RE (Regular Expression Pattern Matching)

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. A **Regular Expressions (RegEx)** is a special sequence of characters that uses a search pattern to find a string or set of strings. It can detect the presence or absence of a text by matching it with a particular pattern, and also can split a pattern into one or more sub-patterns.

MetaCharacters

To understand the RE analogy, MetaCharacters are useful, important, and will be used in functions of module re. Below is the list of metacharacters.

\ – Backslash

The backslash (\) makes sure that the character is not treated in a special way. This can be considered a way of escaping metacharacters

[] – Square Brackets

Square Brackets ([]) represent a character class consisting of a set of characters that we wish to match. For example, the character class [abc] will match any single a, b, or c.

We can also specify a range of characters using – inside the square brackets. For example,

- [0, 3] is sample as [0123]

^ – Caret

Caret (^) symbol matches the beginning of the string i.e. checks whether the string starts with the given character(s) or not. For example –

- ^g will check if the string starts with g such as geeks, globe, girl, g, etc.

$ – Dollar

Dollar($) symbol matches the end of the string i.e checks whether the string ends with the given character(s) or not. For example –

- s$ will check for the string that ends with a such as geeks, ends, s, etc.

. – Dot

Dot(.) symbol matches only a single character except for the newline character (\n). For example –

- a.b will check for the string that contains any character at the place of the dot such as acb, acbd, abbb, etc
- .. will check if the string contains at least 2 characters

| – Or

Or symbol works as the or operator meaning it checks whether the pattern before or after the or symbol is present in the string or not. For example –

- a|b will match any string that contains a or b such as acd, bcd, abcd, etc.

? – Question Mark

Question mark (?) checks if the string before the question mark in the regex occurs at least once or not at all. For example –

- ab?c will be matched for the string ac, acb, dabc but will not be matched for abbc because there are two b. Similarly, it will not be matched for abdc because b is not followed by c.

* – Star

Star (*) symbol matches zero or more occurrences of the regex preceding the * symbol. For example –

- ab*c will be matched for the string ac, abc, abbbc, dabc, etc. but will not be matched for abdc because b is not followed by c.

+ – Plus

Plus (+) symbol matches one or more occurrences of the regex preceding the + symbol. For example –

- ab+c will be matched for the string abc, abbc, dabc, but will not be matched for ac, abdc because there is no b in ac and b is not followed by c in abdc.

{m, n} – Braces

Braces match any repetitions preceding regex from m to n both inclusive. For example –

- a{2, 4} will be matched for the string aaab, baaaac, gaad, but will not be matched for strings like abc, bc because there is only one a or no a in both the cases.

(<regex>) – Group

Group symbol is used to group sub-patterns. For example –

- (a|b)cd will match for strings like acd, abcd, gacd, etc.

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

Import the `re` module:

```
import re
```

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)
```

Example 2

```
import re

 s = 'PythonorOther: A computer program portal for users'

 match = re.search(r'portal', s)

 print('Start Index:', match.start())

print('End Index:', match.end())
```

Output

```
Start Index: 34

End Index: 40
```

Special Sequences

A special sequence is a `\` followed by one of the characters in the list below, and has a special meaning:

Character Description:

\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

\d Returns a match where the string contains digits (numbers from 0-9)

\A Returns a match if the specified characters are at the beginning of the string

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

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)
```

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())
```

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)
```

Match Object

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

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 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)
```