Manipulación de cadenas de caracteres con Expresiones Regulares

Julio Waissman Vilanova

2025-2

💡 ¿Qué son las expresiones regulares?

🧐 Definición

- Son una secuencia de caracteres que **forman un patrón de búsqueda**.
- Se utilizan para encontrar, reemplazar y manipular texto.
- Piensa en ellas como un lenguaje para describir patrones de texto.

🔍 ¿Para qué sirven?

- **Validación de datos**: Verificar si un correo electrónico o un número de teléfono tienen el formato correcto.
- **Búsqueda y reemplazo**: Encontrar todas las menciones de una palabra y reemplazarlas por otra.
- **Extracción de información**: Sacar datos específicos de un archivo de registro (logs) o de un documento.
- **Análisis de texto**: Contar la frecuencia de palabras o identificar patrones.

🛠️ Componentes básicos

Caracteres literales

- Son caracteres que se interpretan tal cual.
- Ejemplo: `gato` busca la secuencia de caracteres "gato".
- `hola mundo` busca exactamente "hola mundo".

Metacaracteres

- Son caracteres con un significado especial.
- `.` **(punto)**: Coincide con cualquier carácter (excepto una nueva línea).
  - Ejemplo: `c.sa` coincide con "casa", "cosa", "cisa", etc.
- `*` **(asterisco)**: Coincide con **cero o más** repeticiones del carácter anterior.
  - Ejemplo: `go*le` coincide con "gle", "gole", "goole", "gooole", etc.
- `+` **(más)**: Coincide con **una o más** repeticiones del carácter anterior.
  - Ejemplo: `go+le` coincide con "gole", "goole", "gooole", pero NO con "gle".
- `?` **(interrogación)**: Coincide con **cero o una** repetición del carácter anterior. Es decir, el carácter es opcional.
  - Ejemplo: `colou?r` coincide con "color" y "colour".

Metacaracteres (cont.)

- `|` **(barra vertical)**: Actúa como un **"O" lógico**.
  - Ejemplo: `(gato|perro)` coincide con "gato" o "perro".
- `^` **(acento circunflejo)**: Coincide con el **inicio** de la línea.
  - Ejemplo: `^Hola` solo coincide si la línea comienza con "Hola".
- `$` **(signo de dólar)**: Coincide con el **final** de la línea.
  - Ejemplo: `mundo$` solo coincide si la línea termina con "mundo".

Conjuntos de caracteres `[...]`

- Coinciden con **cualquier carácter dentro del conjunto**.
- Ejemplo: `[aeiou]` coincide con cualquier vocal.
- `[0-9]` coincide con cualquier dígito.
- `[a-zA-Z]` coincide con cualquier letra, mayúscula o minúscula.
- `[^...]`: Si el `^` está al inicio del conjunto, **niega** la coincidencia.
    - Ejemplo: `[^aeiou]` coincide con cualquier carácter que NO sea una vocal.

Cuantificadores

- Indican cuántas veces debe repetirse el carácter o grupo anterior.
- `{n}`: Coincide con **exactamente n** repeticiones.
  - Ejemplo: `a{3}` coincide con "aaa".
- `{n,}`: Coincide con **n o más** repeticiones.
  - Ejemplo: `a{2,}` coincide con "aa", "aaa", "aaaa", etc.
- `{n,m}`: Coincide con un mínimo de **n** y un máximo de **m** repeticiones.
  - Ejemplo: `a{1,3}` coincide con "a", "aa", o "aaa".

Abreviaciones comunes en Python para expresiones regulares

- `\d` : Coincide con cualquier dígito (0-9).
  - Equivalente a `[0-9]` .
  - Ejemplo: `\d{3}` coincide con `123` , `456` , etc.
- `\D` : Coincide con cualquier carácter que **no sea un dígito**.
  - Equivalente a `[^0-9]` .
  - Ejemplo: `\D` coincide con "a", "!", " ", etc.
- `\w` : Coincide con cualquier carácter alfanumérico (letras, dígitos y guion bajo).
  - Equivalente a `[a-zA-Z0-9_]` .
  - Ejemplo: `\w+` coincide con `hola` , `usuario123` , `_variable` .
- `\W` : Coincide con cualquier carácter que **no sea alfanumérico**.

Abreviaciones comunes en Python para expresiones regulares (cont.)

- `\s` : Coincide con cualquier carácter de espacio en blanco (espacio, tabulación, salto de línea).
  - Equivalente a `[ \t\n\r\f\v]` .
  - Ejemplo: `hola\smundo` coincide con "hola mundo".
- `\S` : Coincide con cualquier carácter que **no sea un espacio en blanco**.
- `\b` : Coincide con un **límite de palabra**. Es una posición, no un carácter.
  - Ejemplo: `\bcat\b` solo coincide con la palabra completa "cat", no con "caterpillar".
- `(?i)` : Modificador para hacer la búsqueda **insensible a mayúsculas y minúsculas**.
  - Se utiliza al inicio de la expresión.
  - Ejemplo: `(?i)python` coincide con "Python", "PYTHON", "python", etc.

