P-8 Regular expression( <https://www.programiz.com/python-programming/regex>)

(<https://www.w3schools.com/python/python_regex.asp>)

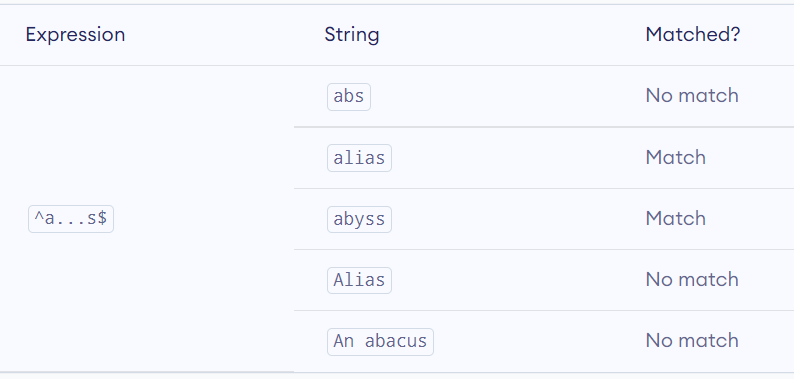
A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

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

^a...s$

The above code defines a RegEx pattern. The pattern is: any five letter string starting with a and ending with s.

A pattern defined using RegEx can be used to match against a string.



Python has a module named re to work with RegEx. Here's an example:

import re

pattern = '^a...s$'

test\_string = 'abyss'

result = re.match(pattern, test\_string)

if result:

print("Search successful.")

else:

print("Search unsuccessful.")

Here, we used re.match() function to search pattern within the test\_string. The method returns a match object if the search is successful. If not, it returns None.

**Specify Pattern Using RegEx**

To specify regular expressions, metacharacters are used. In the above example, ^ and $ are metacharacters.

**MetaCharacters**

Metacharacters are characters that are interpreted in a special way by a RegEx engine. Here's a list of metacharacters:

**[]** **.** **^** **$** **\*** **+** **?** **{}** **()** **\** **|**

**[] - Square brackets**

Square brackets specifies a set of characters you wish to match.

Ex. , [abc] will match if the string you are trying to match contains any of the a, b or c.

You can complement (invert) the character set by using caret ^ symbol at the start of a square-bracket.

[^abc] means any character except a or b or c.

[^0-9] means any non-digit character.

**^ - Caret**

The caret symbol ^ is used to check if a string starts with a certain character.

Ex. ^a

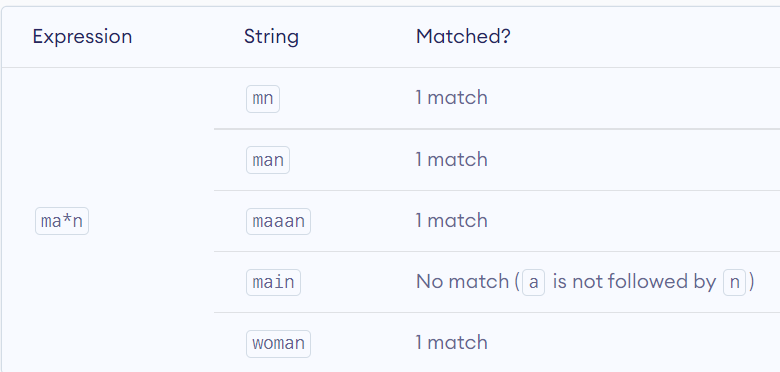
**$ - Dollar**

The dollar symbol $ is used to check if a string ends with a certain character.

E. a$

\* - **Star**

The star symbol \* matches zero or more occurrences of the pattern left to it.



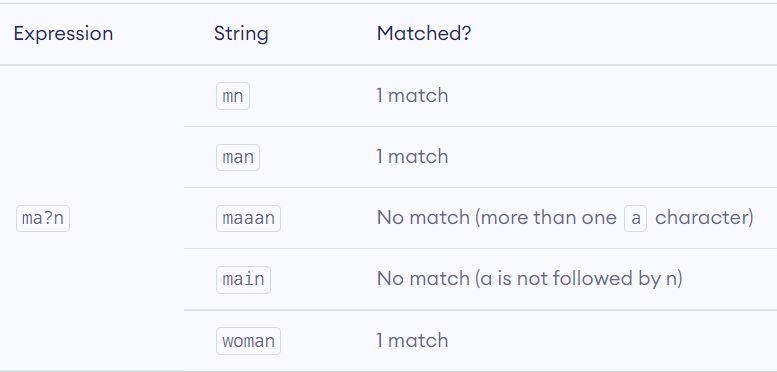
**+ - Plus**

The plus symbol + matches one or more occurrences of the pattern left to it.



