A RegEx is a powerful tool for matching text, based on a pre-defined pattern. It can detect the presence or absence of a text by matching it with a particular pattern, and also can split a pattern into one or more sub-patterns. The Python standard library provides a re module for regular expressions. Its primary function is to offer a search, where it takes a regular expression and a string. Here, it either returns the first match or else none.

\w – matches a word character

\d – matches digit character

\s – matches whitespace character (space, tab, newline, etc.)

\b – matches a zero-length character

\A Returns a match if the specified characters are at the beginning of the string

---

```
import re

txt = "The rain in Spain"

#Check if the string starts with "The":

x = re.findall("\AThe", txt)

print(x)

if x:

    print("Yes, there is a match!")

else:

   print("No match")
```

```
['The']
Yes, there is a match!
```

```
match = re.search(r'portal', 'A computer science \ portal for Education')

print(match) print(match.group())

print('Start Index:', match.start())

print('End Index:', match.end())
```

```
<re.Match object; span=(21, 27), match='portal'>
portal
Start Index: 21
End Index: 27
```

#case sensitive words.

```
print(re.findall(r'[Ee]ducation', 'Education of education: \ A computer science portal for education'))
```

```
['Education', 'education', 'education']
```

#Ranges

```
print('Range',re.search(r'[a-zA-Z]', 'x'))
```

```
Range <re.Match object; span=(0, 1), match='x'>
```

```
x = range(3, 6)
for n in x:
  print(n)
```

```
3
4
5
```

```
x = range(3, 20, 2)
for n in x:
  print(n)
```

```
3
5
7
9
11
13
15
17
19
```

#Negation

```
print(re.search(r'[^a-z]', 'c'))
```

```
None
```

```
print(re.search(r'C[^l]', 'Class'))
```

```
None
```

#Beginning and End of String

 # Beginning of String

```
match = re.search(r'^is', 'This is the month')

print('Beg. of String:', match)

match = re.search(r'^is', 'is the month')

print('Beg. of String:', match)
```

 # End of String

```
match = re.search(r'education$', 'Compute science portal for education')

print('End of String:', match)
```

```
Beg. of String: None
Beg. of String: <re.Match object; span=(0, 2), match='is'>
End of String: <re.Match object; span=(27, 36), match='education'>
```

```
print('Any Character', re.search(r'p.th.n', 'python 3'))
```

```
Any Character <re.Match object; span=(0, 6), match='python'>
```

 # example of a word with an alternative spelling – color or colour.

```
print('Color',re.search(r'colou?r', 'color'))

print('Colour',re.search(r'colou?r', 'colour'))
```

```
Color <re.Match object; span=(0, 5), match='color'>
Colour <re.Match object; span=(0, 6), match='colour'>
```

#regular expression to identify the date (mm-dd-yyyy).

```
print('Date{mm-dd-yyyy}:', re.search(r'[\d]{2}-[\d]{2}-[\d]{4}','13- 07-2023'))
```

```
Date{mm-dd-yyyy}: <re.Match object; span=(0, 10), match='13-07-2023'>
```

#Consider a scenario where both three digits, as well as four digits, are accepted.

```
print('Three Digit:', re.search(r'[\d]{3,4}', '189'))

print('Four Digit:', re.search(r'[\d]{3,4}', '2145'))
```

```
Three Digit: <re.Match object; span=(0, 3), match='189'>
Four Digit: <re.Match object; span=(0, 4), match='2145'>
```

#Open-Ended Ranges

#No limit for a character repetition.

#We can set the upper limit as infinitive.

#A common example is matching street addresses.

```
print(re.search(r'[\d]{1,}','5th Floor, B-218, \ Sector-136, Noida, Uttar Pradesh - 201405'))
```

```
<re.Match object; span=(0, 1), match='5'>
```

#Shorthand characters

```
print(re.search(r'[\d]+','5th Floor, B-218,\ Sector-136, Noida, Uttar Pradesh - 201405'))
```

```
<re.Match object; span=(0, 1), match='5'>
```

#Grouping

```
grp = re.search(r'([\d]{2})-([\d]{2})-([\d]{4})', '12-07-2023') print(grp)
```

```
<re.Match object; span=(0, 10), match='12-07-2023'>
```

#Return a tuple of matched

```
grp = re.search(r'([\d]{2})-([\d]{2})-([\d]{4})','14-07-2023')
print(grp.groups())
```

```
('14', '07', '2023')
```

#Retrieve a single group

```
grp = re.search(r'([\d]{2})-([\d]{2})-([\d]{4})','14-07-2023')
print(grp.group(3))
```

```
2023
```

```
grp = re.search(r'(?P[\d]{2})-(?P[\d]{2})-(?P[\d] {4})','14-07-2023')
print(grp.group('dd'))
```

```
14
```

```
grp = re.search(r'(?P[\d]{2})-(?P[\d]{2})-(?P[\d] {4})','14-07-2023')
print(grp.groupdict()
```

```
{'dd': '14', 'mm': '07', 'yyyy': '2023'}
```

#Lookahead
#In the case of a negated character class

```
print('negation:', re.search(r'n[^e]', 'Python'))
print('lookahead:', re.search(r'n(?!e)', 'Python'))
```

```
negation: None
lookahead: <re.Match object; span=(5, 6), match='n'>
```

#Lookahead can also disqualify the match if it is not followed by a particular character.
#This process is called a positive lookahead, and can be achieved by simply replacing ! character
 # with = character.

```
print('positive lookahead', re.search(r'n(?=e)', 'jasmine'))
```

```
positive lookahead <re.Match object; span=(5, 6), match='n'>
```

#Substitution

```
print(re.sub(r'([\d]{4})-([\d]{4})-([\d]{4})-([\d]{4})',r'\1\2\3\4', '1111-2222-3333-4444'))
```

```
1111222233334444
```

#Compiled RegEx

```
regex = re.compile(r'([\d]{2})-([\d]{2})-([\d]{4})')
```

 # search method

```
print('compiled reg expr', regex.search('13-07-2023'))
```

 # sub method

```
print(regex.sub(r'\1.\2.\3', '13-07-2023'))
```

```
compiled reg expr <re.Match object; span=(0, 10), match='13-07-2023'>
13.07.2023
```