**Regular Expressions**

Regular Expressions:

A regular expression is a string that contains special symbols and characters to find and extract the information needed by us

from the given data.

A regular expression helps us to search information, match, find and split information as per our requirements.

A regular expression is also called simple regex.

Regular expression are available not only in python but alos in many languages like java,perl,AWK etc..

​

Python provides re module that stands for regular expression. This module contains methods like

compile()

search()

match()

findall()

split() etc..

​

Regular expression are used to perform the following important operations

1. matching strings

2. seaching for strings

3.Finding all strings

4.Splitting a string into pieces

5.Replacing strings

​

which is used to find the information in the available data.

​

for example if we create regular expression then it may look like this

reg = r'm\w\w'

​

where r is used to represent it is raw string so as to use the escape sequences also.

​

​

str1 **=**'Asha\nTalari'

print(str1)

str2 **=** r'Asha\nTalari'

print(str2)

​

#First step is to create regular expression or a pattern to search in the string.

​

reg = r'm\w\w'

here r - represent raw string

here m - indiates that to search for the character starting with character 'm'

here w - indicates that to search for any alpha or numeric character and since 2 \w is provided then it means 2 alpha numeric

        character

  so the meaning of this expression is any word starting from m and followed by 2 alpha numeric characters

once the expression is created then it needs to be compiled and then use this expression for searching .

the result of the search function will be stored in result object which can be printed using group method

​

​

​

​

​

*#First compile the regular expression , once compiled it can be used for many strings also.*

**import** re

prog **=** re.compile(r'm\w\w')

​

str **=** 'got,mat cat,mat mat bat'

result **=** prog.search(str)

print(result)

print(result.group())

​

​

<re.Match object; span=(4, 7), match='mat'>

mat

str **=** 'Operating system format mat'

result **=** prog.search(str)

print(result.group())

mat

*# #here instead of compiling first and then using it , we can directly use in the search function*

*# #syntax of search function is*

*# result = re.search('expression','string')*

*# this is equivalent to*

*# prog = re.compile('expression')*

*# result = prog.search('string')*

**import** re

str **=** 'got man sun mop run'

result **=** re.search(r'm\w\w',str)

print(result.group())

man

*#here even though there are 2 strings which starts with m , only first occurance is only displayed .*

*#if we wanted all the occurances than we can us findall() method , here the result will be returned as list and hence can be iterated*

*#using loops*

​

**import** re

str **=** 'man sun mop run'

result **=** re.findall(r'm\w\w',str)

print(result)

​

**for** e **in** result:

print(e)

​

['man', 'mop']

man

mop

*#if we wanted to search the string whose starting character is provided in the search expression then we need to use match method*

*#the match method will return resultant string only if it is presnt else None will be returned*

*#since None can not be used on group method , checking with if condiction will be good*

**import** re

str **=** 'man got man sun mop run'

result **=** re.match(r'm\w\w',str)

**if** result **is** **not** **None**:

print(result.group())

**else**:

print(result)

man

*#Split method -that splits the given string into pieces according to the regular expression and returns the pieces as elements*

*# of a list*

*# re.split(r'\W+',str)*

​

*# here \W - which is reverse of \w , if \w is for alpha nummeric characters(A-Z,a-z,0-9) then \W is for non alpha numeric characters*

*# so this means to split the string at non alpha numeric character , + idicates to match 1 or more occurances indicated by W*

**import** re

str **=** 'This : is the ; "core" python\'s book'

result **=** re.split(r'\W+',str)

print(result)

​

**for** e **in** result:

print(e)

['This', 'is', 'the', 'core', 'python', 's', 'book']

This

is

the

core

python

s

book

*#if we wanted to find some string and replace with new string then use the method sub*

*# sub(regular expression, new string,string)*

**import** re

str **=** 'kumbamela will be conducted at Ahmedabad in india'

result **=** re.sub(r'Ahmedabad','Allahadabad',str)

print(result)

kumbamela will be conducted at Allahadabad in india

#Sequence characters used in regular expression along with their meanings

​

\d --- Represnts any digit (0 to 9)

\D --- represnts any non digit

\s --- represnts white space e.g.\t\n\r\f\v

\S --- represnts non white space character

\w --- represnts any alphanumeric (A to Z, a to z, 0 to 9)

\W --- reprents any non alphanumeric

\b --- reprents a space around words

\A --- matches only at start of the string

\Z --- matches only at end of the string

​

​

Each of these sequence represents a single character matched in the string for e.g

