**RE methods**

1. *re.match(pattern, string):* Match pattern only at beginning of string.

*Example*:

>>> re.match('My','My name is XYZ')

<\_sre.SRE\_Match object; span=(0, 2), match='My'>

2. *re.search(pattern, string):* Match pattern anywhere in the string (Only the first occurrence of the pattern).

*Example:*

>>> re.search('n','My name is XYZ')

<\_sre.SRE\_Match object; span=(3, 4), match='n'>

3. *re.findall(pattern, string):* Returns a list of all the occurrences of the pattern in the string.

*Example:*

>>> re.findall('n','My name is name')

['n', 'n']

**Special Characters**

a) . - Match any character except newline (returns a string before the ‘.’. No of ‘.’ = no of characters it returns)

*Example:*

>>> re.search('.t','christ')

<\_sre.SRE\_Match object; span=(4, 6), match='st'>

b) ^ - Match the start of the string –

re.search('^M','My name is XYZ')

c) $ - Match the end of the string -

re.search('t$','christ')

d) \d - Matches a digit; or [0-9] –

re.findall('\d','christ12uni556versity66') output: ['1', '2', '5', '5', '6', '6', '6']

e) \D - Matches a non-digit; or [^0-9] –

re.findall('\D','chr55ist66') output: ['c', 'h', 'r', 'i', 's', 't']

f) \s - Matches a whitespace; or [ \t\n\r\f\v] –

re.findall('\s','ch r55 st66\n') output: [' ', '\t', '\n']

g) \S - Matches a non-whitespace; or [^ \t\n\r\f\v] –

re.findall(('\S','ch r55 st66\n') output: ['c', 'h', 'r', 'i', 's', 't']

h) \w - Matches a “word” character; or [a-zA-Z0-9\_] -

re.findall('\w','ch r55 st66\n') output: ['c', 'h', 'r', '5', '5', 's', 't', '6', '6']

i) \W - Matches a Unicode non-“word” character; or [^a-zA-Z0-9\_] -

re.findall('\W','ch r55 st66\n') output: [' ', '\t', '\n']

j) e\* -Matches 0 or more repetitions of expression e.

re.findall('fe\*ls','feels feeels fls') output: ['feels', 'feeels', 'fls']

k) e+ - 1 or more repetitions of expression e.

re.findall('fe+ls','feels feeels fls') output: ['feels', 'feeels']

l) e? - 0 or 1 repetitions of expression e.

re.findall('fe?ls','fels feels fls') output: ['fels', 'fls']

**Character Class**

Rather than matching one particular character we want to match any one of a set of characters. This can be achieved by using a ***character class***—one or more characters enclosed in square brackets.

*Example:*

>>> re.findall('ch[rpt]ist','christ chpist chuist')

['christ', 'chpist']

**\*IMP: Picks one of the character from the character set and matches. Returns the string if true\***

We can use range too [a-z].

***To check whether a number starts with 7,8 or 9***

>>> re.search('^[7-9]','856637287')

<\_sre.SRE\_Match object; span=(0, 1), match='8'>

**\*IMP: Note that inside a character class, apart from \, the special characters lose their special meaning, although in the case of ^ it acquires a new meaning (negation)\***

**Quantifiers**

1. *e*{*m*} **-** Match exactly *m* occurrences of expression *e*

re.findall('fe{2}ls','feels feeels feeps') output: ['feels']

2. e{m,n} - match at least m and at most n occurrences of expression –

re.findall('fe{1,3}ls','feels feeels feeps fels') output: ['feels', 'feeels', 'fels']

3. e{m,} – Greedily matches at least m occurrences of expression e

re.findall('ch.{2,}l','chrwueilqoewul') output: ['chrwueilqoewul']

4. e{,n} - Greedily matches at most n occurrences of expression e

re.findall('ch.{,7}l','chyyuullol') output: ['chyyuullol']

5. e{m,}? - Nongreedily matches at least m occurrences of expression e

re.findall('ch.{2,}?l','chrwueilqoewul') output: ['chrwueil']

6. e{,n}? - Nongreedily matches at most n occurrences of expression e

re.findall('ch.{,7}?l','chyyuullol') output: ['chyyuul']

**Grouping and capturing**

If we want to match any one of two or more alternatives, we use the alternating symbol ‘|’

*Example:*

>>> re.findall('aircraft|airplane|jet','craft airplane jet')

['airplane', 'jet']

For aircraft and airplane, grouping can be done by using – ()

Eg: air(craft|plane)|jet which will **match** either ‘aircraft’ or ‘airplane’ or ‘jet’ and will **capture the grouped part** i.e., either ‘plane’ or ‘craft’ (if airplane or aircraft is matched) or ‘ ‘ (if jet is matched because jet has no group)

*Example:*

>>> re.findall('air(craft|plane)|jet','craft airplane jet')

['plane', ' ']

regex (air(craft|plane)|jet) - will give you **two captures** (eg: (airplane, plane)) if ‘airplane’ or ‘aircraft’ is matched and will give you one capture if jet is matched

*Example:*

>>> re.findall('(air(craft|plane)|jet)','craft airplane jet')

[('airplane', 'plane'), ('jet', '')]

To **switch off** capturing effect, we use ‘?:’ following the opening parenthesis.

*Example:*

>>> re.findall('(air(?:craft|plane)|jet)','craft airplane jet')

['airplane', 'jet']

**Assertions**

1. \b - which asserts that the character that precedes it must be a “word” (\w)

re.findall(r'\bjet\b','jet jetski')

output: ['jet']

2. \A - Matches at the start

re.findall(r'\A\w+','black white')

output: ['black']

3. \B - which asserts that the character that precedes it must be a non- “word” (\W)

re.findall(r'\Bcat\B','certificate')

output: ['cat']

4. \Z - Matches at the start

re.findall('\w+h\Z','catfish')

output: ['catfish']

5. (?=e) = Matches the string with the expression e and after matching the current position is at the beginning of the string. It returns an object if expression is matches, else returns null

re.search('(?=^\w+[a-z].+)','hello')

output: <\_sre.SRE\_Match object; span=(0, 0), match=''>

**Flags**

1. **re.M or re.MULTILINE** - When specified, the pattern character ^ match the beginning of the string and the beginning of each line (immediately following each newline); and the pattern character $ match at the end of the string and at the end of each line (immediately preceding each newline).

re.findall('^a.+s$','a boys name is suhas\nHe is 12 years old\na dog has four legs',flags=re.M)

['a boys name is suhas', 'a dog has four legs']

2. **re.I or re.IGNORECASE –** Indicates case-insensitive matching.

re.findall('^a.+s$','A boys name is suhaS',re.I)

['A boys name is suhaS']

3. **re.S or re.DOTALL** - Make the dot character . match any character, including a newline. Without this flag, a dot will match anything except a newline.

re.findall('.+','once upon a time,\nthere lived a king',re.S)

['once upon a time,\nthere lived a king']