What are regular expresions:

Las expresiones regulares sirven para extraer data especifica de algún archivo o de alguna colección de información basta, o de la que no se conocen los valores exactos, encontrando patrones similares entre los valores, y no enfocándose en el valor exacto como tal, lo que nos da bastante margen para encontrar los valores.

El módulo re:

Para poder utilizar las regular expresions, debemos importar el módulo re, de la librería de Python. Una vez importado podremos utilizar todos los métodos del módulo.

Métodos del módulo re:

re.search(r”aza”,”plaza”, re.IGNORECASE): Este metodo se utiliza para encontrar un resultado específico, exactamente igual dentro de una cadena. En este caso, le pasamos una r, delante del valor que estamos buscando para decirle a Python que el string que estamos buscando no contiene caracteres especiales, y luego le pasamos la cadena en la que deseamos buscar dicho valor, o bien una variable igualada a dicha cadena o archivo. El tercer parámetro con opciones adicionales que podemos agregar a la búsqueda, como ignorar las mayúsculas, por ejemplo.

re.findall(r”cat|dog”,”I like both cats and dogs”): Al contrario de search que solamente devuelva uno de los resultados aunque le pasemos el carácter especial |, la función re.findall nos devuelve ambos. Esta función devolverá una lista con los valores encontrados.

Caracteres especiales:

^:si se introduce delante de la cadena de búsqueda devolverá todas las que comiencen con dicha cadena.

$: si se introduce detrás de la cadena de búsqueda devolverá todas las que terminen con dicha cadena.

.: si se introduce en la cadena de búsqueda sustituye el valor en el que se coloco por cualquier valor, devolviéndonos todas las cadenas que contengan la cadena, con cualquier letra dentro de donde colocamos el punto.

.\*: si se introduce en la cadena de búsqueda sustituye el valor en el que se colocó por cualquier valor, devolviéndonos todas las cadenas que contengan la cadena, con cualquier letra dentro de donde colocamos el punto y con cualquier cantidad de letras incluyendo cero, entre la letra que ese encuentra delante del punto y detrás del asterisco.

El uso de estos caracteres especiales tiene un defecto, que se en coding es lo que se llama greeding. Lo que ocurre es que en vez de enfocarse en una sola palabra, este uso de caracteres aplica para toda la cadena, así que si esto ocurre entre la primera palabra y la ultima de la cadena, esta búsqueda devolverá toda la cadena, con todos los strings que contiene. Esto ocurre porque se le ha pasado un punto delante, lo que nos lleva al siguiente uso de caracteres especiales .

[a-z]\*: Esta combinación de caracteres especiales tiene el mismo efecto que el .\*, solo que en este caso, como estamos especificando que se debe de tratar de una letra ente la a y la z, excluirá los espacios, así que no saltara a la siguiente palabra, resolviendo el problema que causa el uso de .\*.

[a-z]: si se introduce un rango en la cadena de búsqueda, o las letras aceptadas para esta búsqueda en específico, devolverá todas las que contengan los elementos de esta búsqueda. Esto no aplica para espacios en blanco, solo devolverá las que contengan los elementos del rango dado, o de las letras posibles dadas.

[^a-z]: Todo lo que esta introducido en los corchetes, y tiene delante el símbolo ^, se excluirá de la búsqueda en vez de excluirlo, o sea, devolverá todos los valores que coincidan excepto los introducidos en el corchete.

|: re.search(“cat|dog”,”The cat.”): Esta es la manera de buscar dos palabras, o de dar alternativas a la búsqueda, sino encuentra la primera, irá a por la segunda, pero solamente devolverá el primer resultado con el que se encuentra, si queremos que nos devuelva todos los posibles resultados debemos usar la función findall.

a+: El carácter plus se usa para decirle la función que puede buscar cualquier cantidad de esa letra que le hemos pasado, esto lo podemos combinar con otras letras, por lo que nos devolverá la que coincida con las que tengan al menos una letra. Este carácter no significa que puede haber algo entre una letra y la otra, a no ser que usemos la wildcart “.”.

?: El carácter question mark, significa que la letra que le pasemos delante es opcional, puede estar o no.

**NOTA:** De ser necesario encontrar uno de los caracteres especiales, por ejemplo un punto, para que concuerde por ejemplo con un correo electrónico que tiene una extensión .com, solamente debemos poner el carácter \.com delante y este evitara los caracteres especiales.

Backslashes cracters:

\w: Representa una palabra completa

\s: Representa espacios, tabulaciones y nuevas líneas.

\d: Representa dígitos, números.

\b: Representa Word Boundries.

Capturing groupes:

Una vez que le hemos pasado la función o el metodo re.search a un argumento, el resultado de este podemos observar en la siguiente línea que será la ubicación del match object obtenido, o sea que a lo que le pasemos esta función nos devolverá un objeto del tipo match, del modulo re. Este objeto es una tupla con los elementos dentro de la cadena

Referencias: [www.regex.com](http://www.regex.com)