\w is for single any alpha numeric character, suppose if we wanted 0 or more repititions then

we need to use I[\w]\*. I , Iam

​

​

​

*#r'a[\w]\* - here a represents it should start with a and [\w]\* represents repitiations of any alphanumeric chacters*

**import** re

str **=** 'an apple a day keeps the doctor away'

*# result = re.findall(r'a\w\w',str)*

result **=** re.findall(r'a[\w]\*',str)

​

print(result)

​

​

['an', 'apple', 'a', 'ay', 'away']

*#here even ay is getting displayed , which is part of the string of the word and we do not want this then we can provide*

*#the expression which will be distinguised based on spaces using \b - which is space around words*

​

result **=** re.findall(r'\ba[\w]\*\b',str)

print(result)

['an', 'apple', 'a', 'away']

*#if we wanted to get the words starting from numeric digits then we can use \d*

**import** re

str **=** 'The meeting will be conducted on 1st and 21st of every month'

result **=** re.findall(r'\d[\w]\*',str)

print(result)

['1st', '21st']

*#if we wanted to retrive the words with some specified length then we can provide the length*

*#we can even surround by spaces*

str **=** 'an aaaaa apple a day keeps the doctor away'

result **=** re.findall(r'\b\w{5}\b',str)

print(result)

['aaaaa', 'apple', 'keeps']

*#if we use search instead of findall then only first occurance will be displayed*

str **=** 'an aaaaa apple a day keeps the doctor away'

result **=** re.search(r'\b\w{5}\b',str)

print(result.group())

aaaaa

*#to retireve all the words that have atleast 4 characters*

**import** re

str **=** 'one two three four five six seven 8,9,10'

result **=** re.findall(r'\b\w{4,}\b',str)

print(result)

['three', 'four', 'five', 'seven']

*#to retirive all words with 3 or 4 or 5 characters length*

​

**import** re

str **=** 'one two three four five six seven 8,9,10'

result **=** re.findall(r'\b\w{3,5}\b',str)

print(result)

['one', 'two', 'three', 'four', 'five', 'six', 'seven']

*#if we want to retrive single digit from the string*

**import** re

str **=** 'one two three four five six seven 8,9,10'

result **=** re.findall(r'\b\d\b',str)

print(result)

['8', '9']

**import** re

str **=** 'one two three four five six seven 8,9,10'

result **=** re.findall(r'\b\d\d\b',str) *#double digit*

print(result)

['10']

*#if we want to retrieve the last word of the string starting with t*

​

**import** re

str **=** 'one two three one two three'

result **=** re.findall(r'\Ao[\w]\*',str)

print(result)

​

result1 **=** re.findall(r't[\w]\*\Z',str) *#from the end of the string*

print(result1)

['one']

['three']

Repetition Syntax

There are five ways to express repetition in a pattern:

​

A pattern followed by the meta-character \* is repeated zero or more times.

Replace the \* with + and the pattern must appear at least once.

Using ? means the pattern appears zero or one time.

For a specific number of occurrences, use {m} after the pattern, where m is replaced with the number of times the pattern should repeat.

Use {m,n} where m is the minimum number of repetitions and n is the maximum. Leaving out n {m,} means the value appears at least m times, with no maximum.

test\_phrase **=** 'sdsd..sssddd...sdddsddd...dsds...dsssss...sdddd'

​

test\_patterns **=** [ 'sd\*', *# s followed by zero or more d's*

'sd+', *# s followed by one or more d's*

'sd?', *# s followed by zero or one d's*

'sd{3}', *# s followed by three d's*

'sd{2,3}', *# s followed by two to three d's*

]

​

​

**def** multi\_re\_find(patterns,phrase):

'''

Takes in a list of regex patterns

Prints a list of all matches

'''

**for** pattern **in** patterns:

print('Searching the phrase using the re check: %r' **%**(pattern))

print(re.findall(pattern,phrase))

print('\n')

multi\_re\_find(test\_patterns,test\_phrase)

Searching the phrase using the re check: 'sd\*'

['sd', 'sd', 's', 's', 'sddd', 'sddd', 'sddd', 'sd', 's', 's', 's', 's', 's', 's', 'sdddd']

Searching the phrase using the re check: 'sd+'

['sd', 'sd', 'sddd', 'sddd', 'sddd', 'sd', 'sdddd']

Searching the phrase using the re check: 'sd?'

