PYTHON:

## RegEx Functions

The re module offers a set of functions that allows us to search a string for a match:

|  |  |
| --- | --- |
| **Function** | **Description** |
| [findall](https://www.w3schools.com/python/python_regex.asp#findall) | Returns a list containing all matches |
| [search](https://www.w3schools.com/python/python_regex.asp#search) | Returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match anywhere in the string |
| [split](https://www.w3schools.com/python/python_regex.asp#split) | Returns a list where the string has been split at each match |
| [sub](https://www.w3schools.com/python/python_regex.asp#sub) | Replaces one or many matches with a string |

## Metacharacters

Metacharacters are characters with a special meaning:

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Example** | **Try it** |
| [] | A set of characters | "[a-m]" |  |

|  |  |  |  |
| --- | --- | --- | --- |
| \ | Signals a special sequence (can also be used to escape special characters) | "\d" |  |
| . | Any character (except newline character) | "he..o" |  |
| ^ | Starts with | "^hello" |  |
| $ | Ends with | "world$" |  |
| \* | Zero or more occurrences | "aix\*" |  |
| + | One or more occurrences | "aix+" |  |
| {} | Exactly the specified number of occurrences | "al{2}" |  |
| | | Either or | "falls|stays" |  |

|  |  |  |  |
| --- | --- | --- | --- |
| () | Capture and group |  |  |

## Special Sequences

A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Example** | **Try it** |
| \A | Returns a match if the specified characters are at the beginning of the string | "\AThe" |  |

|  |  |  |  |
| --- | --- | --- | --- |
| \b | Returns a match where the specified characters are at the beginning or at the end of a word | r"\bain" r"ain\b" |  |

|  |  |  |  |
| --- | --- | --- | --- |
| \B | Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word | r"\Bain" r"ain\B" |  |

|  |  |  |  |
| --- | --- | --- | --- |
| \d | Returns a match where the string contains digits (numbers from 0-9) | "\d" |  |
| \D | Returns a match where the string DOES NOT contain digits | "\D" |  |
| \s | Returns a match where the string contains a white space character | "\s" |  |
| \S | Returns a match where the string DOES NOT contain a white space character | "\S" |  |
| \w | Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore \_ character) | "\w" |  |
| \W | Returns a match where the string DOES NOT contain any word characters | "\W" |  |
| \Z | Returns a match if the specified characters are at the end of the string | "Spain\Z" |  |

## Sets

A set is a set of characters inside a pair of square brackets [] with a special meaning:

|  |  |  |
| --- | --- | --- |
| **Set** | **Description** | **Try it** |
| [arn] | Returns a match where one of the specified characters (a, r, or n) are present |  |

|  |  |  |
| --- | --- | --- |
| [a-n] | Returns a match for any lower case character, alphabetically between a and n |  |
| [^arn] | Returns a match for any character EXCEPT a, r, and n |  |
| [0123] | Returns a match where any of the specified digits (0, 1, 2, or 3) are present |  |
| [0-9] | Returns a match for any digit between 0 and 9 |  |
| [0-5][0-9] | Returns a match for any two-digit numbers from 00 and 59 |  |
| [a-zA-Z] | Returns a match for any character alphabetically between a and z, lower case OR upper case |  |
| [+] | In sets, +, \*, ., |, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string |  |

JAVASCRIPT:

Regular Expressions:

Methods:

| Methods that use regular expressions | |
| --- | --- |
| **Method** | **Description** |
| [exec](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/exec) | A RegExp method that executes a search for a match in a string. It returns an array of information or null on a mismatch. |
| [test](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/test) | A RegExp method that tests for a match in a string. It returns true or false. |
| [match](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/match) | A String method that returns an array containing all of the matches, including capturing groups, or null if no match is found. |
| [matchAll](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/matchAll) | A String method that returns an iterator containing all of the matches, including capturing groups. |
| [search](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/search) | A String method that tests for a match in a string. It returns the index of the match, or -1 if the search fails. |
| [replace](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/replace) | A String method that executes a search for a match in a string, and replaces the matched substring with a replacement substring. |
| [split](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/split) | A String method that uses a regular expression or a fixed string to break a string into an array of substrings. |
| Flag | Description |
| g | Global search. |
| i | Case-insensitive search. |
| m | Multi-line search. |
| s | Allows . to match newline characters. (Added in ES2018, not yet supported in Firefox) |
| u | "unicode"; treat a pattern as a sequence of unicode code points |
| y | Perform a "sticky" search that matches starting at the current position in the target string. See [sticky](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/sticky) |

var re = /pattern/flags;

or

var re = new RegExp('pattern', 'flags');

Creating RegExp:

RegExp literal:

var re = /ab+c/;

Note: Use if regexp will remain constant:

Regular expression literals provide compilation of the regular expression when the script is loaded. If the regular expression remains constant, using this can improve performance.

