Regular Expressions: Definition

"*provide a more powerful mechanism for pattern matching by enabling you to restrict pattern matches to specific character values, specific ranges and numbers of characters, specific character positions within a term, and so on*"
(IBM Watson Knowledge Base)

Regular Expressions

- are strings that are used to match and manipulate text
- are tools designed to solve a specific problem

1. Verifying Input (e.g., correctly formatted Dates)
2. Finding text matching a pattern (**car** but not **carry**, **scary**)
3. Replacing text matching a pattern (extract URL link and add HTML tags)
4. Splitting text (split into sentences, words)
5. Widely supported: major text editors, analytical platforms (SQL, Tableau, Alteryx ...)

```
\b[Cc][Aa][Rr]\b
```

RegEx simplifies many programming and text processing task

Regular Expressions: History

Neuroscience

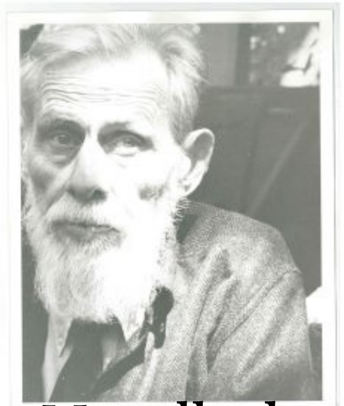
1943 - IDEA

1956 - ALGEBRA

Programming

1968

1979



Mcculloch

https://archives.library.illinois.edu/thought-collective/cyberneticians/warren-s-mcculloch/

Pitts

https://en.wikipedia.org/wiki/Walter_Pitts

Kleene

https://nationalmedals.org/laureate/stephen-c-kleene/

Thompson

https://simple.wikipedia.org/wiki/Ken_Thompson

Aho

https://en.wikipedia.org/wiki/Alfred_Aho

Neuroscience:
nervous system models

Algebra notation:
regular sets/regular expressions

Unix:
Ed text editor
1973 – stand-alone grep

Egrep
Extended RegEx

Regular Expressions and Finite State Automata

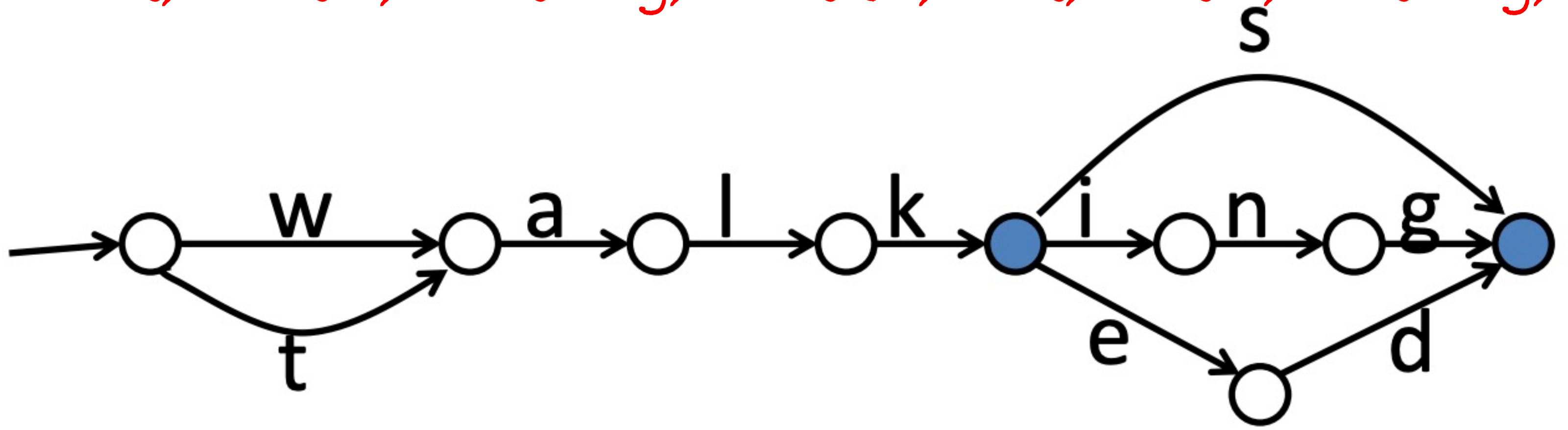
Can we express Morphology via Regular expressions or Finite State Automata?