🚀 Ejemplos prácticos

📧 Validación de correo electrónico

- Expresión: `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`
- Análisis:
  - `^`: Inicio de la línea.
  - `[a-zA-Z0-9._%+-]+`: Uno o más caracteres válidos para el nombre de usuario.
  - `@`: El símbolo arroba literal.
  - `[a-zA-Z0-9.-]+`: Uno o más caracteres válidos para el dominio.
  - `\.`: El punto literal (se escapa con `\`).
  - `[a-zA-Z]{2,}`: Dos o más letras para el TLD (ej. com, org, es).
  - `$`: Fin de la línea.

📞 Validación de número de teléfono (ej. de México)

- Expresión: `^\d{10}$`
- Análisis:
  - `^`: Inicio de la línea.
  - `\d`: Es una forma corta de `[0–9]`, coincide con un dígito.
  - `{10}`: Exactamente 10 repeticiones.
  - `$`: Fin de la línea.
- Coincide con números como: `5512345678`

🌐 Extracción de URLs

- Expresión: `(https?:\/\/)?(www\.)?[\da-zA-Z.-]+\.[a-zA-Z]{2,6}`
- Análisis:
  - `(https?:\/\/)?` : El protocolo "http" o "https" es opcional.
  - `(www\.)?` : El "www." es opcional.
  - `[\da-zA-Z.-]+` : El dominio principal (uno o más dígitos, letras o puntos/guiones).
  - `\.` : El punto literal.
  - `[a-zA-Z]{2,6}` : El TLD (2 a 6 letras).

Uso de expresiones regulares con Pandas 🐼

- Pandas integra la funcionalidad de expresiones regulares a través de los métodos de la columna `.str`.
- Solo funciona para `Series` de tipo string u `object`.
- Esto permite aplicar patrones de búsqueda directamente sobre Series de manera eficiente.

Filtrar filas que coinciden con un patrón

Puedes usar el método `.str.contains()` para filtrar filas basándote en si el texto de una columna coincide con una expresión regular.

```
import pandas as pd

# Crear un DataFrame de ejemplo
data = {'producto': ['Manzana roja', 'Uva verde', 'Pera amarilla',
                     'Naranja', 'manzanita Granny Smith']}
df = pd.DataFrame(data)

# Filtrar productos que contengan la subpalabra "Manzan"
# ignorando mayúsculas y minúsculas
df[df['producto'].str.contains('manzan', case=False, na=False)]
```

Extraer información específica

El método `.str.extract()` es ideal para extraer partes específicas de una cadena que coinciden con grupos de captura definidos en la expresión regular. Los grupos de captura se crean con paréntesis `()`.

```
import pandas as pd

# Crear un DataFrame con datos de ventas
data = {'orden': ['ID-453-2023', 'ID-121-2024', 'ID-987-2023', 'ID-765-2024']}
df = pd.DataFrame(data)

# Extraer el año de la orden
df['año'] = df['orden'].str.extract(r'-(\d{4})$')
```

Reemplazar texto basado en un patrón

Puedes usar `.str.replace()` para reemplazar todas las coincidencias de un patrón por una nueva cadena.

```
import pandas as pd

# Crear un DataFrame con precios
data = {'precio': ['€15.50', '$10.00', '€22.75', '¥500']}
df = pd.DataFrame(data)

# Eliminar el símbolo de la moneda
df['precio_sin_moneda'] = df['precio'].str.replace(r'[^0-9.]', '', regex=True)
```

Ahora a ustedes

```
import pandas as pd

# Crear un DataFrame con telefonos en diferentes formatos
data = {'tel': ['(662) 123-4567', '662-123-4567', '662 123 4567',
                '6621234567', 'sin telefono', '2345']}
df = pd.DataFrame(data)
```

- Crea una serie nueva llamada `tel_estandar` que contenga los números de teléfono en el formato `6621234567`, es decir, solo los dígitos.
- Igualmente, los números que no sean de 10 dígitos o que no se puedan estandarizar deben ser marcados como `NA`.

✨ Algunos consejos sobre regex

- Empieza con patrones sencillos y ve añadiendo complejidad.
- Usa herramientas en línea como **regex101.com** para probar tus expresiones y entender cada componente.
- A veces, un enfoque más simple es mejor. No siempre necesitas la expresión más compleja.
- Los prompts ayudan mucho a generar espresiones regulares, pero siempre hay que revisarlos y verificarlos

📚 Recursos adicionales

- **Pandas Documentation**: Guía oficial sobre el manejo de texto en Pandas.
- **Regex101**: Herramienta en línea para probar y depurar expresiones regulares.
- **RegExr**: Otra herramienta en línea para aprender, construir y probar expresiones regulares.
- **Cheat Sheet de Expresiones Regulares**: Una referencia rápida para los componentes más comunes de las expresiones regulares.