

# **Tutorial de expresiones regulares: Manipulación de cadenas de texto**

**Pablo Ruiz Fabo**

**Université de Strasbourg – Laboratoire LiLPa**

# Público previsto

- Un público no técnico, como humanistas que quieren aprender a manipular cadenas de texto de forma eficaz en un editor de texto o en otra aplicación que soporte expresiones regulares.

# Plan del tutorial

- I: “Teoría”, en estas diapositivas
- II: Ejercicios, en un directorio aparte

Un archivo de texto llamado `ejemplos.txt` permite aplicar los ejemplos de cada diapositiva

# Introducción

# Expresiones regulares (Regex)

- Herramienta (o “lenguaje”) para encontrar y modificar cadenas de texto que correspondan a un patrón de caracteres
  - palabras con un sufijo determinado
  - rimas
  - palabras que empiezan con un prefijo dado
  - fechas
  - sílabas ...
- Una expresión regular puede representar múltiples cadenas de texto

# Sitios y editores para probar las regex

## Sitios

- [regexpal.com](http://regexpal.com)
- [regex101.com](http://regex101.com)
- [regexpr.com](http://regexpr.com)

## Editores

- Geany (usado en estas slides) [multiplataforma]
- Sublime Text [multiplataforma]
- Notepad++ [Windows]

# Variedades de regex

- Existen muchas variedades de regex.
  - Simples, más antiguas:
    - Regex básicas
    - Regex extendidas
- Variedades usadas en esta clase:
  - Librerías actuales, similares a PCRE (Perl-compatible regular expressions)

# Elementos del lenguaje

Veremos esto con más detalle después ...

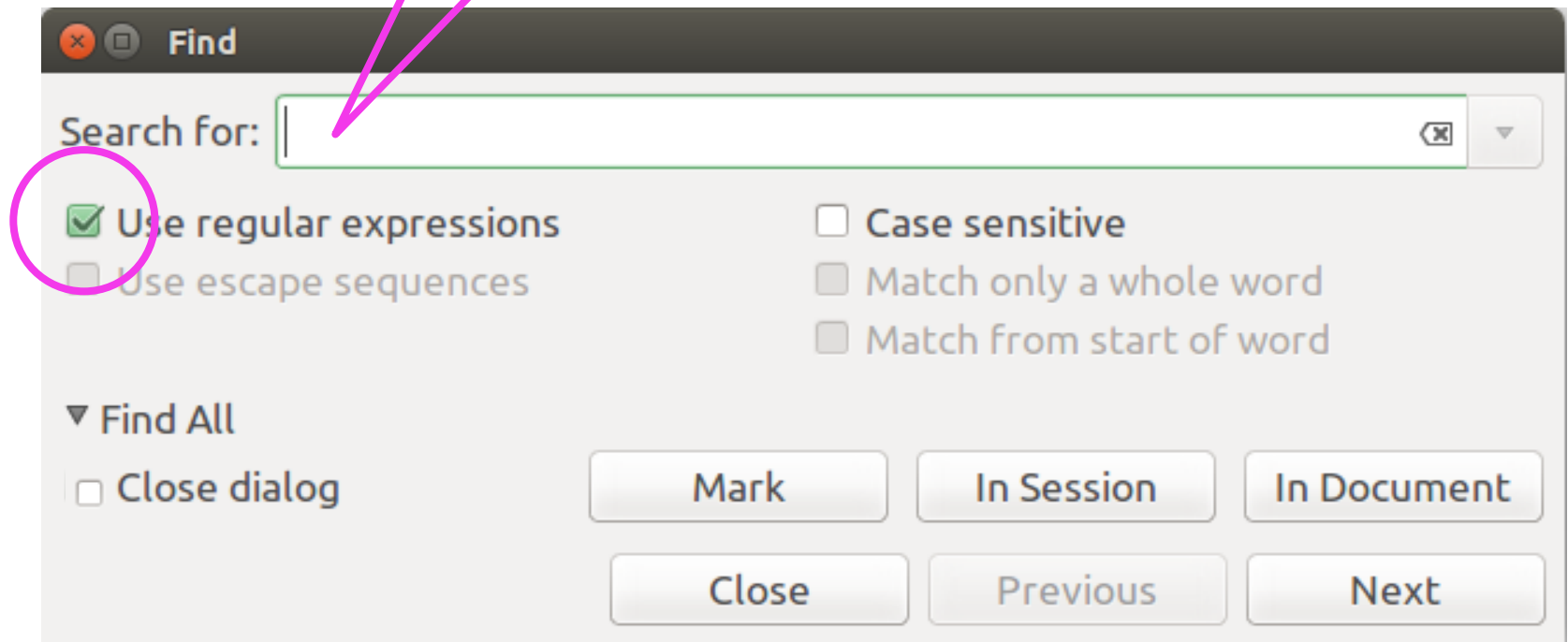
- **Literales:** en general, un carácter se corresponde a sí mismo
- **Caracteres especiales:** Representan conjuntos de caracteres o llevan a cabo ciertas funciones
  - **Abreviaturas de clase** de caracteres
  - **Operadores y cuantificadores:** modifican la forma en que se buscan las correspondencias (p. ej. número máximo de correspondencias a encontrar)
  - **Anclas (anchors):** especifican una posición, como el principio o el final de la línea



# Editor Geany: Búsqueda de patrones

Ctrl + F

Regex aquí



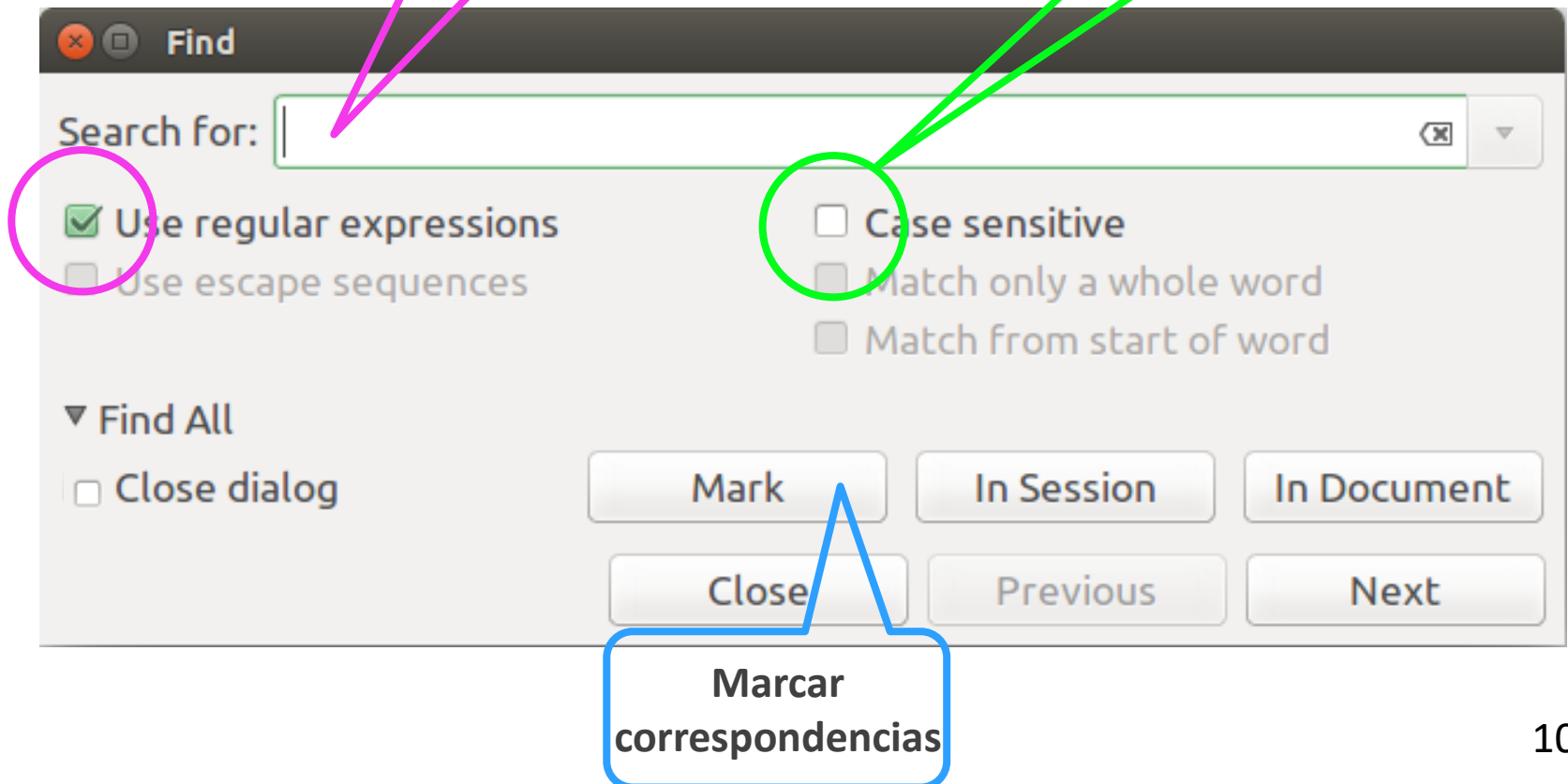
# Editor Geany:

## Búsqueda de patrones

Ctrl + F

Regex aquí

Considerar  
mayúscula/  
minúscula



# Literales

# Literales: En general, un carácter se corresponde a sí mismo

The image displays two screenshots of a text editor window titled 'jgdb\_juego-de-hacer-versos.txt'. The text in the editor is as follows:

```
53 La manera que tiene  
54 sobre todo en verano  
55 de ser un paraíso.  
56 Aunque, de cuando en cuando,  
57  
58 si alguna de esas nubes  
59 que las carga el diablo  
60 uno piensa en la historia  
61 de estos últimos años.
```

In the first screenshot, the search string 'la' is entered in the 'Find' dialog. The 'Case sensitive' checkbox is unchecked. The search results highlight 'La' on line 53 and 'la' on lines 59 and 60. The 'la' on line 60 is circled in blue.

In the second screenshot, the 'Case sensitive' checkbox is checked. The search results highlight only 'La' on line 53. The 'La' on line 53 is circled in blue.