- Morphemes are typically arranged in a certain order:
  - WORK-ING versus ***ING-WORK**
  - WORK-ED versus ***WORK-S-ED**
- Closed class morphemes: inflections (-s, -ed, -ing)

Regular Language

- expressed with regular expressions or deterministic finite automata
- a set of strings with characters or set of symbols

walk, walk-s, walk-ing, walk-ed, talk, talk-s, talk-ing, talk-ed



Finite State Automata

(w|t)alk(s|ed|ing)?

Regular Expression

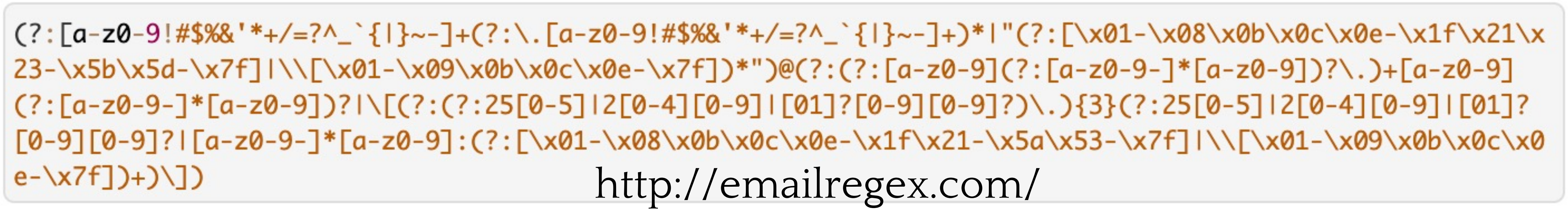
Alexander Frazer & Liane Guilou. 2016. Finite State Morpphology.

When Not To Use Regular Expressions

1. Parsing HTML: html.parser, beautiful soup library

2. Parsing URL path: python urllib.parse; urlparse

3. Parsing Emails: python email.parser, mail-parser



```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x
23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9]
(?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?
[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0
e-\x7f])+)\])
```

