

# Regular expressions for translators and interpreters

Constantin Orasan

*Helped by:* Diptesh Kanojia, Hadeel  
Saadany, Leonardo Zilio, Shenbin Qian

28<sup>th</sup> March 2022

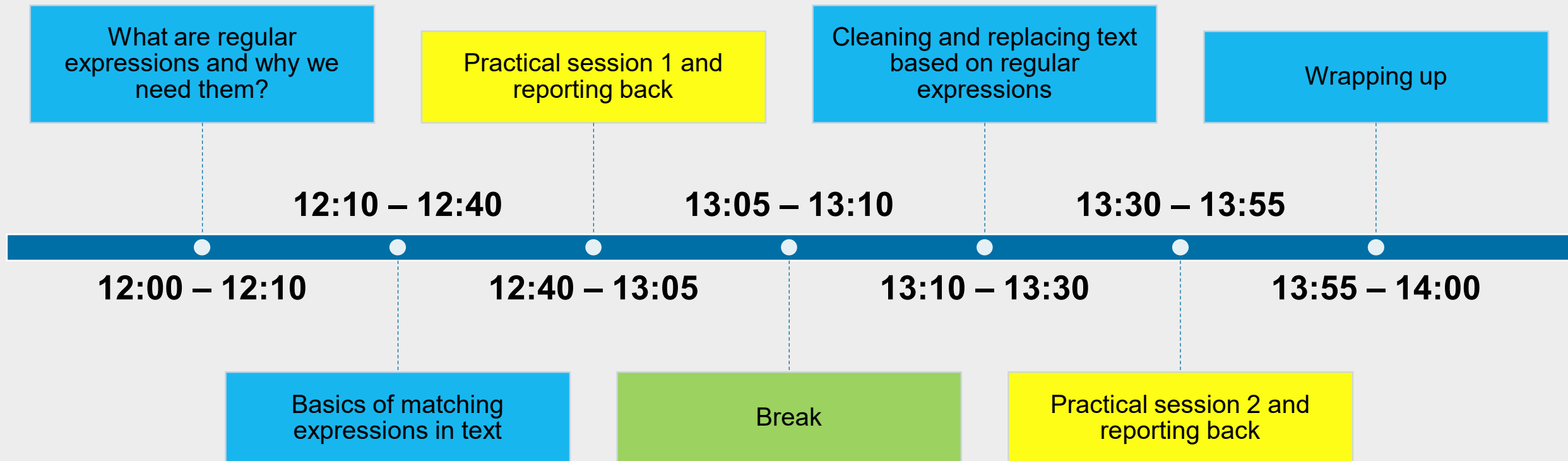


CENTRE FOR  
TRANSLATION  
STUDIES

UNIVERSITY OF SURREY

While waiting please open the following link  
[PollEv.com/corasan](https://PollEv.com/corasan)