? - **Question Mark**

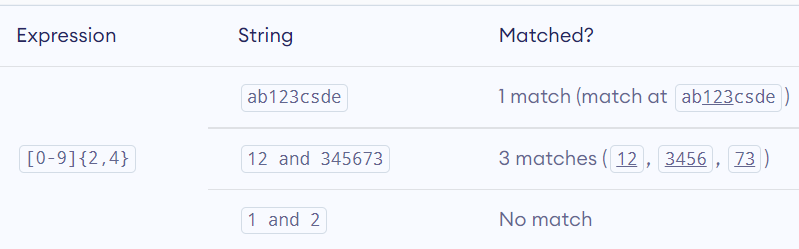
The question mark symbol ? matches zero or one occurrence of the pattern left to it.



{} – **Braces**

Consider this code: {n,m}. This means at least n, and at most m repetitions of the pattern left to it.





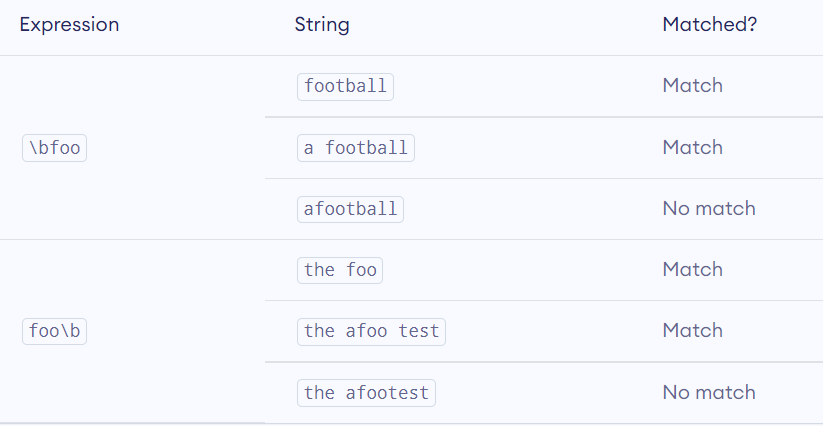
**| - Alternation**

Vertical bar | is used for alternation (or operator).

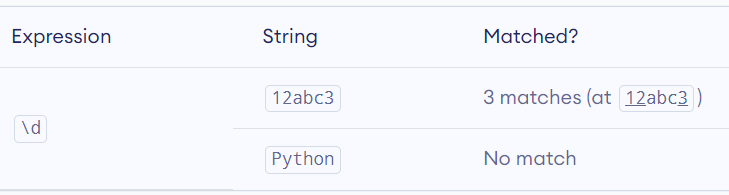
() – **Group**

Parentheses () is used to group sub-patterns. For example, (a|b|c)xz match any string that matches either a or b or c followed by xzim

**\b** – Matches if the specified characters are at the beginning or end of a word.



**\d** - Matches any decimal digit. Equivalent to [0-9]



**Python RegEx**

Python has a module named re to work with regular expressions. To use it, we need to import the module.

import re

The module defines several functions and constants to work with RegEx.

**re.findall()**

The re.findall() method returns a list of strings containing all matches.

**Example 1: re.findall()**

# Program to extract numbers from a string

import re

string = 'hello 12 hi 89. Howdy 34'

pattern = '\d+'

result = re.findall(pattern, string)

print(result)

# Output: ['12', '89', '34']

If the pattern is not found, re.findall() returns an empty list.

**re.split()**

The re.split method splits the string where there is a match and returns a list of strings where the splits have occurred.

**Example 2: re.split()**

import re

string = 'Twelve:12 Eighty nine:89.'

pattern = '\d+'

result = re.split(pattern, string)

print(result)

# Output: ['Twelve:', ' Eighty nine:', '.']

If the pattern is not found, re.split() returns a list containing the original string.

**re.sub()**

The syntax of re.sub() is:

re.sub(pattern, replace, string)

The method returns a string where matched occurrences are replaced with the content of replace variable.

**Example 3: re.sub()**

# Program to remove all whitespaces

import re

# multiline string

string = 'abc 12\

de 23 \n f45 6'

# matches all whitespace characters

pattern = '\s+'

# empty string

replace = ''

new\_string = re.sub(pattern, replace, string)

print(new\_string)

# Output: abc12de23f456

If the pattern is not found, re.sub() returns the original string.

**re.subn()**

The re.subn() is similar to re.sub() except it returns a tuple of 2 items containing the new string and the number of substitutions made.

**Example 4: re.subn()**

# Program to remove all whitespaces

import re

# multiline string

string = 'abc 12\

de 23 \n f45 6'

# matches all whitespace characters

pattern = '\s+'

# empty string

replace = ''

new\_string = re.subn(pattern, replace, string)

print(new\_string)

# Output: ('abc12de23f456', 4)

**re.search()**

The re.search() method takes two arguments: a pattern and a string. The method looks for the first location where the RegEx pattern produces a match with the string.

If the search is successful, re.search() returns a match object; if not, it returns None.

match = re.search(pattern, str)

**Example 5: re.search()**

import re

string = "Python is fun"

# check if 'Python' is at the beginning

match = re.search('\APython', string)

if match:

print("pattern found inside the string")

else:

print("pattern not found")

# Output: pattern found inside the string