['sd', 'sd', 's', 's', 'sd', 'sd', 'sd', 'sd', 's', 's', 's', 's', 's', 's', 'sd']

Searching the phrase using the re check: 'sd{3}'

['sddd', 'sddd', 'sddd', 'sddd']

Searching the phrase using the re check: 'sd{2,3}'

['sddd', 'sddd', 'sddd', 'sddd']

Character Sets

Character sets are used when you wish to match any one of a group of characters at a point in the input. Brackets are used to construct character set inputs. For example: the input [ab] searches for occurrences of either a or b. Let's see some examples:

test\_phrase **=** 'sdsd..sssddd...sdddsddd...dsds...dsssss...sdddd'

​

test\_patterns **=** ['[sd]', *# either s or d*

's[sd]+'] *# s followed by one or more s or d*

​

multi\_re\_find(test\_patterns,test\_phrase)

Searching the phrase using the re check: '[sd]'

['s', 'd', 's', 'd', 's', 's', 's', 'd', 'd', 'd', 's', 'd', 'd', 'd', 's', 'd', 'd', 'd', 'd', 's', 'd', 's', 'd', 's', 's', 's', 's', 's', 's', 'd', 'd', 'd', 'd']

Searching the phrase using the re check: 's[sd]+'

['sdsd', 'sssddd', 'sdddsddd', 'sds', 'sssss', 'sdddd']

Exclusion

We can use ^ to exclude terms by incorporating it into the bracket syntax notation. For example: [^...] will match any single character not in the brackets. Let's see some examples:

test\_phrase **=** 'This is a string! But it has punctuation. How can we remove it?'

Use [^!.? ] to check for matches that are not a !,.,?, or space. Add a + to check that the match appears at least once. This basically translates into finding the words.

result **=** re.findall('[^!.? ]+',test\_phrase)

print(result)

['This', 'is', 'a', 'string', 'But', 'it', 'has', 'punctuation', 'How', 'can', 'we', 'remove', 'it']

Character Ranges

As character sets grow larger, typing every character that should (**or** should **not**) match

could become very tedious. A more compact format using character ranges lets you define a

character set

to include all of the contiguous characters between a start **and** stop point.

The format used **is** [start**-**end].

​

Common use cases are to search **for** a specific range of letters **in** the alphabet.

For instance, [a**-**f] would **return** matches **with** any occurrence of letters between a **and** f.

​

Let's walk through some examples:

​

test\_phrase **=** 'ASHA This is an example sentence. Lets see if we can find some letters.'

​

test\_patterns**=**['[a-z]+', *# sequences of lower case letters*

'[A-Z]+', *# sequences of upper case letters*

'[a-zA-Z]+', *# sequences of lower or upper case letters*

'[A-Z][a-z]+'] *# one upper case letter followed by lower case letters*

multi\_re\_find(test\_patterns,test\_phrase)

Searching the phrase using the re check: '[a-z]+'

['his', 'is', 'an', 'example', 'sentence', 'ets', 'see', 'if', 'we', 'can', 'find', 'some', 'letters']

Searching the phrase using the re check: '[A-Z]+'

['ASHA', 'T', 'L']

Searching the phrase using the re check: '[a-zA-Z]+'

['ASHA', 'This', 'is', 'an', 'example', 'sentence', 'Lets', 'see', 'if', 'we', 'can', 'find', 'some', 'letters']

Searching the phrase using the re check: '[A-Z][a-z]+'

['This', 'Lets']

test\_phrase **=** 'This is a string with some numbers 1233 and a symbol #hashtag'

​

test\_patterns**=**[ r'\d+', *# sequence of digits*

r'\D+', *# sequence of non-digits*

r'\s+', *# sequence of whitespace*

r'\S+', *# sequence of non-whitespace*

r'\w+', *# alphanumeric characters*

r'\W+', *# non-alphanumeric*

]

​

multi\_re\_find(test\_patterns,test\_phrase)

Searching the phrase using the re check: '\\d+'

['1233']

Searching the phrase using the re check: '\\D+'

['This is a string with some numbers ', ' and a symbol #hashtag']

Searching the phrase using the re check: '\\s+'

[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']

Searching the phrase using the re check: '\\S+'

['This', 'is', 'a', 'string', 'with', 'some', 'numbers', '1233', 'and', 'a', 'symbol', '#hashtag']

Searching the phrase using the re check: '\\w+'

['This', 'is', 'a', 'string', 'with', 'some', 'numbers', '1233', 'and', 'a', 'symbol', 'hashtag']

Searching the phrase using the re check: '\\W+'

[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' #']

**%%**writefile file1.py

"This is the file created for testing regular expression\n"

"There are many lines added in this file"

​

​

Writing file1.py

**with** open('file1.py','r') **as** filehand:

**for** line **in** filehand:

result **=** re.findall(r'f[\w]\*',line)

**if** len(result)**>**0:

print(result,line)

​

['file', 'for']

['file']

​