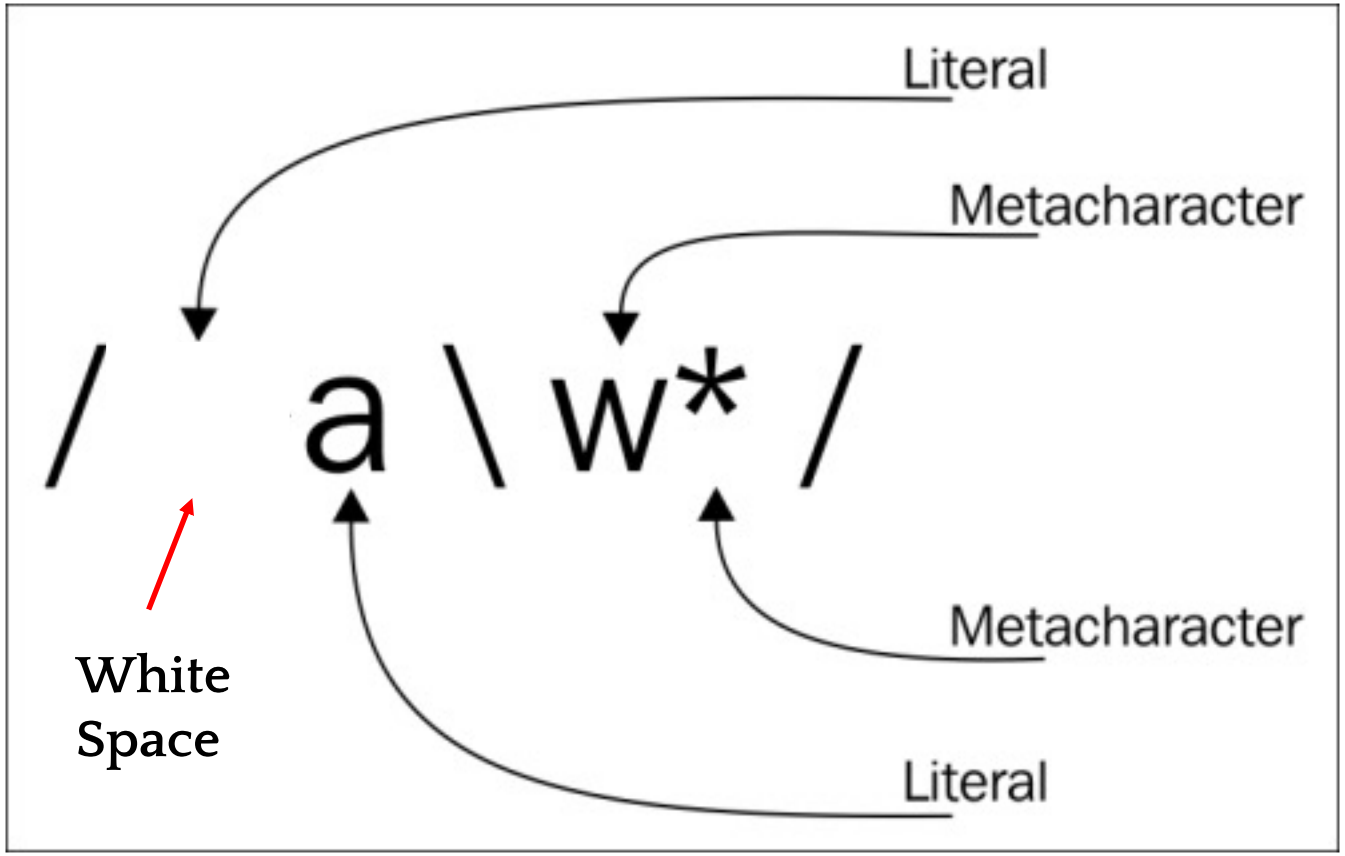
http://emailregex.com/

Regular Expressions Syntax

A regular expression is a pattern of text that consists of

1) <u>ordinary characters</u> (for example, letters a through z or numbers 0 through 9)
2) <u>special characters</u> known as metacharacters.

Match: any word starting with **a**

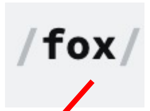


Literals: Space and "a"

