# Python RegEx

‹ Previous
Next ›

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 in Python

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

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)
```

Try it Yourself »

## RegEx Functions

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

| Function | Description |
|---|---|
| findall | Returns a list containing all matches |
| search | Returns a Match object if there is a match anywhere in the string |
| split | Returns a list where the string has been split at each match |
| 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]" | Try it » |
| \ | Signals a special sequence (can also be used to escape special characters) | "\d" | |

| | | | |
|---|---|---|---|
| ^ | Starts with | "^hello" | Try it » |
| $ | Ends with | "planet$" | Try it » |
| * | Zero or more occurrences | "he.*o" | Try it » |
| + | One or more occurrences | "he.+o" | Try it » |
| ? | Zero or one occurrences | "he.?o" | Try it » |
| {} | Exactly the specified number of occurrences | "he.{2}o" | Try it » |
| \| | Either or | "falls\|stays" | Try it » |
| () | 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" | Try it » |
| \b | Returns a match where the specified characters are at the beginning or at the end of a word<br>(the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\bain"<br>r"ain\b" | Try it »<br>Try it » |
| \B | Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word<br>(the "r" in the beginning is making sure that the string is being treated as a "raw string") | r"\Bain"<br>r"ain\B" | Try it »<br>Try it » |
| \d | Returns a match where the string contains | "\d" | |

| | | | |
|---|---|---|---|
| \D | Returns a match where the string DOES NOT contain digits | "\D" | Try it » |
| \s | Returns a match where the string contains a white space character | "\s" | Try it » |
| \S | Returns a match where the string DOES NOT contain a white space character | "\S" | Try it » |
| \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" | Try it » |
| \W | Returns a match where the string DOES NOT contain any word characters | "\W" | Try it » |
| \Z | Returns a match if the specified characters are at the end of the string | "Spain\Z" | Try it » |

# 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 | Try it » |
| [a-n] | Returns a match for any lower case character, alphabetically between `a` and `n` | Try it » |
| [^arn] | Returns a match for any character EXCEPT `a` , `r` , and `n` | Try it » |
| [0123] | Returns a match where any of the specified digits ( `0` , `1` , `2` , or `3` ) are present | Try it » |
| [0-9] | Returns a match for any digit between `0` and `9` | Try it » |
| [0-5][0-9] | Returns a match for any two-digit numbers from `00` a | |

| | | |
|---|---|---|
| [a-zA-Z] | Returns a match for any character alphabetically between `a` and `z`, lower case OR upper case | Try it » |
| [+] | In sets, `+`, `*`, `.`, `\|`, `()`, `$`,`{}` has no special meaning, so `[+]` means: return a match for any `+` character in the string | Try it » |

# 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)
```

Try it Yourself »

The list contains the matches in the order they are found.

If no matches are found, an empty list is returned:

## Example

Return an empty list if no match was found:

```
import re

txt = "The rain in Spain"
```

Try it Yourself »

# The search() Function

The `search()` function searches the string for a match, and returns a Match object 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())
```

Try it Yourself »

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

## Example

Make a search that returns no match:

```
import re

txt = "The rain in Spain"
```

Try it Yourself »

# 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)
```

Try it Yourself »

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)
```

Try it Yourself »

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)
```

Try it Yourself »

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
```

Try it Yourself »

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"
```

Try it Yourself »

## 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)
```

Try it Yourself »

## 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())
```

Try it Yourself »

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