Regular Expression

Regular expression is a tool to match the pattern in the text, by using this we can match, find and replace the text or word into the strings. so we need to understand what I pattern and How to right that pattern.

**^** = beginning of the line (example – “^python” 🡪 it indicates that our line strt with python)

**$**  = End of the line(example if you email ends with **.com** you can mention as **com$)**

**.** = single character except to new line it can be alphabets or number and also it can

be operation +, -, /,\*

**[]** = it will match any single character (example [1-9] 🡪 it will look for single character from 1-9)

**^**[1-9] = it match single character other than the number inside the bracket

‘A**\*** ‘ - it matches 0 or more occurrence of preceding expression (example it can be A, AA, AAA, AAAA and so on)

‘A**+** ‘ = it matches 1 or more occurrence of preceding expression (example it can be A, AA, AAA, AAAA and so on)

‘A**?** ‘ = is matches 0 or 1 occurrence of preceding expression

**{}** = it indicates multiplications, examples a below

‘A**{2}**’ = it match exactly 2 occurrence of AA

‘A**{2,}**’ = it match 2 occurrence of AA or more than that

‘A**{2, 5}**’ = it match from 2 to 5 occurrence of AA

A**|**B = it matches either A or B

**()** = it indicates group the regular expression

**\s** = it matches the space

**\S** = it matches the non-white space(any character other than the space)

**\d** = it matches single character that is digits

**\D** = it matches single character which are non-Digits

**\w** = it matches any single character alphabets, number and underscore( \_ )

**Replace Function using string function**

Str1 = my name is Deepak

# I want to replace Deepak with Modi

Str1.repace(“Deepak”, “Modi”)

O/P

My name is Modi

**Drawback on string function**

Str1 = the road here is broad

# I want to replace road with rd

Str1.replace(“road”,”rd”)

O/P

The rd here is brd

**In regular expression, this can be over-come by using index operation**

Str1 = the road here is broad

Srt1[0:17]+str1[17:].replace(“road”, “rd”)

O/P

The rd here is broad

This is what about why regular expression

## **Pattern of the functions**

Variable name = re.match/search/finall(pattern,string)

## **Match function in regular expression:**

Match function is used to match any text or word in the string will go with example

*import re*

*str1 = "Python class at FITA"*

*match = re.match(r"Python \w\w\w\w\w", str1)*

*print(match)*

*print(match.group(0)) # it is to print only the matched object*

*O/P*

*<re.Match object; span=(0, 12), match='Python class'>*

*Python class*

## **Search function in regular expression:**

search function is used to match any text or word in the string will go with example

*import re*

*str1 = "Python class at FITA"*

*search1 = re.search(r"Python \w\w\w\w\w", str1)*

*print(search1)*

*print(search1.group(0)) # it is to print only the matched object*

*O/P*

*<re.Match object; span=(0, 12), match='Python class'>*

*Python class*

The difference between match and search is match finds at the beginning of the string but search finds entire string in the below example I have changed the string and search it returns first match of that pattern

*import re*

*str1 = " FITA Python class"*

*search1 = re.search(r"Python \w\w\w\w\w", str1)*

*print(search1)*

*print(search1.group(0)) # it is to print only the matched object*

*O/P*

*<re.Match object; span=(0, 12), match='Python class'>*

*Python class*

## **Findall function in regular expression:**

findall function is used to match all the content or word in the string will go with example

*EX1:*

*import re*

*str1 = "Python class at FITA and the version is Python 3\_7\_4"*

*findall1 = re.findall(r"Python \w\w\w\w\w", str1)*

*print(findall1)*

*O/P*

*['Python class', 'Python 3\_7\_4']*

*EX2:*

*import re*

*str1 = "we need to inform him with the atest information"*

*for i in re.finditer("inform", str1):*

*x = i.span() # here it will tell us the starting index and index of the requested string*

*print(x)*

*O/P*

*(11, 17)*

*(37, 43)*

## **Greedy method in regular expression:**

In this method it will look for additional expression then the compiled one will check with an example

*Ex:*

*mystr = "abcadname = 'deepak' abdjkadfasdfname = 'kumar' adsfasdaf"*

*m = re.search(r'name = \'(.\*)\' ', mystr) # \ is an escape char, dot(.) ref to single character and \* is it match 0 or more preceding expression*

*print(m)*

**I was looking for the word “” name = ‘Deepak’ “” but the output was as below it was searched still the second quotes ( ‘ )**

*O/P*

*match="name = 'deepak' abdjkadfasdfname = 'kumar'*

here is \* and + are greedy and we want to make our greedy into non-greedy

*EX:*

*mystr = "abcdname = 'deepak' abdjkadfasdfname = 'kumar' adsfasdaf"*

*m = re.search(r'name = \'(.\*?)\'', mystr) # \ is an escape char, dot(.) ref to single character and \* is it match 0 or more preceding expression*

*print(m.group())*

*O/P*

*name = 'deepak'*

## **Lookahead method in regular expression:**

We will go with an example, here we are going to find first bb followed by a

*EX:*

*mystr = 'ababaaabbaaaabbababaaa'*

*m = re.search(r'b(?=b)',mystr) #(?= this is the lookahead operator*

*print(m)*

O/P:

*span=(7, 8), match='b'*

## **Look behind method in regular expression:**

We will go with an example, here we are going to find first b which is preceded by 3 a

*EX:*

*mystr = 'ababaababbaaaabbababaaa'*

*m = re.search(r'(?<=a{3})b',mystr) #(?<= this is the lookbehind operator and this {3} is for multiplication*

*print(m)*

O/P:

*span=(14, 15), match='b'*

## **‘-ve’ Lookhead method in regular expression:**

Here we are going to find first b which is not followed by an a

*Ex:*

*mystr = 'ababaababbaaaabbababaaa'*

*m = re.search(r'b(?!a)',mystr) #(?! this is the -ve lookahead operator*

*print(m)*

*O/P*

*span=(8, 9), match='b'*

## **‘-ve’ Look behind method in regular expression:**

Here we are going to find a which does not have b before init

*EX1*

*mystr = 'ababaababbaaaabbababaaa'*

*m = re.search(r'(?<!b)a',mystr) #(?<! this is the -ve lookbehind operator*

*print(m)*

*O/P*

*span=(0, 1), match='a'*

Here we are going to find b which does not have a before init

*EX1*

*mystr = 'ababaababbaaaabbababaaa'*

*m = re.search(r'(?<!a)b',mystr) #(?<! this is the -ve lookbehind operator*

*print(m)*

*O/P*

*span=(9, 10), match='b'*