Regular Expressions

Regular expressions are also called as REs, or regexes, or regex.

Regular expressions are available in re module.

Using Regular expressions we can specify the rules for the set of possible strings that you want to match; this set might contain **English sentences**, or **e-mail addresses**, or **TeX commands**, or **anything you like**.

Various methods of Regular Expressions?

1. re.match()
2. re.search()
3. re.findall()
4. re.split()
5. re.sub()
6. re.compile()

re.match(*pattern*, *string*):

This method finds match if it occurs at start of the string.

Example:

```
>>> import re
>>> st = "This is Naveen"
>>> result = re.match(r"Naveen",st)
>>> print(result)
Output : None
```

Example

```
>>> import re
>>> st = "Naveen is here"
>>> result = re.match(r"Naveen",st)
>>> print(result)
output : <_sre.SRE_Match object; span=(0, 6), match='Naveen'>
```

Note: To print the matching string we'll use method group() (It helps to return the matching string).

Example

```
>>> import re
>>> st = "Naveen is here"
>>> result = re.match(r"Naveen",st)
>>> print(result.group(0))
```

```
Output : Naveen
```

There are methods like start() and end() to know the start and end position of matching pattern in the string.

Example

```
>>> import re
>>> st = "Naveen is from Sathya Naveen is teaching Python"
>>> result = re.match(r"Naveen",st)
>>> print(result.start())
Output: 0
>>> print(result.end())
Output: 6
```

re.search(*pattern*, *string*):

It is similar to match() but it doesn't restrict us to find matches at the beginning of the string only.

Example:

```
>>> import re
>>> st = "This is Naveen"
>>> result = re.search(r"Naveen",st)
>>> print(result)
Output : <_sre.SRE_Match object; span=(8, 14), match='Naveen'>
>>> print(result.group(0))
```

```
Output: Naveen
```

re.findall (*pattern, string*):

It helps to get a list of all matching patterns. It has no constraints of searching from start or end.

Example:

```
>>> import re

>>> st = "Naveen is from Sathya Naveen is teaching Python"

>>> result = re.findall(r"Naveen",st)

>>> print(result)

Output: ['Naveen', 'Naveen']
```

re.split(*pattern, string*, [*maxsplit=0*]):

This method helps to split *string* by the occurrences of given *pattern*.

Example

```
>>> import re

>>> st = "kumar"

>>> result = re.split(r"m",st)

>>> print(result)

Output : ['ku', 'ar']
```

Example

```
>>> import re

>>> st = "Programming"

>>> result = re.split(r"m",st)

>>> print(result)

Output: ['Progra', '', 'ing']
```

Method split() has another argument "maxsplit". It has default value of zero. In this case it does the maximum splits that can be done, but if we give value to maxsplit, it will split the string.

Example:

```
>>> import re

>>> st = "Programming"

>>> result = re.split(r"m",st,maxsplit=1)

>>> print(result)

Output: ['Progra', 'ming']
```

re.sub(*pattern, repl, string*):

It helps to search a pattern and replace with a new sub string. If the pattern is not found, *string* is returned unchanged.

Example

```
>>> import re
>>> st = "Current rocking programming is Java"
>>> result = re.sub(r"Java","Python",st)
>>> print(result)
Output: Current rocking programming is Python
```

What are the most commonly used operators?

Get the Complete ref @ https://docs.python.org/2/library/re.html

Operators	Description
.	Matches with any single character except newline '\n'.
?	match 0 or 1 occurrence of the pattern to its left
+	1 or more occurrences of the pattern to its left
*	0 or more occurrences of the pattern to its left
\w	Matches with a alphanumeric character whereas \W (upper case W) Matches non alphanumeric character.
\d	Matches with digits [0-9] and /D (upper case D) matches with non-digits.
\s	Matches with a single white space character (space, newline, return, tab, form) and \S (upper case S) matches any non-white space character.

\b	boundary between word and non-word and /B is opposite of /b
[..]	Matches any single character in a square bracket and [^..] matches any single character not in square bracket
\	It is used for special meaning characters like \. to match a period or \+ for plus sign.
^ and $	^ and $ match the start or end of the string respectively
{n,m}	Matches at least n and at most m occurrences of preceding expression if we write it as {,m} then it will return at least any minimum occurrence to max m preceding expression.
a| b	Matches either a or b
( )	Groups regular expressions and returns matched text
\t, \n, \r	Matches tab, newline, return

Problem 1: Return the first word of a given string

Solution-1 Extract each character **(**using “**\w**“**)**

```
>>> import re

>>> s1 = "I am learning python with Naveen"

>>> result = re.findall(r".",s1)

>>> print(result)
```

Output: ['I', ' ', 'a', 'm', ' ', 'l', 'e', 'a', 'r', 'n', 'i', 'n', 'g', ' ', 'p', 'y', 't', 'h', 'o', 'n', ' ', 'w', 'i', 't', 'h', ' ', 'N', 'a', 'v', 'e', 'e', 'n']

Above, space is also extracted, now to avoid it use "\w" instead of ".".

```
>>> import re
>>> s1 = "I am learning python with Naveen"
>>> result = re.findall(r"\w",s1)
>>> print(result)
```

