RegEx or Regular Expressions

A RegEx or Regular Expression, is a sequence of characters that forms a search pattern.

* RegEx can be used to check if a string contains the specified search pattern.

RegEx module

Python has a built-in module called 're', which can be used to work with Regular Expressions.

```
import re
```

RegEx in python

When you have imported the 're' module, you can start using regular expressions.

RegEx functions

The 're' module offers a set of functions that allows us to search a string for a match.

Function	Description
findall	Returns a list containing all the matches.
search	Returns a match object if there is a match anywhere in the string
split	Returns a split where the string has been split at each match.
sub	Replaces one or many matches with a string.

RegEx :- (regex101.com for practise)

* Extracting required data from the given text by identifying a common pattern.
* mainly used on huge data
* Regex can only be applied on string.

ex:- a=1234

re.findall("2", a)

↳ X error as a is int

re.findall('2', str(a))

↳ '2'

Ex:-

text = "Innomatics is one of #### the best @@@@ Data science Institute &&&& in Hyderabad"

Remove special characters #, @, & from this string

1)

```
a = text.replace('#', '')
b = a.replace('@', '')
c = b.replace('&', '')
Print(c)
```

↳

"Innomatics is one of the best Data science Institute in Hyderabad".

2)

```
special = "#@&"
s = ""
for i in text:
    if i in special:
        continue
    else:
        s += i
print(s)
```

↳ "Innomatics is one of the best Data science Institute in Hyderabad".

3)

```
text.replace("#","").replace("@","").replace("&","")
```

↳ "Innomatics is one of the best Data science Institute in Hyderabad".

4) Using RegEx

```
import re
re.sub("[#@&]", "", text)
```

↳ "Innomatics is one of the best Data science Institute in Hyderabad".

Meta characters (we will see some we commonly use)

\w - characters or alphanumeric

\W - special characters

\d - digits

\s - spaces

\b - boundary

+ (plus) - repetition (one or more occurances)

* (Asterix)

. (dot) - Any character (except newline character)

( ) (paranthesis)

[ ] (square bracket) - a set of characters

{ } (curly brackets) - Exactly the specified no. of occurences

| (Pipe) - Either or

^ (caret) - Not the listed

""" """ - triple quotation for paragraphs

if there is ' (Apostrophe) like shuba's in our text then " " single quotations will not work. so we use """ """ (double quotes)

normally we use " " for simple text.

ex:- data = """What is Python? ... Copy from online"""

if we want to print each character from above text

```
for i in range(len(data)):
    print(data[i])
```

(or)

```
a=[]
for i in data:
    a.append(i)
print(a)
```

(or)

Using RegEx

findall(pattern, text)

```
print(re.findall("\w", data))
```

if we want to print each word.

```
print(data.split())
```

Using regex # \w+ will seperate when it finds spaces

```
print(re.findall("\w+", data))
```
↳ ['what', 'is', ...]

if we want words and spaces \w*

```
print(re.findall("\w*", data))
```
↳ ['what', ' ', 'is', ...]

```
print(re.findall("\w\w", data))
```
↳ ['wh', 'at', 'is', ...]

```
print(re.findall("\w\w\w", data))
```
↳ ['wha', 'pyt', 'hon'....]

(or)

```
print(re.findall("\w{3}", data))
```
↳ ['wha', 'pyt', 'hon',...]

extract the words which starts with letter 'p' or 'P'.

1)
```
l = data.split()
l1 = []
for i in l:
    if i[0] == 'p' or i[0] == 'P':
        l1.append(i)
```

```
        else:
            continue
    print(ll)
```
↳ ['Python', 'python', 'programming', ...]

2)
```
for i in data.split():
    if i.startswith("p") or i.startswith("P"):
        print(i)
```
↳

Python
python
programming
:

using regex

```
# \b boundary between the words
# better use r(raw) for any kind of data
# here the boundary is space
print(re.findall(r"\bp\w+", data)
```
↳ ['programming', 'program', ...]

```
# if we want to use multiple conditions
# we use [ ] for multiple conditions
print(re.findall(r"\b[pP]\w+", data)
```
↳ ['python', 'Python', 'Programming', ...]

find word Python

```
print(re.findall(r"Python", data))
```

↳ ['Python', 'Python', ...]

extract the words which are starting with vowels.

```
print(re.findall(r"\b[aAeEiIoOuU]\w+", data))
```

↳ ['is', 'Executive', 'is', 'an', ...]

extract words which start with consonant.

'^' (caret) represents not the values mentioned

\s is for space

```
print(re.findall(r"\b[^aAeEiIoOuU]\w+", data))
```

↳ ['what', 'Python', 'summary', ..., 'Level', ...]

Ex:-

```
emails = "my mail id abc@gmail.com and xyz@yahoo.com
          and shuba@innomatics.in"
```

extract only email id

'@' and '.' are common for email id

```
Print (re.findall (r"\w+@\w+.\w+", emails))
```

↳ ['abc@gmail.com', 'xyz@yahoo.com', 'shuba@innomatics.in']

```
# only extract domains from emails.
# it will print only selected words that are in
  the paranthesis
print (re.findall (r"\w+@(\w+)\.\w+", emails))
```

↳ ['gmail', 'yahoo', 'innomatics']

```
# only name of email id
  print (re.findall (r"(\w+)@\w+\.\w+", emails)
```

↳ ['abc', 'xyz', 'shuba']

Ex:-

```
mobile = "my mobile number is 9160054949"
# extract only mobile number
# \d for digits then for multiple we use +

Print (re.findall (r"\d+", mobile)
```