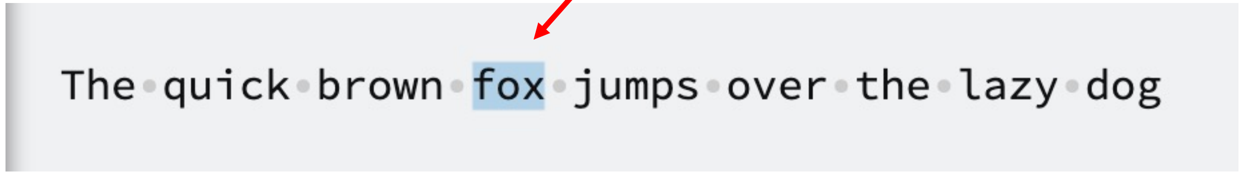
Metacharacters: \w and *

Literals

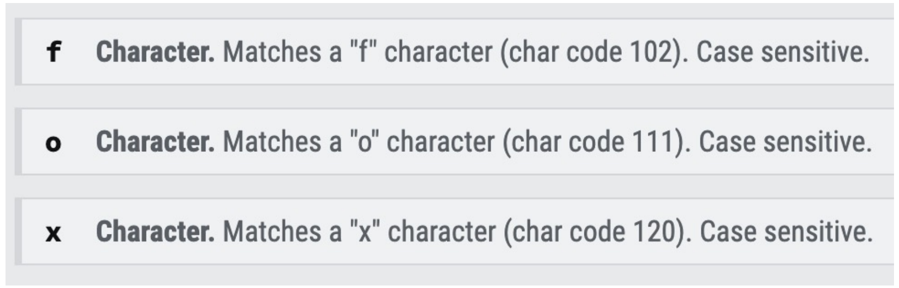
The simplest form of pattern matching



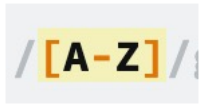
```
/fox/
```



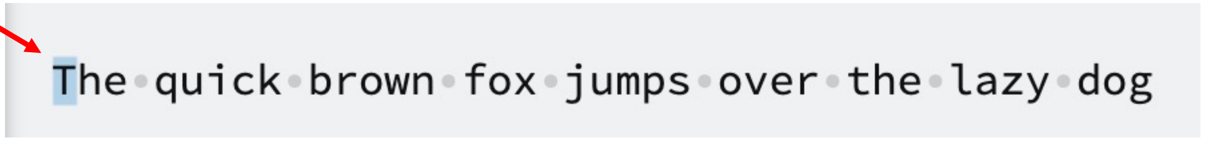
```
The quick brown fox jumps over the lazy dog
```



f	**Character.** Matches a "f" character (char code 102). Case sensitive.
o	**Character.** Matches a "o" character (char code 111). Case sensitive.
x	**Character.** Matches a "x" character (char code 120). Case sensitive.

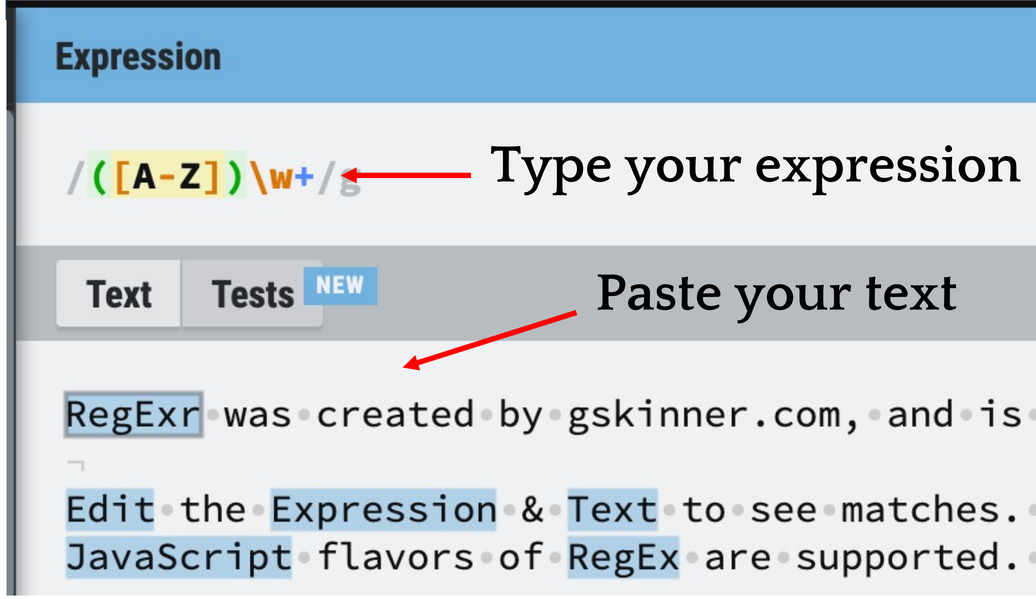


```
/[A-Z]/
```



```
The quick brown fox jumps over the lazy dog
```

