Regular Expressions (re module)

Regular expression is a special string or a sequence which defines the search pattern.
Regular expressions are used for searching pattern within string.

Applications of regular expression

1. Input Validations
    a. Password validation
    b. Username validation
    c. Email validation
    d. Mobile no validation
    e. URL validation
    f. Domain name validation
2. Parsers
3. Machine Learning (Chat Bots)
4. Search Engine

To work with regular expressions python provides a predefined module called “re”.

This module provides the following functions

1. match
2. search
3. findall
4. fullmatch
5. split
6. compile
7. sub

How to create regular expression pattern?

1. r’string’
2. using compile function

Example:

```
import re
```

```
str1="Best Language to Learn is Python"

# 1st method creating pattern
m=re.search(r'Learn',str1)
print(m)
m=re.search(r'Java',str1)
print(m)

#2nd method of creating pattern
p=re.compile(r'Python') # Creating pattern
m=p.search(str1)
print(m)
```

Output:

```
<re.Match object; span=(17, 22), match='Learn'>
None
<re.Match object; span=(26, 32), match='Python'>
```

re.match(pattern, string, flags=0)
If zero or more characters at the beginning of string match the regular expression pattern, return a corresponding Match. Return None if the string does not match the pattern;

Match object returned by successful matches and searches.

1. Slice notation → tuple (startindex,stopindex)
2. Search value

Example:

```
>>> str1="python"
>>> m=re.match(r'p',str1)
>>> print(m)
<re.Match object; span=(0, 1), match='p'>
>>> m=re.match(r'y',str1)
>>> print(m)
None
>>> m=re.match(r'P',str1)
>>> print(m)
None
>>> m=re.match(r'P',str1,re.IGNORECASE)
>>> print(m)
<re.Match object; span=(0, 1), match='p'>
```

re.search(pattern, string, flags=0)
Scan through string looking for the first location where the regular expression pattern produces a match, and return a corresponding Match. Return None if no position in the string matches the pattern;

```
>>> s1="python oracle mysql python java .net java"
>>> m=re.search(r'java',s1)
>>> print(m)
<re.Match object; span=(27, 31), match='java'>
>>> m=re.search(r'python',s1)
>>> print(m)
<re.Match object; span=(0, 6), match='python'>
>>> m=re.search(r'django',s1)
>>> print(m)
None
```

re.findall(pattern, string, flags=0)
Return all non-overlapping matches of pattern in string, as a list of strings or tuples. The string is scanned left-to-right, and matches are returned in the order found. Empty matches are included in the result.

```
>>> s1="python java .net java python oracle python"
>>> list1=re.findall(r'python',s1)
>>> print(list1)
['python', 'python', 'python']
>>> list2=re.findall(r'java',s1)
>>> print(list2)
['java', 'java']
>>> list3=re.findall(r'django',s1)
>>> print(list3)
[]
```

re.fullmatch(pattern, string, flags=0)
If the whole string matches the regular expression pattern, return a corresponding Match. Return None if the string does not match the pattern;

```
>>> str1="python"
>>> m=re.fullmatch(r'python',str1)
>>> print(m)
<re.Match object; span=(0, 6), match='python'>
>>> str1="python language"
>>> m=re.fullmatch(r'python',str1)
>>> print(m)
None
```

Pattern is created with special characters.

.
(Dot.) In the default mode, this matches any character except a newline. If the DOTALL flag has been specified, this matches any character including a newline.

Example:

```
>>> str1="python language"
>>> list1=re.findall(r'g',str1)
>>> print(list1)
['g', 'g']
>>> list2=re.findall(r'.',str1)
>>> print(list2)
['p', 'y', 't', 'h', 'o', 'n', ' ', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e']
```

Example:

```
import re
namesList=["naresh","ramesh","kishore","raman","kiran"]
for name in namesList:
    m=re.match(r'.a',name)
    if m!=None:
        print(name)
```

Output:

```
naresh
ramesh
raman
```

Example:

```
>>> s1="python is high level programming language"
>>> list1=re.findall(r'..',s1)
>>> print(list1)
['py', 'th', 'on', ' i', 's ', 'hi', 'gh', ' l', 'ev', 'el', ' p', 'ro', 'gr', 'am', 'mi', 'ng', ' l', 'an', 
'gu', 'ag']
>>> list2=re.findall(r'p..',s1)
>>> print(list2)
['pyt', 'pro']
```

^
(Caret.) Matches the start of the string, and in MULTILINE mode also matches immediately after each newline.

$
Matches the end of the string or just before the newline at the end of the string, and in MULTILINE mode also matches before a newline.

Example:
```
import re
namesList=["naresh","ramesh","kishore","raman","kiran"]
for name in namesList:
    m=re.search(r'n$',name)
    if m!=None:
        print(name)
```

Output:
```
raman
kiran
```

Example:
```
import re
namesList=["naresh","ramesh","kishore","raman","kiran"]
for name in namesList:
    m=re.search(r'^r',name)
    if m!=None:
        print(name)
```

Output:
```
ramesh
raman
```

*

Causes the resulting RE to match 0 or more repetitions of the preceding RE, as many repetitions as are possible. ab* will match ‘a’, ‘ab’, or ‘a’ followed by any number of ‘b’s.

Example:

```
import re
namesList=["naresh","ramesh","kishore","raman","kiran"]
for name in namesList:
    m=re.search(r'^r.*n$',name)
    if m!=None:
        print(name)
```

Output:

```
raman
```

Example:

```
>>> str1="a ab abb acb abbbbb ax"
>>> list1=re.findall(r'ab*',str1)
>>> print(list1)
['a', 'ab', 'abb', 'a', 'abbbbb', 'a']
```

+

Causes the resulting RE to match 1 or more repetitions of the preceding RE. ab+ will match ‘a’ followed by any non-zero number of ‘b’s; it will not match just ‘a’.