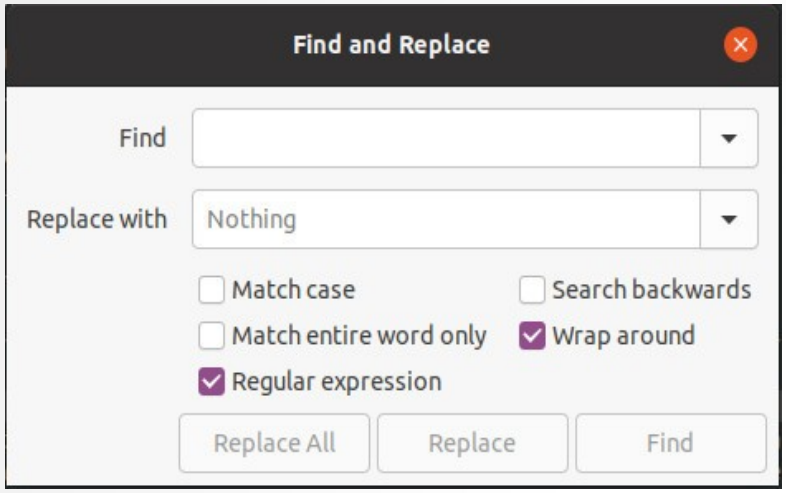
Regular Expressions

Library Carpentry

https://librarycarpentry.org/lc-data-intro/

Regular Expressions (regex)

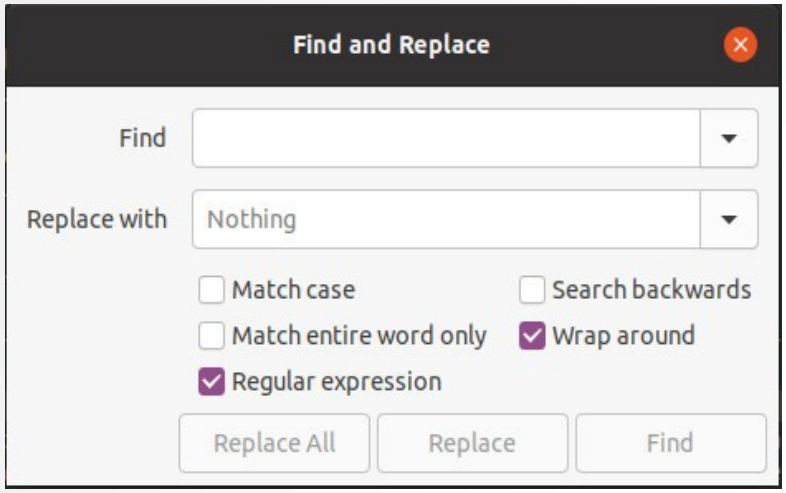


- How much do you know about regular expressions a.k.a. regex?

- - no idea?
- - a vague idea?
- - know what they are?

https://librarycarpentry.org/lc-data-intro/

Regular Expressions (regex)



- A sequence of characters that define a pattern for matching strings (rather than only matching an exact string)
- Used to match/find/replace patterns of text
- Example use: find email addresses or phone numbers in text
- Regular expressions can match a string of zero length or many lines of text
- Used in programming languages, “find and replace” functionality in text editors, OpenRefine, Google Sheets, grep at the commandline etc.