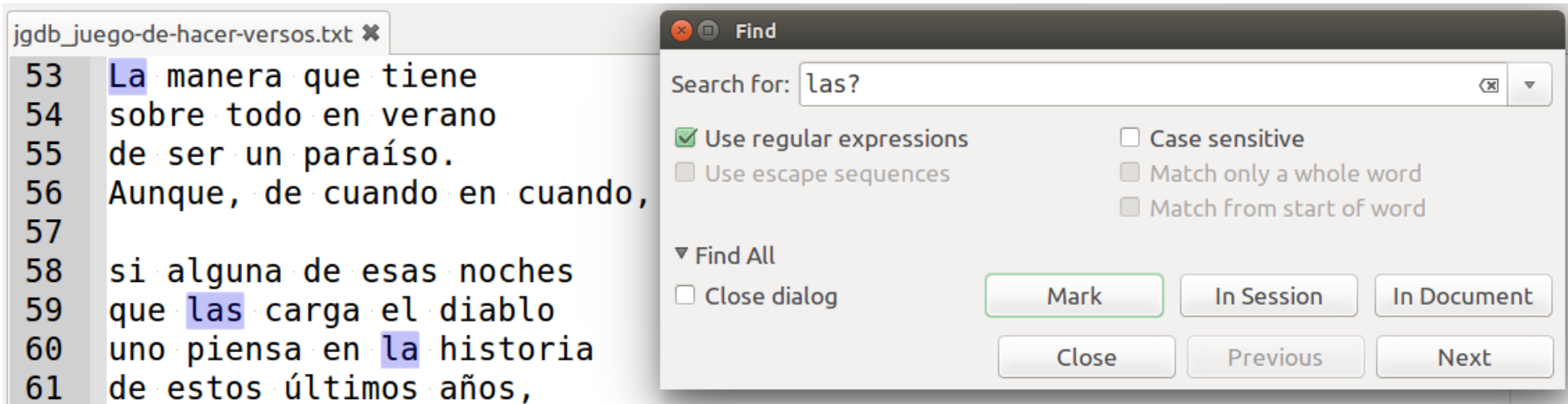
# Cuantificadores

# Cuantificadores

- Funciones más avanzadas
- Veremos ejemplos más adelante
- Cuantificadores
  - cuántas veces queremos reconocer una expresión (slides siguientes)
- Modos de cuantificación
  - buscar la cadena mínima o la máxima que corresponda a la expresión (Slide 24)

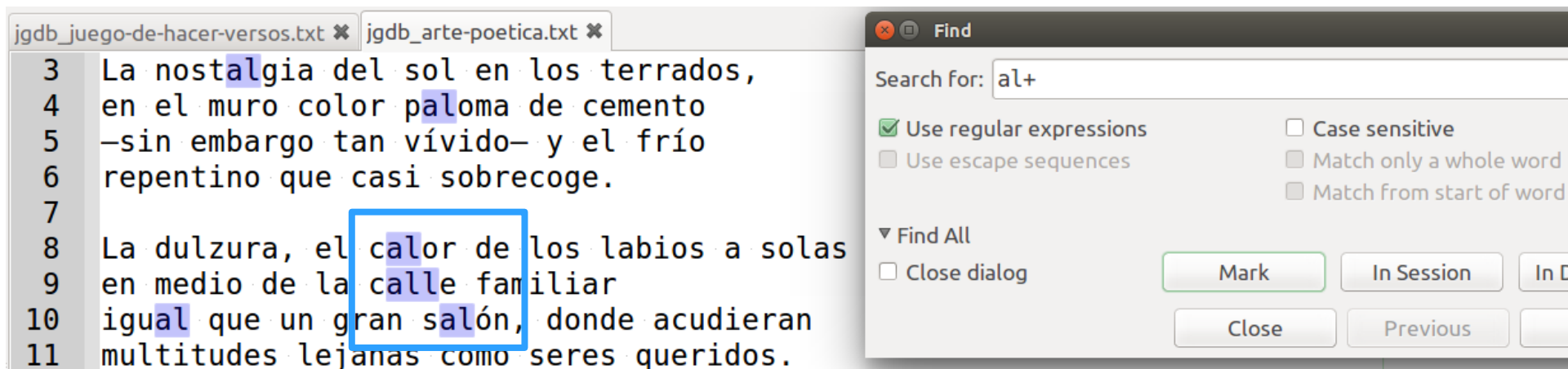
# Cuantificador ? :

## 0 o 1 ocurrencia



*las?* corresponde a *la* y *las*

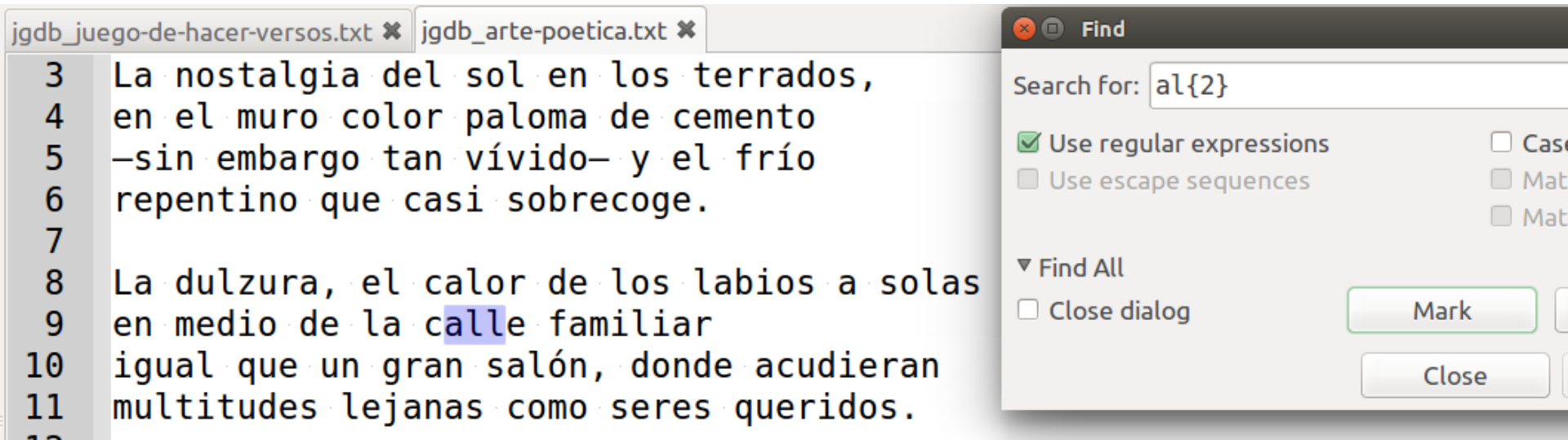
# Cuantificador + : 1 o más veces



***al+*** corresponde a ***al*** y ***all***, y también ***allllllllll*** ... (para cualquier número de *l*)



# {x}: exactamente x veces



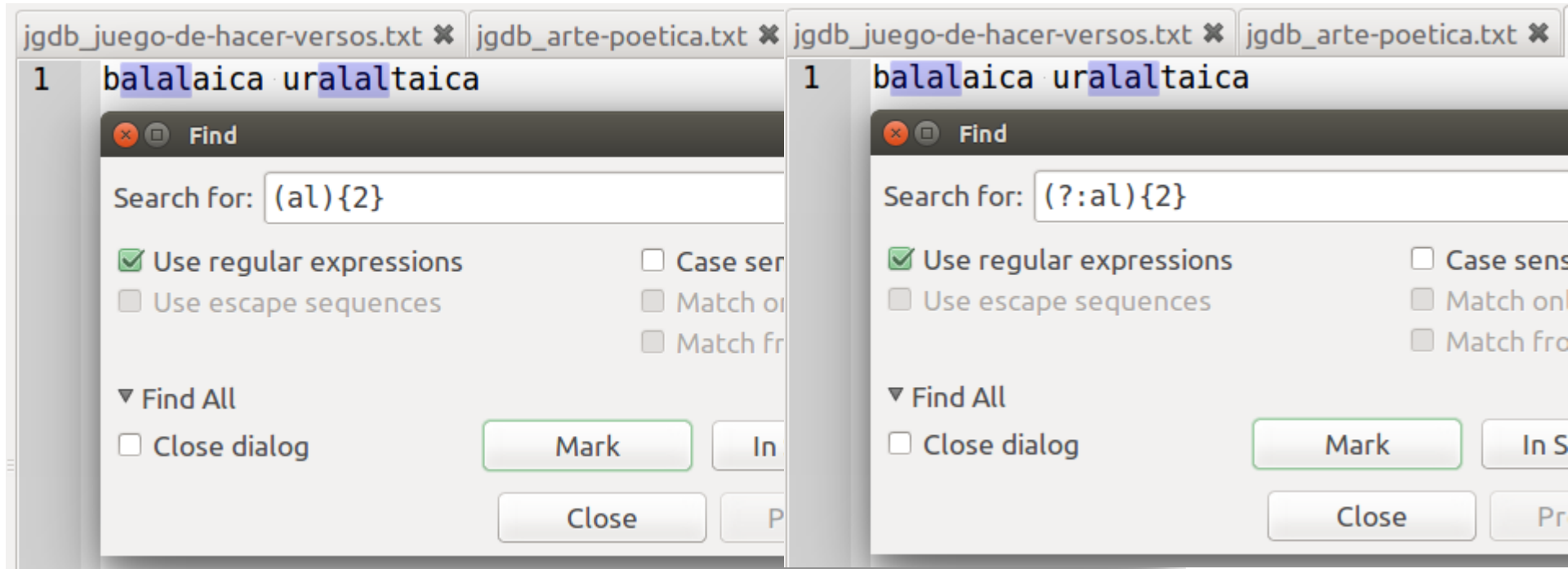
***`al{2}`*** corresponde a ***`all`***

**{x}: exactamente x veces**

- ¿Cómo encontrar *ala!*?

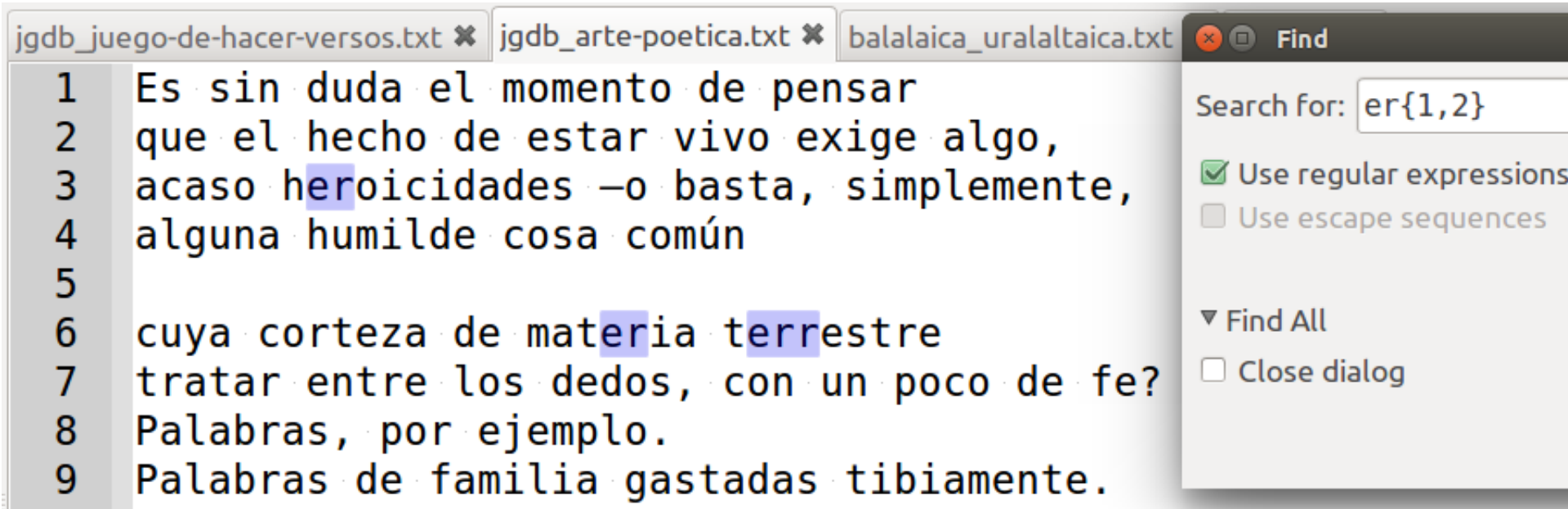
# {x}: exactamente x veces

- ¿Cómo encontrar *ala!*?