# Planned structure for today's session



**Part 1: What  
are regular  
expressions?**



# How well do you know regular expressions?

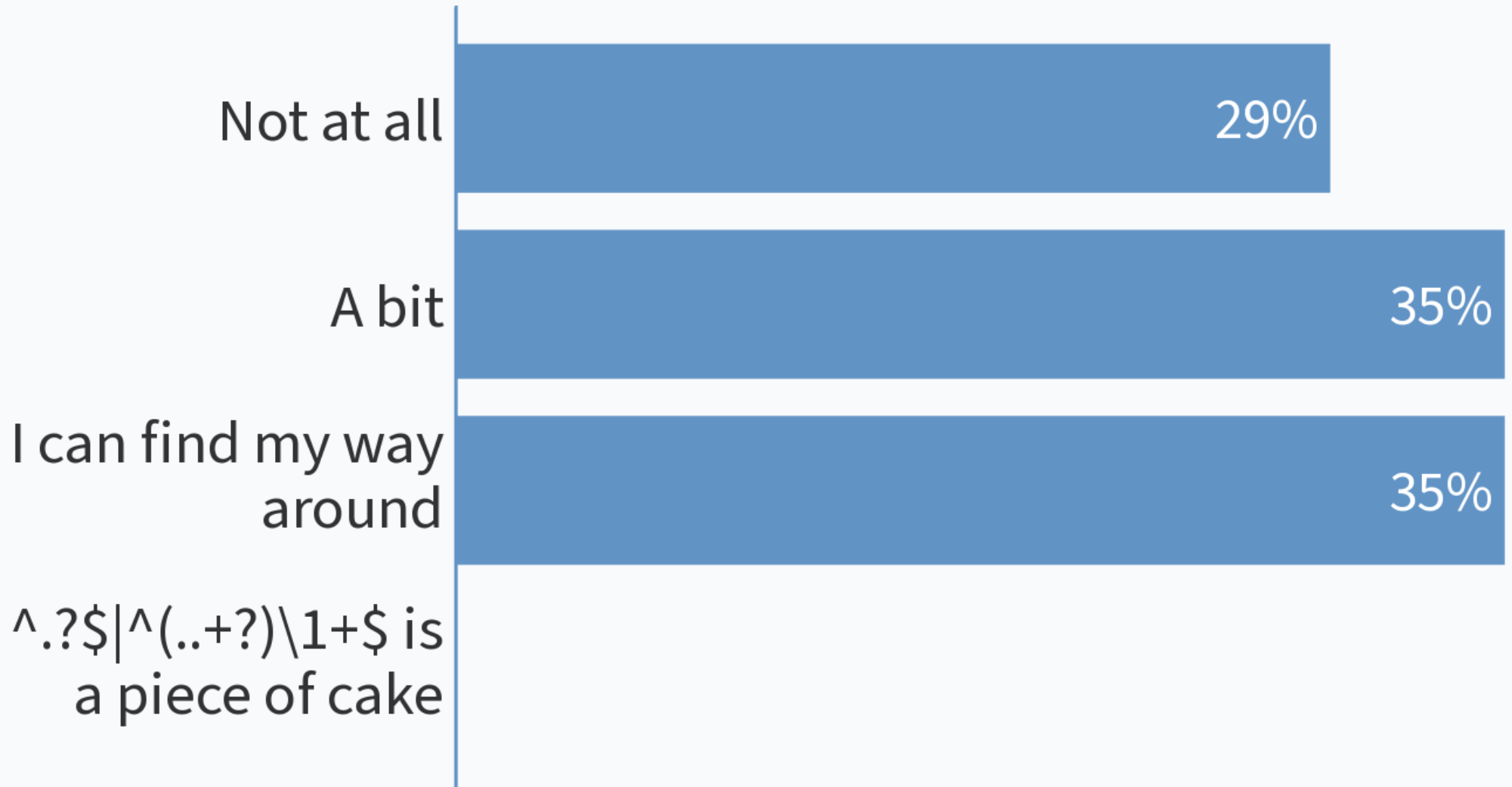
Not at all

A bit

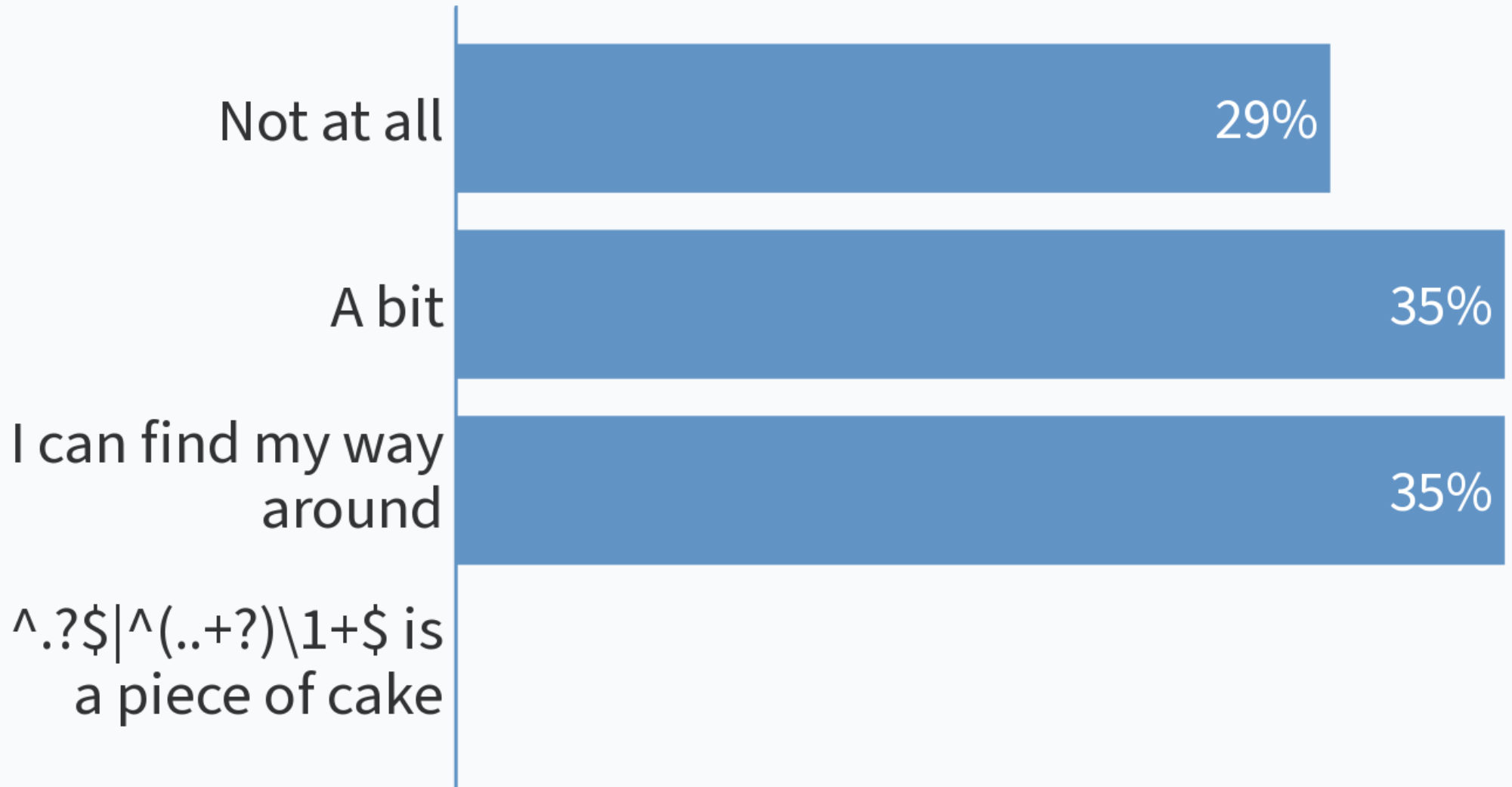
I can find my way around

`^.{?}$|^(..+?)\1+$` is a piece of cake

# How well do you know regular expressions?

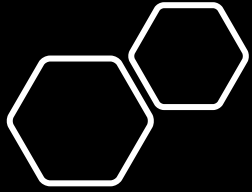


# How well do you know regular expressions?



# What are regular expressions?

- » “search-and-replace function on steroids”
- » allow to assess whether a text contains a certain sequence of characters (**matches the pattern**)
- » sometimes referred as **wildcard characters**
  
- » Examples of use:
  - Search for several words/forms of words (e.g. singular and plural) at once
  - Search for different forms of the same word (e.g. *London-based* vs *London based*)
  - Filter texts that fulfils certain conditions
  - Clean a corpus of text built from the web
  - Convert numerical expressions between language specific representations (e.g. 1.45 vs 1,45)
  - Extensively used behind the scenes by CAT tools (e.g. recognise various expressions, split text into words)



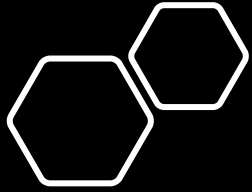
# Regular expressions in Trados

» Using “Advanced display filters” we can filter segments based on conditions applied to source and target segments

The screenshot displays the Trados software interface. The main window is titled "Translation Results - English-German" and shows a list of segments. The "Advanced Display Filter" dialog box is open, highlighting the "Content" tab. The "Source" field contains the regular expression `^[1]{1,2};[0-9]{2} [AP]M`. The "Regular Expression" checkbox is checked, and the "Case Sensitive" checkbox is unchecked. The "Filters applied" section shows the source filter as `Source:"^[1]{1,2};[0-9]{2} [AP]M"; Regular Expression:"True"`. The status bar at the bottom indicates that 6 segments are filtered out of 43 total segments.

Segment ID	Source	Target
1	11:00 AM - 11:30 AM	AT 11:00 - 11:30
17	11:00 AM - 11:30 AM	AT 11:00 - 11:30
19	11:30 AM - 1:00 PM	
21	1:00 PM - 2:00 PM	
37	11:00 AM - 11:30 AM	
39	11:30 AM - 1:00 PM	
42	1:00 PM - 2:00 PM	





# Regular expressions in Trados

» Using “Advanced display filters” we can filter segments based on conditions applied to source and target segments

The screenshot shows the 'Advanced Display Filter' dialog box in Trados. The 'Source' field contains the regular expression `^([1,2]:[0-9]{2}) [AP]M`. The 'Regular Expression' checkbox is checked. The status bar at the bottom indicates 'Filtered 6 of 43 segments'.


