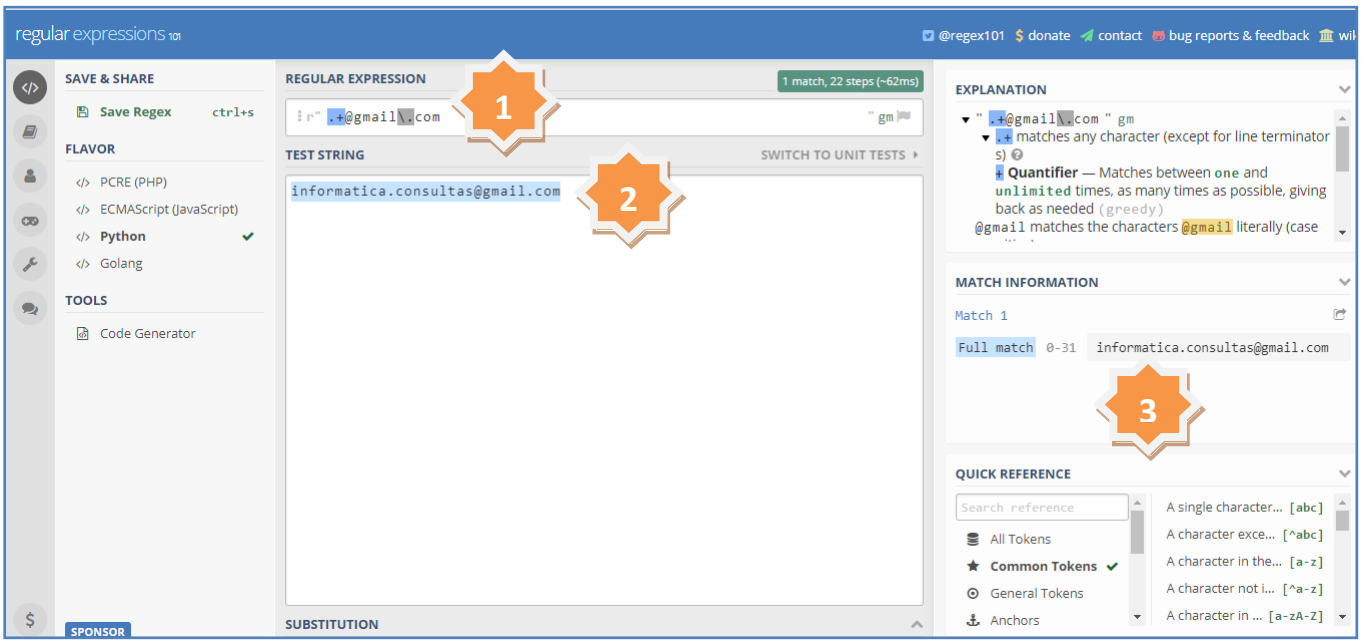
Cómo usar Regex101.com para construir expresiones regulares

