

Regular Expression Cheat Sheet

Guidelines:

- Use raw strings for your regexes, e.g. `r"\d+"`, to avoid backslash problems
- Use this syntax to store a regular expression for repeated use: `p = re.compile(<re>)`
- `p.match(str)` matches the beginning of the string
- `p.search(str)` finds the leftmost match
- `p.findall(str)` returns a list of all matching substrings

Special Characters

- `\` escape special characters
- `.` matches any character
- `^` matches beginning of string
- `$` matches end of string
- `[sb-d]` matches any chars 's', 'b', 'c' or 'd'
- `[^a-c6]` matches any char except 'a', 'b', 'c' or '6'
- `R|S` matches either regex R or regex S
- `()` creates a capture group and indicates precedence

Quantifiers

- `*` 0 or more (append `?` for non-greedy)
- `+` 1 or more (append `?` for non-greedy)
- `?` 0 or 1 (append `?` for non-greedy)
- `{m}` exactly m occurrences
- `{m, n}` from m to n. m defaults to 0, n to infinity
- `{m, n}?` from m to n, as few as possible

Special sequences

- `\A` start of string
- `\b` matches empty string at word boundary (between `\w` and `\W`)
- `\B` matches empty string not at word boundary
- `\d` digit
- `\D` non-digit
- `\s` whitespace: `[\t\n\r\f\v]`
- `\S` non-whitespace
- `\w` alphanumeric: `[0-9a-zA-Z_]`
- `\W` non-alphanumeric
- `\Z` end of string
- `\g<id>` matches a previously defined group

Extensions

- `(?iLmsux)` Matches empty string, sets re.X flags
- `(?:...)` Non-capturing version of regular parentheses
- `(?P<name>...)` Creates a named capturing group.
- `(?P=name)` Matches whatever matched previously named group
- `(?#...)` A comment; ignored.
- `(?=...)` Lookahead assertion: Matches without consuming
- `(?!...)` Negative lookahead assertion
- `(?<=...)` Lookbehind assertion: Matches if preceded
- `(?<!...)` Negative lookbehind assertion
- `(?(id)yes|no)` Match 'yes' if group 'id' matched, else 'no'