. - Any Character Except New Line

\d - Digit (0-9)

\D - Not a Digit (0-9)

\w - Word Character (a-z, A-Z, 0-9, \_)

\W - Not a Word Character

\s - Whitespace (space, tab, newline)

\S - Not Whitespace (space, tab, newline)

\b - Word Boundary (means start or an word), **beginning or at the end of a word**

\B - Not a Word Boundary(not at start of word), N**OT at the beginning (or at the end) of a word**

^ - Beginning of a String

$ - End of a String

[] - Matches Characters in brackets

[^ ] - Matches Characters NOT in brackets

| - Either Or

( ) - Group

Quantifiers:

\* - 0 or More

+ - 1 or More

? - 0 or One

{3} - Exact Number

{3,4} - Range of Numbers (Minimum, Maximum)

**re.finditer(pattern, string, flags=0)**

--- Return an iterator yielding match objects over all non-overlapping matches for the RE pattern in string. The string is scanned left-to-right, and matches are returned in the order found

Once got match object we can call below methods on it

start()-----> start index of mattc

end() -------> end+1 index of match

group()----> returns matched string

**Flags**

re.compile('pattern\_want\_to\_search',re.I)

re.I--------> Ignore case sensitive flag

1. search(pattern, string)----- This methods returns a match object
2. Findall(pattern, string) --- this methods reruns all metched as string in list
3. re.sub(pattern, repl, string, count=0, flags=0)

pattern ----- Pattern that you want to replace with new value

string ----- String on which replacement will be done

repl -------- New value of replacement string

Return the string obtained by replacing the leftmost non-overlapping occurrences of pattern in string by the replacement repl. If the pattern isn’t found, string is returned unchanged. repl can be a string or a function; if it is a string, any backslash escapes in it are processed. That is, \n is converted to a single newline character, \r is converted to a carriage return, and so forth

**Greedy and Non-greedy**

\*?, +?, ??

The '\*', '+', and '?' qualifiers are all greedy; they match as much text as possible.

if the RE <.\*> is matched against '<a> b <c>', it will match the entire string, and not just '<a>'.

Adding ? after the qualifier makes it perform the match in non-greedy or minimal fashion; as few characters as possible will be matched. Using the RE <.\*?> will match only '<a>'

**Example: 1**

s = '<html><head><title>Title</title>' -------------- one sample string

**Greedy search**

re.match('<.\*>', s) ------------------ this will match the whole string as this is greedy search

<html><head><title>Title</title>

**Non Greedy search**

To make on greedy search the solution is to use the non-greedy qualifiers **\*?, +?, ??, or {m,n}?,** which match as little text as possible.

[]

It can be used in multiple ways

1. To match listed individually, e.g. [amk] will match 'a', 'm', or 'k'.
2. Ranges of characters can be indicated by giving two characters and separating them by a '-', for example [a-z]. For ranging it picks one character as starting and other as end.

e.g-

[0-9A-Fa-f] will match characters in 0 to 9 range , a-f , A-F(any hexadecimal digit)

**[0-9a-zA-Z.-]** ---- will match alpha numeric containing ‘.’ And ‘-‘

1. Special characters lose their special meaning inside sets. For example, [(+\*)] will match any of the literal characters '(', '+', '\*', or ')'.
2. Character classes such as \w or \S (defined below) are also accepted inside a set, although the characters they match depends on whether ASCII or LOCALE mode is in force
3. Characters that are not within a range can be matched by complementing the set. If the first character of the set is '^', all the characters that are not in the set will be matched
4. We can use [] to search for specific words/pattern also by enclosing them in quote.

Search for words 'fox', 'dog', 'horse'

patterns = [ 'fox', 'dog', 'horse' ]

text = 'The quick brown fox jumps over the lazy dog.'

for pattern in patterns:

    print('Searching for "%s" in "%s" ->' % (pattern, text),)

    if re.search(pattern,  text):

        print(re.findall(pattern,  text))

    else:

        print('Not Matched!')

**Specific/any character matching at end with []**

1. [^b]at

It will match any string that doesn’t start with ‘b’ but end with ‘at’

1. \d[-.]

Match aby string which have digits followed by ‘-‘ or ‘.’

