/text/ ->regular expression

/abc/ -> exact match abc

[ ] -> collection of characters

/[abc]/ -> any character of a, b or c

[a-z] -> alphabets -lowercase

[A-Z] -> Uppercase alphabets

[a-zA-Z]-> both uppercase and lowercase alphabets

[a-zA-Z0-9] -> lowercase, uppercase alphabets and numbers

[0-9] -> numbers

{min-character, max-character} ->

{8} -> should repeat 8 times

{8,} -> minimum 8 characters

{,30} ->up to 30 characters

{8,30} ->between 8 and 30 characters

/ experssion / -> regular experssion is enclosed within / /

/ / -> regular expression literal

\ -> escape sequence

/ he\/she / he/she

. -> single character

\* -> multiple character , 0 or multiple

+ -> one or multiple

^ -> check from the start

$ -> check at the end

?= -> check the pattern, irrespective of position

@ -> special character @

\-, \. , \$, \\* -> special character

[K1shor@gmail.com](mailto:K1shor@gmail.com)

email: /^([a-z0-9])[a-z0-9\\_\-\.\$]+\@+([a-z])+\.+([a-z])+$/

/^([a-z0-9])[a-z0-9\\_\-\.\$]+\@+([a-z])+\.+(com)$/

/^([a-z])[a-z0-9\\_\-\.]+\@+([a-z])+\.+([a-z])+$/

name: /^([a-zA-Z])+$/

age: /^([0-9])$/

password: /^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[\@\!\\_\-\$\%\^\&\\*]).{8,30}$/

/^(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[\@\!\\_\-\$\%\^\&\\*])$/)

|  |  |
| --- | --- |
| **Metacharacter** | **Description** |
| **^** | Matches the starting position within the string. In line-based tools, it matches the starting position of any line. |
| **.** | Matches any single character (many applications exclude [newlines](https://en.wikipedia.org/wiki/Newline), and exactly which characters are considered newlines is flavor-, character-encoding-, and platform-specific, but it is safe to assume that the line feed character is included). Within POSIX bracket expressions, the dot character matches a literal dot. For example, a.c matches "abc", etc., but [a.c] matches only "a", ".", or "c". |
| **[ ]** | A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] specifies a range which matches any lowercase letter from "a" to "z". These forms can be mixed: [abcx-z] matches "a", "b", "c", "x", "y", or "z", as does [a-cx-z].  The - character is treated as a literal character if it is the last or the first (after the ^, if present) character within the brackets: [abc-], [-abc]. Note that backslash escapes are not allowed. The ] character can be included in a bracket expression if it is the first (after the ^) character: []abc]. |
| **[^ ]** | Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z] matches any single character that is not a lowercase letter from "a" to "z". Likewise, literal characters and ranges can be mixed. |
| **$** | Matches the ending position of the string or the position just before a string-ending newline. In line-based tools, it matches the ending position of any line. |
| **( )** | Defines a marked subexpression. The string matched within the parentheses can be recalled later (see the next entry, \*n*). A marked subexpression is also called a block or capturing group. **BRE mode requires \( \)**. |
| **\*n*** | Matches what the *n*th marked subexpression matched, where *n* is a digit from 1 to 9. This construct is vaguely defined in the POSIX.2 standard. Some tools allow referencing more than nine capturing groups. Also known as a backreference. **backreferences are only supported in BRE mode** |
| **\*** | Matches the preceding element zero or more times. For example, ab\*c matches "ac", "abc", "abbbc", etc. [xyz]\* matches "", "x", "y", "z", "zx", "zyx", "xyzzy", and so on. (ab)\* matches "", "ab", "abab", "ababab", and so on. |
| **{*m*,*n*}** | Matches the preceding element at least *m* and not more than *n* times. For example, a{3,5} matches only "aaa", "aaaa", and "aaaaa". This is not found in a few older instances of regexes. **BRE mode requires \{*m*,*n*\}**. |

**Examples:**

* .at matches any three-character string ending with any character ending with "at", including "hat", "cat", "bat", "4at", "#at" and " at" (starting with a space).
* [hc]at matches "hat" and "cat".
* [^b]at matches all strings matched by .at except "bat".
* [^hc]at matches all strings matched by .at other than "hat" and "cat".
* ^[hc]at matches "hat" and "cat", but only at the beginning of the string or line.
* [hc]at$ matches "hat" and "cat", but only at the end of the string or line.
* \[.\] matches any single character surrounded by "[" and "]" since the brackets are escaped, for example: "[a]", "[b]", "[7]", "[@]", "[]]", and "[ ]" (bracket space bracket).
* s.\* matches s followed by zero or more characters, for example: "s", "saw", "seed", "s3w96.7", and "s6#h%(>>>m n mQ".