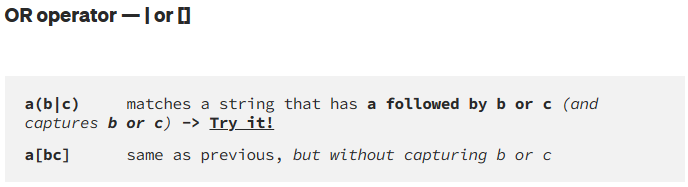
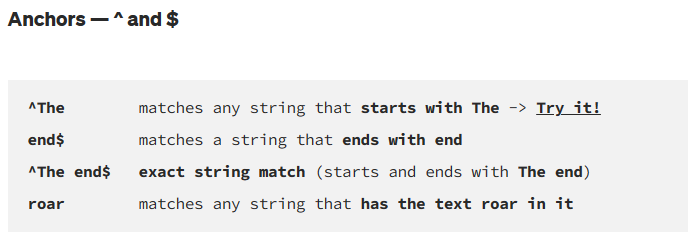
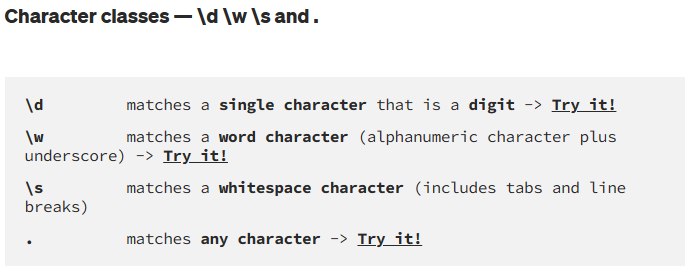
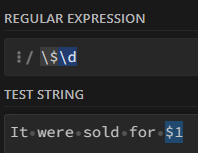
**RegEx**

****

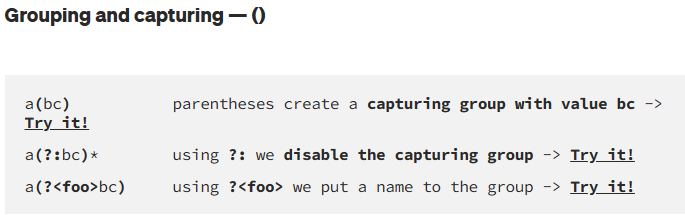
****

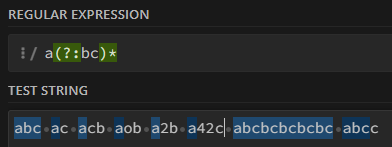
* **\d \w and \s also present their negations with \D, \W and \S respectively For example, \D will perform the inverse match with respect to that obtained with \d**
* **In order to be taken literally, you must escape the characters ^.[$()|\*+?{\ with a backslash \ as they have special meaning.**

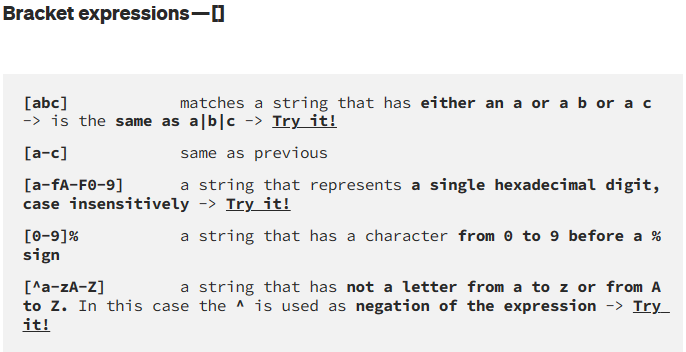
****

**Flags**

* **g (global)** does not return after the first match, restarting the subsequent searches from the end of the previous match
* **m (multi-line)** when enabled ^ and $ will match the start and end of a line, instead of the whole string
* **i (insensitive)** makes the whole expression case-insensitive (for instance /aBc/i would match AbC)