Filters applied:  
Source: "^([1,2]:[0-9]{2}) [AP]M"; Regular Expression: "True"













Filtered 6 of 43 segments

# Regular expressions in SketchEngine








CONCORDANCE Account expires in January 2023 »  
Get more space +

British National Corpus (BNC) i

Written Medium Periodi... x CQL [word = "colou?r"] • 4,177  
128.89 per million tokens • 0.0037% i 

            KWIC + i ☆

Details Left context KWIC Right context

21	<input type="checkbox"/>	<span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">i</span> Written books a... ing and dying the paper a copper <b>colour</b> . </s><s> Bhimji had used art docu 
22	<input type="checkbox"/>	<span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">i</span> Written books a... the partial form and the affects of <b>colour</b> on a flat place. </s><s> Her conter 
23	<input type="checkbox"/>	<span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">i</span> Written books a... ary works are energetic, vibrant in <b>colour</b> and convey to the viewer the simul 
24	<input type="checkbox"/>	<span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">i</span> Written books a... s were working class or women of <b>colour</b> . </s><s> Feminism for Beginners 
25	<input type="checkbox"/>	<span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">i</span> Written books a... ta Dart. </s><s> While both these <b>colour</b> books provide hours of pleasant br 
26	<input type="checkbox"/>	<span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">i</span> Written books a... 'temper amento mediterráneo, de <u>color</u> , de entusiasmo'. </s><s> Gallic re 
27	<input type="checkbox"/>	<span style="border: 1px solid #ccc; border-radius: 50%; padding: 2px;">i</span> Written books a... t the performance style and vocal <b>colour</b> which these groups consciously or 



# Part 2: Matching expressions in text

---



# What are regular expressions

- » a special sequence of characters that specifies a **search pattern** in text
- » has a specialised syntax
- » it is a “programming” language on its own (and there are several varieties of it)
- » we will use the .NET flavour of regular expressions which is widely used (including by SDL Trados)
  
- » we will start by using <https://regexr.com/>

**Notation:** I will use `...` notation to represent regular expressions

(e.g. `text` or  
`(.*)@(.*)\.com`)

# Matching an exact string

- » A string matches itself (i.e. simple find string operation)
- » The matching is case sensitive  
`butter` vs `Butter`
- » But some characters have special meaning and they have to be treated specially

```
Expression
/butter/g

Text Tests

Uri: https://www.anchorbutter.co.uk/butter/
File Name: C:\Downloaded Web Sites\www.anchorbutter.co.uk\butter\index.htm
Content Type: text/html
-
Uri: https://www.anchorbutter.co.uk/cookie-policy/
File Name: C:\Downloaded Web Sites\www.anchorbutter.co.uk\cookie-policy\index.htm
Content Type: text/html
-
Uri: https://www.anchorbutter.co.uk/cream/
File Name: C:\Downloaded Web Sites\www.anchorbutter.co.uk\cream\index.htm
Content Type: text/html
-
Butter ←
```

# Escaping special characters

- » If we want to match `\butter\` we need to have `\\butter\\`
- » Notice the `\\`. We need to escape character `\` using `\`
- » If the characters have a special meaning (meta-characters) we need to escape them in order to match them (e.g. `\.`, `\[`, `\(`, etc.)

```
Expression
/\\butter\\/g

Text Tests

Uri: https://www.anchorbutter.co.uk/butter/
File Name: C:\Downloaded Web Sites\www.anchorbutter.co.uk\butter\index.htm
Content Type: text/html
-
Uri: https://www.anchorbutter.co.uk/cookie-policy/
File Name: C:\Downloaded Web Sites\www.anchorbutter.co.uk\cookie-policy\index.htm
Content Type: text/html
-
Uri: https://www.anchorbutter.co.uk/cream/
File Name: C:\Downloaded Web Sites\www.anchorbutter.co.uk\cream\index.htm
Content Type: text/html
-
Butter
```

# Meta-characters

- » The power of regular expressions comes from meta-characters
- » the meta-character `.` (dot) will match any **single** character
- » to match the `.` (dot) character we need to escape it `\.`

Expression

```
/./g
```

Text Tests **NEW**

RegExp was created by gskinner.com, and  
Edit the Expression & Text to see match  
PCRE & JavaScript flavors of RegExp are  
mode.

Expression

```
/\./g
```

Text Tests **NEW**

RegExp was created by gskinner.com, and  
Edit the Expression & Text to see match  
PCRE & JavaScript flavors of RegExp are  
mode.

# Matching sets of characters

- » the meta-characters `[` and `]` will indicate a set of characters to match
  - can either enumerate the characters individually `[abcd]`
  - can indicate a range `[a-d]`
  - it will match only **one character** from the list/range
- » meta-characters listed inside `[` and `]` lose their special nature and are treated as simple characters. e.g. `[ab.]` matches a, b or .
- » if we want to match `-` in the set we need to put it first to avoid declaring a range `[a-c]` vs `[-ac]`
- » `^` will indicate which characters not to match if it appears first after `[` e.g. `^[a-c]` will match anything but a, b or c.
- » if we want to match a string which does not contain `-` we have `[-]`



