

LS4_200314

Expressions régulières (= rationnelles)

Suite de caracteres qui définit un ensemble de mots.

La plus part des symboles (caractères) ne représentent qu'eux memes. Les autres symboles sont appelés metacaractères ou caractères spéciaux.

Les caracteres spéciaux

- '.' représente n'importe quel caractère excepté le caractère retour à la ligne ('\n')
Exemple :
'a' : représente les suites de deux caractères, formées d'un caractère quelconque (excepté \n) suivi d'un 'a'.
- '\d' qui représente les caractères numériques
Exemple : 'a\d' : {'a0','a1','a2',..., 'a9'}
- '\w' qui représente les caractères alphanumériques ('a'...'z','A'...'Z','0'...'9')
- '\s' qui représente les caractères d'espacements
- '\D', '\W', '\S' sont les complémentaires respectifs de '\d', '\w' et '\s'
- '[' : represente l'un des symboles situés entre les crochets
Remarque : Entre [] les spécificités perte leur rôles, sauf '^'
Exemple :
[abc] = {'a','b','c'}
[db] = {'\','d','b'}

Exemple :

On veut tous les mots formés de 'a' ou de 'b' : {a,b}*

{a,b}* = [ab]*

- '[^]' : représente la négation d'un des symboles
Exemple :
[^ab] : n'importe quel caractère qui soit ni 'a', ni 'b'
- [a-t]: n'importe quel caractere de 'a' à 't' {'a','b',..., 't'}
- [a-t0-3]: n'importe quel caractere de 'a' à 't' ou de '0' à '3'
- '|' symbolise le 'ou'
Exemple :

$\backslash d / b' : \{'O', '1', \dots, '9', 'b'\}$

- '\$' signifie la fin de la chaîne

Exemple :

'a+\$': cherche une chaîne de caractère qui termine par n'importe quelle suite de a

- '^' : signifie le début de chaîne

Exemple :

'^bb' : chaîne qui commencent par 'bb'

- '\b' : signifie le début ou la fin d'un mot

Exemple :

'\bc\wlc\b' : les mots commençant et terminant par c

'\bc\w*c' : les mots commençant par c*

-Caractères de répétition

- '*' représente la répétition de **0** à "l'infini" du caractère ou du groupe qui le **précède**

Exemple :

'a*' : { \emptyset , 'a', 'aa', 'aaa', ...}

- '+' représente la répétition de **1** à "l'infini" du caractère ou du groupe qui le **précède**

- '?' répétition de 0 à 1 du caractère ou groupe qui le représente (absence ou présence du caractère qui le précède)

- {m,n} : placé après un caractère signifie que le caractère est répété au moins m fois et au plus n fois.

Remarque :

{0,} = correspond à ''

{1,} = correspond à '+'

{,3} = répétition au plus de 3 fois Exemple :

'a{2,4} : {'aa', 'aaa', 'aaaa'}

Groupe

() : définit un groupe, sont numérotés dans l'ordre d'apparition (lecture gauche droite). Et peuvent être appelés par leur numéro précédé du symbole '\

Exemple :

'(\d)\w*1' : chaînes qui commencent par un chiffre c, suivi de n'importe quelle suite de caractères alphanumériques, suivi de c.

On peut échapper un caractère spécial en mettant un '\' devant.

Exemple :

'\d' = suite de 2 caractères : '\' puis 'd'

Les méthodes

Elles se trouvent dans le module 're'.

```
import re

re.match(expr_reg, chaine [,flag]) :
```

- si il existe un **préfixe** de la chaîne contenu dans l'expression régulière passée en paramètre retourne objet de type `expr_reg`
- sinon return 'None'

Exemple :

```
>>>re.match('ba','banane')
retourne un objet car 'ba' préfixe de banane

>>>re.match('na','banane')
None

>>>re.match('ba*c+', 'baaacca')
le préfixe retenu est 'baaacc'
```

Remarque :

Les caractères de répétitions induisent un comportement 'glouton'

Pour ne pas avoir ce comportement il faut alors ajouter '?' à la suite du caractère de répétition

```
>>>re.match('ba*c+?', 'baaacc')
le préfixe retenu est 'baaac'
```

Pour obtenir le préfixe retenu, on a la méthode `group()` qui s'applique à l'objet retourné par la méthode `match`

```
>>>o = re.match('ba*c+', 'baaacca')
>>>o.group()
'baaacc'
```

S'il y a des groupes dans l'expression :

```
>>> o = re.match('a(\d\d)[A-Z](\d\d)', 'a71C98ttt')
>>> o.group(0)
'a71C98'
>>> o.group(1)
'71'
>>> o.group(2)
'98'
```

Flags

`re.IGNORECASE == re.I` ignore la casse, ignore la casse du deuxième paramètre

```
>>>re.match('ab','Ab', re.I).group()
'Ab'
```

`re.MULTILINE = re.M` permet que le '^' signifie début de ligne et le '\$' signifie la fin de ligne (utile pour le `search()`)

`re.DOTALL = re.s` : permet que le '.' signifie n'importe quel caractère (y compris '\n')

`re.search(expr_reg, chaine, [,flags])` : recherche la première sous chaîne maximale qui est incluse dans l'expression régulière. Retourne objet de type expression régulière si succès, sinon `None`

`raw`

```
>>>re.match(r'a\n')
lis littéralement a \ n ne prend pas en compte le retour de ligne
```