https://regexr.com/



```
/([A-Z])\w+/g
```

Type your expression

Paste your text

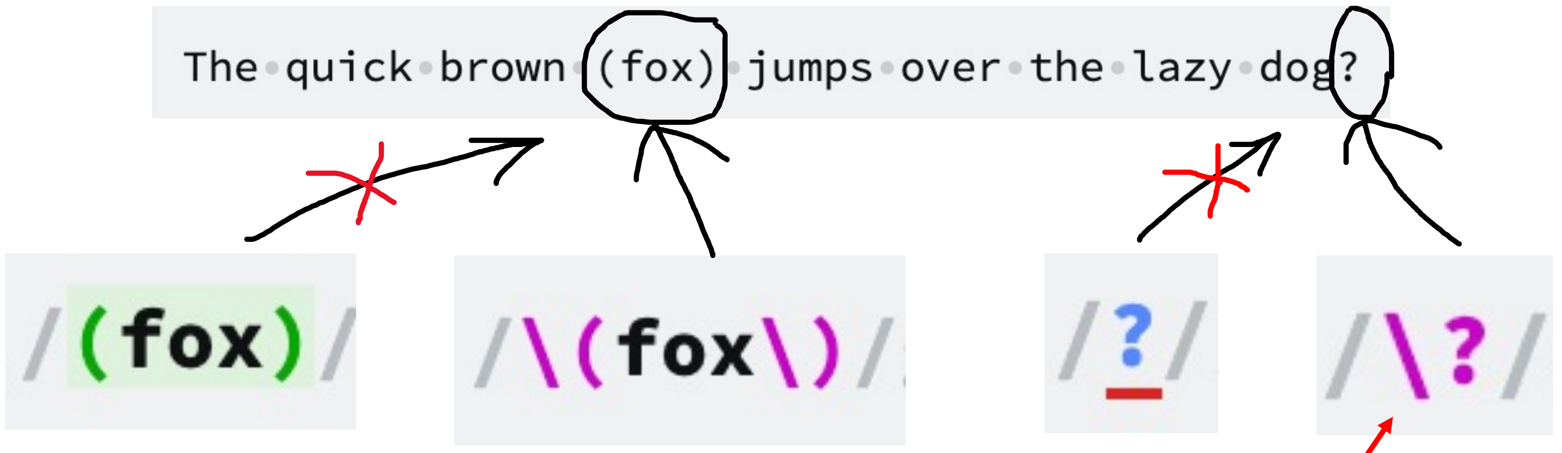
```
RegExr was created by gskinner.com, and is
Edit the Expression & Text to see matches.
JavaScript flavors of RegEx are supported.
```

Character Classes

[a-z]	lower cases only
[A-Z]	upper cases only
[0-9]	digits only

12 Metacharacters

- Backslash \
- Caret ^
- Dollar sign $
- Dot .
- Pipe symbol |
- Question mark ?
- Asterisk *
- Plus sign +
- Opening parenthesis (
- Closing parenthesis)
- Opening square bracket [
- The opening curly brace {



Backslash (escape)

from our previous slide

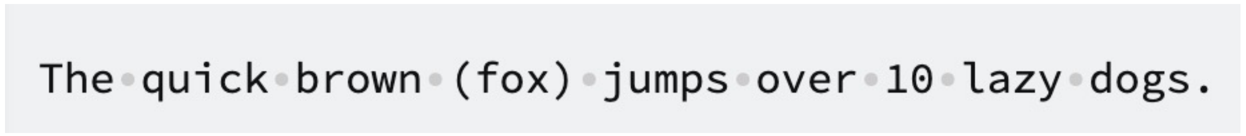
[a-z]
[A-Z]
[0-9]