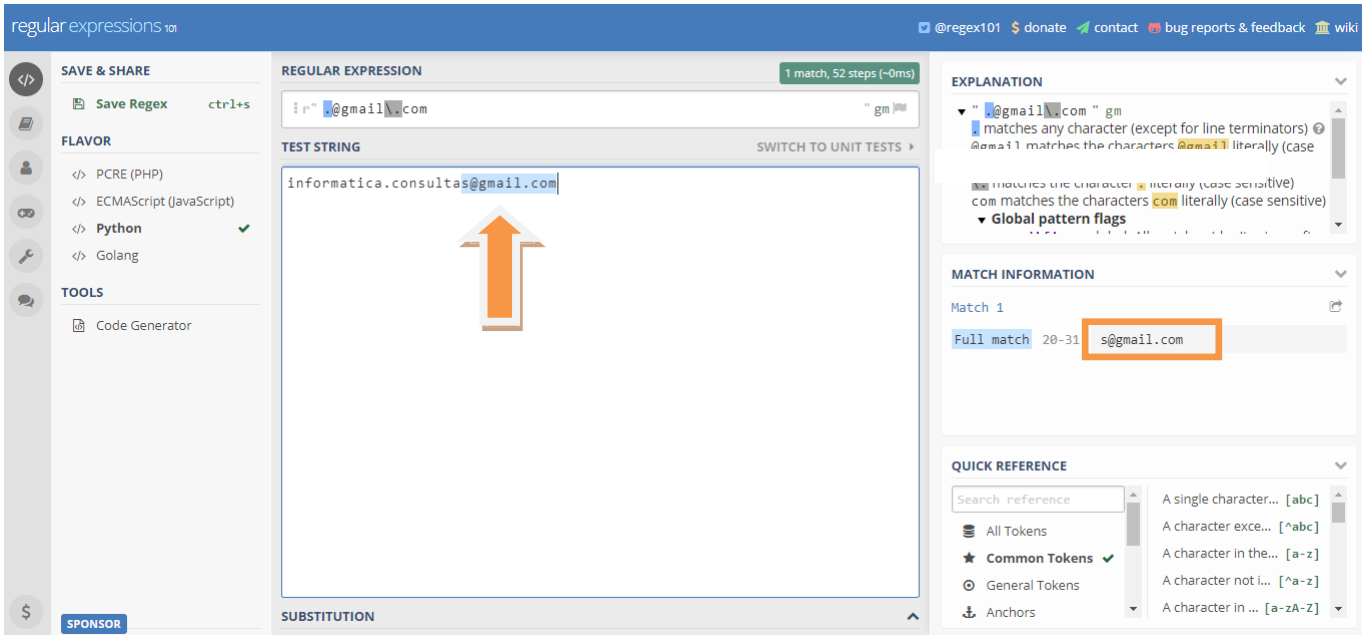
Mas información de expresiones regulares en Python: https://github.com/python/cpython/blob/3.8/Lib/re.py

Lo que veremos funciona en cualquier lenguaje. Entonces, para empezar con la validación de un correo la expresión regular sería algo como: .+@gmail.com. Probamos:



1. Se escribe la expresión regular,
2. Se escribe el texto en donde vamos a realizar la búsqueda,
3. Se emite el resultado, como vemos en el ejemplo, encontró que en el texto de búsqueda sí existe un correo electrónico con el dominio.

Cómo es que funciona esta expresión regular? Primero utilice un punto . significa que espera cualquier carácter (letras, números o símbolos) pero un solo . espera un solo carácter. Cómo se vería la expresión regular si solo usamos el . sin el signo + del paso anterior?:

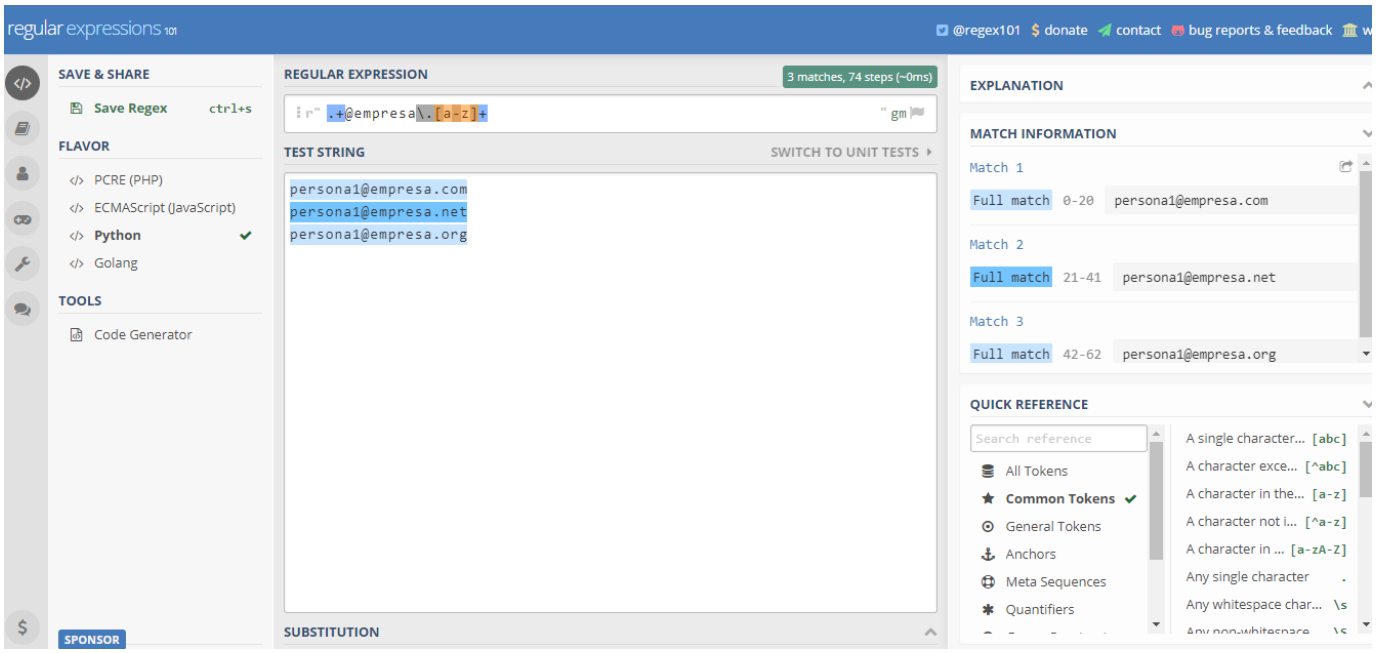


Como se muestra, solo encontró un carácter (la letra **s** que es la que está junto a la **@**), entonces el signo + le dice a la expresión regular que el patrón que esta antes que él (a la izquierda) puede repetirse más de una vez.

Luego está un texto que es el dominio que buscamos gmail.com, pero aquí hay un problema, el dominio lleva un . y como vimos antes el . le dice que representa cualquier carácter, entonces tomaría como válidos otros dominios, para evitar esto

se usa un carácter de escape \ antes del . esto hace que el . o cualquier otra comando de expresión regular sea ignorado y tratado como texto.

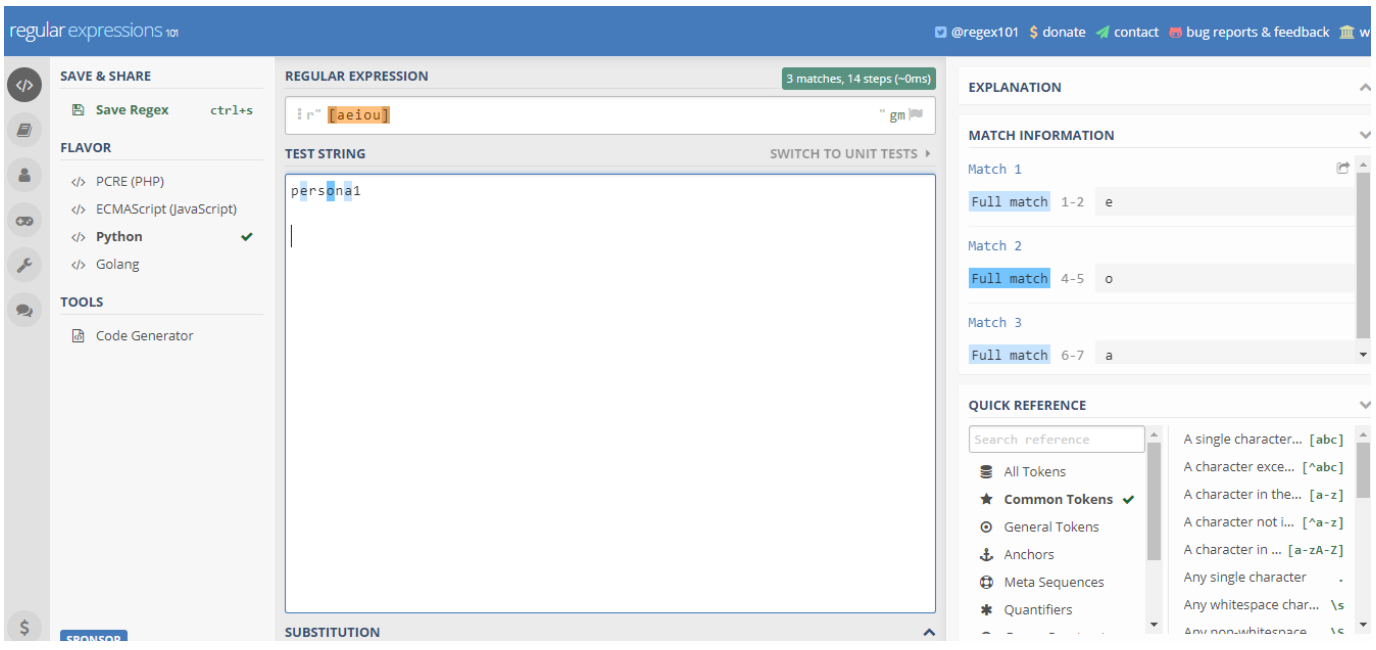
Vamos a ajustar un poco este código para que pueda encontrar un email con el dominio empresa pero en versiones .com .net .org etc. Cambiemos la regex por esta .+@empresa\.[a-z]+



