What are the most commonly used operators?

Regular expressions can specify patterns, not just fixed characters. Here are the most commonly used operators that helps to generate an expression to represent required characters in a string or file. It is commonly used in web scrapping and  text mining to extract required information.

|  |  |
| --- | --- |
| **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 |

For more details on  meta characters “(“, “)”,”|” and others details , you can refer this link (<https://docs.python.org/2/library/re.html>).

Now, let’s understand the pattern operators by looking at the below examples.

Some Examples of Regular Expressions

Problem 1: Return the first word of a given string

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

Code

import re

result=re.findall(r'.','AV is largest Analytics community of India')

print result

**Output:**

['A', 'V', ' ', 'i', 's', ' ', 'l', 'a', 'r', 'g', 'e', 's', 't', ' ', 'A', 'n', 'a', 'l', 'y', 't', 'i', 'c', 's', ' ', 'c', 'o', 'm', 'm', 'u', 'n', 'i', 't', 'y', ' ', 'o', 'f', ' ', 'I', 'n', 'd', 'i', 'a']

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

Code

result=re.findall(r'\w','AV is largest Analytics community of India')

print result

**Output:**

['A', 'V', 'i', 's', 'l', 'a', 'r', 'g', 'e', 's', 't', 'A', 'n', 'a', 'l', 'y', 't', 'i', 'c', 's', 'c', 'o', 'm', 'm', 'u', 'n', 'i', 't', 'y', 'o', 'f', 'I', 'n', 'd', 'i', 'a']

**Solution-2**Extract each word**(**using “**\***” or “**+**“**)**

Code

result=re.findall(r'\w\*','AV is largest Analytics community of India')

print result

**Output:**

['AV', '', 'is', '', 'largest', '', 'Analytics', '', 'community', '', 'of', '', 'India', '']

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 “**+**“.

Code

result=re.findall(r'\w+','AV is largest Analytics community of India')

print result

**Output:**

['AV', 'is', 'largest', 'Analytics', 'community', 'of', 'India']

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

Code

result=re.findall(r'^\w+','AV is largest Analytics community of India')

print result

**Output:**

['AV']

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

Code

result=re.findall(r'\w+$','AV is largest Analytics community of India')

print result

**Output:**

[‘India’]

Problem 2: Return the first two character of each word

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

Code

result=re.findall(r'\w\w','AV is largest Analytics community of India')

print result

Output:

['AV', 'is', 'la', 'rg', 'es', 'An', 'al', 'yt', 'ic', 'co', 'mm', 'un', 'it', 'of', 'In', 'di']

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

Code

result=re.findall(r'\b\w.','AV is largest Analytics community of India')

print result

**Output:**

['AV', 'is', 'la', 'An', 'co', 'of', 'In']

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

To explain it in simple manner, I will again go with a stepwise approach:

**Solution-1**Extract all characters after “@”

Code

result=re.findall(r'@\w+','abc.test@gmail.com, xyz@test.in, test.first@analyticsvidhya.com, first.test@rest.biz')

print result

**Output:** ['@gmail', '@test', '@analyticsvidhya', '@rest']

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

result=re.findall(r'@\w+.\w+','abc.test@gmail.com, xyz@test.in, test.first@analyticsvidhya.com, first.test@rest.biz')

print result

**Output:**

['@gmail.com', '@test.in', '@analyticsvidhya.com', '@rest.biz']

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

Code

result=re.findall(r'@\w+.(\w+)','abc.test@gmail.com, xyz@test.in, test.first@analyticsvidhya.com, first.test@rest.biz')

print result

**Output:**

['com', 'in', 'com', 'biz']

Problem 4: Return date from given string

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

**Solution:**

Code

result=re.findall(r'\d{2}-\d{2}-\d{4}','Amit 34-3456 12-05-2007, XYZ 56-4532 11-11-2011, ABC 67-8945 12-01-2009')

print result

**Output:**

['12-05-2007', '11-11-2011', '12-01-2009']

If you want to extract only year again parenthesis “( )” will help you.

