RegEx Functions

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. Python provides the built-in function called **re** which can be used to work with Regular Expressions. The below syntax is used to load **re** module.

```
import re
```

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 Math object if there is a match anywhere in the
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
[]	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	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	Any digit, equivalent to [0-9]	“\d”
\D	Any non-digit, equivalent to [^ 0-9]	“\D”
\s	Any whitespace, equivalent to [\t\n\r\f\v]	“\s”
\S	Any non-whitespace, equivalent to [^ \t \n \r \f \v]	“\S”
\w	Any Alphanumeric character, equivalent to [a-zA-Z0-9_]	“\w”
\W	Any Non-alphanumeric character, equivalent to [^ a-zA-Z0-9_]	“\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
[a-n]	Returns a match for any lower case character, alphabetically
[^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
[0-9]	Returns a match for any digit between 0 and 9
[0 - 5]	Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a
[+]	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. If no match found, return an empty list.

```
In [1]:
import re
str = "the weather is very pleasant."
x = re.findall("ea", str)
y = re.findall("a very", str)

In [2]: x
Out[2]: ['ea', 'ea']
```

```
In [3]: y
Out[3]: []
```

• • •

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.

```
In [4]:
import re
str = "the weather is very pleasant."
x = re.search("we", str)

In [5]: x
Out[5]: <_sre.SRE_Match object; span=(4, 6), match='we'>

In [6]: x.start()
Out[6]: 4

In [7]: x.end()
Out[7]: 6
```

• • •

The split() Function

The **`split()`** function returns a list where the string has been split at each match. we can control the number of occurrences by specifying the **`maxsplit`** parameter

```
In [8]:
import re
str = "the weather is very pleasant."
x = re.split("\s", str)
y = re.split("\s", str, 1)

In [9]: x
Out[9]: ['the', 'weather', 'is', 'very', 'pleasant.']

In [10]: y
Out[10]: ['the', 'weather is very pleasant.']
```

• • •

The sub() Function

The **`sub()`** function replaces the matches with the text of your choice. You can control the number of replacements by specifying the **`count`** parameter.

```
In [11]:
import re
str = "the weather is very pleasant."
x = re.sub("\s", "_", str)
y = re.sub("\s", "_", str, 2)

In [12]: x
Out[12]: 'the_weather_is_very_pleasant.'

In [13]: y
Out[13]: 'the_weather_is very pleasant.'
```

• • •

Match Object

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

```
In [14]:
import re
str = "the weather is very pleasant."
x = re.search("weather", str)

In [15]: x.span()
Out[15]: (4, 11)

In [16]: x.string
Out[16]: 'the weather is very pleasant.'

In [17]: x.group()
Out[17]: 'weather'
```