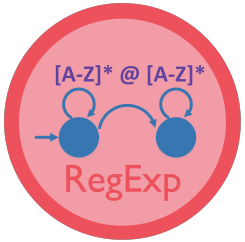
# Expresiones Regulares



# Expresiones Regulares

Lenguaje creado para facilitar

el análisis de texto

# Expresiones Regulares

Útil para:

- Verificar que una cadena de texto concuerde con un patrón
- Obtener pedazos importantes de un texto
- Realizar substituciones en strings

# Importación

Necesitamos importar el módulo re

```
'import re'
```

# Página para probar nuestras expresiones

https://regex101.com/

# Metacaracteres

Caracteres que toman valor dependiendo del contexto

. -> Corresponde cualquier caractér menos al salto de línea

a. -> av, ae, a3, a@, a2

# Metacaracteres

| -> Corresponde cualquiera de las dos opciones

(a|b) -> a , b

Se puede acumular para más opciones

(a|b|c) -> a , b , c

# Metacaracteres

^ -> Corresponde al inicio del texto
^hola -> hola a todos los programadores

$ -> Corresponde con el final del texto
programadores$ ->hola a todos los programadores

# Cuantificadores

Definen la repetición de uno o más caracteres

* -> Se puede repetir 0 o más veces
trata de tomar el mayor número de repeticiones

b* -> ’’, b, bb, bbbbb, bbbbbb

# Cuantificadores

```
+ -> Se puede repetir 1 o más veces
     trata de tomar el mayor número de
repeticiones
     3+ -> 3, 333, 333333

? -> Se puede repetir 0 o 1 vez
     d? -> '' , d
```

# Cuantificadores

{n} -> Se puede repetir n veces

a{3} -> aaaa

{n,m} -> Se puede repetir de n a m veces

w{2,5} -> ww , www , wwww , wwwww

w{,4} -> '' , w , www , wwww

w{2,} -> ww , wwwwwww , wwwwwwwwwwwwwww

# Cuantificadores

Podemos agrupar con paréntesis

```
ho(la)+ -> hola, holala, holalala
```

# Rango

[] -> Especifica un rango de valores

[12a] -> 1, 2, a

[@2 3c] -> @, 2, " ", 3, c

[a-z] -> a, b, c, ...., z

[A-Z] -> A, B, C, ...., Z

[0-9] -> 0, 1, 2, ...., 10

[A-Za-z] -> A, a, B, b, ..., Z, z

[A-Za-z0-9] -> todas las letras y números

[A-Za-z0-9._] -> todos los caracteres válidos en un correo electrónico

# Rango

^ -> Sirve para negar un rango

[^a-z] -> 2, A, @, 8, c, ....

[^A-Za-z] -> 3, 6, @, 1, ....

Dentro de los rangos los metacaracteres como .,+,? son considerados como caracteres normales

# Rango

^ -> Sirve para negar un rango

[^a-z] -> 2, A, @, 8, c, ....

[^A-Za-z] -> 3, 6, @, 1, ....

Dentro de los rangos los metacaracteres como .,+,? son considerados como caracteres normales

# Rango

^ -> Sirve para negar un rango

[^a-z] -> 2, A, @, 8, c, ....

[^A-Za-z] -> 3, 6, @, 1, ....

Dentro de los rangos los metacaracteres como .,+,? son considerados como caracteres normales

# Alias

Son formas resumidas de declarar rangos de caracteres

(digitos)
\d -> equivale a [0-9]

(no digitos)
\D -> equivale a [^0-9]

(nombres o palabras)
\w -> equivale a [a-zA-Z0-9_]

(no nombres ni palabras)
\W -> equivale a [^a-zA-Z0-9_]

# Alias

(salto de linea ,tabulación o espacio)
\s -> equivale a [\n\t ]
(no salto de linea ni tabulación ni espacio)
\S -> equivale a [^\n\t ]
\b -> frontera de palabra
\B -> todo menos frontera de palabra

() -> Los paréntesis sirven para agrupar segmentos de la expresión regular

# METODOS

`patron.findall(texto)` - Retorna una lista que contiene todos las concordancias generadas. En caso de existir grupos, retorna una lista con los grupos sin incluir el 0

`patron.finditer(texto)` - Retorna un iterador que va a ir retornando objetos Match de cada concordancia.

# METODOS

`re.match(patron, texto)` - Se utiliza para ver si hay match al principio del texto

`re.search(patron, texto)` - Se utiliza para ver si hay match en cualquier parte del texto

`re.findall(patron, texto)` - Regresa una lista de todas las subcadenas que hayan hecho match

# METODOS

re.sub(patron, reemplazo, texto) - Regresa el texto con el reemplazo en vez del patrón

re.sub(patron, reemplazo, texto, máximo) - Regresa el texto con el reemplazo hecho una cantidad de veces, indicada en máximo

re.compile(patron) - Compila un patron en un objeto de tipo expresion regular

# METODOS

`re.compile(exprReg)` - Genera un objeto de tipo patrón

`patron.search(texto)` - Genera un objeto match de la primer concordancia encontrada en el texto, en caso de no haber ninguna concordancia, retorna None

`patron.sub(rempl, texto)` - Genera una cadena que reemplace las concordancias por un valor de reemplazo preestablecido