The Perl Regular Expression (RegEx) Pattern Matching Cheat Sheet

Concatenation: any sequence of forward-slash delimited characters: `/even/` or `m/even/

The binding operator (`~=`) compares a value against a RegEx: `$string =~ /odd/`

+ means “one or more”, e.g., one or more a’s: `/ela+/`

Parentheses are used to group, e.g., one or more ela’s: `/(ela)+/`

Characters can be escaped with a backslash, e.g., one or more closing parentheses: `/ela\)+/`

The alternation character is a vertical bar, e.g., ’a’ or ’b’ or ’c’: `/a|b|c/`

Character classes are an alternation shorthand notation: `[/abc]/`

Character class ranges are also possible: `/[0-9]/` or `/[a-zA-Z]/`

Ranges can be inverted, e.g., anything but a vowel: `/[^aeiouAEIOU]/`

Other shorthand includes a single digit: `/\d/`, a single space character: `/\s/` or a single alphanumeric word character: `/\w/`

The `\d`, `\s` and `\w` shorthand can be inverted with `\D`, `\S` and `\W`, respectively

Specific repeat-counts are possible: `/\w{2}/` or `/\w{2,4}/` or `/\w{2,}/`

A single element is there “zero or one” time(s): `/bart?/`

A single element is there “zero, one or more” time(s): `/bart*/`

Any single character (except newline): `/bar./`

Word boundaries are matchable: `/\beven\b/`

The start and end of a string/line: `/^hello$/`

A blank string/line: `/^$/`

A string/line of space characters (potentially): `/^s*$/`

The binding operator (`=~`) checks for a match, and can be negated with `!~`

As well as grouping, the parentheses also remember what was successfully matched in the built-in after-match variables: `$1, $2, $3`, and so on ...

In Perl, every RegEx is greedy by default – switch off greediness with `?/: `(.+?)`, Bart/

Alternative delimiters (including any bracketing pair) can be specified after the initial “m”: `m{http://(.+)}`

Substitutions are possible (using the `s` prefix), as are translations (using the `tr` prefix) and deletions (using the `d` qualifier). Patterns can match globally with the `g` qualifier and case-insensitively with the `i` qualifier

More information on Perl’s RegEx technology is in the following manual pages: `perlre, perlrequick, perlretut`, as well as the excellent *Mastering Regular Expressions, 2nd Edition* by Jeffrey E. F. Friedl.