- Tanto  $(a!){2}$  como  $(? :a!){2}$  valen ...  
(ver slide 60 para la diferencia)

# **{x,y} : Entre x e y veces**



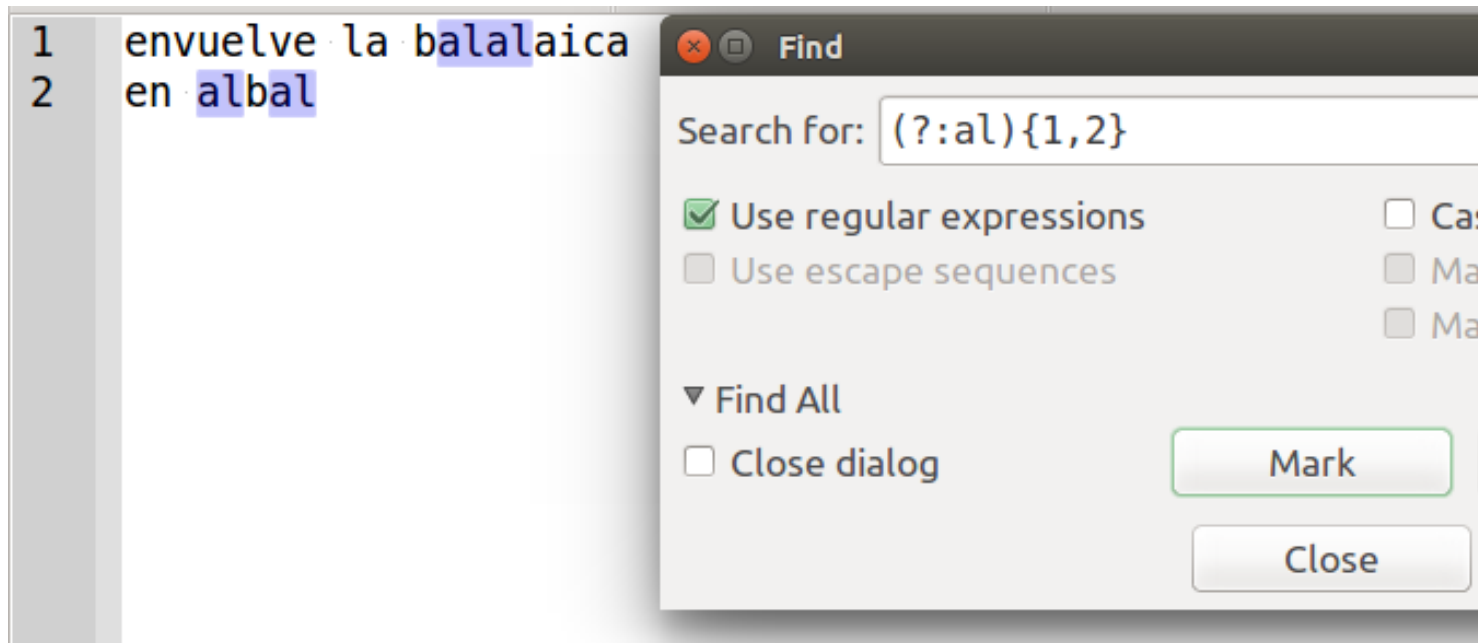
The screenshot shows a text editor with three tabs: 'jgdb\_juego-de-hacer-versos.txt', 'jgdb\_arte-poetica.txt', and 'balalaica\_uralaltaica.txt'. The active tab is 'jgdb\_arte-poetica.txt', which contains a poem. The text is as follows:

```
1 Es sin duda el momento de pensar
2 que el hecho de estar vivo exige algo,
3 acaso heroïcidades –o basta, simplemente,
4 alguna humilde cosa común
5
6 cuya corteza de materia terrestre
7 tratar entre los dedos, con un poco de fe?
8 Palabras, por ejemplo.
9 Palabras de familia gastadas tibiamente.
```

On the right side, there is a 'Find' dialog box. The 'Search for:' field contains the text 'er{1,2}'. Below this field, there are two checkboxes: 'Use regular expressions' (which is checked) and 'Use escape sequences' (which is unchecked). At the bottom of the dialog box, there are two buttons: 'Find All' and 'Close dialog'.

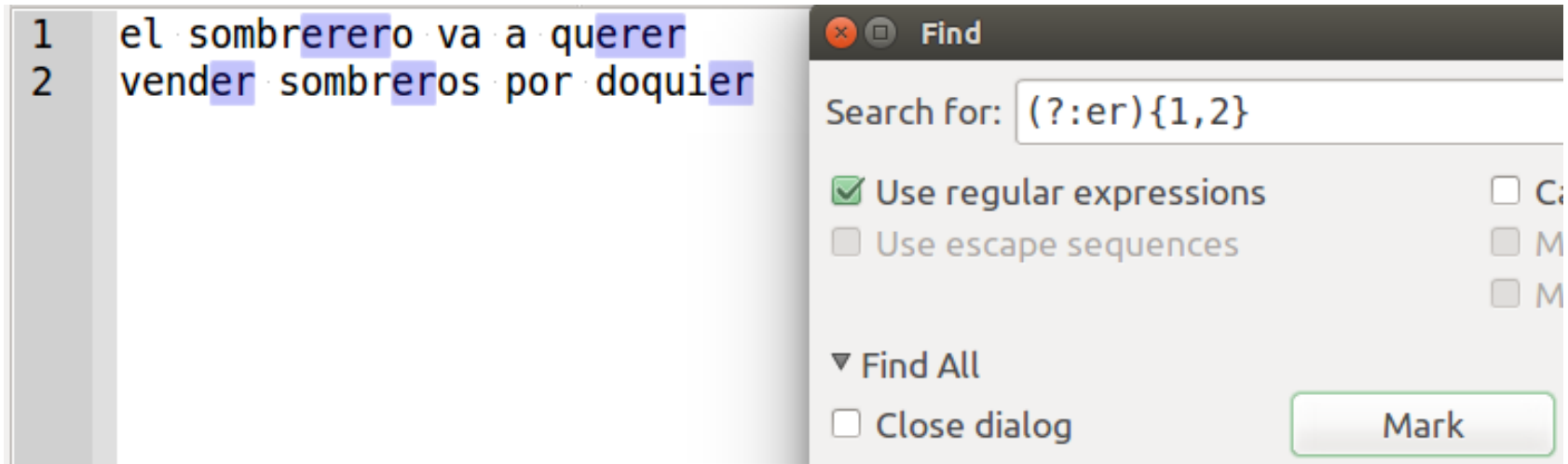
***er{1,2}*** corresponde a ***er*** y ***err***

# {x,y} : Between x and y times



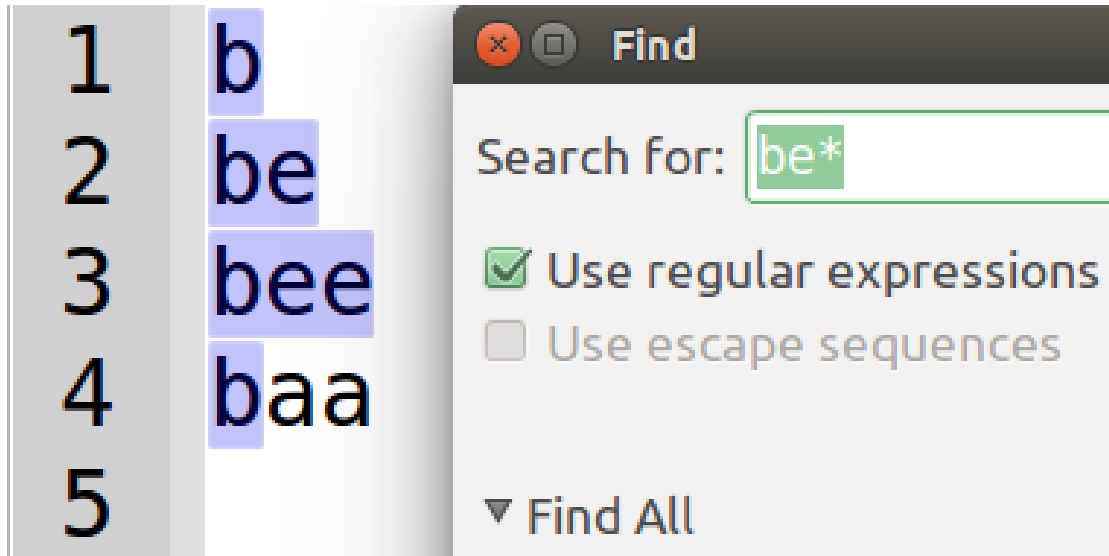
*(?:al){1,2}* corresponde a ***al*** y ***alal***

# $\{x,y\}$ : Entre $x$ e $y$ veces



$(?:er){1,2}$  corresponde a **er** y **erer**

**\* : 0 o más veces**



***be\**** corresponde a ***b***, y a ***b*** seguida de todo número de ***e***

# Modos de cuantificación para + y \* : Greedy vs. Lazy

- **Greedy:** No parar hasta la última correspondencia posible (modo por defecto)



Fuente : I. Taylar, [CC-BY](#)



