La clave para este tipo de operaciones son las expresiones regulares (regex). ¿No sabes lo que son las expresiones regulares? Vamos a ver qué dice Wikipedia sobre las expresiones regulares. Se define expresión regular como:

Una secuencia de caracteres que forma un patrón de búsqueda, principalmente utilizada para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones. Por ejemplo, el grupo formado por las cadenas Handel, Händel y Haendel se describe con el patrón "H(a|ä|ae)ndel". La mayoría de las formalizaciones proporcionan los siguientes constructores: una expresión regular es una forma de representar a los lenguajes regulares (finitos o infinitos) y se construye utilizando caracteres del alfabeto sobre el cual se define el lenguaje

Otra gran definición de expresión regular la aporta [regular-expressions.info](http://www.regular-expressions.info/):

Una expresión regular es una cadena de caracteres especial que define una búsqueda de patrones. Puedes pensar en las expresiones regulares como un tipo de comodín con esteroides. Seguramente estés familiarizado con este tipo de “comodines” como cuando utilizas .txt para buscar archivos de texto en un gestor de archivos. El equivalente a eso convertido en una expresión regular sería.

Sé que el concepto de expresión regular puede sonar un poco confuso. Por ello, vamos a ver unos cuantos ejemplos de expresiones regulares para entenderlas un poco mejor.

Ejemplos de expresiones regulares

En esta sección analizaremos unos cuantos ejemplos de expresiones regulares para entenderlas un poco mejor. Digamos que tienes esta expresión regular:

/abder/

Con esto simplemente estaremos indicando que queremos buscar el texto explícito abder.

¿Y esta expresión regular?

/a[nr]t/

Puedes interpretar esta expresión regular como lo siguiente: busca un patrón el cual su primera letra sea la a y su última letra sea la t, y entre dichas letras contenga la letra n o la r. Por lo tanto, casos de éxito para este patrón, serían las palabras inglesas ant y art.

Permíteme un pequeño test llegados a este punto. ¿Cómo escribirías una expresión regular para buscar palabras que empiecen con ca y terminen con alguno de estos caracteres tbr? Exacto, sería algo así:

/ca[tbr]/

Es posible que ya hayas visto expresiones regulares que comienzan con un acento circunflejo, esto significa que buscan una cadena que comience con el string mencionado después del ^. Por la tanto, con una expresión como la siguiente, estaríamos buscando un string que comience con This.

/^This/

Por lo tanto, con el texto

My name is Abder  
This is Abder  
This is Tom

Y basándonos en la expresión regular anterior /^This/, las cadenas resultantes serían:

This is Abder  
This is Tom

¿Qué tenemos que hacer si queremos encontrar una cadena que termine por algún carácter en especial? En ese caso, utilizamos el símbolo $. Aquí tienes un ejemplo:

Abder$

Por lo tanto, utilizando el texto anterior (el de las tres líneas), obtendríamos estas cadenas resultantes:

My name is Abder  
This is Abder

Muy bien, ¿qué piensas de esta expresión?

^[A-Z][a-z]

Sé que parece complejo al principio, pero vamos a analizarlo parte por parte.

Como ya hemos visto antes, el acento circunflejo ^ busca una cadena que comience por el string mencionado después. [A-Z] hace referencia a las mayúsculas. Por lo que, si leemos esta parte de la expresión regular: ^[A-Z], estamos indicando que buscamos una cadena que empiece por mayúsculas. La última parte, [a-z] hace referencia a que queremos encontrar cadenas que empiecen por mayúsculas y que luego sigan con minúsculas.

Entonces, ¿cuál de las siguientes cadenas coincidiría con la expresión regular de antes? Si no estás seguro, puedes usar Python, tal y como veremos en la siguiente sección.

abder  
Abder  
ABDER  
ABder

Las expresiones regulares son un tema muy amplio, y estos ejemplos son sólo para proporcionarte una ligera idea de qué son y por qué las usamos.

Expresiones regulares en Phyton

Aquí viene la parte divertida. Vamos a ver cómo trabajar en Python con expresiones regulares. El módulo que utilizaremos para hacerlas funcionar es el módulo [re](https://docs.python.org/3/library/re.html#module-re).

El primer ejemplo fue encontrar la palabra Abder. En Python, lo haríamos del siguiente modo:

import re  
text = 'My name is Abder'  
match\_pattern = re.match(r'Abder', text)  
print match\_pattern

Si ejecutas el script de Python de arriba, obtendrás: None!

El script funciona perfectamente, el problema viene en cómo funciona la función match(). Si nos vamos a la documentación del módulo re, veremos que lo que la función match() hace es:

Si cero o más caracteres al comienzo de la cadena coinciden con el patrón de la expresión regular, devolverá el objeto coincidente correspondiente. Si devuelve none es que la cadena no coincide con el patrón; ten en cuenta que esto es diferente para una coincidencia de longitud cero.

Ahá, podemos ver que match() solo devuelve un resultado si se encuentra la coincidencia AL PRINCIPIO del string.

No te preocupes, podemos utilizar la función search(), que es, basándose en la documentación:

Escanea una cadena en busca de la primera ubicación en la que el patrón de la expresión regular produce una coincidencia, y devuelve el objeto coincidente correspondiente. Si devuelve none es que ninguna posición en la cadena coincide con el patrón. Ten en cuenta que esto es diferente para una coincidencia de longitud cero.

Si cambiamos el script anterior, utilizando search() en lugar de match(), obtendremos algo así como:

\_sre.SRE\_Match object at 0x101cfc988

Así es, obtendremos un objeto match.

Si lo que queremos es que nos proporcione el resultado en forma de string, tenemos que utilizar la función group(). Si quieres ver la coincidencia completa, utiliza group(0). Tal que así:

print match\_pattern.group(0)