****

****

****

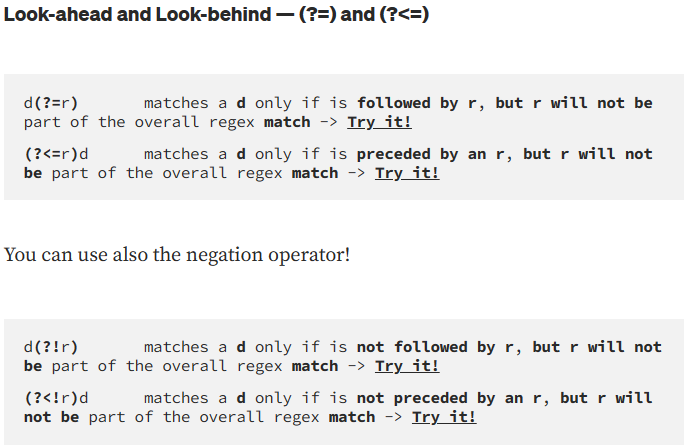
In regular expressions, "**greedy**" and "**lazy**" are two quantifier behaviours that affect how matching is performed. Quantifiers are used to specify how many times a character or group of characters should be repeated in a regex pattern.

1. **Greedy Matching:** Greedy quantifiers try to match as much text as possible while still allowing the overall regex pattern to match. They are denoted by the following characters:
   * \* (asterisk): Matches zero or more occurrences of the preceding character or group.
   * + (plus): Matches one or more occurrences of the preceding character or group.
   * ? (question mark): Matches zero or one occurrence of the preceding character or group.
   * {m,n} (curly braces): Matches at least m and at most n occurrences of the preceding character or group.

For example, in the regex pattern a.\*b, the \* is a greedy quantifier, and it will match as much text as possible between the letter 'a' and the letter 'b', even if there are multiple occurrences of 'b' in between. So, given the string "abcdbefb", the pattern a.\*b will match the entire string "abcdbefb" because it starts from the first 'a' and ends at the last 'b'.

1. **Lazy Matching:** Lazy quantifiers, also known as "non-greedy" or "reluctant" quantifiers, try to match as little text as possible while still allowing the overall regex pattern to match. **They are denoted by adding a ? immediately after a greedy quantifier**:
   * \*?: Makes the preceding character or group match zero or more occurrences, but as few as possible.
   * +?: Makes the preceding character or group match one or more occurrences, but as few as possible.
   * ??: Makes the preceding character or group match zero or one occurrence, but as few as possible.
   * {m,n}?: Makes the preceding character or group match at least m and at most n occurrences, but as few as possible.

For example, in the regex pattern a.\*?b, the \*? is a lazy quantifier, and it will match as little text as possible between the letter 'a' and the letter 'b'. So, given the string "abcdbefb", the pattern a.\*?b will match only "ab" because it starts from the first 'a' and ends at the first 'b' encountered.



Back-references are a powerful feature in regular expressions (regex) that allow you to refer to previously matched groups within the same regex pattern. They are used to match repeated or similar patterns in a string.

In regex, a capturing group is a portion of a pattern enclosed in parentheses (). When a capturing group matches a substring in the input string, the matched substring is remembered and can be referred to later using a back-reference.

The syntax for a back-reference is \n, where n is the index of the capturing group you want to refer to. The first capturing group is referred to as \1, the second capturing group as \2, and so on.

Here's an example to illustrate the concept of back-references:

Let's say you want to match a string that consists of two consecutive identical words, such as "hello hello" or "foo foo". You can use a capturing group to match the first word, and then use a back-reference to refer to that captured word in the second part of the pattern.

The regex pattern would be: (\w+)\s+\1

Let's break it down:

(\w+): This is the first capturing group. It matches one or more word characters (letters, digits, or underscores) and captures them.

\s+: This matches one or more whitespace characters (spaces, tabs, or newlines) that separate the two words.

\1: This is the back-reference to the first capturing group. It ensures that the same text that was captured in the first group is matched here.

So, the pattern (\w+)\s+\1 would match strings like "hello hello", "foo foo", "123 123", etc., where the same word appears consecutively.

Back-references are a powerful tool in regex that allow you to match repeated or similar patterns in a concise and efficient way, and they are widely used in many regex applications.