# Modos de cuantificación para + y \* : Greedy vs. Lazy

- **Lazy:** Parar en la correspondencia mínima posible



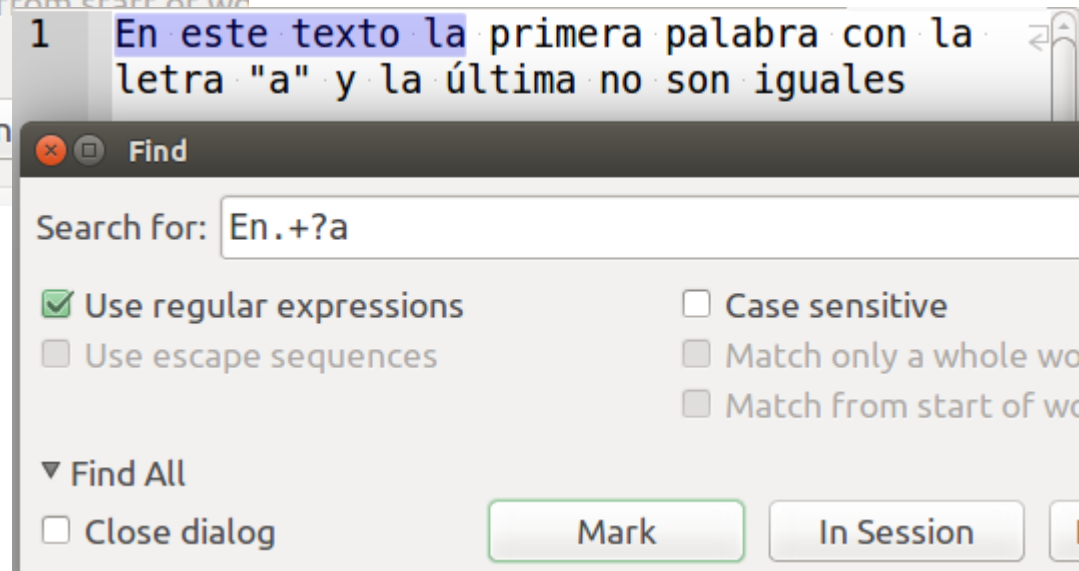
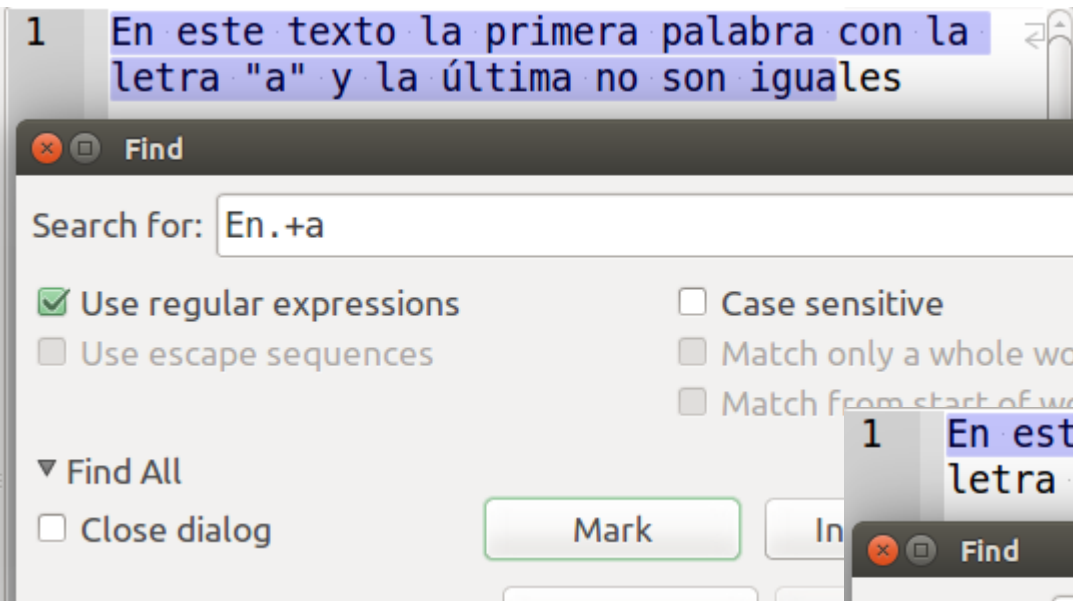
[Fuente](#), por S. Laube (dominio público)

# Matching modes for + and \*:

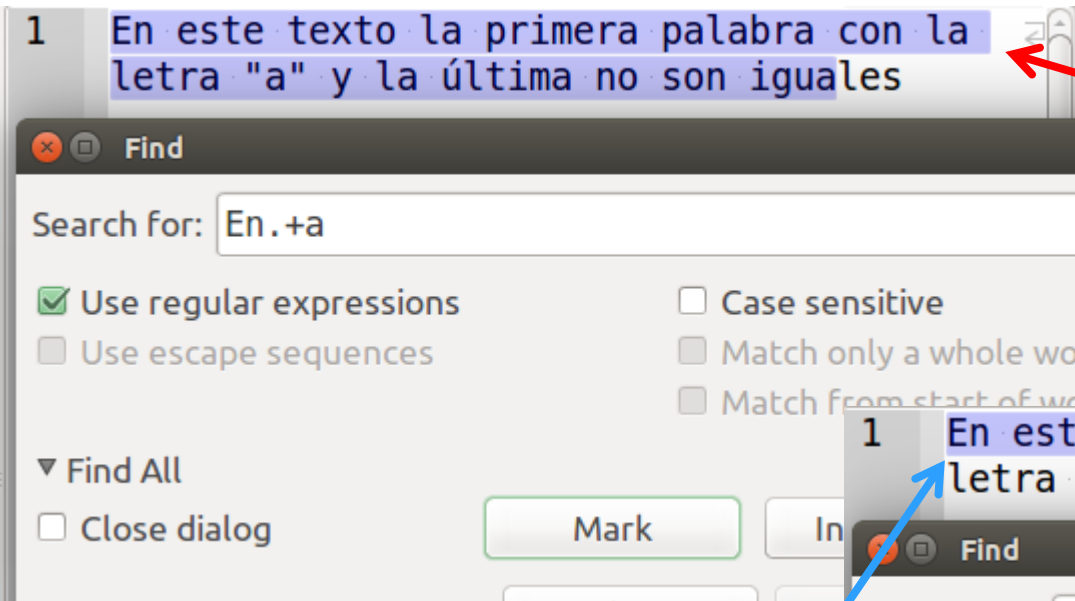
## Greedy vs. Lazy

- **Greedy:** Parar en la última correspondencia posible (**modo por defecto**)
- **Lazy:** Parar en la primera correspondencia posible
- Para hacer que el cuantificador esté en modo lazy, añadir un signo de interrogación:
  - Greedy: +            \*
  - Lazy:    +?        \*?

# Modos de cuantificación para + y \* : Greedy vs. Lazy

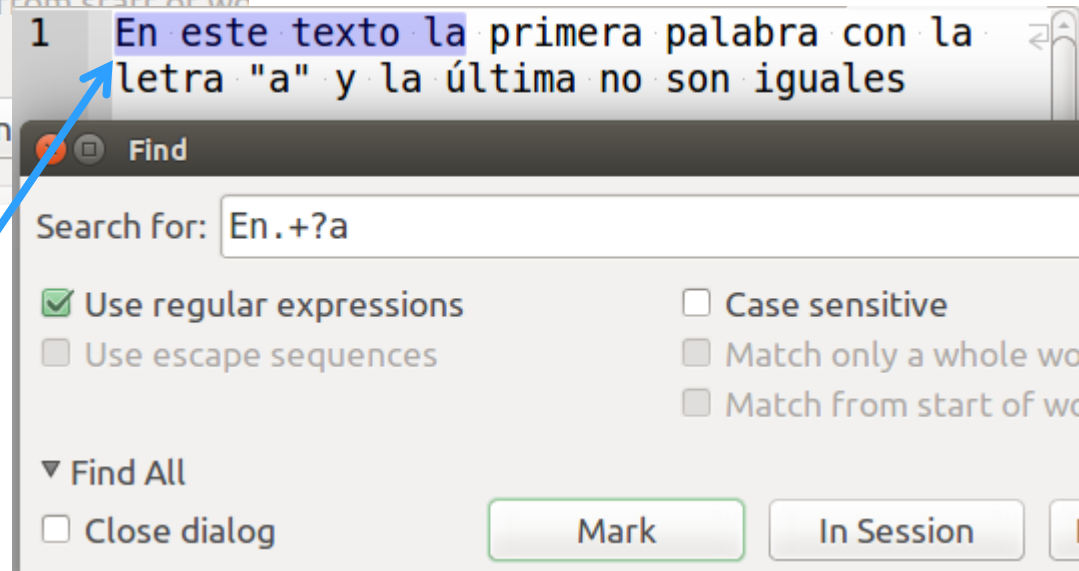


# Modos de cuantificación para + y \* : Greedy vs. Lazy



**Greedy:** Cubre desde En hasta la última ocurrencia de la letra a; sigue aceptando caracteres mientras quede una a más a la derecha en la cadena

**Lazy:** Cubre desde En hasta la primera ocurrencia de la letra a; deja de aceptar caracteres en cuanto encuentra una a

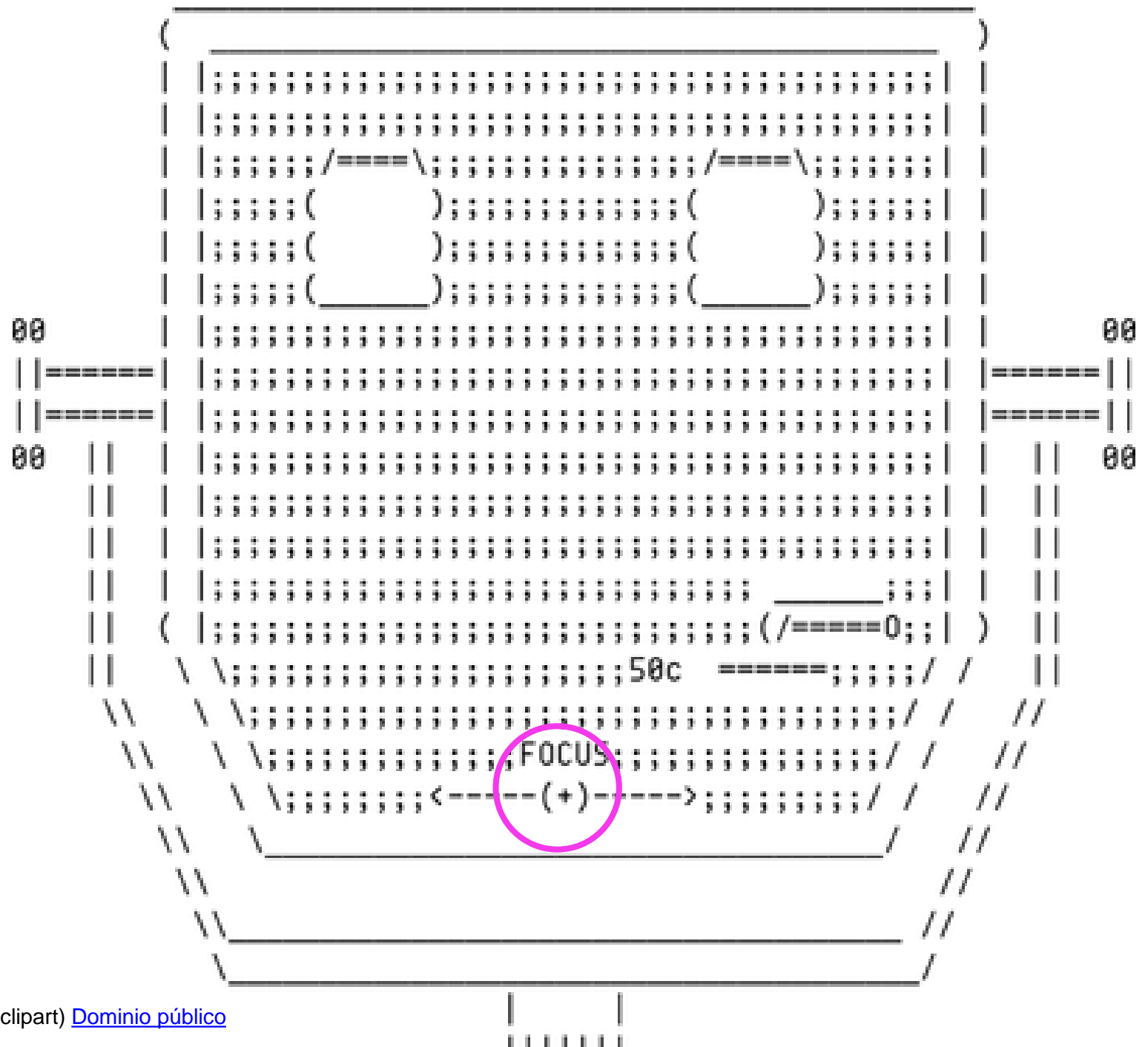


# Cuantificadores: Resumen

Cuantificador	Comportamiento
$x?$	$x$ es opcional (0 or 1 ocurrencias)
$x^*$	$x$ es opcional (0 o cualquier número de ocurrencias)
$x^+$	$x$ ocurre al menos una vez
$x\{y\}$	$x$ ocurre $y$ veces exactamente
$x\{w,z\}$	$x$ ocurre entre $w$ veces y $z$ veces
$*?$ $+?$	El signo ? hace que el cuantificador se comporte de modo lazy
$(?:xxx)$ (Nota: esto no es un cuantificador)	Para hacer que el cuantificador tenga alcance sobre una secuencia de más de un carácter, poner la secuencia dentro de $(?: \quad )$ y añadir el cuantificador tras el paréntesis de cierre