Constructor of RegExp:

var re = new RegExp('ab+c');

Note: Use if regexp will be changing:

Using the constructor function provides runtime compilation of the regular expression. Use the constructor function when you know the regular expression pattern will be changing, or you don't know the pattern and are getting it from another source, such as user input.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Char | Name | Example | | Details | |
|  |  | String Literal | New RegExp() | Matches  Doesn’t Match | Further Explanation |
| \ | Backslash as deonting special Char | \b | (“[\\b](file:///\\b)”) | A backslash that precedes a non-special character indicates that the next character is special  Matches C:\  for regexp must escape at string literal and regexp level | |
| \ | Backslash as escape char | /C:\\/ | “/C:\\\\” |  |  |
| ^ | Begin Input | /^A/ |  | =“An E”  != “an A” | beginning of input. If (multiline flag=true), immediately after a line break |
| ^ | Complemented Char |  |  |  |  |
| $ | End of Input | /t$/ | =”eat”  !=”eating” | |  |
| \* | Preceeding Exp 0 or more | /bo\*/ | =”booooed”  =”bird”  !=”goat” | |  |
| + | Preceeding Exp 1 or more | /a+/ | =”candy”  =”caaaandy”  !=”booooed” | |  |
| ? | Non-greedy modifier | /\d+/ vs. /\d+?/ | 1st: ‘123abc’  =’123’ | With ‘?’:  ‘123abc’  = ‘1 | After \* + ? or {} returns fewest possible |
| . | Decimal Point | /.n/ | =”an”, “on”  !=”nay” | | Any single char != newline. If “dotAll”=true, also = newline |
| () | Capturing Parentheses | /(foo)/  /(foo) \1 \2/ | =”foo” in “foo bar”  =”foo & foo” in “foo bar foo” | |  |
| (?:x) | Non-capturing | /(?:foo){1,2} vs.  /foo {1,2} | =”foo” in “foo”  =” last ‘o’” in “foo” | | |
| X(?=y) | X only if y is after | /Jack(?=Sprat)/ | =”Jack” in “Jack Sprat”  !=”Jack” in “Jack Black” | | |
| X(?!y) | X only is y is NOT after | /Jack(?!Sprat)/ | =”Jack” in “Jack Black”  !=”Jack” in “Jack Sprat” | | |
| (?<=y)x | X only if Y is before | /(?<=Jack)Sprat/ | =”Sprat” in “Jack Sprat”  !=”Sprat” in “Dave Sprat” | | |
| (?<!y)x | X only if Y is NOT before | /(?<!Jack)Sprat/ | =”Sprat” in “Dave Sprat”  !=”Sprat” in “Jack Sprat” | | |
| X|y | X if no X then Y | /green|red/ | =”green” in “big green red”  =”red” in “big red” | | |
| {n} | N occurences of previous exp | /a{2}/ | =”aa” in “aaaa” (first 2 a’s)  !=”a” in “a” | | |
| {n,} | At least N occurences | /a{2,}/ | =”aaaa” in “aaaa”  != “a” in “a” | | |
| {n, m} | At Least N but no more than M | /a{2,4}/ | =”aa” in “baad” = “aaaa” in “caaaad”  Only “aaaa” in “daaaaaaaaaaaad” | | |
| [xyz] | Char set, . \* aren’t special | /[a-z]+/ | All in “test.in.ng” | | |
| [^xyz] | Negated char set | /[^abc]/ | =”r” in “barc” | | |
| [\b] | Backspace |  |  | | |
| \b | Word boundary | Given string “moon” | /\bm/ = because m is at beginning  /oo\b/ != because oo is in middle  /on\b/ = because on is at the end | | |
| \B | NON word boundary | /\B../ | =”oo” in “noonday”  =”ye” in “possibly yesterday  =” “ in “bigspace “ | | |
| \cX | Control char where X is A-Z | /\cM/ | Matches control-M | | |
| \d | Digits [0-9] |  |  | | |
| \D | NON Digits |  |  | | |
| \f | Form feed |  |  | | |
| \n | Newline |  |  | | |
| \r | Carriage Return |  |  | | |
| \s | Whitespace | Includes space, tab, form feed, line feed |  | | |
| \S | NON whitespace |  |  | | |
| \t | Tab | (U+0009) |  | | |
| \v | Vertical Tab | (U+000B) |  | | |
| \w | Alphanumeric + underscore | [A-Za-z0-9\_] |  | | |
| \W | Any NON word char |  |  | | |
| \n | Back reference to last substring |  | For example, /apple(,)\sorange\1/ matches 'apple, orange,' in "apple, orange, cherry, peach." | | |
| \0 | NULL | (U+0000) |  | | |
| \xhh | Two hexadecimal digits | |  | | |
| \uhhhh | Four hexadecimal digits | |  | | |
| \ | Escaping | /a\\*/  /\/ex\// | Matches ‘a\*’ not ‘\*b’  Matches ‘/ex/’ | | |
|  |  |  |  | | |
|  |  |  |  | | |
|  |  |  |  | | |
|  |  |  |  | | |
|  |  |  |  | | |