Expresiones regulares: Respuestas

1. Usando el texto de la Declaración de los Derechos Humanos, que está en el archivo `derechos_humanos.txt` en los materiales del tutorial:

1.1. ¿Cuántas palabras terminan en "tad"?
`tad\b` Haciendo *Mark* o *Find in Document* dan 15 resultados

1.2. ¿Cuál es el sufijo nominal más frecuente entre los siguientes: *-tad*, *-dad*, *-ión*, o *-iento*?

`tad\b`	15
`dad\b`	32
`ión\b`	78
`iento\b`	13

1.3.

a) ¿Cuántas veces aparece la preposición "de"?
`\bde\b` 136 resultados

b) ¿Haciendo una búsqueda sensible a mayúsculas obtendríamos el mismo resultado?
`\b[Dd][Ee]\b` Para que haya 136 resultados con búsqueda sensible a mayúsculas, haría falta una expresión que también liste las mayúsculas

1.4. Dar una expresión que encuentre palabras relacionadas con la noción de “libertad” y solamente tales palabras (p. ej. el nombre "libertad", el adjetivo “libre”, el verbo “liberar” etc.)

```
\blib[er]\w+
```

¿Deberíamos cambiar la expresión si la palabras “libro”, “librería”, o el nombre propio “Delibes” fueran parte del texto?

Sí, porque la expresión de arriba también encontraría *libro* and *librería*

Una opción para evitar encontrar *libro* sería la siguiente:

```
\blibe?r(?:a|e|emente|tad)e?s?
```

Para evitar encontrar *librería*, se podría usar lookahead negativo `X(?!Y)` , que quiere decir *X no seguido de Y*:

```
\blibe?r(?:a|e(?!r)|emente|tad)e?s?
```

`e(?!r)` en el grupo de no captura *(?:)* con la alternancia (con pipe) evitaría que *librería* sea encontrada por la expresión.

2. Dar una expresión regular que encuentre todas las palabras que empiezan con "b" (mayúscula o minúscula) en la oración siguiente, y solo esas palabras:

La boa que vive en el baobab de Bob cerca de la casa de Rob

```
\b[Bb]
```

¿Qué expresión encontraría las palabras que terminan con "b" (independientemente de mayúscula/minúscula)?

```
[Bb]\b
```

3. Dada la expresión regular *^[^^]*

a) proporcionar una cadena de caracteres que no coincida con la expresión

Cualquier cadena de caracteres que empiece con ^ no corresponde a la expresión, p. ej. `^koala`

b) proporcionar una cadena de caracteres que coincida con la expresión

`koala` se corresponde con la expresión (la *k* de *koala* corresponde)

4. Dada la expresión regular *^[^$]+$*

a) proporcionar una cadena de caracteres que no coincida con la expresión

Cualquier cadena que contenga *$* no podrá corresponder a la regex

b) proporcionar una cadena de caracteres que coincida con la expresión

Cualquier cadena que no contenga el carácter *$* corresponde a esta expresión

5. Transforma "adoro las expresiones regulares" en "las expresiones regulares, las adoro", sin usar símbolos literales en el contexto de la expresión.

(El *contexto* es la expresión de búsqueda (casilla *find*), donde indicamos los grupos dentro del patrón a buscar, si es necesario. Y la sustitución es la expresión que usamos en la casilla *replace*, en la cual podemos hacer referencia a los grupos con la notación \1, \2, …, \9).

Contexto ^(\w+) (\w+) (\w+) (\w+)

Sustituciónnt \2 \3 \4, \2 \1

6. Doblar *nuestros ingresos*. Dicho de otro modo, dada la cadena *nuestros ingresos*, proporcionar una expresión regular que escribe *nuestros ingresos nuestros ingresos*.

Contexto `^(.+)$`

Sustitución `\1 \1`

7. Dar dos expresiones regulares que empiecen y terminen con la letra *a*, y que se comporten de la forma descrita a continuación para la oración *abra el sistema*:

a) la primera expresión debe corresponder solo a la primera palabra, *abra*
b) la segunda expresión debe corresponder a la oración completa, *abra el sistema*

a) `a.+?a` corresponde solo a *abra* dado el cuantificador lazy `+?`
   Una forma equivalente (pero que puede llegar a ejecutarse más rápido) de escribir esto es `a[^a]+a`
b) `a.+a` corresponde a la oración entera, dado el cuantificador en modo greedy `+` (que es el modo por defecto de los cuantificadores)
   Otra forma (que puede llegar a ejecutarse más rápido) sería `a[^\n]+a`

8. Transformar una *boa* en un *baobab* sin usar caracteres literales en el contexto de la regex. En otras palabras, dar una regex que reescribe la palabra *boa* como *baobab* sin usar símbolos literales en el contexto.

