Python RegEx

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

When you have imported the re module, you can start using regular expressions:

### **Example**

Search the string to see if it starts with "The" and ends with "Spain":

**import re  
  
txt = "The rain in Spain"  
x = re.search("^The.\*Spain$", txt)**

## **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** |
| [] | 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** |
| \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 (the "r" in the beginning is making sure that the string is being treated as a "raw string") | 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 (the "r" in the beginning is making sure that the string is being treated as a "raw string") | 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** |
| [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 |

## **The findall() Function**

The findall() function returns a list containing all matches.

### **Example**

Print a list of all matches:

import re  
  
txt = "The rain in Spain"  
x = re.findall("ai", txt)  
print(x)

Return an empty list if no match was found:

import re  
  
txt = "The rain in Spain"  
x = re.findall("Portugal", txt)  
print(x)

## **The search() Function**

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:

### **Example**

Search for the first white-space character in the string:

import re  
  
txt = "The rain in Spain"  
x = re.search("\s", txt)  
  
print("The first white-space character is located in position:", x.start())

Make a search that returns no match:

import re  
  
txt = "The rain in Spain"  
x = re.search("Portugal", txt)  
print(x)

If no matches are found, the value None is returned:

## **The split() Function**

The split() function returns a list where the string has been split at each match:

### **Example**

Split at each white-space character:

import re  
  
txt = "The rain in Spain"  
x = re.split("\s", txt)  
print(x)

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

### **Example**

Split the string only at the first occurrence:

import re  
  
txt = "The rain in Spain"  
x = re.split("\s", txt, 1)  
print(x)

## **The sub() Function**

The sub() function replaces the matches with the text of your choice:

### **Example**

Replace every white-space character with the number 9:

import re  
  
txt = "The rain in Spain"  
x = re.sub("\s", "9", txt)  
print(x)

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

### **Example**

Replace the first 2 occurrences:

import re  
  
txt = "The rain in Spain"  
x = re.sub("\s", "9", txt, 2)  
print(x)

## **Match Object**

A Match Object is an object containing information about the search and the result.

**Note:** If there is no match, the value None will be returned, instead of the Match Object.

### **Example**

Do a search that will return a Match Object:

import re  
  
txt = "The rain in Spain"  
x = re.search("ai", txt)  
print(x) #this will print an object

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

### **Example**

Print the position (start- and end-position) of the first match occurrence.

The regular expression looks for any words that starts with an upper case "S":

import re  
  
txt = "The rain in Spain"  
x = re.search(r"\bS\w+", txt)  
print(**x.span()**)

### **Example**

Print the string passed into the function:

import re  
  
txt = "The rain in Spain"  
x = re.search(r"\bS\w+", txt)  
print(**x.string**)

### **Example**

Print the part of the string where there was a match.

The regular expression looks for any words that starts with an upper case "S":

import re  
  
txt = "The rain in Spain"  
x = re.search(r"\bS\w+", txt)  
print(**x.group()**)

**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.

Exercises

1. Write a Python program to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

import re

def is\_allowed\_specific\_char(string):

charRe = re.compile(r'[^a-zA-Z0-9.]')

string = charRe.search(string)

return not bool(string)

print(is\_allowed\_specific\_char("ABCDEFabcdef123450"))

print(is\_allowed\_specific\_char("\*&%@#!}{"))

1. Write a Python program that matches a string that has an a followed by zero or more b's.

import re

def text\_match(text):

patterns = '^ab\*?'

if re.search(patterns, text):

return 'Found a match!'

else:

return('Not matched!')

print(text\_match("a"))

Write a Python program that matches a string that has an a followed by one or more b's.

import re

def text\_match(text):

patterns = '^ab+?'

if re.search(patterns, text):

return 'Found a match!'

else:

return('Not matched!')

print(text\_match("a"))

1. Write a Python program that matches a string that has an a followed by zero or one 'b'.

import re

def text\_match(text):

patterns = 'ab?'

if re.search(patterns, text):

return 'Found a match!'

else:

return('Not matched!')