Con una sola expresión regular encontramos el dominio empresa en las versiones .com .net y .org. El cambio fue sustituir **com** por [a-z] los signos [y] se usan para definir un rango, en este caso le dice que puede ir cualquier carácter de la ***a*** a la ***z*** y el signo + al final le dice que ese patrón se puede repetir más de una vez.

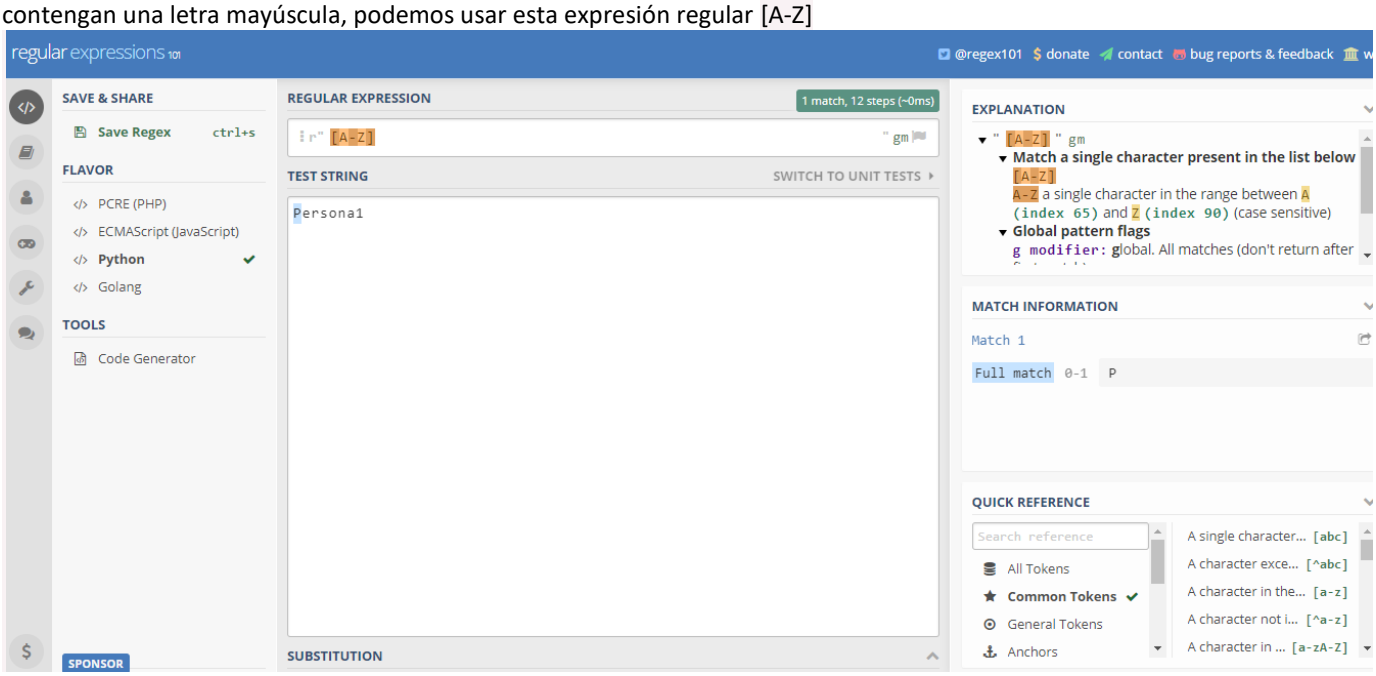
Caracteres, grupos de caracteres y rangos

Las expresiones regulares pueden contener palabras, grupos o rangos, por ejemplo una regex pueden usarse para buscar una palabra específica pero también podemos definir que contenga un rango de caracteres o grupo usando los signos [y], por ejemplo la expresión regular [aeiou] puede usarse para saber si el texto contiene una vocal.