## Select the expressions matched by *blt-[0-9]*

blt

BLT

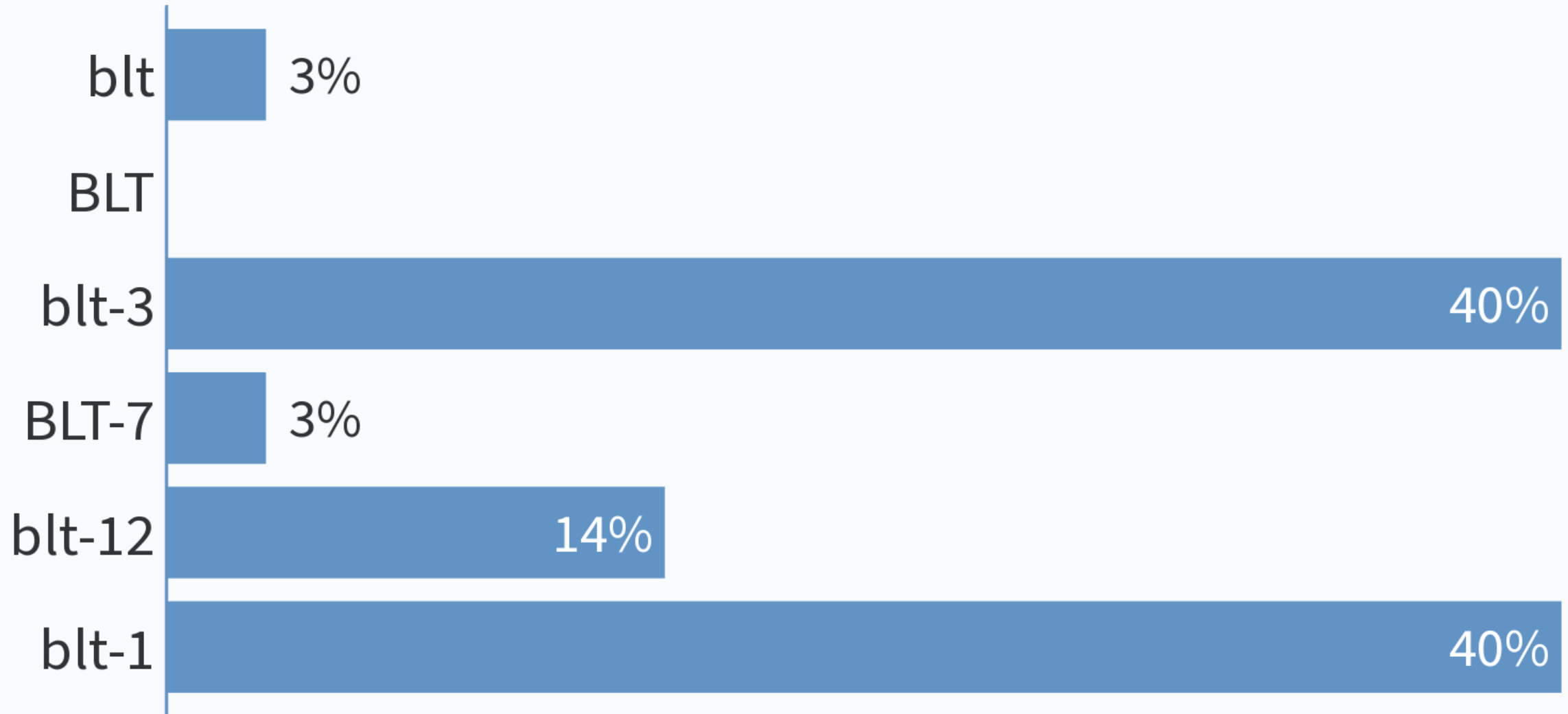
blt-3

BLT-7

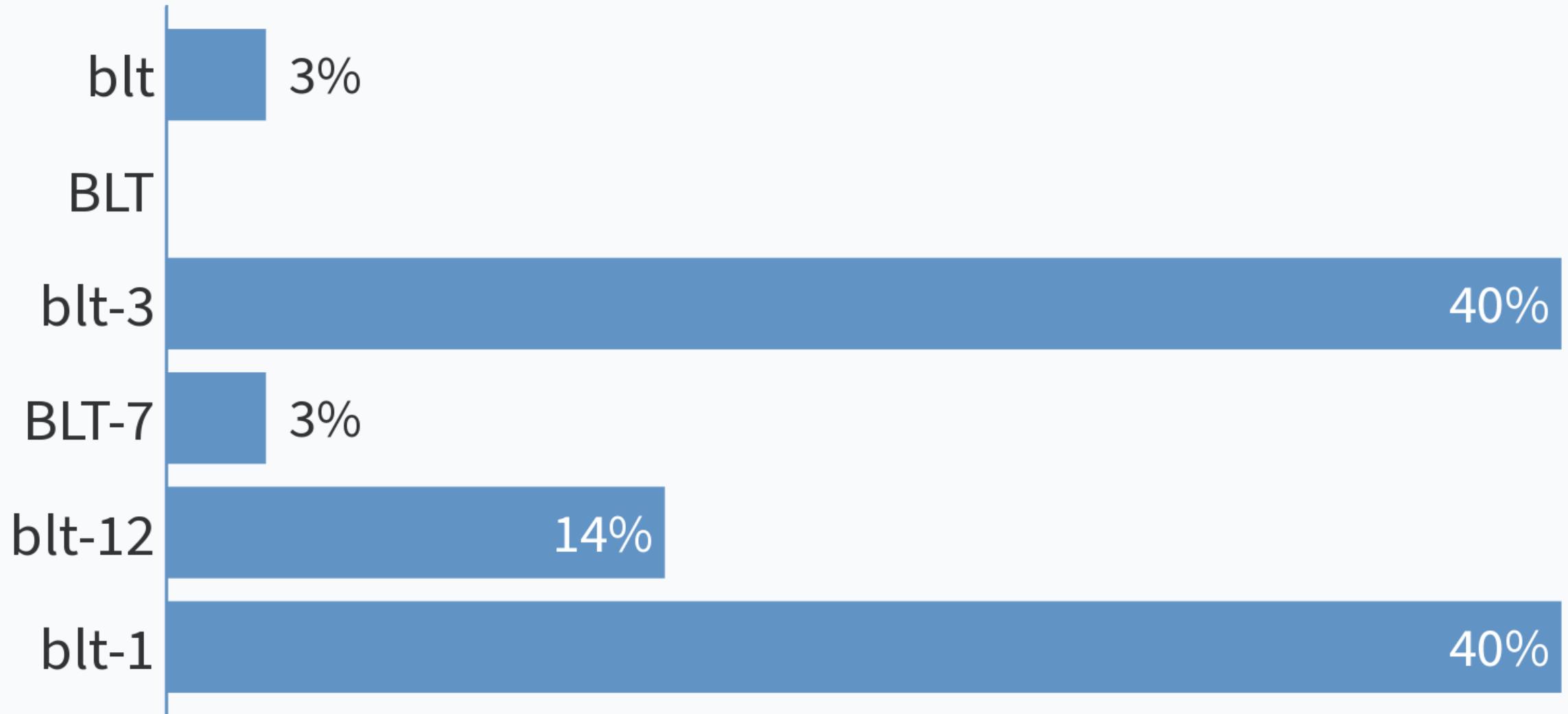
blt-12

blt-1

## Select the expressions matched by *blt-[0-9]*



## Select the expressions matched by *blt-[0-9]*



## Expression

JavaScript ▾

```
/blt-[0-9]/g
```

Text

Tests

4 mat

```
Uri: https://www.anchorbutter.co.uk/globalassets/images/food-ideas/blt-2.jpg  
File Name: C:\Downloaded Web Sites\www.anchorbutter.co.uk\globalassets\images\food-ideas\blt-2.jpg  
Content Type: image/jpeg  
  
Uri: https://www.anchorbutter.co.uk/globalassets/images/food-ideas/blt-3.jpg?  
...=500%h=400&quality=70&mode=crop
```

# Examples

- » `ABCD` matches the string *ABCD* , but not *AB1D*
- » `AB.D` matches both *ABCD* and *AB1D* because `.` matches any character.
- » `AB[A-D]D` matches the following strings *ABAD* , *ABBD* , *ABCD* , *ABDD* but nothing else.
- » `1.1` matches *101*, *111*, *1,1*, *1a1*, ...
- » `summar[is]e` matches both *summarise* and *summarize*
- » `20[01][0-9]` matches years between 2000 and 2019
  
- » Write in [pollev.com/corasan](https://pollev.com/corasan) the regular expression which matches both *gray* and *grey*

Write an expression which matches both *gray* and *grey*

gr[ea]y \gr[ae]y\  
/gr[ae]y/ [a-b]gr(a|e)y  
gr[ae]y  
/gr[a-e]y/

# Repeating sequences

- » \* repeats an expression 0 or unspecified number of times e.g. `a*` matches a sequence of 0 or many letters a
- » + repeats an expression 1 or more times e.g. `a+` matches a sequence of 1 or many letters a
- » ? repeats an expression 0 or 1 times. Indicates something optional. e.g. `home-?brew` matches either *homebrew* or *home-brew*.
- » {n} where *n* is a number which indicates that an expression appears exactly *n* times. e.g. `a{3}` matches *aaa*
- » {m, n} where *m* and *n* are integer repeats an expression at least *m* times and at most *n* times. If *n* is missing it is considered unlimited.