Brackets and hyphen are metacharacters to define a range

`[0-9a-zA-Z]` alphanumeric class

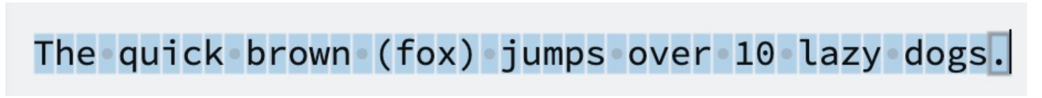
`[^0-9]` negation (match non-digits)

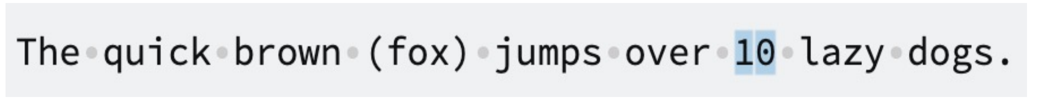
Predefined Character Classes

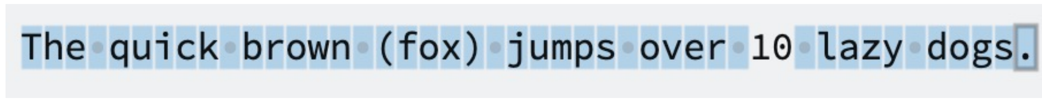

The quick brown (fox) jumps over 10 lazy dogs.

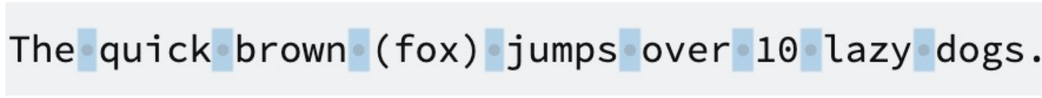
Matching one single character only

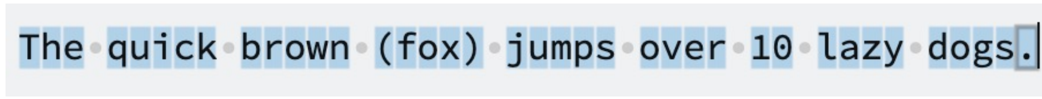
`.`	This element matches any character except newline `\n`
`\d`	This matches any decimal digit; this is equivalent to the class `[0-9]`
`\D`	This matches any non-digit character; this is equivalent to the class `[^0-9]`
`\s`	This matches any whitespace character; this is equivalent to the class `[ \t\n\r\f\v]`
`\S`	This matches any non-whitespace character; this is equivalent to the class `[^ \t\n\r\f\v]`
`\w`	This matches any alphanumeric character; this is equivalent to the class `[a-zA-Z0-9_]`
`\W`	This matches any non-alphanumeric character; this is equivalent to the class `[^a-zA-Z0-9_]`

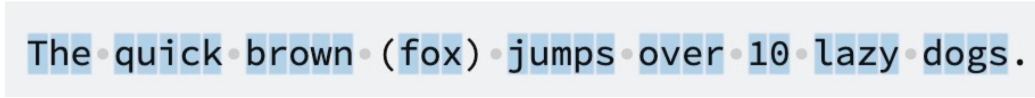

The quick brown (fox) jumps over 10 lazy dogs.

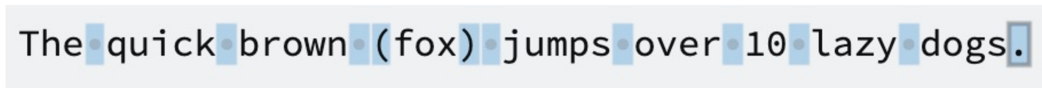

The quick brown (fox) jumps over 10 lazy dogs.


The quick brown (fox) jumps over 10 lazy dogs.

