Regular Expressions In Python

- **While developing the real time application like web-applications in the User Interface we need to perform some sort of validation based on some pattern then we have to use regular expression**
- **Reading the data from data based some specific pattern**
- **Finding and replacements options in ms-word …**

Match():

match() will search for the “pattern” in the first word of target. If the first word of target matched with given pattern then it will return “first word” to the “Match” Object otherwise it will return “None”

Example :

```
import re

p=re.compile("m\w\w")     #we can specify the patter
print("Type is : ",type(p)) # <class 're.Pattern'>

m=p.match("man mad map jug jun ram sai manasa mom")
print("Type is : ",type(m))  #<class 're.Match'>
print("Match Object : ",m)
```

search():

search() will search for the “pattern” in the entire target. If any matches are existed with given pattern then it will return that “matched string” to the “Match” Object otherwise it will return “None”

```
import re

p=re.compile("m\w\w")   # <class 're.Pattern'>
m=p.search("gun jug mam mom map dad dam") #<class 're.Match'>
print("Match Object : ",m)
```

findall()

findall( ) return list collection with matched string from target whatever string are matched with given pattern . If None of the match is exists then it will return an empty list []

```
import re
p=re.compile("m\w\w")
lst=p.findall("dad mom mam gun jug mad map nan")
print("Matched String : ",lst)  #[mom,mam,mad,map]
```

Alternate way for match( ) | search( ) | findall( )

```
import re

'''
p=re.compile("m\w\w")
m=p.match("mom mad mam dad jug gun nan") '''

#re.match(pattern,target)
m=re.match("m\w\w","mom mad mam dad jug gun nan")
print("Match Object : ",m)

'''
p=re.compile("m\w\w")
m=p.search("mom mad mam dad jug gun nan") '''
```

```
#re.search(pattern,target)
m=re.search("m\w\w","dom mad mam dad jug gun nan")
print("Match Object : ",m)

'''
p=re.compile("m\w\w")
lst=p.findall("mom mad mam dad jug gun nan") '''

lst=re.findall("m\w\w","mom mad mam dad jug gun nan")
print("Matches ",lst)

# match( ) --> Match Object | None
# search( ) -> Match Object | None
# findall( ) -> list Collection
```

#Match Object

```
import re

m=re.match("m\w\w","mom dad mam sir anu cnu jug")
print("Type is : ",type(m)) #<class 're.Match'>
print("Match Object : ",m)

print("Match : ",m.group())
print("Starting index pos of match : ",m.start())
print("Ending index pos+1 of match : ",m.end())
```

Finditer() :

it will return callable_iterator Object . it is the collection of Match Objects

```
import re
import time

citr=re.finditer("m\w\w","mom dad mam sir mad anu mug cnu jug")
print("Type is : ",type(citr))  #<class 're.callable_iterator'> | iterator
print("citr Object : ",citr)
#callable_iterator is the collection of Match objects

for m in citr:
   time.sleep(1)
   print(m)
   print("- "*40)
   print("Match : ",m.group())
   print("Starting index : ",m.start())
   print("Ending index : ",m.end())
```

Output:

```
Type is :  <class 'callable_iterator'>
citr Object :  <callable_iterator object at 0x00000054500B34C0>
<re.Match object; span=(0, 3), match='mom'>
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :  mom
Starting index :  0
Ending index :  3
<re.Match object; span=(8, 11), match='mam'>
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :  mam
Starting index :  8
Ending index :  11
<re.Match object; span=(16, 19), match='mad'>
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :  mad
Starting index :  16
Ending index :  19
<re.Match object; span=(24, 27), match='mug'>
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :  mug
Starting index :  24
Ending index :  27
```

Working with Predefined Patterns

\w → any alpha| numerical values
\W → except Alpha | numerical values
\d → only digits
\D → except Digits
\s → only spaces
\S → except Spaces
^ → Target start with
$ → Target ends with
. → any thing

Example :

```
import re
import time

citr=re.finditer("\d","a1 $7A *B")

for m in citr:
   time.sleep(1)
   print("Match : ",m.group( ))
   print("Index : ",m.start())
   print("- "*30)
```

Example :

```
import re
import time

citr=re.finditer("\D","a1 $7A *B")

for m in citr:
   time.sleep(1)
   print("Match : ",m.group( ))
   print("Index : ",m.start())
   print("- "*30)
```

output:

```
Match :  a
Index :  0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :
Index :  2
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :  $
Index :  3
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :  A
Index :  5
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :
Index :  6
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :  *
Index :  7
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Match :  B
```

```
Index :  8
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Example:

