

Regex 101

Anda Stoica, Technical University of Cluj-Napoca

What is a regular expression (regex)?

- Regex = pattern describing the format of a string
- Used for pattern identification, string parsing, input validation
- Generally used for matching strings with a defined format (e.g. dates, urls, etc.)
- Language-independent (Java, Python, ...)
- Case sensitive!



Anchors - ^, &

- `^Start` → matches any string **starting** with “Start”
- `end$` → matches any string **ending** with “end”
- `^Start end$` → matches the string “Start end” **exactly**
- `abc` → matches any string that has abc as a **substring**



Quantifiers - $*$, $+$, $?$, $\{ \}$

- $*$ \rightarrow **zero or more**
- $+$ \rightarrow **one or more**
- $?$ \rightarrow **zero or one**
- $\{ \mathbf{x} \}$ \rightarrow **x times**
- $\{ \mathbf{x}, \}$ \rightarrow **x or more**
- $\{ \mathbf{x}, \mathbf{y} \}$ \rightarrow **min x, max y times**



Quantifiers - *, +, ?, { }

- abc^* → ab followed by **zero or more** c
- abc^+ → ab followed by **one or more** c
- $abc?$ → ab followed by **zero or one** c
- $abc\{2\}$ → ab followed by **2** c
- $abc\{2, \}$ → ab followed by **2 or more** c
- $abc\{2, 5\}$ → ab followed by **min 2, max 5** c
- $a(bc)^*$ → a followed by **zero or more** copies of the sequence bc



OR operator - |

- $a(b | c) \rightarrow$ a followed by **either b or c**

Bracket expression can be used instead:

- $a[bc] \rightarrow$ a followed by **either b or c**



Bracket expressions - []

- [abc] → string that is **either a or b or c** (in any order)
- [a-c] → string that is **either a or b or c**
- [a-fA-F0-9] → string that represents a hex digit
- [a-z] % → string that has a character **from a-z before a % sign**
- [a-z] * → string that has **zero or more characters from a-z**
- [^a-zA-Z] → string that is **not a character from a-z or A-Z** (here ^ is negation)



Escaping special characters

Special characters `^ . [$ () | * + ?` need to be escaped (use `\`) to be taken literally (e.g. `\$` if we want to match the `$` character)

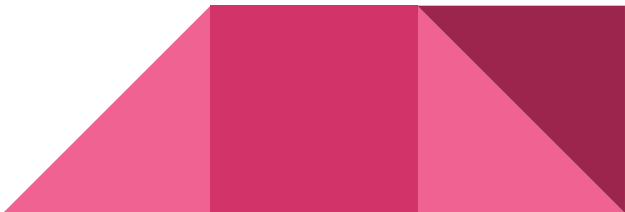
- `\[[a-z]*\]` → string of the format “[abcd]”



Character classes

- `\d` → any digit
- `\w` → any word character (alphanumeric and `_`)
- `\s` → whitespace
- `.` → any character

Negations:

- `\D` → any non-digit character
 - `\W` → any non-word character
 - `\S` → any non-whitespace character
- 

Word boundaries

- `\b` → performs a “**whole words only**” search (represents boundaries of words)
- `\B` → **negation** of `\b`

Examples:

- `\babc\b` → match word “abc”
- `\Babc\B` → match sequence “abc” which is contained inside a word (not at start or at end of word)



Email example

Easy regex for checking the validity of an email address:

```
^([_A-Za-z0-9-.\\"+]+)@([a-zA-Z0-9_-]+\.)\.[a-zA-Z]{2,})$
```

Username @ domain . top-level domain

Note: Use () for grouping. When parsing the string, we can extract each group separately.

Useful resources

- <https://regex101.com/>
 - Website for testing out regular expressions
 - Also provides explanations for the regexes you write
- <https://www.regular-expressions.info/>
 - Lots of tutorials and information about anything regex-related
- StackOverflow :)

