Expresiones regulares

Es una notación para expresar cadenas de texto. Se usa para especificar patrones de búsqueda y de substitución.

Tienen un formato bastante estandarizado en diversas aplicacioens, entre ellas, grep, MS Word, Word Perfect, perl, PHP, etc.

En python están implementadas en el módulo **re**.

Ejemplos:

`'hoy'`	Cualquier cadena que contenga 'hoy'
`'ali'`	Cualquier cadena que contenga 'ali' (no necesariamente una palabra; por ejemplo 'salida')
`'[Hh]oy'`	'Hoy' o 'hoy'
`'[abc]'`	'a' o 'b' o 'c'
`'[0123456789]'`	cualquier dígito
`'[A-Z]'`	cualquier letra entre A y Z (las mayúsculas)
`'[0-5]'`	cualquier dígito entre 0 y 5
`'[^a-z]'`	cualquier carácter excepto las minúsculas (^ = negación)
`'^The'`	cualquier cadena que comience por 'The' (^ = inicio de cadena)
`'paz$'`	cualquier cadena que acabe por 'paz' ($ = final de cadena)
`'^abc$'`	la cadena 'abc' (sin ningún otro carácter antes o después)
`'\bcasa'`	una palabra que empice por 'casa' (\b = límite de palabra)
`'baa*'`	una cadena que comience por 'ba' y vaya seguida de *n* caracteres 'a' (donde $n \geq 0$) (por ej. 'basta', 'baal', 'baaa', etc.)
`'c[aeiou]*'`	la cadena 'c' seguida por una serie (potencialmente) vacía de vocales
`'a(bc)*'`	una cadena que contenga 'a' seguido de una serie potencialmente nula de 'bc's: 'a', 'abc', 'abcbc', etc.
`'c[aeiou]+'`	la cadena 'c' seguida por una serie vacía de vocales (al menos ha de haber una)
`'casas?'`	'casa' o 'casas' (? indica opcionalidad)
`'a?b+$'`	una cadena que acabe en una 'a' opcional seguida por una o más 'b's
`'d{2}'`	'dd'
`'d{2,4}'`	entre dos y cuatro 'd's
`'d{3,}'`	al menos tres 'd's
`'el|la'`	'el' o 'la'
`'(b|cd)ef'`	una cadena que contenga 'bef' o 'cdef'
`'cit(y|ies)'`	una cadena que contenga 'city' o 'cities'
`'(a|b)*c'`	una cadena que tiene una secuencia de 'a's o de 'b's acabada en una 'c'.
`'c.t'`	una cadena que contenga 'c' seguida de un caracter y una 't' (el punto significa cualquier cara´cter)
`'a.[0-9]'`	una cadena que tenga una 'a' seguida de otro carácter y un número.
`'^.{3}$'`	una cadena de tres caracteres exactamente
`'^[a-zA-Z]'`	una cadena que empiece por una letra (no un número)
`'%[^a-zA-Z]'`	una cadena con un caracter que no es una letra entre dos signos de porcentaje
`'(\$|¥)[0-9]+'`	una cadena que contiene el signo del dolar o del yen, seguido por al menos un dígito (por ej. '$35') (el signo de escape '\' obliga a interpretar '$' como el signo del dolar')
`'^\w+$'`	una cadena formada exclusivamente por caracteres alfanuméricos (\w = cualquier carácter alfanumérico o '_')
`'casa\s'`	una cadena que contenga 'casa' seguido por un espacio (\s = un espacio, un tabulador o un salto de párrafo)
`'casa\S'`	una cadena que contenga 'casa' seguida de otro carácter que no sea un espacio (\S = cualquier carácter que no es un espacio en blanco, tabulador o salto de párrafo)
`'(?<=c)t'`	una cadena que contenga 't' precedida de 'c' (?<= ... contexto previo)
`'Ana (?=\sMaria)'`	una cadena que contenga 'Ana' seguido de un espacio y de 'María' (?= .... contexto siguiente)
`'Anu(?!bis)'`	Una cadena que contenga 'Anu', pero que no vaya seguido de 'bis' (= Anubis') (?! ... no está en el contexto siguiente)
`r'(\b\w+)\s+\1'`	La cadena contiene una palabra repetida

En python es conveniente pre-compilar las expresiones regulares para obtener mayor eficiencia en su uso:

```
patron = re.compile('c[ei]')     # 'c' seguida de 'e' o 'i'
```

Funciones útiles:

match	Determina si la expresión regular se corresponde con el principio de una cadena.
search	Busca una parte de una cadena que se corresponda con la expresión regular
split	Divide la cadena en una lista, usando como separador la expresión regular
sub	en una cadena, substituye por otra cadena todos los casos de coincidencia de una expresión regular
subn	igual que sub(), pero permite limitar el número de substituciones

Tras aplicar una función de coincidencia:

group	Devuelve la cadena coincidente con la expresión
start	Indica la posición inicial de la coincidencia
end	Indica la posición final de la coincidencia
span	Indica las posiciones inicial y final de la coincidencia

Ejemplos:

```
>>> re.match('c[aei]', 'casa') ==> OK
>>> re.match('c(asa|osa)','cosaco') ==> OK

>>> re.search('casas?','la casa blanca') ==> OK
>>> m = re.search('casas?','la casa blanca')
>>> m.span() ==> (3,7)

>>> re.split('/','Alemania/Berlin/Bayern Munich') ==>
['Alemania','Berlin','Bayern Munich']

>>> re.sub('ro','ra','caro') ==> 'cara'
>>> re.sub('(?<=c)t','z','pacto') ==> 'paczo'
>>> re.sub(' 'Ana(?=\sMaria) ','Eva','Ana Maria') => 'Eva Maria')

>>> re.match('Anu(?!bis)','Anular') ==> OK
>>> re.sub('Anu(?!bis)','imo','Anular') ==> 'imolar'

>>> frase = 'el coche está en la la calle'
>>> p = re.search(r'(\b\w+)\s+\1',frase)
>>> p.span() ==> (17, 22)
>>> p.group() ==> 'la la'
```