https://librarycarpentry.org/lc-data-intro/

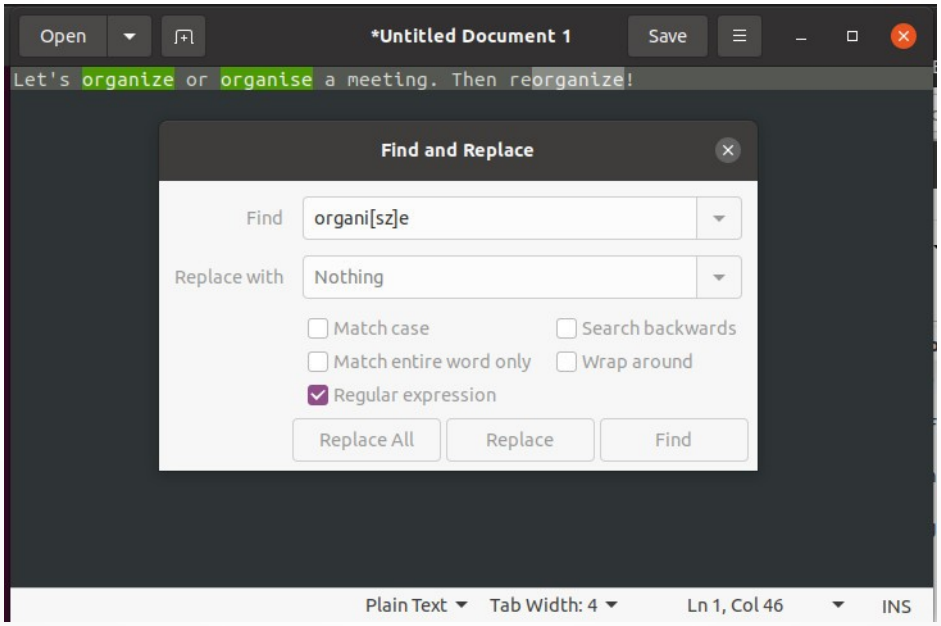
Regular Expressions (regex)

- Match by types of characters (e.g. upper case letters, digits, spaces, etc.)
- Match patterns that repeat any number of times
- Capture the parts of the original string that match your pattern (can be used in replacement)
- Consist of literal characters (“a” to match the letter “a”) and metacharacters (“.” to match a non-specific single character)
- Backlash “\” is used to escape (make the literal character of a metacharacter)
  - - “\.” matches a period instead of any single character

https://librarycarpentry.org/lc-data-intro/

Regex use case

- Find multiple spellings of same word:
- *organi[sz]e* matches both organise and organize (even if part of the words reorganise, reorganize, organises, organizes, organised, organized, etc.)



https://librarycarpentry.org/lc-data-intro/

Regex use case

- Pull all of the phone numbers out of a directory of files

```
$ ls contact_info/
gio.txt  lauren.txt
$ cat contact_info/lauren.txt
Name: Lauren
Phone: (940) 369-7613
$ grep -r -h -o -P '\(\d{3}\) \d{3}-\d{4}' contact_info/
(940) 369-7613
(555) 555-1234
$
```

grep is a commandline program for printing matches found in files

options used:

- r recursively search files in the directory
- h don't print filenames before match
- o only print matching part of line
- P use Perl-compatible regex

https://librarycarpentry.org/lc-data-intro/

Regular Expression Syntax

- Use literal characters to match exactly
  - `a` matches “a”; 1 matches “1”; , matches “,”
- `.` matches any single character
  - `.ow` matches “tow”, “cow”, “low”
- `\` escapes the following special character
  - `\.com` matches “.com”; .com matches “.com” or “bcom” or “4com” etc.
- `*` matches the preceding element zero or more times
  - `ab*c` matches “ac”, “abc”, “abbbc”
- `+` matches the preceding element one or more times
  - `ab+c` matches “abc”, “abbbc” but not “ac”
- `?` matches when the preceding character appears zero or one time
  - `cows?` matches “cow” or “cows”

https://librarycarpentry.org/lc-data-intro/

Regex Syntax – specific count

- `{VALUE}` matches the preceding character the number of times defined by VALUE
  - `a{5}bc` matches “aaaaabc”
- `{VALUE,VALUE}` matches the preceding character a number of times given in the range of values
  - `0{1,9}33` matches between one and nine zeroes in length, i.e. “000033”

https://librarycarpentry.org/lc-data-intro/

Regex Syntax - Opposites

- \d matches any single digit
  - \d matches “9” or “0”
- \D matches a non-digit (opposite of \d)
  - \D matches “a” or “,”
- \w matches any word character (equivalent to [A-Za-z0-9_])
  - \w matches “A”, “8”, “_”
- \W matches any non-word character (opposite of \w)
  - \W matches “-” or “ “
- \s matches any space, tab, or newline
  - \s matches “ “
- \S matches a character that is not a space, tab, nor newline (opposite of \s)
  - \S matches “a”

https://librarycarpentry.org/lc-data-intro/

Regex Syntax - Positions

- `^` indicates a position at the start of the line. What you put after the caret only matches if it starts the line.