# Escapar Caracteres







# Encontrar caracteres especiales

- ¿Cómo encontrar caracteres especiales?
- Deben ser escapados (precedidos) por el carácter backslash:

**\****x** corresponde al carácter especial **x**

- ¿Cómo encontrar el backslash? (Pista: También es un carácter especial)

# Encontrar caracteres especiales

- ¿Cómo encontrar caracteres especiales?
- Deben ser escapados (precedidos) por el carácter backslash:

**\****x** corresponde al carácter especial **x**

- ¿Cómo encontrar el backslash? (Pista: También es un carácter especial)

**\\** se corresponde con **\**

# Clases de caracteres

# Clases de caracteres:

## Definición

- Clase de caracteres: un conjunto de caracteres
- Una clase de caracteres se define como un conjunto de signos entre corchetes, como **[aeoiuáéíóú]**

# Clases de caracteres:

## Abreviaturas

- Existen formas abreviadas de representar clases de caracteres de uso frecuente:
  - Blancos: **\s** (incluye tabulaciones, espacios y saltos de línea)
  - Números: **\d**
  - Caracteres de palabra: **\w**
    - Nota: \w encuentra caracteres acentuados en muchas aplicaciones, pero no en todas las aplicaciones que soportan expresiones regulares.  
En Geany y notepad++ \w encuentra caracteres acentuados, en SublimeText no

# Abreviatura negada de una clase de caracteres: El complemento de la clase

- La alternancia **mayúscula/minúscula** para la abreviatura de una clase de caracteres representa el **complemento** de la clase
- Dada una clase  $\backslash x$ ,  $\backslash X$  es todo lo que no está en  $\backslash x$  y vice-versa
- $\backslash s$  = blancos ;  $\backslash S$  = no blancos
- $\backslash w$  = caracteres de palabra ;  
 $\backslash W$  no caracteres de palabra
- $\backslash d$  = dígitos ;  $\backslash D$  todo menos un dígito

# Clases de caracteres negadas [ ^ ]

- Para negar una clase, escribir el carácter ^ como el símbolo inicial dentro de los corchetes:
  - [aeiou] Todas las vocales sin acentos
  - [ ^aeiou] Todo menos vocales sin acentos

# Clases de caracteres negadas [^ ]

- Las abreviaturas de clase también pueden ser negadas con el símbolo ^
- Da el mismo resultado que la alternancia entre mayúsculas y minúsculas (slide 38 )
- [^\d] es igual a \D (todo salvo un número)
- [^\s] es igual a \S (todo salvo un blanco)



**Otros caracteres  
importantes**

# Otros caracteres importantes

- `\t` tabulación
- `\n` salto de línea (ocurre al final de la línea)
- `\r` retorno de carro (a veces ocurre al final de la línea, por ejemplo en Windows, el final de línea se representa con `\r\n`)

# Clases de caracteres que corresponden a cada abreviatura

- `\w` `[A-Za-z0-9_]`
- `\W` `[^A-Za-z0-9_]`
- `\s` `[ \t\r\n\v\f]`
- `\S` `[^ \t\r\n\v\f]`
- `\d`

# Clases de caracteres que corresponden a cada abreviatura

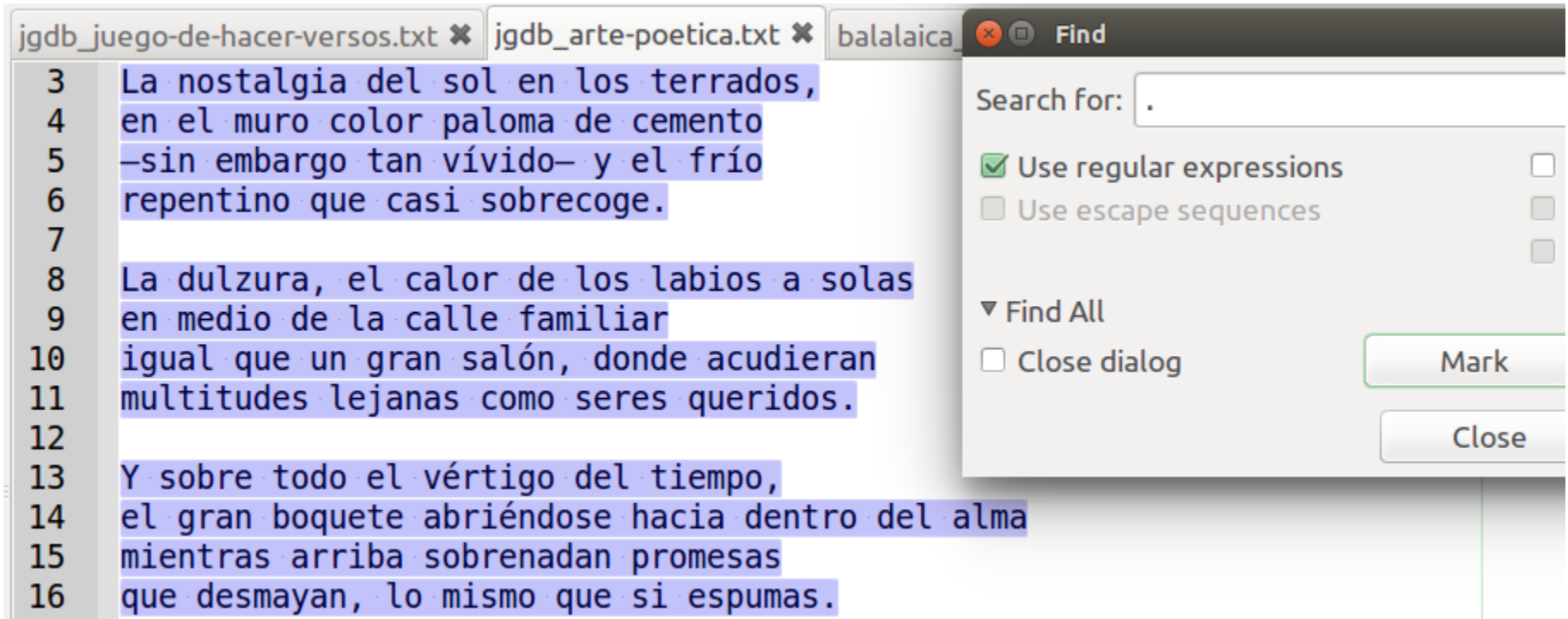
- `\w` `[A-Za-z0-9_]`
- `\W` `[^A-Za-z0-9_]`
- `\s` `[ \t\r\n\v\f]`
- `\S` `[^ \t\r\n\v\f]`
- `\d` `[0-9]`
- `\D`

# Clases de caracteres que corresponden a cada abreviatura

- `\w` `[A-Za-z0-9_]`
- `\W` `[^A-Za-z0-9_]`
- `\s` `[ \t\r\n\v\f]`
- `\S` `[^ \t\r\n\v\f]`
- `\d` `[0-9]`
- `\D` `[^0-9]`

**Comodín universal:**  
**El punto**

# El punto corresponde a (casi) cualquier carácter



- Nota: El punto corresponde a cada uno de los caracteres separadamente.
- Con la función *Mark* en Geany, da la impresión de que cada línea es una correspondencia, pero cada carácter es un match separado (se puede verificar haciendo click en *Next*)
- En [regex101.com](http://regex101.com) se ve más claro que un punto = un carácter

# El punto corresponde a (casi) cualquier carácter

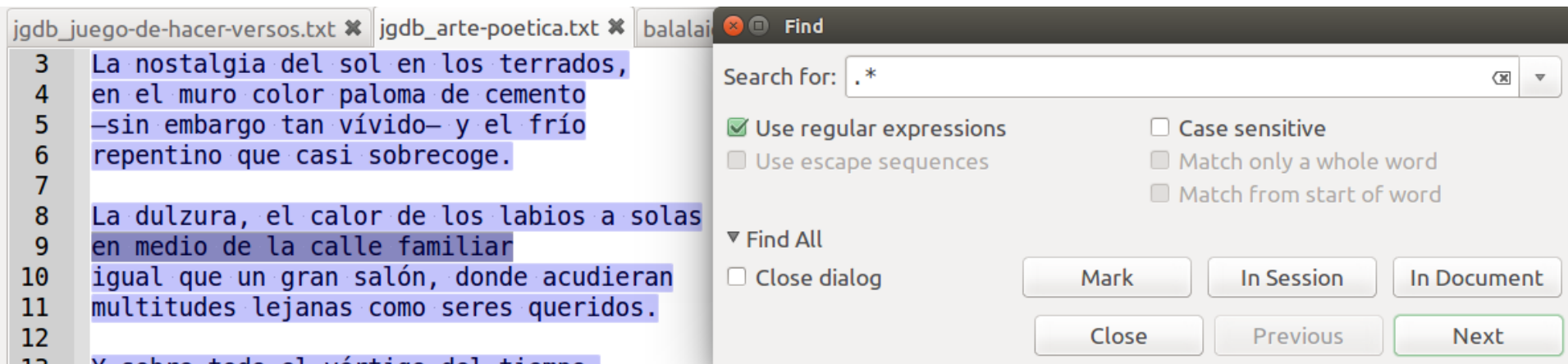
¿Qué expresión correspondería a cada línea  
(La línea entera)?



# El punto corresponde a (casi) cualquier carácter

¿Qué expresión correspondería a cada línea (La línea entera)?