# | (OR operator)

- » | is the *or* operator: defines alternative options
- » It has very low priority, so you may need to use parenthesis to adjust the priority of the operations. For example if we want to match both organization and organisation we can have `organi(s|z)ation`.

Expression

```
/organi(s|z)ation/g
```

Text Tests **NEW**

organisation  
organization

Expression

```
/organis|zation/g
```

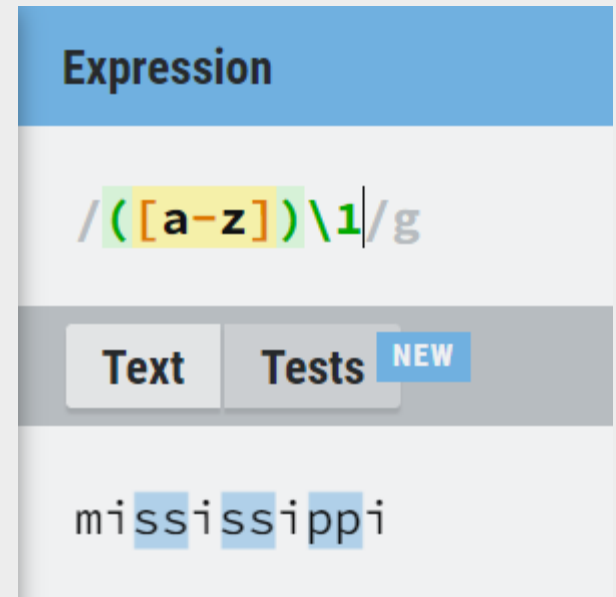
Text Tests **NEW**

organisation  
organization



# Creating groups

- » Groups are marked by ( and )
- » Groups are used to
  - Group things together
  - Retrieve specific parts of the matched string
  - Set the priority of matching
- » It is possible to refer to a group by using \1, \2. **Note:** counting starts from 1 and you need to count the number of ( opened.



The screenshot shows a web-based interface for testing regular expressions. At the top, there is a blue header with the word "Expression". Below this, the regular expression `/([a-z])\1/g` is entered in a text field. The `[a-z]` part is highlighted in yellow, and the `\1` part is highlighted in green. Below the text field, there are two buttons: "Text" and "Tests". The "Tests" button is highlighted in blue and has a "NEW" label next to it. Below the buttons, the word "mississippi" is displayed in a monospace font, with the letters 'i', 's', 's', 'i', 'p', 'p', 'i' highlighted in blue, indicating the matches found by the regular expression.

# Boundaries

- » `^` matches the beginning of the line
- » `$` matches the end of the line
- » `\b` word boundary, where words are defined as a sequence of alphanumeric characters. It is a zero-width assertion (i.e. no actual character is matched)
- » `\B` negation of `\b`: the current position is not a word boundary

Expression

```
/\b[a-z]+\b/g
```

Text Tests **NEW**

this·is·a·test·

Expression

```
/\b[a-z]+\B/g
```

Text Tests **NEW**

this·is·is·a·test·

# Practical session 1

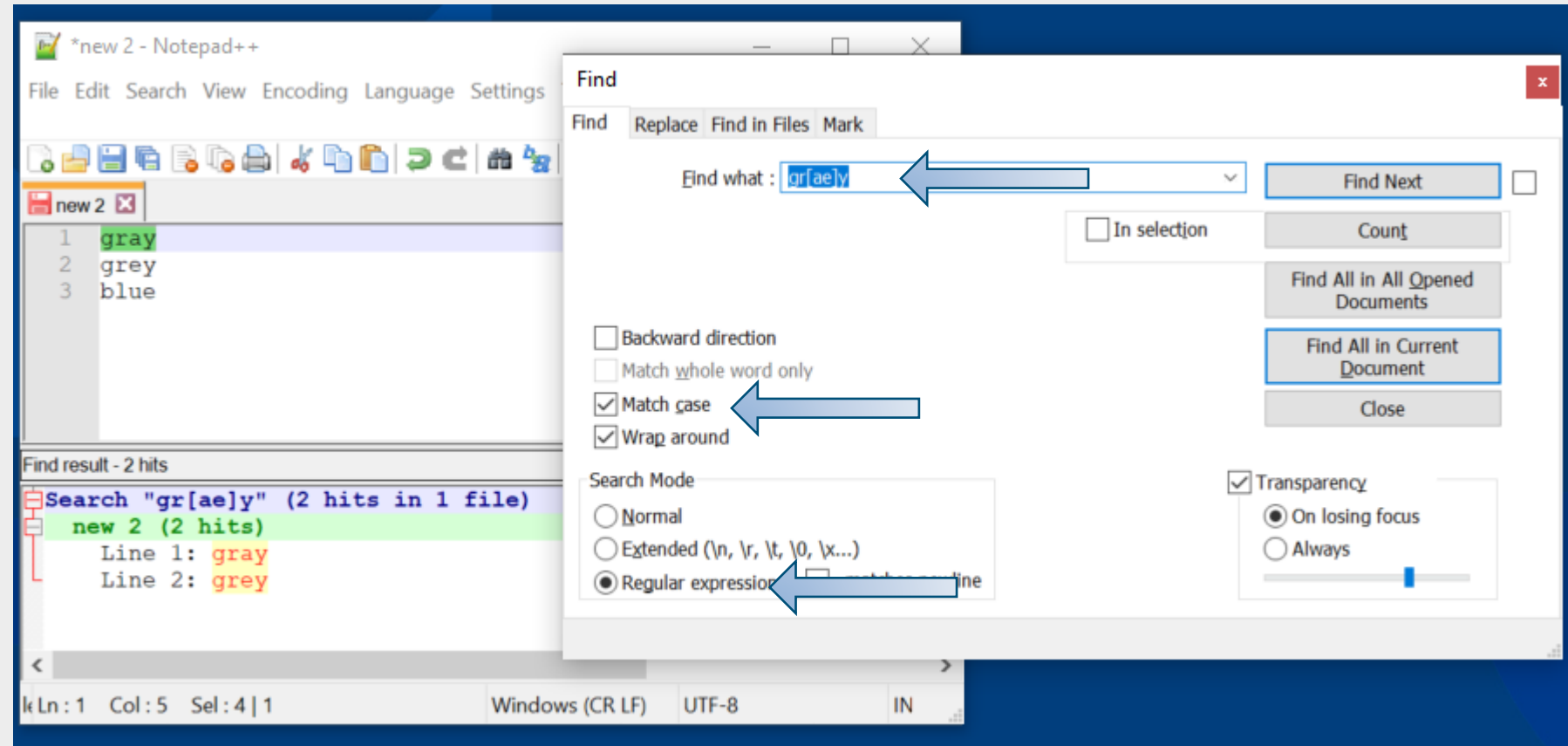
- » Match both *color* and *colour*. How can you match both capitalised and lower-case words?
- » What kind of words the following match:
  - `^[0-9]+\.[0-9]+$`
  - `[A-Z]+\$\$`
  - `^[0-9]{4}$`
  - `^[0-9]+-[a-z]{3,5}$`
  - `^[a-z]{5,}-[a-z]{2,3}-[a-z]{1,6}$`
  - `(ed|ing)$`
- » Match time: 1:00 AM, 2:34PM,
- » More difficult match a time after 1pm when expressed using a 24h clock (e.g. a time after 12:00)