Code

result=re.findall(r'\d{2}-\d{2}-(\d{4})','Amit 34-3456 12-05-2007, XYZ 56-4532 11-11-2011, ABC 67-8945 12-01-2009')

print result

**Output:**

['2007', '2011', '2009']

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

**Solution-1**Return each words

Code

result=re.findall(r'\w+','AV is largest Analytics community of India')

print result

**Output:**

['AV', 'is', 'largest', 'Analytics', 'community', 'of', 'India']

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

Code

result=re.findall(r'[aeiouAEIOU]\w+','AV is largest Analytics community of India')

print result

**Output:**

['AV', 'is', 'argest', 'Analytics', 'ommunity', 'of', 'India']

Above you can see that it has returned “argest” and “ommunity” from the mid of words. To drop these two, we need to use “\b” for word boundary.

**Solution- 3**

Code

result=re.findall(r'\b[aeiouAEIOU]\w+','AV is largest Analytics community of India')

print result

**Output:**

['AV', 'is', 'Analytics', 'of', 'India']

In similar ways, we can extract words those starts with constant using “^” within square bracket.

Code

result=re.findall(r'\b[^aeiouAEIOU]\w+','AV is largest Analytics community of India')

print result

**Output:**

[' is', ' largest', ' Analytics', ' community', ' of', ' India']

Above you can see that it has returned words starting with space. To drop it from output, include space in square bracket[].

Code

result=re.findall(r'\b[^aeiouAEIOU ]\w+','AV is largest Analytics community of India')

print result

**Output:**

['largest', 'community']

Problem 6: Validate a phone number (phone number must be of 10 digits and starts with 8 or 9)

We have a list phone numbers in list “li” and here we will validate phone numbers using regular

**Solution**

Code

import re

li=['9999999999','999999-999','99999x9999']

for val in li:

if re.match(r'[8-9]{1}[0-9]{9}',val) and len(val) == 10:

print 'yes'

else:

print 'no'

**Output:**

yes

no

no

Problem 7: Split a string with multiple delimiters

**Solution**

Code

import re

line = 'asdf fjdk;afed,fjek,asdf,foo' # String has multiple delimiters (";",","," ").

result= re.split(r'[;,\s]', line)

print result

**Output:**

['asdf', 'fjdk', 'afed', 'fjek', 'asdf', 'foo']

We can also use method **re.sub()**to replace these multiple delimiters with one as space ” “.

Code

import re

line = 'asdf fjdk;afed,fjek,asdf,foo'

result= re.sub(r'[;,\s]',' ', line)

print result

**Output:**

asdf fjdk afed fjek asdf foo

Problem 8: Retrieve Information from HTML file

I want to extract information from a HTML file (see below sample data). Here we need to extract information available between <td> and </td> except the first numerical index. I have assumed here that below html code is stored in a string **str**.

Sample HTML file (str)

<tr align="center"><td>1</td> <td>Noah</td> <td>Emma</td></tr>

<tr align="center"><td>2</td> <td>Liam</td> <td>Olivia</td></tr>

<tr align="center"><td>3</td> <td>Mason</td> <td>Sophia</td></tr>

<tr align="center"><td>4</td> <td>Jacob</td> <td>Isabella</td></tr>

<tr align="center"><td>5</td> <td>William</td> <td>Ava</td></tr>

<tr align="center"><td>6</td> <td>Ethan</td> <td>Mia</td></tr>

<tr align="center"><td>7</td> <td HTML>Michael</td> <td>Emily</td></tr>

**Solution:**

Code

result=re.findall(r'<td>\w+</td>\s<td>(\w+)</td>\s<td>(\w+)</td>',str)

print result

**Output:**

[('Noah', 'Emma'), ('Liam', 'Olivia'), ('Mason', 'Sophia'), ('Jacob', 'Isabella'), ('William', 'Ava'), ('Ethan', 'Mia'), ('Michael', 'Emily')]

You can read html file using library urllib2 (see below code).

Code

import urllib2

response = urllib2.urlopen('')

html = response.read()