ou finir par un « . ». De plus, il ne peut à aucun endroit y avoir 2 « . » consécutifs.
- Le « *domaine* » peut contenir des lettres minuscules & majuscules, des chiffres et les caractères : « . » et « - ». il doit comprendre au moins 4 caractères.
- Le « *domaine* » **ne peut pas** commencer ou finir par un « . ». De plus, il ne peut à aucun endroit y avoir 2 « . » consécutifs.
- L'« *extension* » peut contenir des lettres minuscules & majuscules uniquement et 2 au minimum.

Note *: Pour détecter la présence de 2 « . » consécutifs en REGEX, il faut mettre en œuvre des « **groupes non capturants** » qui n'ont pas été abordés en cours…*
On ne va donc pas détecter les « . . » dans les adresses mail !!!

> ***Indication*** *: Le « nom_utilisateur » doit commencer et se terminer par un caractère :* `[\w!#$%&'*+=?^{|}~\-\/]` *(pas de « . »). Entre ce 1er et dernier caractère, il doit y avoir un nombre de caractères quelconque (de 0 à n) :* `[\w!#$%&'*+=?^{|}~\-\/.]` *(avec un « . »)… Même raisonnement pour le « domaine ». L'« extension » est encore plus simple !*

Motif REGEX : .....

.....

Échantillon de test :

```
olivier.eckle@uha.fr
Abc@example.com
Abc@10.42.0.1
user+mailbox/department=shipping@example.com
Abc@def@example.com
Abc.123@example.com
Fred Bloggs@example.com
Joe.\\Blow@example.com
!#$%&'*+-/=?^_.{|}~@example.com
```

F. Une adresse IP V4.

Hypothèses :

- Adresse valide pour une machine ou une passerelle mais pas pour un masque...
- Format : « nnn.nnn.nnn.nnn » où : 000 < « nnn » < 255

> ***Indication*** *: Le motif pour « nnn » doit envisager les cas suivants :*
>
> **25***(0 à 5)*, **2***(0 à 4)(0 à 9)*, **1***(0 à 9)(0 à 9)*, **0***(0 à 9)(0 à 9)*
>
> facultatif

Motif REGEX : .....

.....

.....

.....

Échantillon de test :

100.010.001.000	100.10.1.0
255.244.233.222	256.0.0.0
999.999.999.999	192.168.0.254

> ***Indication*** *: Codez le motifs REGEX pour « nnn » et dupliquez-le avec « . » entre ceux-ci…*

G. Un masque IP V4.

Hypothèses :

- Adresse de masque valide…

> ***Rappel*** *: Si l'on un masque en binaire (ex. :* 11111111.11111111.11111111.00000000*), Pour être valide, il ne peux pas y avoir de « mélange » de 1 et 0… Tous les 1 doivent forcément être sur la gauche des 32 bits tandis que tous les 0 doivent être sur la droite…*

– Format : « nnn.nnn.nnn.nnn » où : « nnn » est une valeur forcément dans la liste :
[254, 252, 248, 240, 224, 192, 128, 0]
Sachant qu'à droite de cette valeur, on a forcément 0 et à gauche, forcément 255…

Indication *: Le motif pour « nnn.nnn.nnn.nnn » doit envisager les cas suivants :*
*255.255.255.***val***, 255.255.***val***. 000, 255.***val***.000.000, ou encore* **val***.000.000.000 !!!*

facultatifs facultatifs facultatifs

Motif REGEX : .....

.....

.....

.....

Échantillon de test :

255.0.000.0	248.000.0.0
255.192.0.00	128.0.00.000
255.255.240.0	256.255.255.0
255.255.255.128	255.248.248.0
255.255.128.1	128.0.0.0