Regular Expressions (re module)

What is regular expression?
A regular expression is special string which define search pattern or match pattern.

A Regular Expression or RegEx is a special sequence of characters that uses a search pattern to find a string or set of strings.

Applications of regular expressions

1. Input Validations
    a. Mobile number validation
    b. Email validation
    c. Password validation
    d. IFSC code validation
    e. Vehicle number validation
    f. Username validation
2. Data Extraction and Parsing
3. Chatbots
4. Search Engines
5. Database Applications
6. Network Applications
7. Command line operations

To work with regular expression python provides a predefined module (re module).
This module provides set of functions to work with regular expressions.

1. match()
2. search()
3. findall()

4. fullmatch()
5. split()
6. compile()

How to defined regular expression pattern?

Regular expression is special string, which define search pattern. This string is created using special sequence of characters.

Regular expression is created in two ways

1. r'sequence of character'
2. compile()

```
>>> import re
>>> s1=r'nit'
>>> print(s1,type(s1))
nit <class 'str'>
>>> s2=re.compile('nit')
>>> print(s2,type(s2))
re.compile('nit') <class 're.Pattern'>
```

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.

```
# Finding pattern at the begining of string

import re

s1=r'n'
```

```
p1=re.compile('n')
str1="naresh"

m1=re.match(s1,str1)
print(m1)

m2=p1.match(str1)
print(m2)

str2="suresh"
m3=re.match(s1,str2)
print(m3)

str3="Naresh"
m4=re.match(s1,str3)
print(m4)
m5=re.match(s1,str3,re.I)
print(m5)
```

Output

```
<re.Match object; span=(0, 1), match='n'>
<re.Match object; span=(0, 1), match='n'>
None
None
<re.Match object; span=(0, 1), match='N'>
```

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;

```
import re

m=re.search(r'java','python java c++ java python java')
print(m)
m=re.search(r'java','python c++ oracle')
print(m)
```

Output

```
<re.Match object; span=(7, 11), match='java'>
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.

```
import re

str1="python java python java java oracle"
list1=re.findall(r'java',str1)
print(list1)
list2=re.findall(r'mysql',str1)
print(list2)
```

Output

```
['java', 'java', 'java']
[]
```

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;

```
import re

str1="python"
m=re.fullmatch(r'python',str1)
print(m)
m=re.fullmatch(r'python','python python')
print(m)
```

Output

```
<re.Match object; span=(0, 6), match='python'>
None
```

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.

```
import re

str1="python 3.13 1a 1b 1c 1d \n"

list1=re.findall(r'3',str1)
print(list1)
list2=re.findall(r'.',str1)
print(list2)
list3=re.findall(r'13',str1)
print(list3)
list4=re.findall(r'..',str1)
print(list4)
```

```
list5=re.findall(r'1.',str1)
print(list5)
list6=re.findall(r'.3',str1)
print(list6)
list7=re.findall(r'.',str1,re.DOTALL)
print(list7)
```

Output

```
['3', '3']
['p', 'y', 't', 'h', 'o', 'n', ' ', '3', '.', '1', '3', ' ', '1', 'a', ' ', '1', 'b', ' ', '1', 'c', ' ', '1', 'd', ' ']
['13']
['py', 'th', 'on', ' 3', '.1', '3 ', '1a', ' 1', 'b ', '1c', ' 1', 'd ']
['13', '1a', '1b', '1c', '1d']
[' 3', '13']
['p', 'y', 't', 'h', 'o', 'n', ' ', '3', '.', '1', '3', ' ', '1', 'a', ' ', '1', 'b', ' ', '1', 'c', ' ', '1', 'd', ' ', '\n']
```

^

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

Example:

```
import re

str1='python'
```

```
m=re.search(r'^p',str1)
print(m)
str2="java"
m=re.search(r'^a',str2)
print(m)
```

Output

```
<re.Match object; span=(0, 1), match='p'>
None
```

$

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.

```
import re

str1='python'
m=re.search(r'n$',str1)
print(m)
str2="java"
m=re.search(r'a$',str2)
print(m)
```

Output

```
<re.Match object; span=(5, 6), match='n'>
<re.Match object; span=(3, 4), match='a'>
```

*

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.