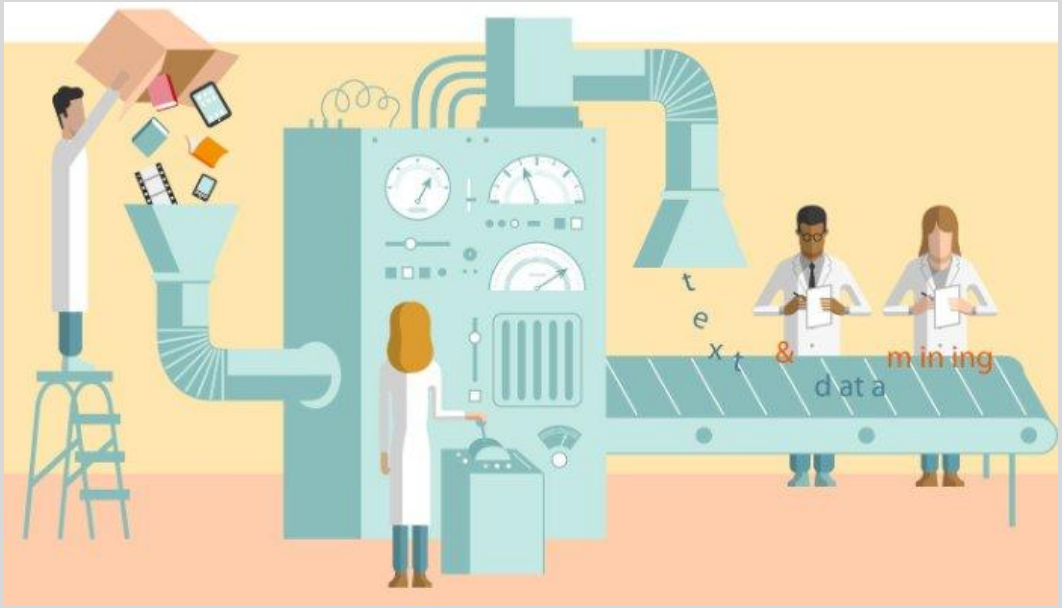

DigitalHouse > Coding School

DATA SCIENCE

Expresiones regulares

1. Introducir el concepto de expresiones regulares: comprender qué son, para qué sirven y cuáles son sus campos de aplicación

2. Presentar los patrones de búsqueda, así como los caracteres utilizados para definir funcionalidades especiales

3. Aplicar los contenidos a un ejercicio práctico

Expresiones regulares

¿Qué son las expresiones regulares y para qué sirven?

Una expresión regular es una **secuencia de caracteres** que define un **patrón de búsqueda** de texto. Las regex constituyen lenguaje muy flexible que sirve para identificar y extraer información de un cuerpo de caracteres **no estructurado**.

Expresiones regulares

```
import re

# Nuestro string
cuil = """ Javier 20-28490848-8
Laura - 27 33495648 9
Eva23366154122
Diego: 24.18357496.5 """

# Definimos el patrón de búsqueda
pattern = r'([a-z]+)\s?-?:?\s?(\d*)-?\s?\.?(\d*)-?\s?\.?(\d*)'

# Lo compilamos en un objeto regex
regex = re.compile(pattern, flags = re.IGNORECASE)

# Hacemos la búsqueda
m = regex.findall(cuil)
lista_empleados = [' '.join(e) for e in m]

def busqueda_cuil(x):
    m = regex.search(x)
    if m:
        return(print(m.group(1)+':',m.group(2)+m.group(3)+m.group(4)))

for empleado in lista_empleados:
    busqueda_cuil(empleado)
```

```
Javier: 20284908488
Laura: 27334956489
Eva: 23366154122
Diego: 24183574965
```

Metacaracteres especiales

```
saludo = '¡Hola, Digital House!'
print(re.findall('hola',saludo,flags = re.IGNORECASE))
print(re.findall('digital house',saludo,flags = re.IGNORECASE))
```

```
['Hola']
['Digital House']
```

Los **metacaracteres especiales** son aquellos que no se encuentran a sí mismos, sino que indican que se debe *matchear* algo de forma no literal, o que afectan a otras partes de la regex repitiendo caracteres o cambiando su significado.

```
. ^ $ * + ? { } [ ] \ | ( )
```

Algunos metacaracteres frecuentes:

\d Cualquier dígito del 0 al 9
\w Cualquier caracter alfanumérico (A-Z, a-z, 0-9 y _)
\s Cualquier espacio en blanco (espacio, tabulado, nueva línea, etc.)
. Cualquier caracter con excepción de nueva línea (\n)

Y sus complementos:

\D Todo menos cualquier dígito
\W Todo menos cualquier caracter alfanumérico
\S Todo menos cualquier espacio en blanco

Cuantificadores:

***** Cero o más del elemento anterior
+ Uno o más del elemento anterior
{min,max} Definen mínimos y máximos de repetición
? Opcional, o cero o uno del elemento anterior

Ejemplos:

\d+ Encuentra un número o más
.* Encuentra cero o más de cualquier caracter
\w{2,6} Encuentra un conjunto de caracteres alfanuméricos con una longitud que va de 2 a 6 caracteres

¿Qué patrón identifican para los siguientes *strings*?

ana_laura@hotmail.com

juan.perez@gmail.com

florgimenez@yahoo.com.ar

Otros metacaracteres importantes:

[ ] Definen un conjunto de caracteres
[A-Za-z0-9_] Definen rangos

( ) Definen un grupo de captura
| Operador "o"
**** Escapa los caracteres especiales

Metacaracteres de posición:

^ Comienzo de un *string* (o negación dentro de un conjunto, [^])
$ Final de un *string*
\b Límite de palabra

Documentación oficial:

https://docs.python.org/3/library/re.html

https://docs.python.org/3/howto/regex.html

Sitios web para practicar:

https://regexr.com/

https://regex101.com/

Práctica Guiada – Expresiones regulares