    - `^the` day matches “the day” but not “on the day”

- `$` indicates a position at the end of the line. What you put before it only matches if it ends the line.

    - `good luck$` matches “it’s good luck” but not “good luck you”

- `\b` asserts the pattern matches at a word boundary. Putting this either side of an expression stops it matching longer variants of words.

    - `mark` will match “mark”, “marking”, “market”, ”unremarkable”

    - `\bword` matches “word”, “wordless”, and “wordlessly”

    - `comb\b` matches “comb” and “honeycomb” but not “combine”

    - `\brespect\b` will match respect but not respectable or disrespectful

https://librarycarpentry.org/lc-data-intro/

Regex Syntax - Brackets

- Square brackets can be used to define a list or range of characters to be found
- [ABC] matches A or B or C
- [A-Z] matches any upper case letter
- [A-Za-z] matches any upper or lower case letter
- [A-Za-z0-9] matches any upper or lower case letter or any digit

https://librarycarpentry.org/lc-data-intro/

Regex Syntax - Misc

- (...) matches the expression inside the parentheses, defining a group that can later be retrieved, such as for use with replacement, using a \number reference (backlash followed by group number)

    - (\d{3}-){2}\d{4} matches “940-565-3000”

    - ark:.67531.(metapth\d+) matches “ark:/67531/metapth213” creating a group that can be referenced by \1 which gives the value “metapth213”

- | means *or*

    - (help|youth)ful matches “helpful” or “youthful”

- /i renders an expression case-insensitive (equivalent to [A-Za-z]), though this varies by implementation

https://librarycarpentry.org/lc-data-intro/

Regex in Python

```
>>> import re
>>> m = re.search('ark:/67531/(meta(crs|dc|pth)\d+)', 'ark:/67531/metadc87321')
>>> m.group(1)
'metadc87321'
>>> m.group(2)
'dc'
```

https://librarycarpentry.org/lc-data-intro/

Regex Use Case

How do you see yourself using regular expressions?

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression ^[Oo]rgani.e\w* match?

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression ^[Oo]rgani.e\w* match?

organise
Organize
organifer
Organi2ed111

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression [Oo]rgani.e\w+$ match?

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression [Oo]rgani.e\w+$ match?

organiser
Organized
organifer
Organi2ed111

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression ^[Oo]rgani.e\w?$ match?

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression ^[Oo]rgani.e\w?$ match?

organise
Organized
organifer
Organi2ek

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression \b[Oo]rgani.e\w{2}\b match?

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression \b[Oo]rgani.e\w{2}\b match?

organisers
Organizers
organifers
Organi2ek1

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression \b[Oo]rgani.e\b|\b[Oo]rgani.e\w{1}\b match?

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

What will the regular expression
\b[Oo]rgani.e\b|\b[Oo]rgani.e\w{1}\b
match?

organise
Organi1e
Organizer
organifed

https://librarycarpentry.org/lc-data-intro/

Regex in Practice

Let’s do the exercises at
https://ldko.github.io/lc-data-intro/04-exercises/index.html

Regex in Practice

Regex in Google Sheets:

1. Export and unzip the 2017 Public Library Survey (https://github.com/LibraryCarpentry/lc-data-intro/blob/gh-pages/files/PLS_FY17.zip) as a CSV file.
2. Upload the CSV file to Google Sheets and open as a Google Sheet if it doesn't do this by default.
3. Look in the ADDRESS column and notice that the values contain the latitude and longitude in parentheses after the library address.
4 . Construct a regular expression to match and extract the latitude and longitude into a new column named 'LATLONG'. HINT: Look up the function REGEXEXTRACT in Google Sheets. That function expects the first argument to be a string (a cell in ADDRESS column) and a quoted regular expression in the second.

https://librarycarpentry.org/lc-data-intro/