```
import re
import time
citr=re.finditer("\w","a1 $7A *B")
for m in citr:
   time.sleep(1)
   print("Match : ",m.group( ))
   print("Index : ",m.start())
   print("- "*30)
```

Example :

```
import re
import time
citr=re.finditer(".","a1 $7A *B")
for m in citr:
   time.sleep(1)
   print("Match : ",m.group( ),"Index : ",m.start())
```

Output:

```
Match :  a Index :  0
Match :  1 Index :  1
Match :    Index :  2
Match :  $ Index :  3
Match :  7 Index :  4
Match :  A Index :  5
Match :    Index :  6
Match :  * Index :  7
Match :  B Index :  8
```

Note:

Whenever predefine patterns fail to fulfill requirements then we have to define our own patterns called "User defined" Pattern

[abc] → either a or b or c
[aeiou] → either a or e or i or o or u
[a-z] → any lower case letters
[A-Z] → any upper case letters
[a-zA-Z] → any upper case or lower case letters
[^a-zA-Z] → Except alphabets
[589] → either 5 or 8 or 9
[5-9] → from 5 to 9
[0-9] → any digits or \d
[a-zA-Z0-9] → Alpha and Numerical or \w
[^a-zA-Z0-9] → Except Alpha and numerical or \W

Example:

```
import re
import time
citr=re.finditer("[a-z]","a1 $7A *BcD 9")
for m in citr:
   time.sleep(1)
   print("Match : ",m.group()," index : ",m.start())
```

Example 2:

```
import re
import time

citr=re.finditer("[acD$]","a1 $7A *BcD 9")
for m in citr:
   time.sleep(1)
   print("Match : ",m.group()," index : ",m.start())
```

output:

```
Match :  a  index :  0
Match :  $  index :  3
Match :  c  index :  9
Match :  D  index :  10
```

Quantifiers :

No.of.occ of match

"a" → exact one "a"
"a+" → at least one 'a' or any no.of 'a's
"a?" → at most one 'a' i.e either 0 a's or 1 a
"a*" → either 0 a's or any no.of. a's
"a{n} → a for n times
"a{m,n} " → a for min m times and max "n" times

Example:

```
import re
import time

citr=re.finditer("a+","a aab aaab aaaab")
for m in citr:
   time.sleep(1)
   print("Match : ",m.group(),"index : ",m.start())
```

Output:

```
Match :  a index :  0
Match :  aa index :  2
Match :  aaa index :  6
Match :  aaaa index :  11
```

Example:

```
import re
import time
citr=re.finditer("a*","a aab aaab aaaab")
for m in citr:
   time.sleep(1)
   print("Match : ",m.group(),"index : ",m.start())
```

output:

```
Match :  a index :  0
Match :   index :  1
Match :  aa index :  2
Match :   index :  4
Match :   index :  5
Match :  aaa index :  6
Match :   index :  9
Match :   index :  10
Match :  aaaa index :  11
Match :   index :  15
Match :   index :  16
```

App1: regular expression for validating mobile number with following

- It must be starts with 6 or 7 or 8 or 9
- Total no.of. digits should be 10

App-1

```
 [6789][0-9][0-9][0-9]
         [0-9][0-9][0-9]
         [0-9][0-9][0-9]
```

App-2:

```
[6-9][0-9]+   //invalid    input →   68  //valid
                         Input → 7888888888888888888  //valid
[6-9][0-9]*   //invalid
    Input → 9 //valid
    Input → 9898 //valid
    Input → 98888888888888888888888888888888 //valid

[6-9][0-9]{9} → valid   or  [6-9]\d{9}
    Input → 9898  //Error
    Input → 9123456789 //valid
    Input →  91231231239  //Error
```

App2: Patter for validate mobile number

- Need to starts with 6-9
- It may be 10 digit or 11 digit length
- If it is 11 digits it should starts with 0

[6-9][0-9]{9} → For validating 10 digits
0[6-9][0-9]{9} → For validating 11 digits
0?[6-9][0-9]{9} → valid for both 10 digit and 11 digit
0?[6-9]\d{9} → valid

App3: Patter for validate mobile number

- Need to starts with 6-9
- It may be 10 or 11 or 12 digit length
- If it is 11 digits it should starts with 0
- If it is 12 digits it should starts with 91

[6-9][0-9]{9} → 10 digits [only for 10 digits]
0?[6-9][0-9]{9} → 10 digits or 11 digits

91?0?[6-9][0-9]{9} → //invalid
9107999999999 → valid
07999999999 → valid
7999999999 → valid

(0|91)?[6-9][0-9]{9} //valid

Example:

```
import re
import time

target="ramesh 2-12-2020 sudha 12-10-2012 Ram 23-9-98"
citr=re.finditer("\d{1,2}-\d{1,2}-\d{2,4}",target)

for m in citr:
   time.sleep(1)
   print(m.group())
```

Example:

```
import re
import time

target="ramesh 2-JAN-2020 sudha 12-10-2012 Ram 23-Feb-98"
citr=re.finditer("[1-9]{1,2}-[a-zA-Z0-9]+-[0-9]{2,4}",target)

for m in citr:
   time.sleep(1)
   print(m.group())
```

Example:

```
import re
import time

target="hello my dear have a nice day"
match=re.search("^h\w{4}",target)

if match!=None:
   print("Target start with  given pattern ")
else:
   print("Sorry target not start with given pattern")
```

Example :2

```
import re
import time

target="hello my dear have a nice day"
match=re.search("d\w{2}$",target)

if match!=None:
   print("Target ends with  given pattern ")
else:
   print("Sorry target not ends with given pattern")
```

Example:

```
import re
import time

target="hello my dear have a nice Day"
match=re.search("d\w{2}$",target,re.IGNORECASE)

if match!=None:
   print("Target ends with  given pattern ")
else:
   print("Sorry target not ends with given pattern")
```

Example:

```
import re
import time

target="mam madam manasa madhu have a nice day mom"
lst=re.findall(r"\bm\w\w\b",target)
print("Result is : ",lst)
```

here : r is raw string

Validate mail-ID

```
1 |--------------2---------|     |------3-----|
s hashikumarch.sssit @ gmail.com
```

Regular Expression for Only gmail-id

```
[a-zA-Z0-9][a-zA-Z0-9_.]+@gmail[.]com
```

Regular Expression for any mail-id

```
1 |--------------2---------|     |------3-----|
s hashikumarch.sssit @ gmail.com
                       yahoo.com
                         tv9.com
                         v6.com
                         eenadu.net
[a-zA-Z0-9][a-zA-Z0-9_.]+@[a-z0-9]+[.][a-z]+
\w[a-zA-Z0-9_.]+@[a-z0-9]+[.][a-z]+
```

Regular Expression for any mail-id

```
1 |--------------2---------|     |------3-----|
s hashikumarch.sssit @ gmail.com
                       yahoo.com
[a-zA-Z0-9][a-zA-Z0-9_.]+@[a-z0-9]+[.][a-z]+

                         tv9.gov.in
                         v6.edu.uk
                         eenadu.net
[a-zA-Z0-9][a-zA-Z0-9_.]+@[a-z0-9]+[[.][a-z]+]+
```

Fullmatch( ) will return match object if the complete target match with given pattern

Validating Mobile number

```
import re
import time

data=input("Enter u r mobile number ")
match=re.fullmatch("[6-9][0-9]{9}",data)

if match!=None:
   print("Valid Mobile number")
else:
   print("Sorry Invalid Mobile number")
import re
import time
```

Validating Mail-IDs

```
data=input("enter u r mail-id ")
match=re.fullmatch("[a-z0-9A-Z][a-zA-Z0-9._]+@[a-z0-9]+[[.][a-
z]+]+",data)

if match!=None:
   print("Valid mail-id")
else:
   print("Invalid Mail-id")
```

re.sub(pattern,replacement,target) → str

- It will return string after replacing the target will given replacement data which are satisfied by the given pattern

```
import re
import time
m=re.sub
   (r"\b\w{4}\b","***"," Hello my dear happy to see you")
print("type is : ",type(m))
print("Result is : ",m)
```

subn(pattern,replacement,target)-> tuple

- It will return tuple object after replace the target with specified pattern
- t[0] will contain the result
- t[1] will contain total no.of.replacements are done

```
import re
import time

t=re.subn(r"\b\w{3}\b","***"," Hello my dear happy to see you")
print("type is : ",type(t))
print("Result is : ",t[0])
print("No.of.Replacements are : ",t[1])
```

split(pattern,target)-> list

- Used to split the target based on the specified pattern

```
import re
import time

t=re.split(r"\b\w{4}\b"," Hello my dear happy to see you")
print("type is : ",type(t))
print("Result is : ",t)
```