Chapter - 6 : Functions

re.compiler

Compile a regular expression pattern into a regular expression object, which can be used for matching using its match(), search() and other methods.

Syntax

```
sPattern = re.compile(pattern)
result = sPattern.match(string)
```

re.MULTILINE

- When specified, the pattern character '^' matches at the beginning of the string and at the beginning of each line (immediately following each newline); and
- The pattern character '$' matches at the end of the string and at the end of each line (immediately preceding each newline).
- By default, '^' matches only at the beginning of the string, and '$' only at the end of the string and immediately before the newline (if any) at the end of the string.

Example

```
import re

x1="""UttarPradesh Lucknow Gorakhpur
HimachalPradesh Shimla
Bihar Patna Buxar Muzaffarpur"""

srcPtr=re.compile("^\w+")
s1 = srcPtr.findall(x1)

srcPtr=re.compile("^\w+", re.MULTILINE)
s2 = srcPtr.findall(x1)
print (s1)
print (s2)
```

Output

```
['UttarPradesh']
['UttarPradesh', 'HimachalPradesh', 'Bihar']
```

Using r prefix

- When r or R prefix is used before a regular expression, it means raw string. For example, '\n' is a new line whereas r'\n' means two characters: a backslash \ followed by n.
- Backlash \ is used to escape various characters including all metacharacters. However, using r prefix makes \ treat as a normal character.

Example

```
import re
x1="Hello\nWorld"
print (x1)
x2=r"Hello\nWorld"
print (x2)
```

Output

```
Hello
World

Hello\nWorld
```

Example

```
import re
string = '\n and \r are escape sequences.'
result = re.findall(r'[\n\r]', string)
print(result)
```

Output ['\n', '\r']

re.sub()

It will returns a string where matched occurrences are replaced with the content of replace variable.

Syntax re.sub(pattern, replace, string)

Example

```
# Program to remove all whitespaces
import re

# multiline string
string = Name Ajay Age 12\
Hindi 23 \n Eng lish 6'

# matches all whitespace characters
pattern = '\s+'

# empty string
replace = ''

new_string = re.sub(pattern, replace, string)
print(new_string)
```

Output NameAjayAge12Hindi23English6

re.split()

It will splits the string depending on match and returns a list of strings where the splits have occurred.

Syntax split(pattern , string , count of split[optional])

Example-1

```
import re

string = 'Twelve:12 Eighty nine:89'
pattern = '\d+'

result = re.split(pattern, string)
print(result)
```

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

Example-2

```
import re

string = 'Twelve:12 Eighty nine:89'
pattern = '\d+'

result = re.split(pattern, string,1)
print(result)
```

Output ['Twelve:', ' Eighty nine:89']