## **Part 2: Transforming and cleaning data using regular expressions**

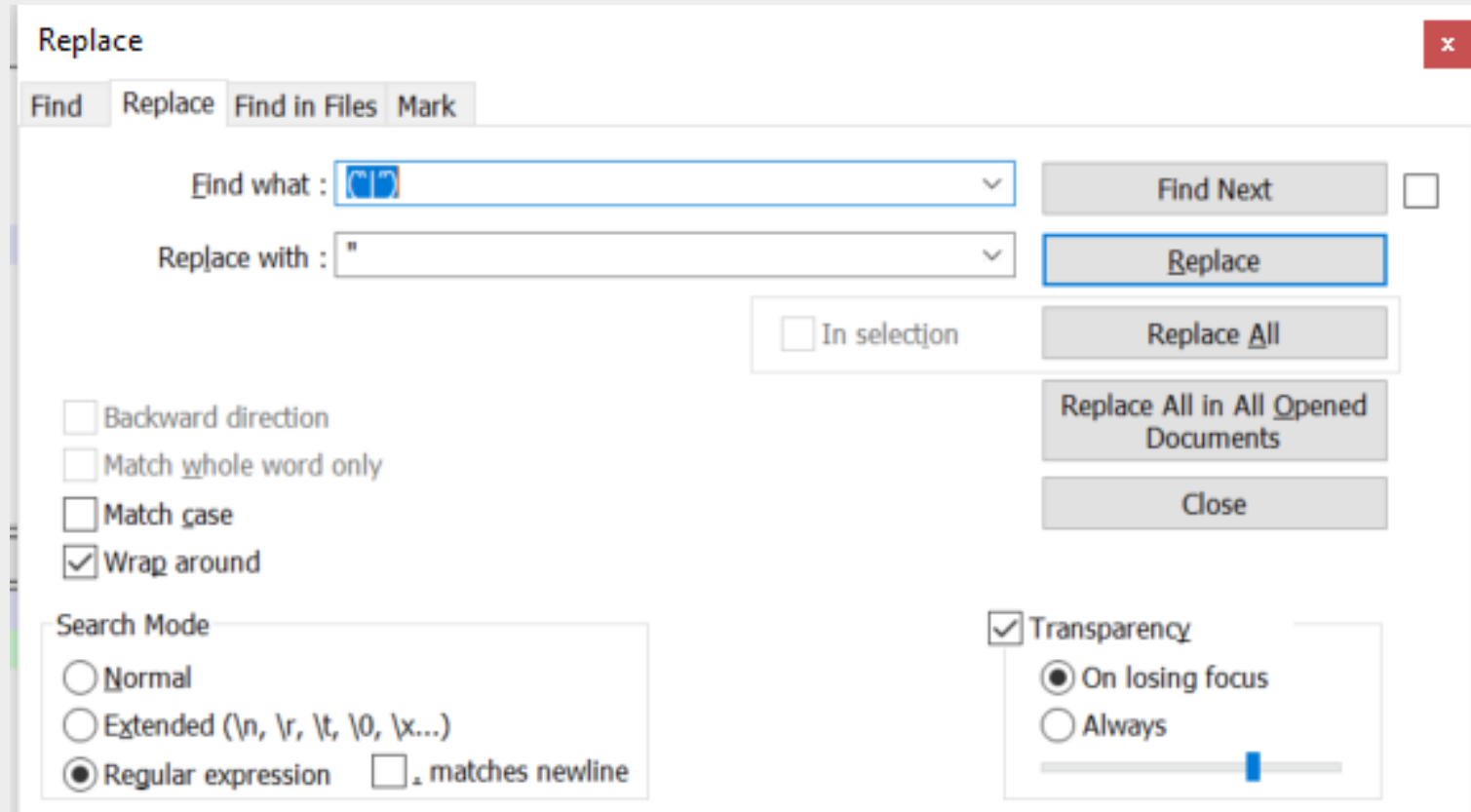
# Notepad++ for regular expressions

- » We will use Notepad++ to clean data
- » Notepad++ is a free text editor that is very powerful (<https://notepad-plus-plus.org/>)
- » It supports regular expressions very well



# Correct smart quotes

- » We have a document which contains smart quotes “ ” . How can we replace them with quotation marks " " ?



Replace

Find Replace Find in Files Mark

Find what : " " Find Next

Replace with : " Replace

In selection Replace All

Backward direction  
 Match whole word only  
 Match case  
 Wrap around

Replace All in All Opened Documents

Close

Search Mode

Normal  
 Extended (\n, \r, \t, \0, \x...)  
 Regular expression  . matches newline

Transparency

On losing focus  
 Always

# Translate blt-X

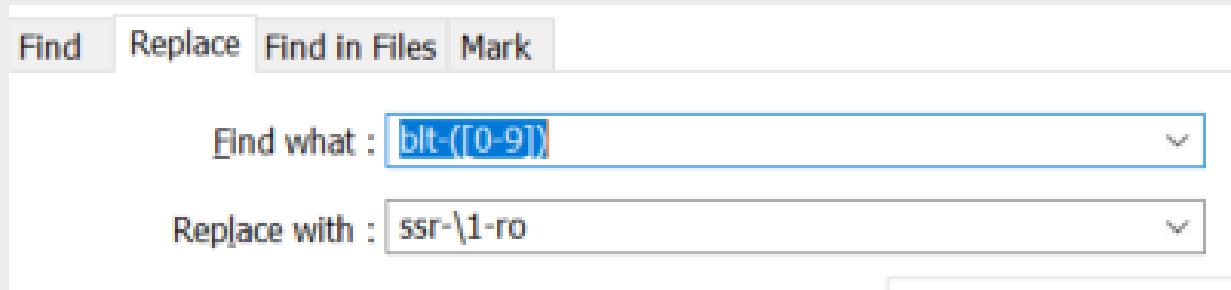
» We need to translate blt-X (where X is a digit) in URIs

Uri: `https://www.anchorbutter.co.uk/globalassets/images/food-ideas/blt-2.jpg`

» The assumption is that *blt* → *ssr*, but we also need to add *-ro* after the number (slightly artificial example, but not impossible), so simple replace of *blt* is not possible

» We match `blt-([0-9])`, where `([0-9])` is a group

» Replace it with `ssr-\1-ro`, where `\1` is reference to group 1 (i.e. copies the text in group 1)



Find Replace Find in Files Mark

Find what : `blt-([0-9])`

Replace with : `ssr-\1-ro`

# Changing capitalisation in glossaries

- » We have a glossary which contains terms and abbreviations. How we can convert all the terms to lower case, but not the abbreviations

E.g.