\*



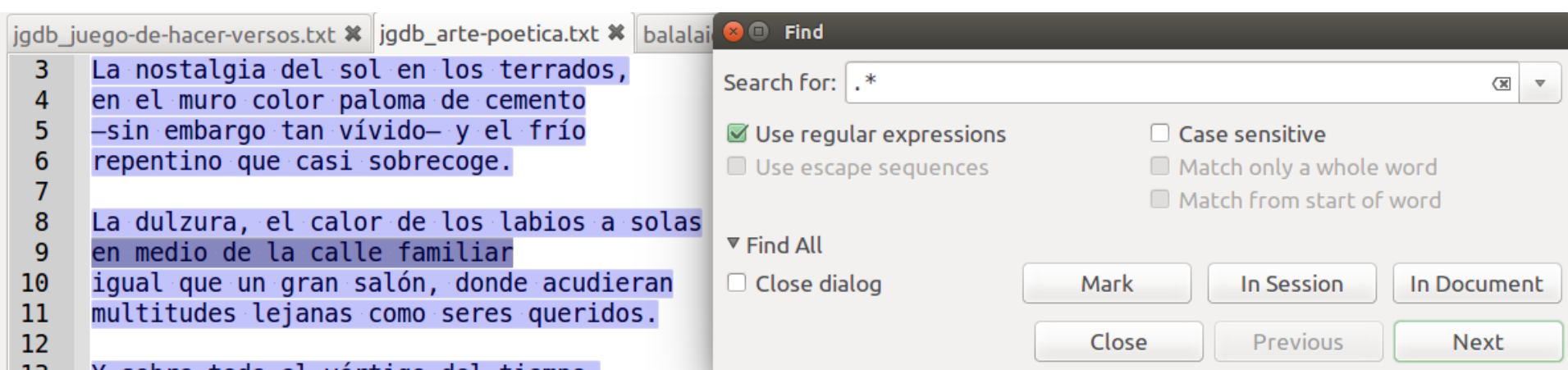
La sombra azul parece igual que para el punto solo, pero haciendo clic en *Next* se ve que ahora es cada LÍNEA lo que se destaca cada vez (no cada carácter)

Por defecto, las regex se evalúan línea a línea

# ¿Por qué *casi* cualquier carácter?

¿Qué expresión correspondería a cada línea  
(La línea entera)?

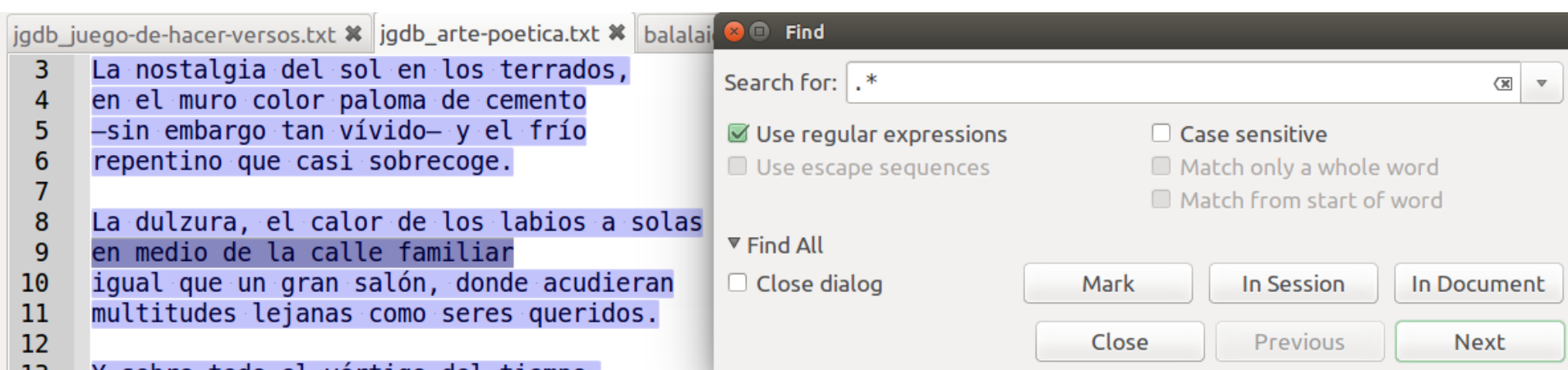
.\*



# ¿Por qué *casi* cualquier carácter?

¿Qué expresión correspondería a cada línea  
(La línea entera)?

.\*



El punto no corresponde a los saltos de línea

**Ancclas (anchors)  
y fronteras de palabra**

# Anclas

- No corresponden a ningún carácter, sino a posiciones en un texto
- El patrón a buscar queda “anclado” a una posición
  - ^ principio de línea
  - \$ final de línea

## Nota: Un símbolo, diferentes significados

- El signo ^ puede significar una de dos cosas en las regexes

## Nota: Un símbolo, diferentes significados

- El signo  $\wedge$  puede significar una de dos cosas en las regexes
  - Dentro de una clase de caracteres, niega la clase ( $[\wedge xyz]$  = todo menos xyz, ver slide 39)
  - En los demás contextos, significa el principio de la línea
- Como en los lenguajes naturales, el contexto desambigua un signo

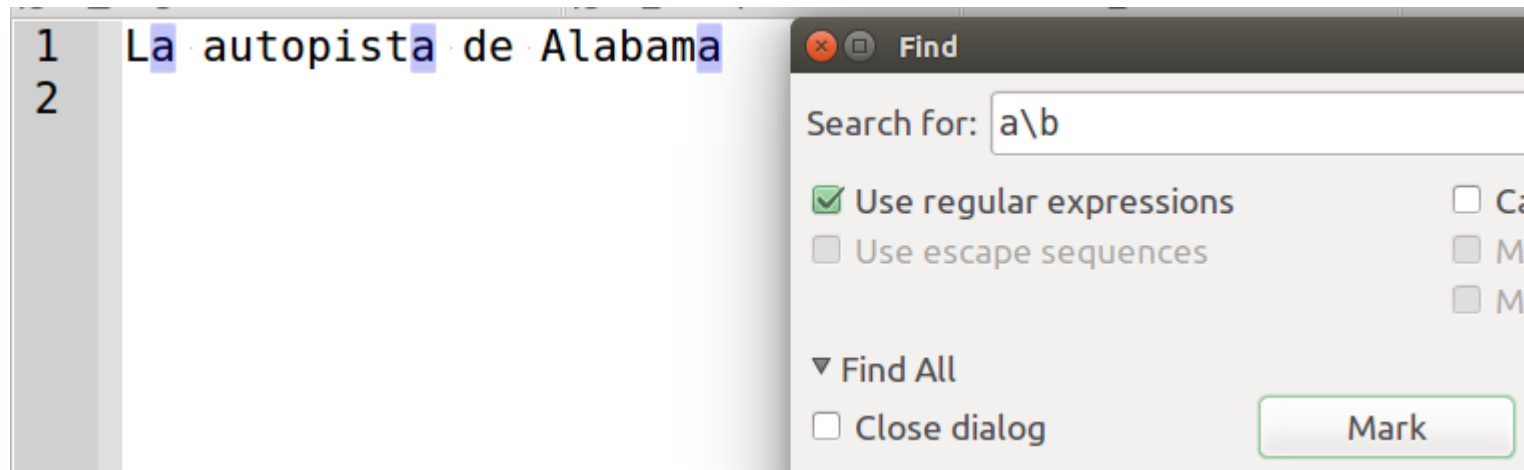
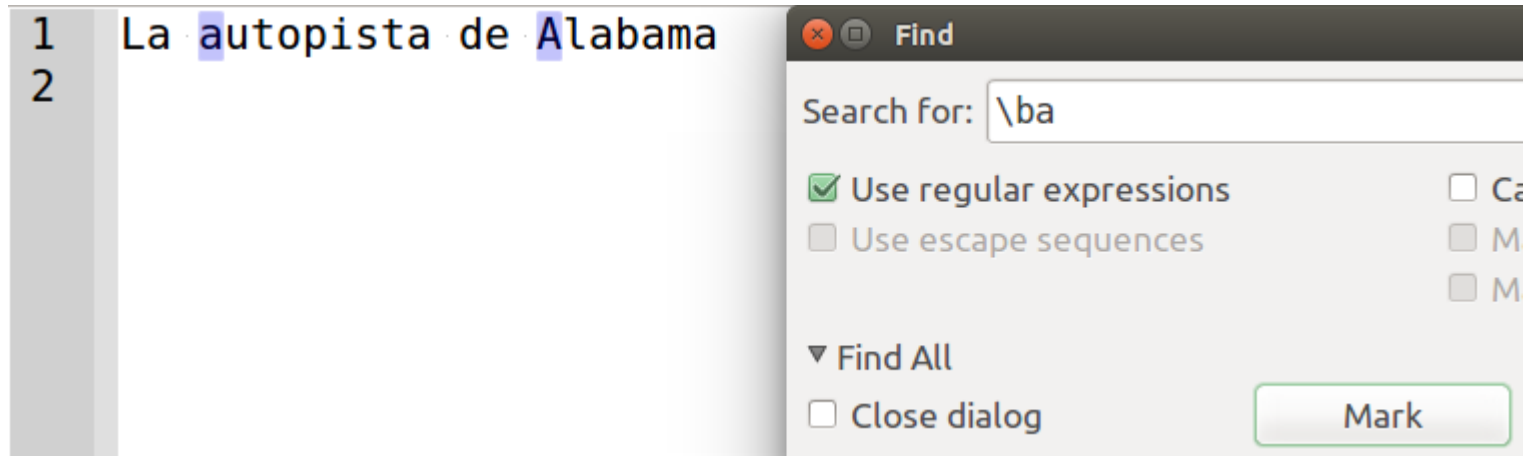
# Fronteras de palabra

- (1) `\b`      Ancla para el principio o final de una palabra
  - (2) `\<`      Ancla para el principio de una palabra
  - (3) `\>`      Ancla para el final de una palabra
- No corresponden a un patrón, pero indican la posición donde debe ser encontrado el patrón, con respecto al principio/fin de las palabras
  - (2) y (3) no están disponibles en todas las aplicaciones (Geany no los usa, pero Sublime Text y Notepad++ sí)



