Regex CheatSheet

Sites úteis: https://regexper.com/; https://regex101.com/
Referências: https://www.rexegg.com/regex-quickstart.html; https://web.stanford.edu/~jurafsky/slp3/2.pdf; https://docs.python.org/3/library/re.html

Princípios básicos sobre regex

- The simplest kind of regular expression is a sequence of simple characters.
- Regular expressions are case sensitive
- "Word" in RE is defined as any sequence of digits, underscores, or letters
- In RE, always go from particular to general.
- The operator precedence hierarchy for regular expressions is:

Parenthesis	()
Counters	* + ? {}
Sequences and anchors	the ^my end$
Disjunction	|

- Anchors are special characters that anchor regular expressions to particular places in a string.
- Patterns can be ambiguous. In these cases, RE always match the largest string they can. (Patterns are "greedy").
- The ? qualifier makes quantifiers "lazy" (non-greedy). *? and +? matches as little text as possible.
- In general, we make regex by fixing two kinds of errors:
  false positives (incorrectly matched ones)
  false negatives (incorrectly missed ones)
- In general, Python 3 regex considers unicode.

Character classes (predefined)

\d	Digit from 0 to 9 | Python 3: Unicode digit.
\D	Character that is not a *digit* as defined by your engine's *\d*
\w	Word character
\W	Character that is not a *word character* as defined by your engine's *\w*
\s	Whitespace character (space, tab, newline, carriage return, vertical tab). Python 3: any Unicode separator
\S	Character that is not a *whitespace character* as defined by your engine's *\s*
.	Any character except line break
\	Escapes special characters: ([{ . ^ * $ \ | + ? / < >

Regular expression
Regex
RE

Classes

[...]	Disjunction of characters
[a-q]	Letter in the range from a to q
[^...]	Negate brackets
[x##]	Symbol with hex code ##

Quantifiers

+	One or more times
*	Zero or more times
?	Zero or one times
{n}	Exactly "n" times
{n,m}	Between "n" and "m" times
{n,}	At least "n" times
{,m}	Up to "m" times

Anchors and Boundaries

^	Start of line
\A	Start of string
$	End of line
\Z	End of string (Python)
\b	Word boundary
\B	Not a word boundary

Logic

|	OR operand
(...)	Capturing group
\1	Contents of Group 1
\2	Contents of Group 2
(?: ...)	Non-capturing group

Escapes

\t	tab
\n	new line
\v	vertical tab

Assertions

(?=...)	Positive lookahead
(?<=...)	Positive lookbehind
(?!...)	Negative lookahead
(?<!...)	Negative lookbehind