Cheatsheet for Regex

A lookup document briefly explaining regex patterns to match texts

|  |  |
| --- | --- |
| Abc | matches letters ‘Abc’ in the same order. This is case-sensitive. |
| 123 | matches numbers ‘123’ in the same order. |
| . | . metacharacter serves as a placeholder to match any character except the newline character. |
| \* | \* metacharacter matches the previous character or group between zero and unlimited times |
| + | * metacharacter matches the previous character or group between one and unlimited times |
| ? | * metacharacter matches the previous character or group between zero and one times |
| {} | * {m,n} - match m to n (both inclusive) instances of previous character * {m,} - match at least m instances of previous character * {,n} - match up to n (including 0 times) instances of previous character * {n} - match exactly n times |
| \w | matches any alpha-numeric characters (lower case letters, upper case letters, numbers and underscore. |
| \W | matches any non-alpha-numeric characters. |
| \d | matches any digit. This equivalent to [0-9] |
| \D | matches any character that is not a digit. |
| \s | matches any white space character (including tab and newline character) |
| \S | matches any non white space character. |
| [\w\d:.\*\_,{?) ] | * Matches any one of the characters inside the brackets * Characters can be listed individually, e.g. [axz] will match 'a', 'x', or 'z' * Character classes such as \w or \d are also accepted inside a set. * Meta characters lose their special meaning inside sets. For example, [(+\*)] will match any of the literal characters '(', '+', '\*', or ')'. |
| [a-z] | * Inside the set, metacharacter ‘-’ defines a range of characters. For example [a-z] will match lower case letters * a-z will literally match ‘a-z’. |
| [^5] | * ^ metacharacter when used as the first character, negates the set of characters i.e. all the characters not in the set will be matched. For example [^5] will match everything except 5. * ^ has no special meaning if it’s not the first character in the set. [5^] will match both 5 and ^. |
| ^ | * Specifies start of sentence for characters succeeding this special character. * Can be used with $ character |
| $ | * Specifies end of sentence for characters preceding this special character. * Can be used with ^ character for full sentence |
| \b | * Matches, without consuming any characters, immediately between a character matched by \w and a character not matched by \w (in either order) |
| \B | * Matches, without consuming any characters, at the position between two characters matched by \w or \W |
| \ | * Characters appearing in text as literal string and also used as regex commands can be matched using the escape character preceding the character(\). |
| | | * Matches either/or characters rather than using character sets[]. |
|  | Operator Hierarchy A regex operator is matched based on an order of hierarchy. When multiple operators are used, this order of operation (below) is followed:   1. (), Parentheses 2. +, \*, {} and ? Quantifiers 3. ^ and $ Sequences and anchors 4. | Disjunction |
| () | * Operators can be applied to more than single characters to a specific group of characters. * Used to change the order in which operators are applied. * When a pattern is catched, it is saved to a register as a group. * Each group can be matched using the group number in a text. (Backreferencing) |
| (?:) | * Does not record as a captured group * Can be used as an operator order changer. * More efficient to use when changing the order of operator precedence and no backreferencing or no grouping is required. |
| a(?=b) | * Returns only ‘a’ if and only if ‘b’ is followed. (Lookahead assertion) * Looks ahead of a character then decides to return match based on the condition |
| a(?!b) | * Returns only ‘a’ if and only if ‘b’ is **not** followed. (Negative Lookahead assertion) |
| (?<=a)b | * Returns ‘b’ if only if ‘a’ precedes ‘b’ |
| (?<!a)b | * Returns ‘b’ if and only if ‘a’ does **not** precede ‘b’ |

Exercise

# Useful Examples

## Extract Date

## Telephone Numbers

* 1. The US telephone format is 000-000-0000.
  2. The Indian phone number format is 00000-00000.

## Extract email

## Extract URLs (both http, https)

## Extract @mentions in tweets

## Extract #tags in tweets

## Name Titles (mr, Mr. Mrs.)

## CEO or CTO followed by name

## Different abbreviations (USA-state vs US-state)

## Different spellings (for example organisation or organization

## Extract all words that are title cased

## Extract all words that do not have title case

## 

## Write an expression to only match numbers between 0 and 249