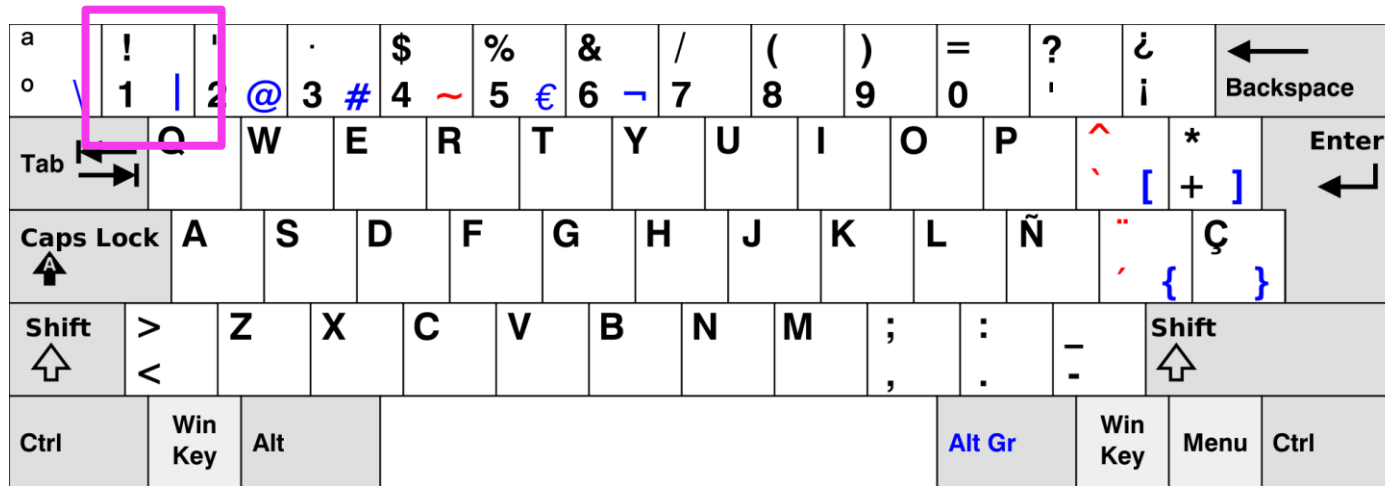
# Fronteras de palabra

Correspondencias a principio o fin de palabra

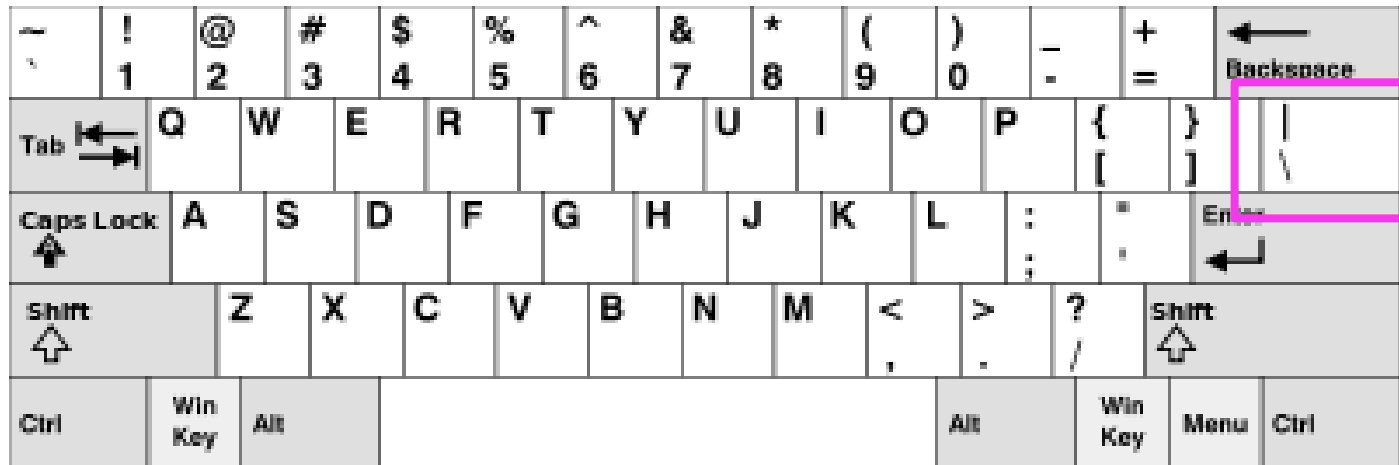


# Alternancias

# El carácter | (“pipe”) introduce una alternancia

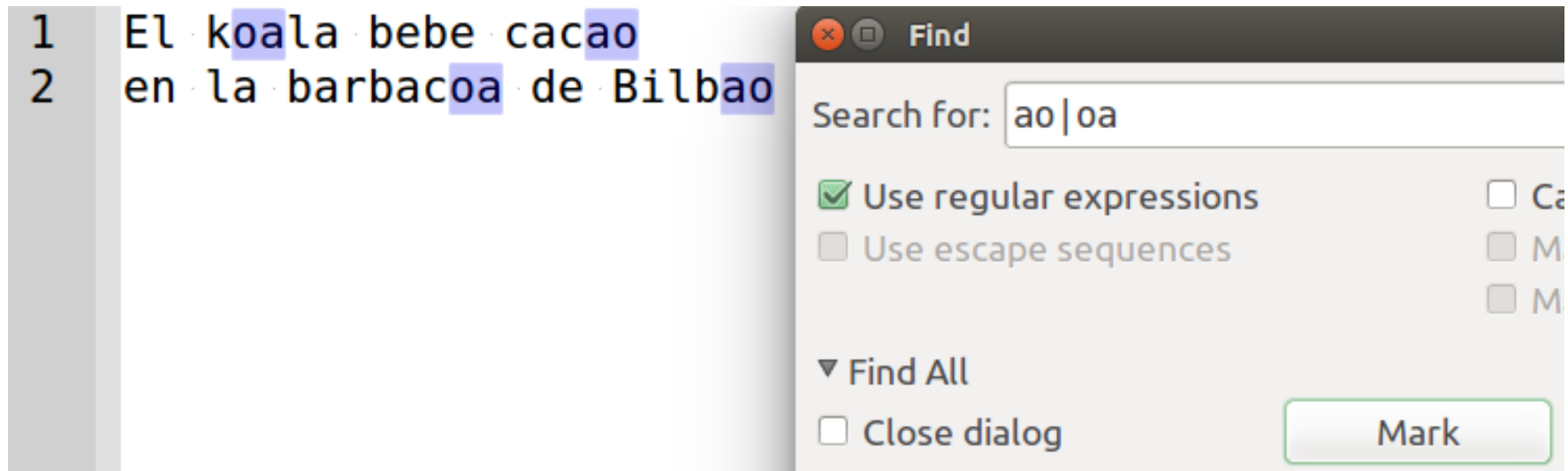


Teclado  
de España  
Alt Gr + 1



Teclado  
de USA

# | Alternancia con “pipe”



***ao|oa*** corresponde a ***ao*** y ***oa***

# **Grupos, Referencias y Sustituciones**

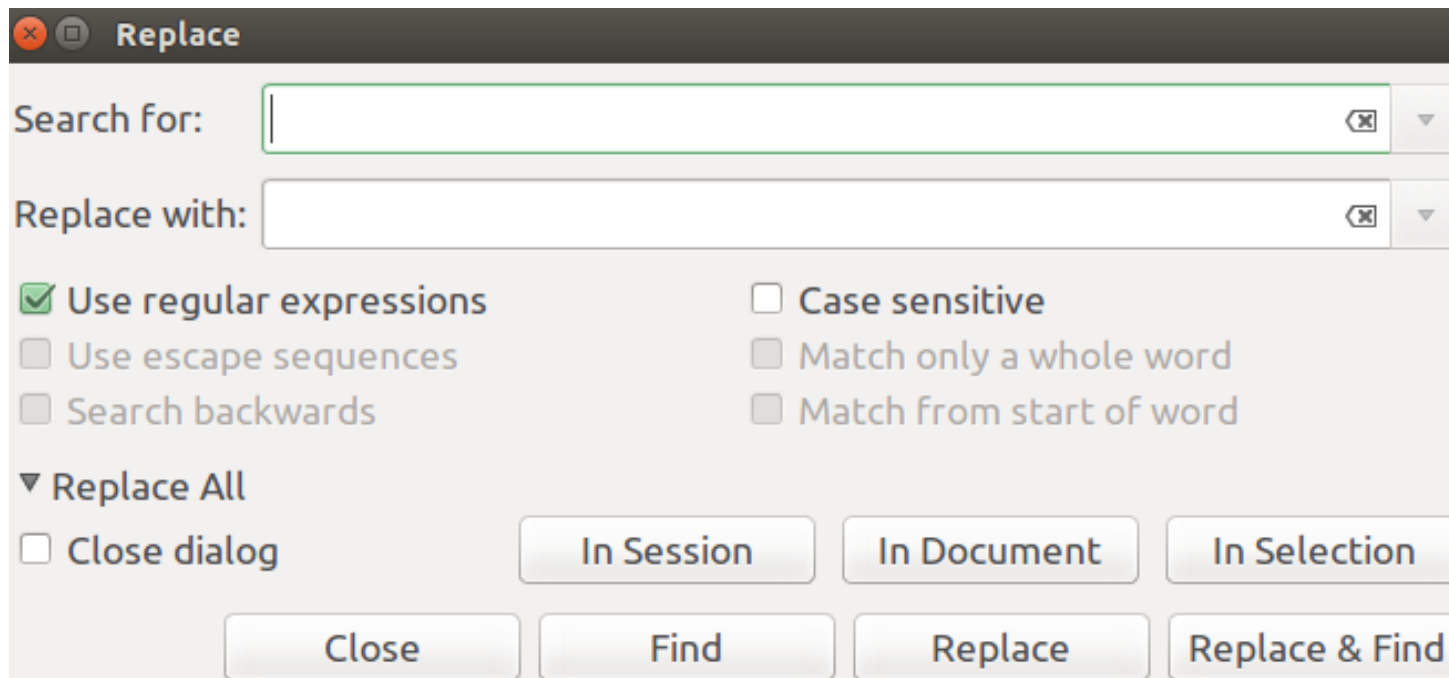
# () Grupos y referencias:

## Reemplazar texto

- Una regex entre **()** es un **grupo**
  - Usando un grupo, almacenamos en el grupo el texto que corresponde a la expresión
  - Normalmente hay un máximo de 9 grupos
- Para acceder al contenido de cada grupo, usamos una **referencia** de **\1** a **\9** (ver abajo)
- Nota: También existen grupos de **no captura**, definidos con **(?: )**
  - Definen el alcance de un operador, pero no almacenan contenido
  - Se usan cuando no necesitamos memorizar (almacenar) el texto que corresponde a un grupo

# () Grupos y referencias: Sustituir texto

Ctrl + H



The image shows a 'Replace' dialog box with a dark title bar. It contains two text input fields: 'Search for:' and 'Replace with:'. Below these are several checkboxes: 'Use regular expressions' (checked), 'Use escape sequences', 'Search backwards', 'Case sensitive', 'Match only a whole word', and 'Match from start of word'. There is a 'Replace All' section with a dropdown arrow and a 'Close dialog' checkbox. At the bottom are four buttons: 'Close', 'Find', 'Replace', and 'Replace & Find'.