Translation memory

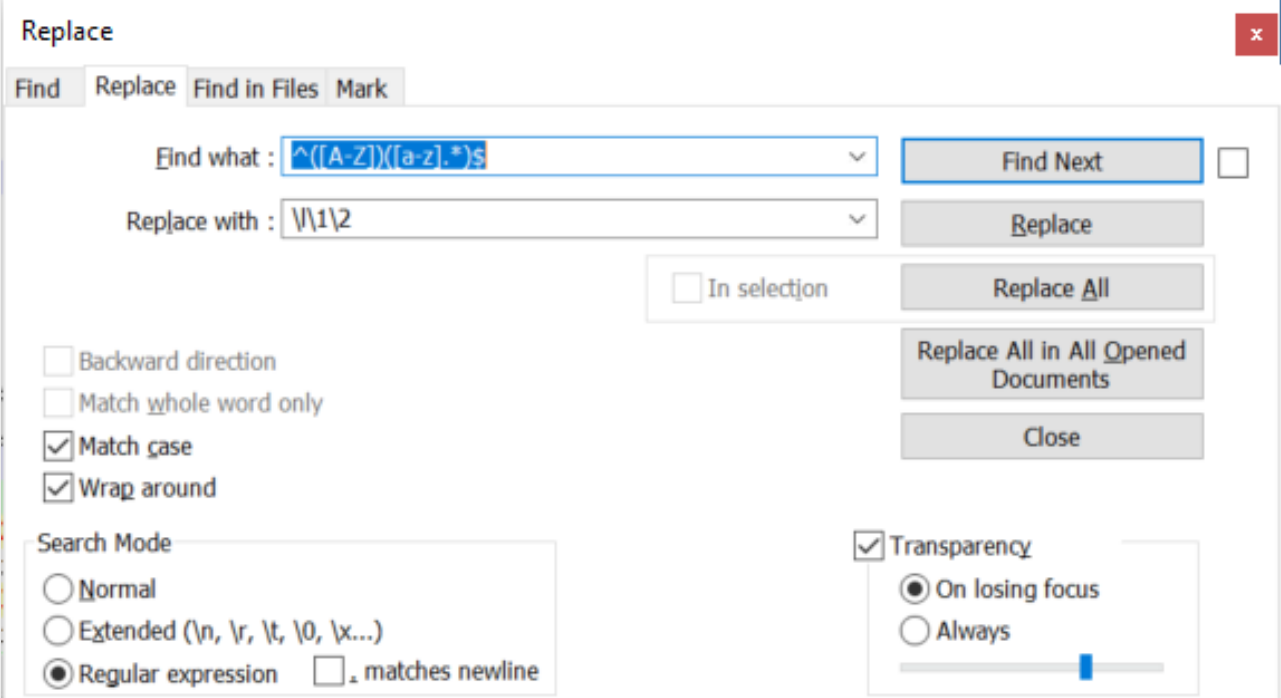
MT

term database

HTML

Computer-aided translation

- » Match `^([A-Z])([a-z].*)$`
- » Replace with `\1\1\2` (`\1` means convert the next character to lowercase)
- » The **Match case** option needs to be selected



Replace

Find Replace Find in Files Mark

Find what : `^([A-Z])([a-z].*)$` Find Next

Replace with : `\1\1\2` Replace

In selection Replace All

Backward direction

Match whole word only

Match case

Wrap around

Search Mode

Normal

Extended (\n, \r, \t, \0, \x...)

Regular expression  . matches newline

Transparency

On losing focus

Always

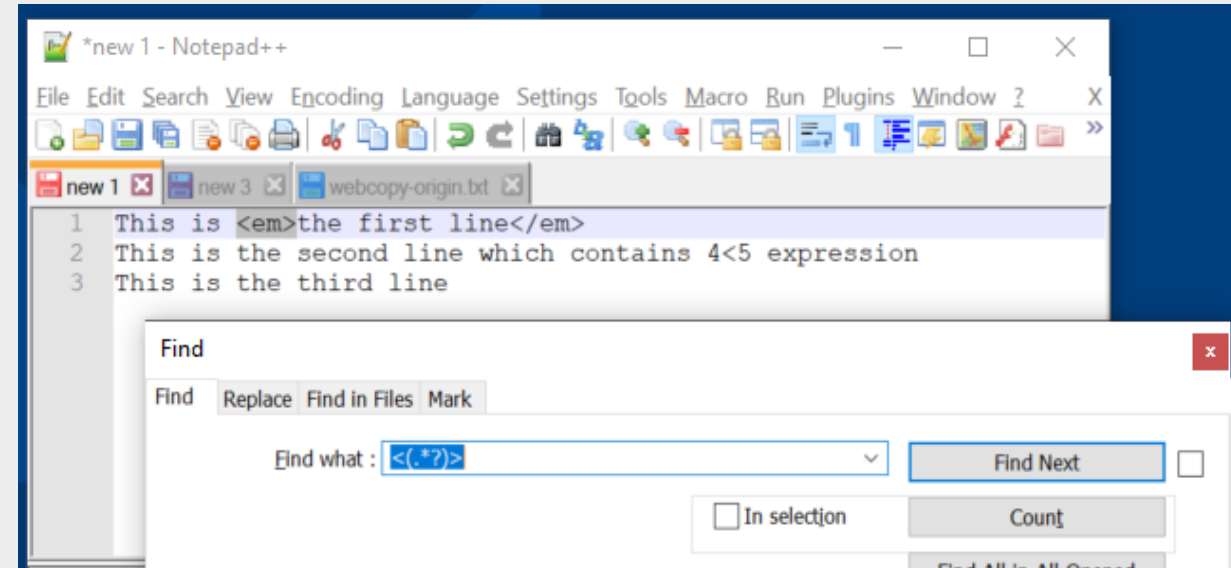
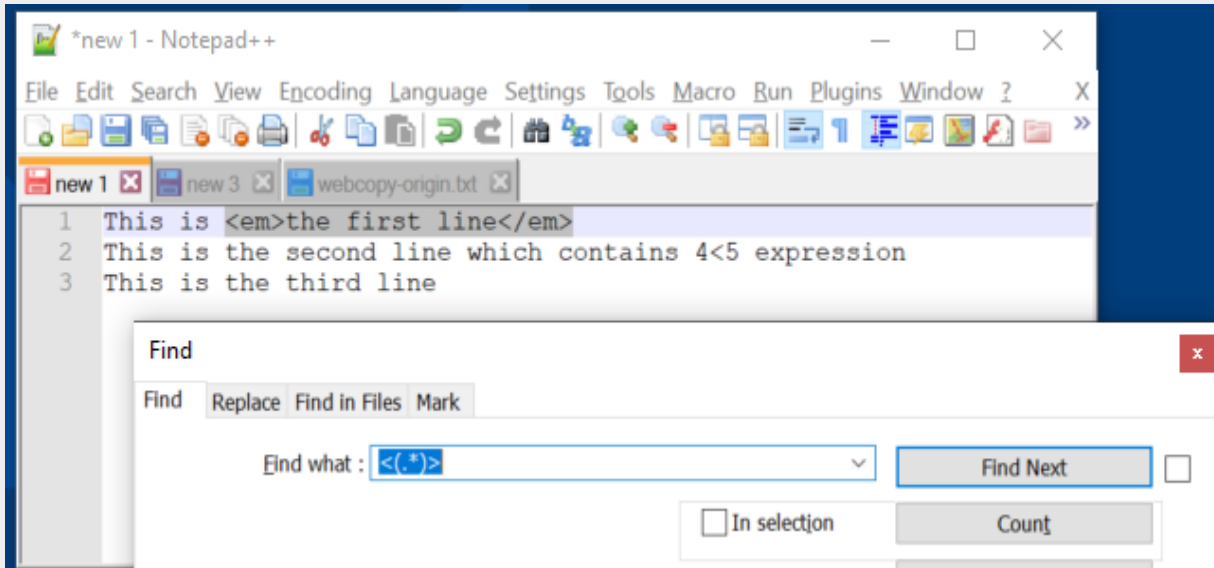
Replace All in All Opened Documents