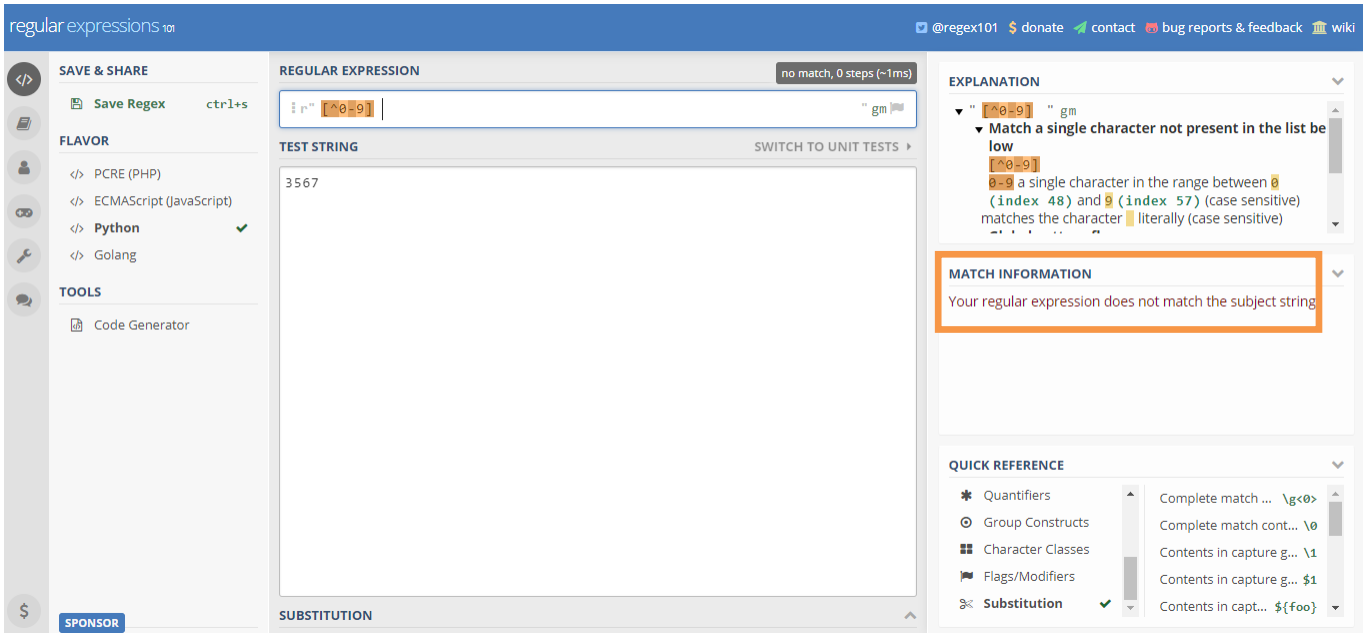


Como vemos la expresión regular encontró las tres vocales en la palabra **Persona1**, y detectó que la palabra contiene vocales. También hay que notar que cada letra [aeiou] se considera de forma independiente y no como una palabra.

Si lo que necesitamos es buscar un rango debemos separar el rango por un - por ejemplo para buscar palabras que contengan una letra mayúscula, podemos usar esta expresión regular [A-Z]



También podemos «negar» el rango buscando el resultado opuesto con el caracter ^ al inicio del rango, por ejemplo la expresión [^0-9] intenta buscar caracteres que no sean números. Al aplicar la expresión a un texto con números, entonces no encuentra ninguna coincidencia.



Cuantificadores de expresiones regulares

Estos se colocan después de un rango, texto o meta-caracter para modificar o definir la cantidad de veces que debe repetirse o encontrarse. Aquí está la lista de ellos:

- ? Coincide 0 ó 1 vez.
- Coincide 0 ó más veces
- + Coincide 1 ó más veces
- {n} Coincide exactamente n veces, donde n es un número entero.
- {n,} Coincide al menos n veces, donde n es un número entero.
- {,m} Coincide un máximo de m veces, donde m es un número entero.
- {n,m} Coincide exactamente de n a m veces, donde n y m son números entero.