Replace

Search for:

Replace with:

☒ Use regular expressions ☐ Case sensitive

☐ Use escape sequences ☐ Match only a whole word

☐ Search backwards ☐ Match from start of word

▼ Replace All

☐ Close dialog

In Session In Document In Selection

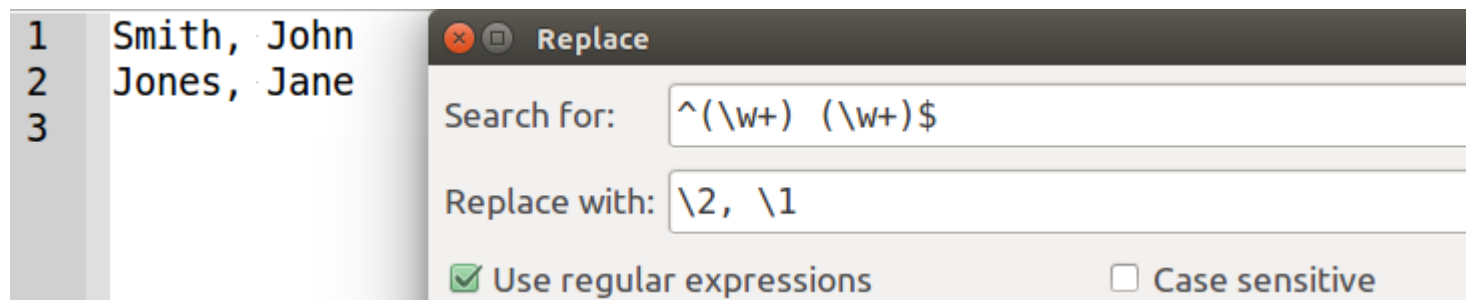
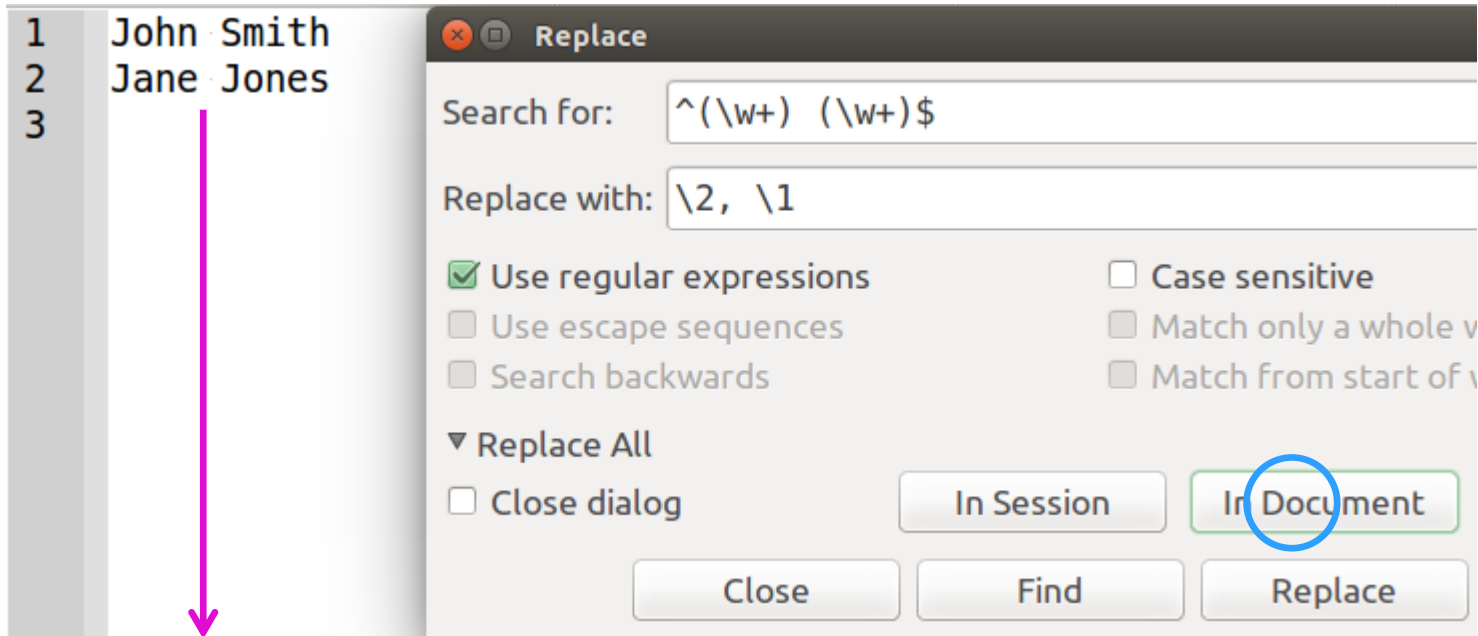
Close Find Replace Replace & Find

# () Grupos y referencias: Sustituir texto

- Ejemplo
  - Cambiar una lista de nombres del formato  
**Nombre Apellido**  
al formato  
**Apellido, Nombre**
  - E.g. *John Smith* se sustituye por *Smith, John*



# () Grupos y referencias: Sustituir texto



John Smith

Grupo 1 es *John*

Grupo 2 es *Smith*

$\wedge (\backslash w+) (\backslash w+) \$$

$\backslash 2, \backslash 1$

Smith, John

Punto de partida: **John Smith**

Objetivo: **Smith, John**

John Smith

$^(\backslash w+) (\backslash w+)\$$

Punto de partida: **John Smith**

Objetivo: **Smith, John**

John Smith

Grupo 1 es *John*

$\wedge(\backslash w+)\ (\backslash w+)\$$

Punto de partida: **John Smith**

Objetivo: **Smith, John**

John Smith

Grupo 1 es *John*

Grupo 2 es *Smith*

$\wedge (\backslash w+) (\backslash w+) \$$

Punto de partida: **John Smith**

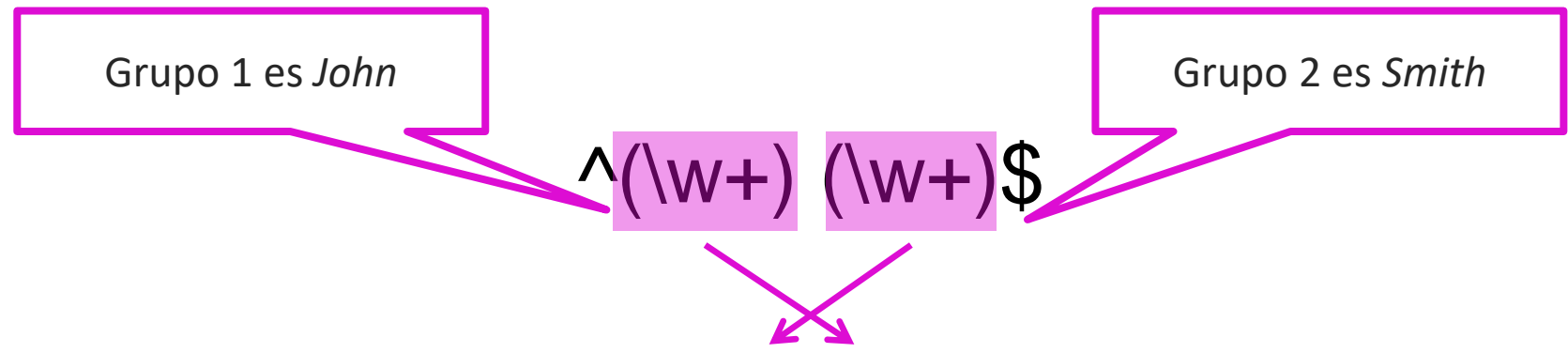
Objetivo: **Smith, John**

John Smith

Grupo 1 es *John*

Grupo 2 es *Smith*

$\wedge (\backslash w+) (\backslash w+) \$$



Punto de partida: **John Smith**

Objetivo: **Smith, John**

John Smith

Grupo 1 es *John*

Grupo 2 es *Smith*

$\wedge (\backslash w+) (\backslash w+) \$$

$\backslash 2 \backslash 1$

Punto de partida: **John Smith**

Objetivo: **Smith, John**

John Smith

Grupo 1 es *John*

Grupo 2 es *Smith*

$\wedge (\backslash w+) (\backslash w+) \$$

$\backslash 2, \backslash 1$



Punto de partida: **John Smith**

Objetivo: **Smith, John**

John Smith

Grupo 1 es *John*

Grupo 2 es *Smith*

$\wedge (\backslash w+) (\backslash w+) \$$

$\backslash 2, \backslash 1$

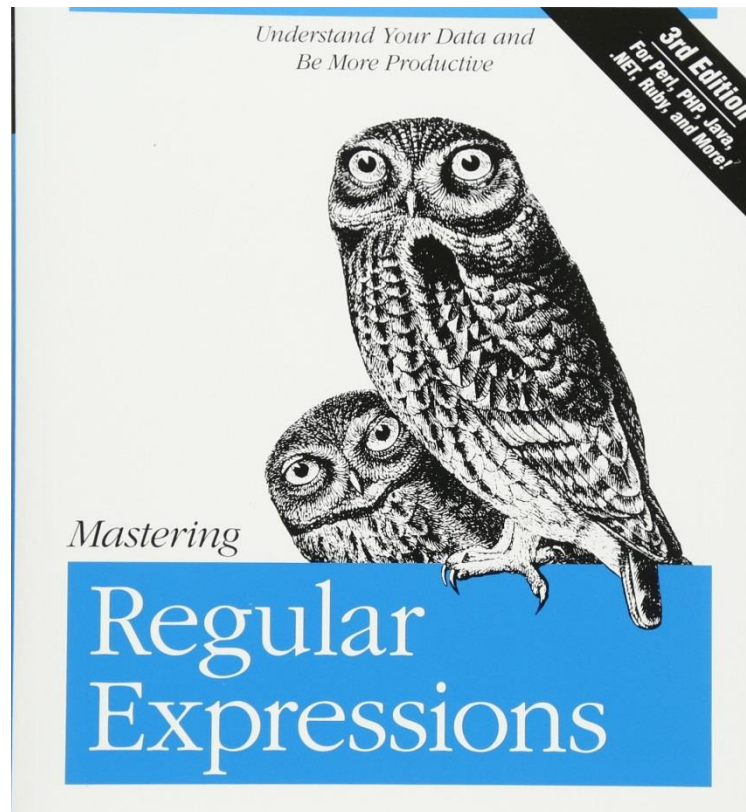
Smith, John

# Información adicional

- Operadores de lookahead
  - lookahead positivo y negativo
    - *a* seguido/no seguido por *b*, **sin incluir *b* en el match**
    - $a(?=b)$                        $a(?!b)$
  - lookbehind positivo y negativo
    - $(?<=b)a$                        $(?<!b)a$
    - *a* precedido/no precedido por *b*, **sin incluir *b* en el match**
- Optimización
  - $a[^b]^*b$  es más rápido que  $a.+?b$  para buscar «*a* seguido de cualquier otro carácter (o nada) hasta la primera ocurrencia de *b*»

# Lectura adicional

- <http://www.regular-expressions.info/>
- Friedl, J.E.F. *Mastering Regular Expressions*. O'Reilly.



# Ejercicios

- Se pueden probar las regex en un editor de texto, o en los sitios de la diapositiva 6
- Por ejemplo, [regex101.com](http://regex101.com) (diapositiva siguiente), da información sobre la secuencia que corresponde a cada grupo de captura en la regex

# regex101.com

The screenshot shows the regex101.com website interface. The browser's address bar displays "Secure | https://regex101.com". The website's header includes the "regular expressions 101" logo and navigation links for "@regex101", "donate", "contact", "bug reports & feedback", and "wiki".

The main content area is divided into three sections:

- REGULAR EXPRESSION**: A text input field contains the regular expression `... / (.) (.) (.) / g`. A status bar indicates "1 match, 12 steps (~0ms)".
- TEST STRING**: A text input field contains the string "a b c". A button labeled "SWITCH TO UNIT TESTS" is located to the right of the input field.
- EXPLANATION**: A dropdown menu is expanded, showing the breakdown of the regular expression `/ (.) (.) (.) / g`. It lists three capturing groups:
  - 1st Capturing Group (.)**: matches any character (except for line terminators).
  - 2nd Capturing Group (.)**: matches any character (except for line terminators).
  - 3rd Capturing Group (.)**: matches any character (except for line terminators).
- MATCH INFORMATION**: A dropdown menu is expanded, showing the match results for "Match 1":

Match	Full match	Group 1.	Group 2.	Group 3.
Match 1	0-5 `a b c`	0-1 `a`	2-3 `b`	4-5 `c`

**Pablo Ruiz Fabo**  
**ruizfabo@unistra.fr**

