# Regular Expression Syntax Reference

This section provides a brief summary of regexp syntax that can be helpful for creating search and issue navigation patterns.

In this section:



### Regex syntax reference

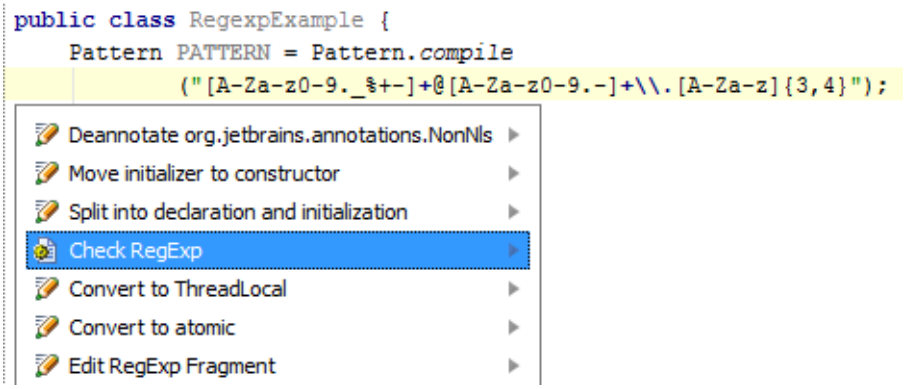
| Character | Description |
|---|---|
| **** | Marks the next character as either a special character or a literal. For example:<br>■ *n* matches the character *n*. "\n" matches a newline character.<br>■ The sequence "\\" matches "\" and "\(" matches "(". |
| **^** | Matches the beginning of input. |
| **$** | Matches the end of input. |
| * | Matches the preceding character zero or more times. For example, "zo*" matches either *z* or *zoo*. |
| + | Matches the preceding character one or more times. For example, "zo+" matches *zoo* but not *z*. |
| ? | Matches the preceding character zero or one time. For example, "a?ve?" matches the *ve* in *never*. |
| . | Matches any single character except a newline character. |
| **(**subexpression**)** | Matches *subexpression* and remembers the match. If a part of a regular expression is enclosed in parentheses, that part of the regular expression is grouped together. Thus a regex operator can be applied to the entire group.<br>■ If you need to use the matched substring within the same regular expression, you can retrieve it using the backreference (\num, where num = 1..n).<br>■ If you need to refer the matched substring somewhere outside the current regular expression (for example, in another regular expression as a replacement string), you can retrieve it using the dollar sign ($num, where num = 1..n).<br>■ If you need to include the parentheses characters into a *subexpression*, use "\(" or "\)". |
| x **\|** y | Matches either *x* or *y*. For example, "z\|wood" matches *z* or *wood*. "(z\|w)oo" matches *zoo* or *wood*. |
| {*n*} | *n* is a nonnegative integer. Matches exactly *n* times. For example, "o{2}" does not match the *o* in *Bob*, but matches the first two o's in *foooood*. |