print(text\_match("ab"))

print(text\_match("abc"))

print(text\_match("abbc"))

print(text\_match("aabbc"))

1. Write a Python program that matches a string that has an a followed by three 'b'.

import re

def text\_match(text):

patterns = '^abbb?'

if re.search(patterns, text):

return('Found a match!')

else:

return ('Not matched!')

print(text\_match("abbb"))

or

import re

def text\_match(text):

patterns = '^ab{3}?'

if re.search(patterns, text):

return('Found a match!')

else:

return ('Not matched!')

print(text\_match("aabba"))

1. Write a Python program that matches a string that has an a followed by two to three 'b'.

import re

def text\_match(text):

patterns = '^ab{2,3}?'

if re.search(patterns, text):

return('Found a match!')

else:

return ('Not matched!')

print(text\_match("abb"))

1. Write a Python program to find sequences of lowercase letters joined with a underscore

import re

def text\_match(text):

patterns = '^[a-z]+\_[a-z]+$'

if re.search(patterns, text):-

return('Found a match!')

else:

return ('Not matched!')

print(text\_match("a\_bb"))

1. Write a Python program to find sequences of one upper case letter followed by lower case letters.

import re

def text\_match(text):

patterns = '^[A-Z]+[a-z]+$'

if re.search(patterns, text):

return('Found a match!')

else:

return ('Not matched!')

print(text\_match("AbbC"))

1. Write a Python program that matches a string that has an 'a' followed by anything, ending in 'b'.

import re

def text\_match(text):

patterns = '^a.\*?b$'

if re.search(patterns, text):

return('Found a match!')

else:

return ('Not matched!')

print(text\_match("addZb"))

1. Write a Python program that matches a word at the beginning of a string.

import re

def text\_match(text):

patterns = '^\w+'

if re.search(patterns, text):

return 'Found a match!'

else:

return('Not matched!')

print(text\_match("The sun shines."))

print(text\_match(" The sun shines."))

1. Write a Python program that matches a word at end of string, with optional punctuation.

import re

def text\_match(text):

patterns = '\w+\S\*$'

if re.search(patterns, text):

return 'Found a match!'

else:

return('Not matched!')

print(text\_match("The sun shines."))

print(text\_match(" The sun shines. "))

1. Write a Python program that matches a word containing 'z'.

import re

def text\_match(text):

patterns = '\w\*z.\w\*'

if re.search(patterns, text):

return 'Found a match!'

else:

return('Not matched!')

print(text\_match("The suzn shines."))

print(text\_match(" The sun shines. "))

1. Write a Python program that matches a word containing 'z', not start or end of the word.
2. Write a Python program to match a string that contains only upper and lowercase letters, numbers, and underscores.
3. Write a Python program where a string will start with a specific number.

import re

def match\_num(string):

text = re.compile(r"^5")

if text.match(string):

return True

else:

return False

print(match\_num('5-2345861'))

print(match\_num('6-2345861'))

1. Write a Python program to remove leading zeros from an IP address.
2. Write a Python program to check for a number at the end of a string.
3. Write a Python program to search the numbers (0-9) of length between 1 to 3 in a given string.
4. Write a Python program to search some literals strings in a string.   
   Sample text : 'The quick brown fox jumps over the lazy dog.'  
   Searched words : 'fox', 'dog', 'horse'

**20.** Write a Python program to replace whitespaces with an underscore and vice versa.

**21.** Write a Python program to find all words starting with 'a' or 'e' in a given string.

**22.** Write a Python program to separate and print the numbers and their position of a given string.

**23.** Write a Python program to replace all occurrences of space, comma, or dot with a colon.

**24.** Write a Python program to find all three, four, five characters long words in a string.

**25.** Write a Python program to find all words which are at least 4 characters long in a string.

**26.** Write a Python program to remove all whitespaces from a string.

**27.** Write a Python program to remove everything except alphanumeric characters from a string.

**28.** Write a Python program to split a string at uppercase letters.

**29.** Write a Python program to do a case-insensitive string replacement.

**30.** Write a Python program to check a decimal with a precision of 2.