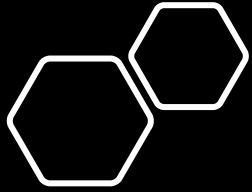
Close



# Cleaning HTML

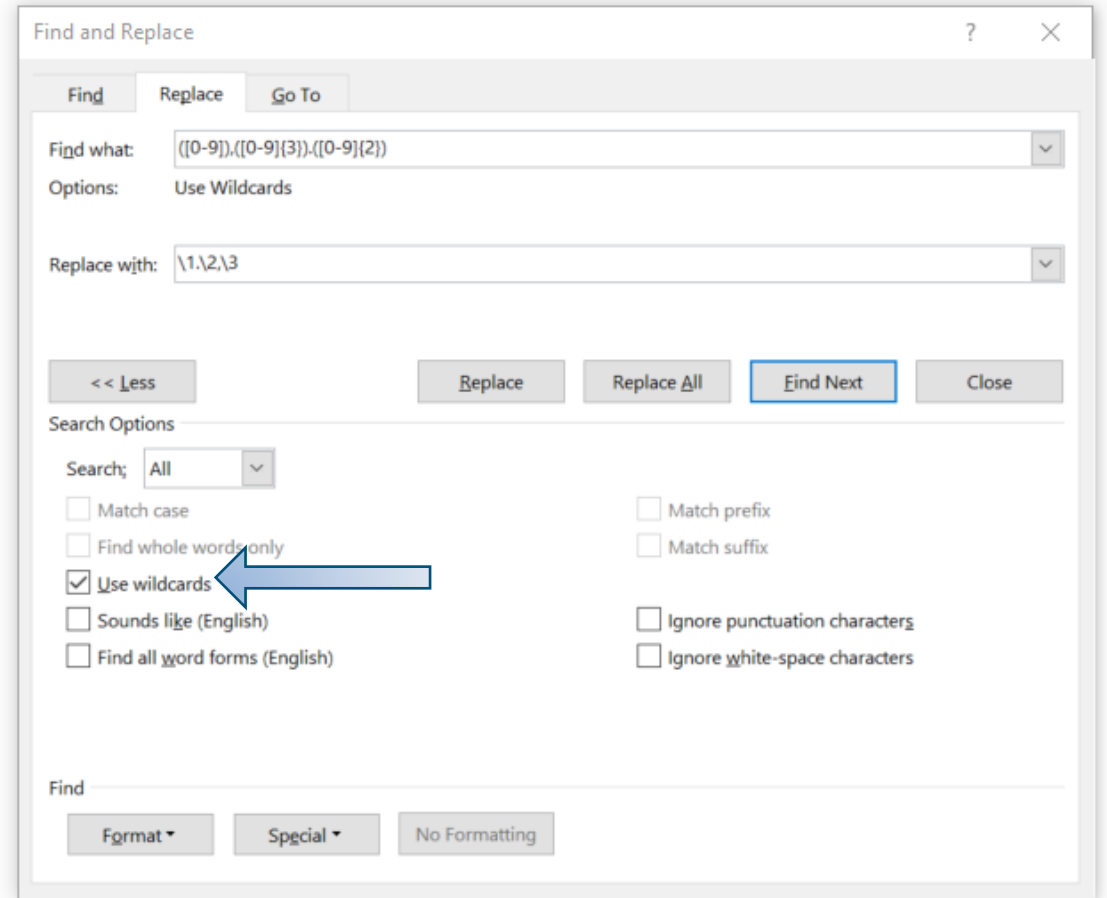
- » An HTML tag is marked by `<>` and used by browsers to control how a text is displayed. E.g. This is an `<em>emphasised</em>` word.
- » At times we need to clean tags from our texts (e.g. corpus that was built from the web)
- » The pattern we should use is `<.*?>`. The `.*?` indicates a **non-greedy** matching





# Regular expressions in Microsoft Word

1.000,00



Example how to use a regular expression in Word to transform how numbers are represented.

Read more about regular expressions in Word at [http://www.gmajor.com/replace\\_using\\_wildcards.htm](http://www.gmajor.com/replace_using_wildcards.htm)

# Practical session 2

1. You are given a date in the format dd/mm/YYYY convert it to yy-mm-dd (e.g. 11/03/2022 → 22-03-11)
2. Convert numbers from the format XX,XXX.XX to XX.XXX,XX
3. We have a file with a list of terms in English where each term is indicated by the tag <en>. The task is to prepare the file to be translated by duplicating the text, but surrounded by the code of the target language (but not translate the text)

E.g.           <en>translation memory</en> →  
                  <en>translation memory</en> <ro>translation memory</ro>

For this activity you can use either Notepad++ or Word.

# Further reading/activities

- » Language independent tutorial about regular expressions  
<https://github.com/zeeshanu/learn-regex>
- » The fantastic world of nerdy regex fun: <https://regexcrossword.com/>
- » Regex Golf: <https://alf.nu/RegexGolf?world=regex&level=r00>
- » Regular expressions in Notepad++ <https://npp-user-manual.org/docs/searching/#regular-expressions>
- » What's that `^ . ? $ | ^ ( . . + ? ) \ 1 + $` : <https://iluxonchik.github.io/regular-expression-check-if-number-is-prime/>

**Thank you**

» Get in touch if you have questions:

[C.Orasan@surrey.ac.uk](mailto:C.Orasan@surrey.ac.uk)

» Slides will be available on

<https://dinel.org.uk/teaching/workshop-on-regular-expressions/>



UNIVERSITY OF  
SURREY