Meta-caracteres

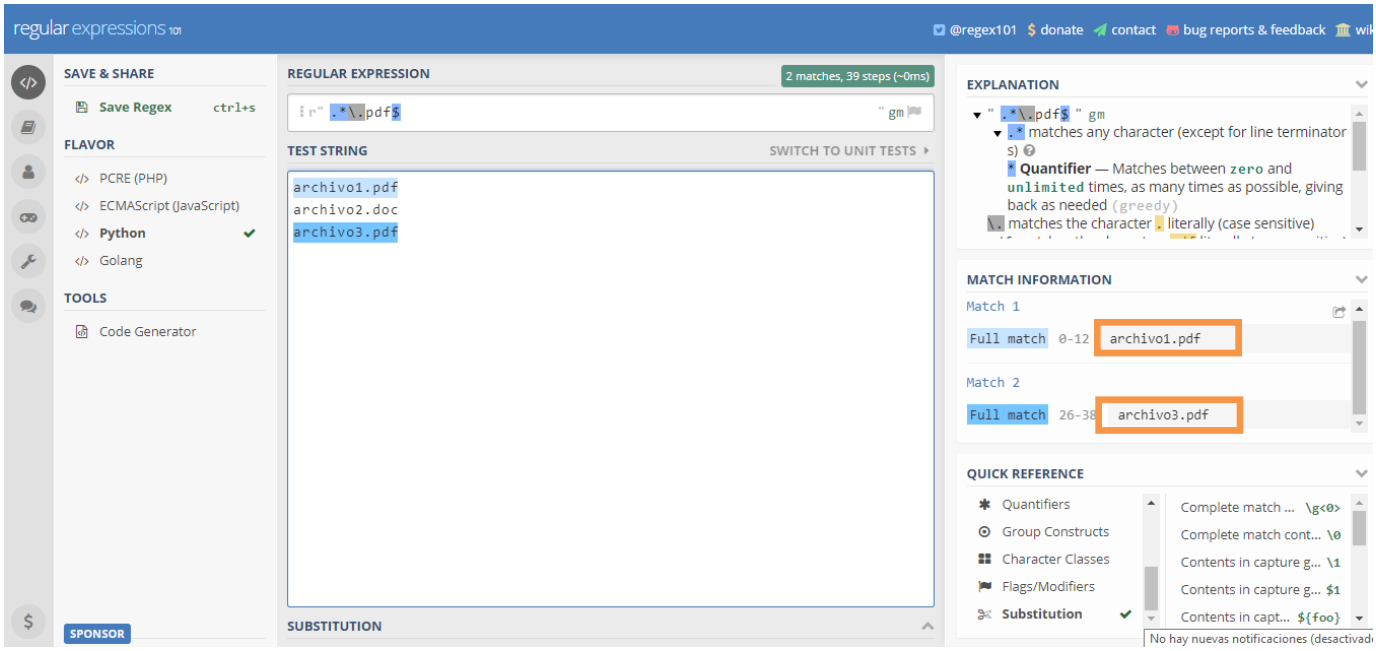
En las **expresiones regulares** existen caracteres especiales con un significado y tratamiento especial, a estos se les llama meta-caracteres o caracteres especiales, aquí hay una lista de ellos:

- . Representa cualquier caracter excepto el salto de línea
- \w Representa cualquier letra o número
- \W Representa cualquier caracter que no sea una letra o un numero.
- \d Representa cualquier numero del 0 al 9
- \D Representa cualquier caracter que no sea un numero del 0 al 9
- \s Representa un espacio en blanco
- \S Representa cualquier caracter que no sea un espacio en blanco.
- $ Representa que ahí finaliza el texto, por ejemplo la expresión com$ busca que **com** sea lo último en el texto. Este caracter solo puede usarse al final de la expresión regular.
- ^ Representa el inicio del texto, por ejemplo la expresión ^hola busca que el texto inicie con **hola**, este caracter solo puede usarse al inicio de la expresión regular.
- \b Representa que ahí inicia o finaliza una palabra, por ejemplo la expresión \b[A-Z][a-z]* busca palabras que inicien con una letra mayúscula y luego lleven cualquier cantidad de letras minúsculas. En cambio la expresión \w*os\b busca palabras que finalicen en **os**.