| Character | Description |
| --- | --- |
| {*n*,} | *n* is a nonnegative integer. Matches at least *n* times. For example, "o{2,}" does not match the *o* in *Bob* and matches all the o's in "foooood." "o{1,}" is equivalent to "o+". "o{0,}" is equivalent to "o*". |
| **{** *n* **,** *m* **}** | *m* and *n* are nonnegative integers. Matches at least *n* and at most *m* times. For example, "o{1,3}" matches the first three o's in "fooooood." "o{0,1}" is equivalent to "o?". |
| **[** *xyz* **]** | A character set. Matches any one of the enclosed characters. For example, "[abc]" matches the *a* in *plain*. |
| **[^** *xyz* **]** | A negative character set. Matches any character not enclosed. For example, "[^abc]" matches the *p* in *plain*. |
| **[** *a-z* **]** | A range of characters. Matches any character in the specified range. For example, "[a-z]" matches any lowercase alphabetic character in the range *a* through *z*. |
| **[^** *m-z* **]** | A negative range characters. Matches any character not in the specified range. For example, "[m-z]" matches any character not in the range *m* through *z*. |
| \b | Matches a word boundary, that is, the position between a word and a space. For example, "er\b" matches the *er* in *never* but not the *er* in *verb*. |
| \B | Matches a non-word boundary. "ea*r\B" matches the *ear* in *never early*. |
| \d | Matches a digit character. Equivalent to [0-9]. |
| \D | Matches a non-digit character. Equivalent to [^0-9]. |
| \f | Matches a form-feed character. |
| \n | Matches a newline character. |
| \r | Matches a carriage return character. |
| \s | Matches any white space including space, tab, form-feed, etc. Equivalent to "[ \f\n\r\t\v]". |
| \S | Matches any nonwhite space character. Equivalent to "[^ \f\n\r\t\v]". |
| \t | Matches a tab character. |
| \v | Matches a vertical tab character. |
| \w | Matches any word character including underscore. Equivalent to "[A-Za-z0-9_]". |
| \W | Matches any non-word character. Equivalent to "[^A-Za-z0-9_]". |
| ****num | Matches *num*, where *num* is a positive integer, denoting a reference back to remembered matches. For example, "(.)\1" matches two consecutive identical characters. |

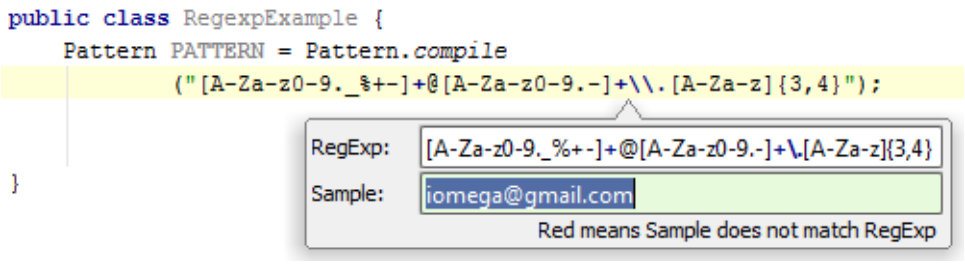
| Character | Description |
|---|---|
| \ *n* | Matches *n*, where *n* is an octal escape value. Octal escape values should be 1, 2, or 3 digits long. For example, "\11" and "\011" both match a tab character. "\0011" is the equivalent of "\001" &*1*. Octal escape values should not exceed 256. If they do, only the first two digits comprise the expression. Allows ASCII codes to be used in regular expressions. |
| \x *n* | Matches *n*, where *n* is a hexadecimal escape value. Hexadecimal escape values must be exactly two digits long. For example, "\x41" matches *A*. "\x041" is equivalent to "\x04" &*1*. Allows ASCII codes to be used in regular expressions. |
| \\$ | Escapes $. |

### Tips and tricks

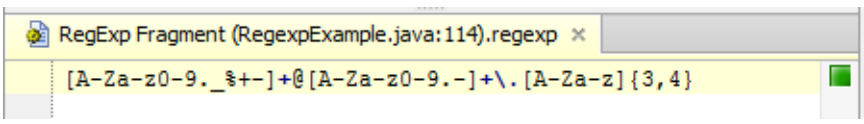
IntelliJ IDEA provides intention actions to check validity of the regular expressions, and edit regular expressions in a scratchpad. Place the caret at a regular expression, and press **Alt+Enter**. The suggestion list of intention actions, available in this context, appears:




- Choose **Check RegExp**, and press **Enter**. The dialog box that pops up, shows the current regular expression in the upper pane. In the lower pane, type the string to which this expression should match. If the regular expression matches the entered string, the background becomes green. If the regular expression doesn't match, then the background is red.




- Choose **Edit RegExp Fragment**, and press **Enter**. The regular expression opens for editing in a separate tab in the editor. However, this is but a scratchpad, and no file is physically created:




  As you type in the scratchpad, all changes are synchronized with the original regular expression. Press **Escape** to close the editor tab.

### See Also

Reference:

- Finding and Replacing Text in File
- Version Control

External Links:

- Regex Tutorial

Web Resources:

- Developer Community