Output: ['I', 'a', 'm', 'l', 'e', 'a', 'r', 'n', 'i', 'n', 'g', 'p', 'y', 't', 'h', 'o', 'n', 'w', 'i', 't', 'h', 'N', 'a', 'v', 'e', 'e', 'n']

Solution-2 Extract each word **(**using "*****" or "**+**"**)**

```
>>> import re
>>> s1 = "I am learning python with Naveen"
>>> result = re.findall(r"\w*",s1)
>>> print(result)
```

output: ['I', '', 'am', '', 'learning', '', 'python', '', 'with', '', 'Naveen', '']

Again, it is returning space as a word because "*" returns zero or more matches of pattern to its left. Now to remove spaces we will go with "+".

```
>>> import re
>>> s1 = "I am learning python with Naveen"
>>> result = re.findall(r"\w+",s1)
>>> print(result)
```

output: ['I', 'am', 'learning', 'python', 'with', 'Naveen']

Solution-3 Extract each word **(**using “**^**“**)**

```
>>> import re
>>> s1 = "I am learning python with Naveen"
>>> result = re.findall(r"^\w+",s1)
>>> print(result)
```

Output: ['I']

If we will use “$” instead of “^”, it will return the word from the end of the string. Let’s look at it.

```
>>> import re
>>> s1 = "I am learning python with Naveen"
>>> result = re.findall(r"\w+$",s1)
>>> print(result)       Output : ['Naveen']
```

Problem 2: Return the first two character of each word

Solution-1 Extract consecutive two characters of each word, excluding spaces (using “**\w**“**)**

```
>>> import re
>>> s1 = "I am learning python with Naveen"
>>> result = re.findall(r"\w\w",s1)
>>> print(result)
```

Output: ['am', 'le', 'ar', 'ni', 'ng', 'py', 'th', 'on', 'wi', 'th', 'Na', 've', 'en']

Solution-2 **Extract** consecutive two characters those available at start of word boundary (using "**\b**"**)**

```
>>> import re
>>> s1 = "I am learning python with Naveen"
>>> result = re.findall(r"\b\w.",s1)
>>> print(result)
```

Output: ['I ', 'am', 'le', 'py', 'wi', 'Na']

Problem 3: Return the domain type of given email-ids

Solution-1 **Extract** all characters after "@"

```
>>> import re
>>> s1 = "naveen@gmail.com mailmenaveenkumar@yahoo.com pythonnaveen@co.in"
>>> result = re.findall(r"@\w+",s1)
>>> print(result)
```

Output: ['@gmail', '@yahoo', '@co']

Above, you can see that ".com", ".in" part is not extracted. To add it, we will go with below code.

```
>>> import re
>>> s1 = "naveen@gmail.com mailmenaveenkumar@yahoo.com pythonnaveen@co.in"
>>> result = re.findall(r"@\w+.\w+",s1)
>>> print(result)

['@gmail.com', '@yahoo.com', '@co.in']
```

Solution – 2 Extract only domain name using “( )”

```
>>> import re

>>> s1 = "naveen@gmail.com mailmenaveenkumar@yahoo.com
pythonnaveen@co.in"

>>> result = re.findall(r"@\w+.(\w+)",s1)

>>> print(result)

Output: ['com', 'com', 'in']
```

Problem 4: Return date from given string

Here we will use “**\d**” to extract digit.

```
>>> import re

>>> s1 = "Hi 007 this is 001 from python"

>>> result = re.findall(r"\d",s1)

>>> print(result)

Output: ['0', '0', '7', '0', '0', '1']
```

Problem 5: Return all words of a string those starts with vowel

Solution-1 Return each words

```
>>> import re

>>> s1 = "I am learning python with Naveen"

>>> result = re.findall(r"\w+",s1)
>>> print(result)
```

Output: ['I', 'am', 'learning', 'python', 'with', 'Naveen']

Solution-2 Return words starts with alphabets (using [])

```
>>> import re

>>> s1 = "I am learning python with Naveen"

>>>result = re.findall(r"[aeiouAEIOU]\w+",s1)

>>> print(result)

['am', 'earning', 'on', 'ith', 'aveen']
```

Example to validate Name

```
import re
user_name = input("Name Please  :")
result = re.match("^[A-Za-z]*$", user_name)
if result == None:
   print("Invalid Name")
else:
   print("Welcome Mr/Miss : ",user_name)
```

Match a string to a numberic sequence of exactly five

```
import re

input = input("Enter an input string:")

m = re.match('\d{5}\Z',input)

if m:

        print("True")
```

```
else:
      print("False")
```

Email validation regex

```
import re
input = input("Enter an input string:")
m = re.match('[^@]+@[^@]+\.[^@]+',input)
if m:
   print("True")
else:
   print("False")
```

To search if an e-mail address is in a string:

```
import re
input = "Contact me by mailmenaveenkumar@gmail.com or at the
office."
m = re.search('[^@]+@[^@]+\.[^@]+',input)
if m:    print("String found.")
else:    print("Nothing found.")
```