Context `(.)(.)(.)`

Replacement `\1\3\2\1\3\1`

9. Dar una expresión que corresponda a todas las palabras que contengan "o" o bien "ó" en el poema del archivo `yurkievich_gloton.txt` que se encuentra en los materiales del curso, y que proviene del libro *Rimbomba* publicado por Saúl Yurkiévich en 1978.

Resulta que en este poema todas las palabras tienen "o" o bien "ó", con lo que una expresión general como \w+ sin más restricciones sería suficiente para encontrar todas las palabras con "o" u "ó" en el caso de este poema.

10. La idea en este ejercicio es tomar un archivo parcialmente marcado en TEI (un estándar ampliamente usado en humanidades, en formato XML) y completar el marcado. El archivo con marcado incompleto es `tei_incomplete_example.xml` en los materiales del curso.

El objetivo es añadir etiquetas `<l>` a los versos y etiquetas `<lg>` a las estrofas:

- **Al principio de cada verso poner <l> y al final de cada verso poner </l>**
- **Al principio de cada estrofa poner <lg> y al final poner </lg>**

Puedes usar varias expresiones regulares.

Hay varias formas de hacer esto. Por ejemplo, a través de los cuatro pasos siguientes:

1. Añadir etiquetas <l> … </l> *a los versos*

Find ^[^<\w\n]+(\w.+)$

Replace <l>\1</l>

2. Añadir etiquetas <lg> … </lg> *a las estrofas, salvo la primera y última estrofa, que tratamos de forma separada (paso 3) porque su contexto en el texto es diferente a las demás*

Find </l>(?:\s{2,})<l>

Replace </l>\n</lg>\n<lg>\n<l>

3. Abordar la primera y última estrofas

3a. Primera estrofa:

Find (<div n.+)

Replace \1\n<lg>

3b. Última estrofa:

Find (</l>)(\s+)( )

Replace \1\n</lg>\2\3

Los pasos de arriba dan un XML como el del archivo `tei_completed.xml` en los materiales del curso.

¿Cómo conseguir la indentación de los archivos?

En primer lugar, la indentación no es un elemento de la sintaxis XML, pero se usa para facilitar la lectura humana del XML, ya que puede facilitar que se vea más clara la jerarquía.

Para la indentación, se puede crear automáticamente. En un terminal de tipo similar a Unix, se puede usar el comando `xmllint --format FILENAME` (en mac, Linux, cygwin, o bash dentro de Windows 10), y si el programa `xmllint` está instalado.

Los editores especializados de XML también crean la indentación. Una versión indentada está en `tei_completed_indented.xml` en los materiales del curso.

11. En la columna Input tenemos marcado HTML incompleto. Está incompleto porque las filas (`<tr>`) no tienen celdas. La tarea consiste en añadir celdas (`<td>`) *del color necesario*, usando expresiones regulares. El color se expresa en cada caso como el valor de un atributo *class* añadido a cada celda, como se ve en la columna *Salida 1*. Después transformaremos el HTML de otras formas, que corresponden a lo que se ve en Salida 2 y Salida 3 más abajo.

1. Desde input (fichero `red_blue_input.html`) a Salida 1 (fichero `red_blue_output_1.html`):

Lo que importa en esta parte del ejercicio es trabajar sobre las palabras “red” y “blue” en la tabla HTML, pero evitando estas palabras en la CSS que está en el encabezado del fichero. Las expresiones deberán ocuparse de eso.

Opción 1

Find ([^.:])(red|blue)

Replace \1<td class="\2">\2</td>

Opción 2 (usando lookbehind)

Find (?<=[>])(red|blue)

Replace <td class="\1">\1</td>

2. Desde Salida 1 (`red_blue_output_1.html`) a Salida 2 (`red_blue_output_2.html`):

Find: >red<

Replace >rouge<

Find >blue<

Replace >bleu<

3. Desde Salida 2 (`red_blue_output_2.html`) a Salida 3 (`red_blue_output_3.html`)

Find red">rouge

Replace blue">rouge

Find blue">bleu

Replace red">bleu