Regular Expression in PYTHON

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

RegEx can be used to check if a string contains the specified search pattern.

RegEx Module

Python has a built-in package called re, which can be used to work with Regular Expressions.

Import the re module:

import re

RegEx Functions

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

|  |  |
| --- | --- |
| match | Returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match at the beginning of the string |
| [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** |  |
| [] | 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+" |  |
| {} | Excactly 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** |
| \A | Returns a match if the specified characters are at the beginning of the string |
| \b | Returns a match where the specified characters are at the beginning or at the end of a word |
| \B | Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word |
| \d | Returns a match where the string contains digits (numbers from 0-9) |
| \D | Returns a match where the string DOES NOT contain digits |
| \s | Returns a match where the string contains a white space character |
| \S | Returns a match where the string DOES NOT contain a white space character |
| \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 | Returns a match where the string DOES NOT contain any word characters |
| \Z | Returns a match if the specified characters are at the end of the string |

## Sets

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Set** | **Description** |  | |  |
| [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 | |  |  |

**MATCH**

### re.match(pattern, string):

This method finds match if it occurs at start of the string returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match. If there are no match, the value None will be returned, instead of the Match Object.

For example, calling match() on the string ‘pet:cat I Love cats’ looking for a pattern pet:\w\w\w’ .Let’s perform it in python now.

import re

s=”pet:cat I love cats”

result = re.match(r'pet:\w\w\w',s)

print(result)

To print the matching string we’ll use method group (It helps to return the matching string). Use “r” at the start of the pattern string, it designates a python raw string.

print(result.group(0)

**Output:pet:cat**

## Search

### re.search(pattern, string):

It is similar to match() but it doesn’t restrict us to find matches at the beginning of the string only The search() function searches the string for a match, and returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match.

If there is more than one match, only the first occurrence of the match will be returned:

Import re

s='pet:cat I love cats pet:cow I love cows'

result = re.search(r'pet:\w\w\w',s)

print(result.group(0))

**Output:pet:cat**

The Match object has properties and methods used to retrieve information about the search, and the result:

.span() returns a tuple containing the start-, and end positions of the match.  
.string returns the string passed into the function  
.group() returns the part of the string where there was a match

SPAN

import re

#Search for an upper case "S" character in the beginning of a word, and print its position:

*str = "The rain in Spain"*

x = re.search(r"\bS\w+", str)

print(x.span())

**Output:(12,17)**

STRING

import re

#The string property returns the search string:

x = re.search(r"\bS\w+", str)

print(x.string())

**Output:** The rain in Spain

Group

import re

#Search for an upper case "S" character in the beginning of a word, and print the word:

str = "The rain in Spain"

x = re.search(r"\bS\w+", str)

print(x.group())

**Output:** spain

### FINDALL

### re.findall (pattern, string):

Ithelps to get a list of all matching patterns. It has no constraints of searching from start or end. If we will use method findall to search ‘pet:\w\w\w’ in given string it will return pet:cat and pet:cow both.it can work like re.search() and re.match() both.

Import re

s='pet:cat I love cats pet:cow I love cows'

result = re.findall(r'pet:\w\w\w',s)

print (result)

**Output:**

['pet:cat', 'pet:cow']

**SPLIT**

### re.split(pattern, string, [maxsplit=0]):

This methods helps to split string by the occurrences of given pattern

Import re

#Split the string at every white-space character:

str = "The rain in Spain"

x = re.split("\s", str)

print(x)

**Output:**

[‘The’,’rain’,’in’,’spain’]

You can control the number of occurrences by specifying the maxsplit parameter

Import re

#Split the string at every white-space character:

str = "The rain in Spain"

x = re.split("\s", str,1)

print(x)

**Output:**

[‘The’,’rain in spain’]

**SUB**

### re.sub(pattern, repl, string):

It helps to search a pattern and replace with a new sub string. If the pattern is not found, string is returned unchanged.

import re

#Replace all white-space characters with the digit "9":

str = "The rain in Spain"

x = re.sub("\s", "9", str)

print(x)

**Output: :The9rain9in9spain**

You can control the number of replacements by specifying the count parameter:

import re

#Replace all white-space characters with the digit "9":

str = "The rain in Spain"

x = re.sub("\s", "9", str,2)

print(x)

**Output: :The9rain9in spain**

### COMPILE

### re.compile(pattern, repl, string):

We can combine a regular expression pattern into pattern objects, which can be used for pattern matching. It also helps to search a pattern again without rewriting it.

import re

pattern=re.compile('flower')

result=pattern.findall('The flower is sunflower')

print(result)

result2=pattern.findall('The flower is sunflower')

print(result2)

**Output:**

['flower', 'flower']

['flower']