1. Match any number starting with 800 or 900

[8-9]00

[8-9]-----> it tell number starting with 8 or 9

00 -----> preceding ‘00’ are literal search, means two 0 after 8 or 9

1. Write a re code which can match aby of below name pattern (name title—Mr. Mr, Mrs Ms followed by space then first name Start will upper case letter , Last name may or may not be be it present then firsat character must be Upper case)

e.g-

Mr. Schafer

Mr Smith

Ms Davis

Mrs. Robinson

Mr. T

Ans

matches=re.finditer(r'M(r|s|rs)\.?\s?[A-Z]\w\*',lines)

for match in matches:

    print(match)

Exmpale 2:

Write re code which will capture below kind of uls

https://www.google.com

http://coreyms.com

https://youtube.com

<https://www.nasa.gov>

https?://(www\.)?\w+(\.\w+)`

Example:

Write a code to extract year ,month,date from below URL

url1= "https://www.washingtonpost.com/news/football-insider/wp/2016/09/02/odell-beckhams-fame-rests-on-one-stupid-little-ball-josh-norman-tells-author/"

Method 1

print(re.findall(r'(\d{4})/(\d{2})/(\d{2})',url1))

Result will be-

[('2016', '09', '02')]

Method 2:

print(re.findall(r'\d{4}/\d{2}/\d{2}',url1))

Result will be in form

['2016/09/02']

**WAS to check is string is a valid IP**

ip='1.20.223.3'

regex = "^((25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])\.){3}(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9]?[0-9])$"

pat = re.search(regex, ip)

print(pat)

Example

Write a Python program to match if two words from a list of words starting with letter 'P'.

Example:

words = ["Python PHP", "Java JavaScript", "c c++"]

output should be – Python and PHP

words = ["Python PHP", "Java JavaScript", "c c++","Pra Pk"]

for w in words:

        m = re.match("(P\w+)\W(P\w+)", w)

        # Check for success

        if m:

            print(m.groups())

**Replacing the patten by somethings or relacing one group by other**

In python there is concept of backreferences (backslash, group number).

his is Python's regex substitution (replace) function. The replacement string can be filled with so-called backreferences (backslash, group number) which are replaced with what was matched by the groups. Groups are counted the same as by the group(...) function, i.e. starting from 1, from left to right, by opening parentheses.

**re.sub(pattens\_in\_groups,\\replacement\_for\_group)**

Exaple:

Write a Python program to convert a date of yyyy-mm-dd format to dd-mm-yyyy format.

import re

def change\_date\_format(dt):

        return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', dt)

dt1 = "2026-01-02"

print("Original date in YYY-MM-DD Format: ",dt1)

print("New date in DD-MM-YYYY Format: ",change\_date\_format(dt1))

**NOTE:**

We are using two \\ b/c one backslash is for group number and other backslash is for replacement.

**Questions**

Write a Python program to find all five characters long word in a string

text = 'The quick brown fox jumps over the lazy dog.'

print(re.findall(r"\b\w{5}\b",text))

print(re.findall('\w{5}',text)) #this will also work but not very acceptable

**Questions:**

Write a Python program to check a decimal with a precision of 2

text1 = "13.4324"

print(re.findall(r'\d{2}\.\d{2}?$',text1))

print(re.search(r'\d{2}\.\d{2}?$',text1))

**Questions:**

Write a Python program to remove the parenthesis in a string

text1 = "example(.com)"

print(re.sub(r'[()]','',text1))

**Questions:**

Write a Python program to remove the parenthes content area in a string

e.g—input -github(.com) then result github

text1 = "example (.com)"

print(re.sub(r'\(.\*?\)','',text1))

**Questions:**

WAP to get the domain name , domain and from email ids

text1 = "myid@gmail.com"

print(re.findall(r'(\w.\*?)@(\w.\*?)\.(\w.\*)',text1))

print(re.findall(r'@(\w.\*?)\.\w\*',text1))

**Accept string that ending with number**

str='ankitrai326'

obj=re.search('.\*[0-9]$',str)

if obj is not None:

    print('accepted')

else:

    print('notaccepted')

accept string starting with vowel

str='animal'

print(re.findall('^[aeiou].\*$',str))

**string starts with a substring**

str="geeks for geeks makes learning fun"

substring='geeks'

print(re.search('^'+substring+'.\*',str))

Find out url from a given string

e.g--

text = '<p>Contents :</p><a href="https://w3resourc&e.com">Python Examples</a><a href="http://github.com">Even More Examples</a>'

output-- ['https://w3resourc&e.com', 'http://github.com']

Note--- in url there may be some special character and symbols

import re

text = '<p>Contents :</p><a href="https://w3resourc&e.com">Python Examples</a><a href="http://github.com">Even More Examples</a>'

result=re.findall(r'https?://.\*?\.com', text)

print(result)

Remove all character except alpha numeric from string.

text1 = '\*\*//Python Exercises// - 12. '

result=re.sub('[^a-zA-Z0-9]', '', text1)

print(result)