↳ ['9160054949']

```
mobile = "my mobile number is 9160054949  12345"
```

{} we can mention the exact count inside the curly brackets.

```
print(re.findall(r"\d{10}", mobile)
    ↳ ['9160054949']

mobile = "my mobile number is 9160054949 1234567890"

# Indian numbers start with 6,7,8,9 i.e 6-9
# so first number should be 6-9 rest 9 numbers
    can be anything

print(re.findall(r"[6-9]{1}\d{9}", mobile))
    ↳ ["9160054949"]

(or)

print(re.findall(r"[6-9]{1}[0-9]{9}", mobile))
    ↳ ['9160054949']

mobile = "my mobile number is 9160054949 1234567890
    916-005-4949 or +91-9160054949 or +91-916-005-
    4949 or +919160054949"

# TO extract patterns like 916-005-4949
# "-" has meaning of range so if we want to use
  '-' better use escape character '\-'.

re.findall(r"[6-9]{1}\d\d\-\d{3}\-\d{4}", mobile)
    ↳ ['916-005-4949', '916-005-4949']
```

for multiple conditions we use | (or)

```
re.findall(r"[6-9]{1}[0-9]{9}|
                [6-9]{1}\d\d\-\d{3}\-\d{4}", mobile)
```

↓

```
['9160054949', '916-005-4949', '+91-9160054949',
 '916-005-4949', '919160054949']
```

extract 91600-54949

```
re.findall(r"[6-9]{1}\d{4}\-\d{5}", mobile)
```

↓

```
['91600-54949']
```

```
Print(re.findall(r"[6-9]{1}[0-9]{9}|[6-9]{1}\d\d
                \-\d{3}\-\d{4}|[+91]{3}\-[6-9]{1}
                [0-9]{9}|[6-9]{1}\d{4}\-\d{5}"
                , mobile)
```

↓

```
['9160054949', '916-005-4949', '+91-9160054949',
 '916-005-4949', '91600-54949', '919160054949']
```

extract +919160054949

```
re.findall(r"[+91]{3}[6-9]{1}[0-9]{9}", mobile)
```

↓

```
[+919160054949]
```

```
Print(re.findall(r"[6-9]{1}[0-9]{9}|[6-9]{1}\d\d
                \-\d{3}\-\d{4}|[+91]{3}\-[6-9]{1}
                [0-9]{9}|[6-9]{1}\d{4}\-\d{5}|
                [+91]{3}[6-9]{1}[0-9]{9}", mobile)
```

```
['9160054949','916-005-4949','+91-9160054949',
 '916-005-4949','91600-54949', '919160054949',
 +"919160054949"]
```

emails

↳ "my mail id is abc@gmail.com and xy2@yahoo.com and shuba@innomatics.in"

print only characters

```
re.findall("\w", emails)
```

↳ ['m',
'y',
:]

print only special characters

```
re.findall("\W", emails)
```

↳

[" ", " ", ... '@', ... '.', ...]

if we want to print everything we use '.'

re.findall(".", emails)

↳ ['m',
'y',
' ',
:
'n']

ip = """ Link-local ..... : fe80. ......
IPv4 Address. .... : 192.168.0.62
Subnet mask. .... : 255.255.255.0
Default Gateway... : 192.168.0.1""""

extract only IP address from the list
IP will have four sets of digits seperated by 3 dots

result = re.findall(r"\d+\.\d+\.\d+\.\d+", ip)

result

↳ ['192.168.0.62', '255.255.255.0', '192.168.0.1']

reviews = """ Good: movie was awesome
Bad: worst movie
Good: Fantastic movie
Bad: bad movie
Good: movie was awesome

Bad: worst movie

Good: movie was good""

extract only Good Reviews

it starts with Good: then can be any length

re.findall(r"Good:.*", reviews)

[alphanumeric character & special character

* as many words including spaces.

['Good: movie was awesome'
'Good: fantastic movie'
'Good: movie was awesome'
'Good: movie was good']

* match() ⟶ match() method

re.match(pattern, text)

* match gives the match object which gives the index of the searched pattern.
* match only works when the searched pattern is in 0th position.

text = "python is a general purpose language"

find 'python' using match()

re.match("python", text)

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

(starting index, ending index)

we have functions start() and End() that we can use on Match object.

result = re.match("python", text)

result.start()

↘ 0 # starting index

result.End()

↘ 6 # ending index

match does not work if the searched pattern is not in 0th position

re.match("general", text)

↘ no result

* **search()**

re.search(pattern, text)

search finds the searched pattern in the entire string. Index of the first match is returned.

```
re.search('general', text)
↳
<re.Match object; span = (12,19), match='general'>

result = re.search('general', text)
result.start()
    ↳ 12
result.end()
    ↳ 19
```

match()	search()	findall()
- The searched pattern should be at 0th position - Returns index	- The searched pattern can be anywhere in the string. - Returns the index of first match	- The searched pattern can be anywhere in the string. - Returns the searched patterns and its multiple occurances

* sub() (mainly used for multiple substitutes at a time)

re.sub(old value, new value, text)

replace "general purpose" with "object oriented"

re.sub("general purpose", "object oriented", text)

↳ "python is a object oriented language"

text = "python is a ####general purpose &&&& language @@@@"

if using replace()

text.replace("#", "").replace("&", "").replace("@", "")

↳ "python is a general purpose language"

if we use sub one condition will solve everything

re.sub(r"[#&@]", "", text)

↳ "python is a general purpose language"

* **split()**

re.split("Seperator", text)

we can use it with multiple seperators.

```
text.split("#")
```

↳

```
['Python is a', '', '', '', 'general---@@@@']
```

```
re.split("[#&@]", text)
```

↳

```
['python is a'
 ''
  ,
 ''
  ,
  :
 'general purpose',
  ,
  :
 '']
```