Ejemplos prácticos.

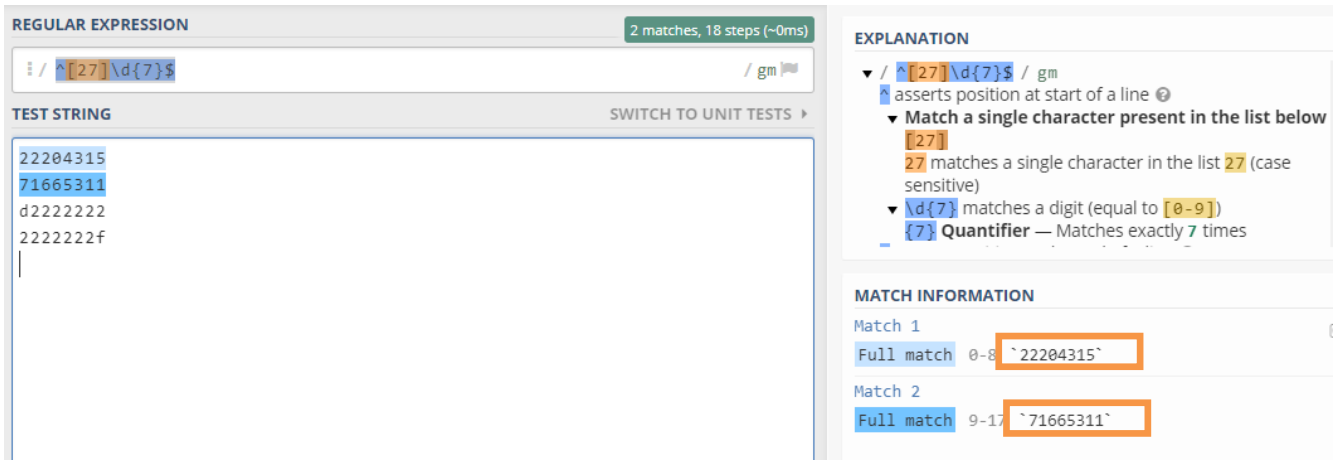
Ahora que ya conoces mucho de **expresiones regulares**, vamos a practicar con algunos ejemplos. Primero vamos a **detectar si un archivo es de tipo pdf**, en base a su nombre y extensión, para ello podemos usar una expresión como esta: .*\.pdf$

Lo que hace la expresión es buscar cualquier caracter, sea numero, letra o símbolo usando el . luego se coloca un * para decir que puede ir cualquier cantidad de caracteres, después sabemos que la extensión de los archivos inicia con un . pero como él . es un caracter especial usamos un signo \ para que lo ignore y trate como un simple . luego, debemos buscar el texto pdf y este debe ser lo último en el texto, no debe haber nada después, esto lo definimos usando el signo $



Ahora si necesitamos validar un número telefónico y nos dicen que solo puede iniciar con un 2 o un 7 y debe de tener exactamente 8 números, podemos usar esta expresión regular: ^[27]\d{7}$

Primero usamos el signo ^ para decir que estamos por definir el inicio de la cadena, luego le decimos que debe llevar un 2 o un 7 con [27] ahora faltan 7 números más para completar los 8 números, entonces usamos \d para decir que continua un número del 0 al 9 y usamos {7} para definir que este último patrón se repite 7 veces, luego usamos $ para definir que aquí debe terminar el texto (después de los últimos 7 números.)


REGULAR EXPRESSION
2 matches, 18 steps (~0ms)
/ ^[27]\d{7}$ / gm
TEST STRING
SWITCH TO UNIT TESTS
22204315
71665311
d2222222
2222222f
EXPLANATION
/ ^[27]\d{7}$ / gm
^ asserts position at start of a line
Match a single character present in the list below
[27]
27 matches a single character in the list 27 (case sensitive)
\d{7} matches a digit (equal to [0-9])
{7} Quantifier — Matches exactly 7 times
MATCH INFORMATION
Match 1
Full match 0-8 `22204315`
Match 2
Full match 9-17 `71665311`