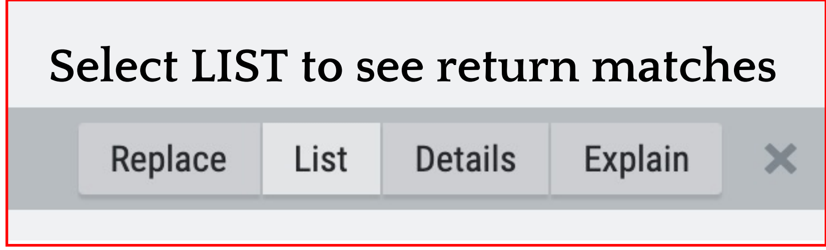

The quick brown (fox) jumps over 10 lazy dogs.


The quick brown (fox) jumps over 10 lazy dogs.

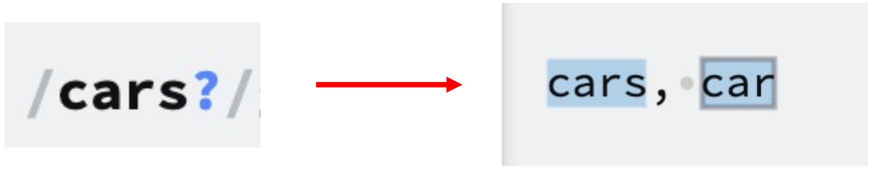

The quick brown (fox) jumps over 10 lazy dogs.


The quick brown (fox) jumps over 10 lazy dogs.

Quantifiers

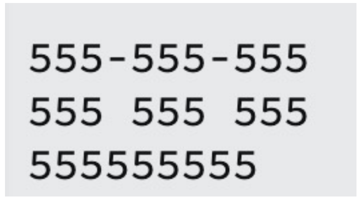

Select LIST to see return matches

`?`	Question mark	Optional (0 or 1 repetitions)
`*`	Asterisk	Zero or more times
`+`	Plus sign	One or more times
`{n,m}`	Curly braces	Between *n* and *m* times

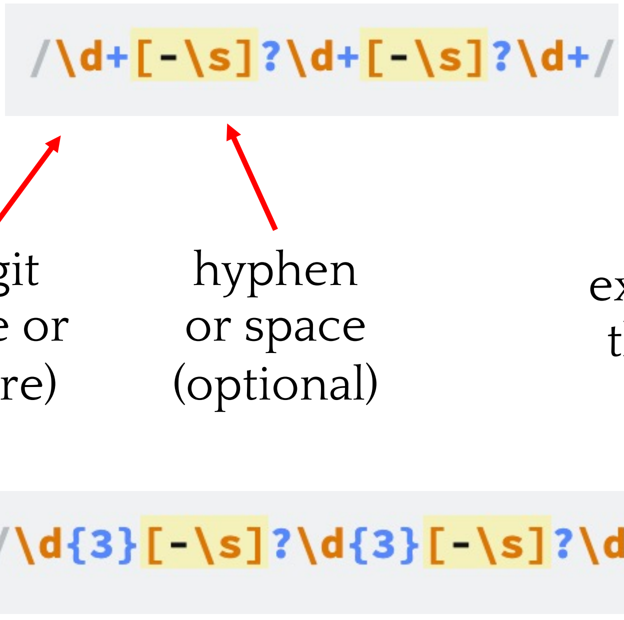


```
/cars?/
```

```
cars, car
```



```
555-555-555
555 555 555
555555555
```



```
/\d+[-\s]?\d+[-\s]?\d+/
```

```
/\d{3}[-\s]?\d{3}[-\s]?\d{3}/
```

Boundary Matchers

`hello, helloed, or Othello`

`^`	Matches at the beginning of a line
`$`	Matches at the end of a line
`\b`	Matches a word boundary
`\B`	Matches the opposite of `\b`. Anything that is not a word boundary
`\A`	Matches the beginning of the input
`\Z`	Matches the end of the input

`/^hello/`

^ is used for negation only inside the character class set [^]

`/\bhello\b/`

https://regexone.com/

RegexOne
Learn Regular Expressions with simple, interactive exercises.