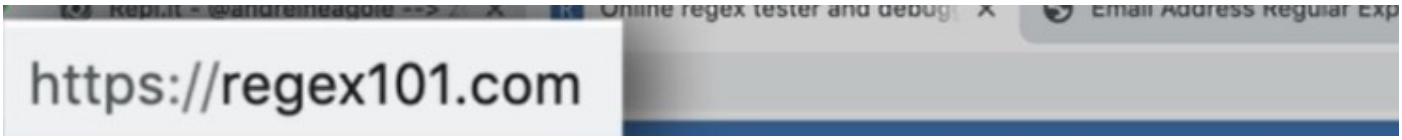
REGULAR EXPRESSIONS IN PYTHON



- 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 is kind of tool for various strings manupulation.

- Use of re.search() and re.match() –

- re.search() and re.match() both are functions of re module in python. The function searches for some substring in a string and return a match object if found, else it returns none.

re.search() vs re.match()

- Both return first match of a substring found in the string, but re.match() searches only in the first line of the string and return match object if found, else return none. But if a match of substring is found in some other line other than the first line of string (in case of a multi-line string), it returns none.

regular expression example.py > ...

```
1  import re
2  pattern = r'eggs' # r stand for row string
3  if re.match(pattern,'hokoeggseggshelloeggs'): #match function used here for searching
4      print('match found')
5  else:
6      print('match not found')
7
8  #output: match not found
```

```
import re
pattern = r'eggs' # r stand for row string
if re.match(pattern,'eggseggshelloeggs'): #match function used here for searching
    print('match found')
else:
    print('match not found')

#output: match found
```

- While re.search() searches for the whole string even if the string contains multi-lines and tries to find a match of the substring in all the lines of string.

ar expression example.py > ...

```
import re
pattern = r'eggs' # r stand for row string
if re.search(pattern,'hokoeggseggshelloeggs'): #match function used here for searching
    print('match found')
else:
    print('match not found')

#output: match found
```

- The Match object has properties and methods used to retrieve information about the search, and the result:

`.span()` returns a tuple containing the start-, and end positions of the match.
`.string` returns the string passed into the function
`.group()` returns the part of the string where there was a match

regular expression example 7.py

```python
import re
pattern='bed'
test_string = 'Early to bed early to rise'

a=re.search(pattern,test_string)
print(a.start()) #start of pattern in the string, index number

print(a.end()) #end of pattern in the string

print(a.span()) #span tell where (from,to) , the pattern exist in the string

print(a.group())  #group gives the part of string where the pattern is existing

b= re.fullmatch(pattern,test_string)
print(b) #pattern should exactly match otherwose it will give None for fullmatch func

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 10: Python Debug Cons

```
Mohd.Uzair@UzairPC MINGW64 /g/python_practice
$  env C:\\Users\\Mohd.Uzair\\AppData\\Local\\Programs\\Python\\Python38-32\\python.exe c:\\Users\\Mohd.Uzair\\.vscode\\extensions\\ms-pyt
pythonFiles\\lib\\python\\debugpy\\launcher 58395 -- "g:\\python_practice\\regular expression example 7.py"
9
12
(9, 12)
bed
None
```

FIND AND REPLACE USING REGULAR EXPRESSION

- If you want to replace a string that matches a regular expression instead of perfect match, use the **`sub ()`** of the re module.
- In re.sub(), specify a regular expression pattern in the first argument, a new string in the second argument, and a string to be processed in the third argument.

```python
'''find  and replace the pattern in string'''

import re
pattern = r'Uzair'
test_string ='Hi I am Uzair. I love my name Uzair. The meaning of uzair I don\'t know'
if re.search(pattern,test_string):

    new_string= re.sub(pattern,'chicken',test_string)
#sub func is used to replace on string to other string
print(new_string)

#here in output we will see that only 'Uzair' is replaced and not the 'uzair'.
```

```
\regular expression ex 2 .py"
Hi I am chicken. I love my name chicken. The meaning of uzair I don't know
```

You can control the number of replacements by specifying the `count` parameter:

Example

Replace the first 2 occurrences:

```
import re

txt = "The rain in Spain"
x = re.sub("\s", "9", txt, 2)
print(x)
```

The findall() Function

The `findall()` function returns a list containing all matches.

Example

Print a list of all matches:

```
import re

txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)
```

split () function

The `split ()` function returns a list where the string has been split at each match:

```
'''split function'''
import re
pattern=r"\s" #'\s' represents white spaces
test_string = "Lakshyadeep is one of the beautiful union territory"
y = re.split(pattern,test_string)
print(y)
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Mohd.Uzair@UzairPC MINGW64 /g/python_practice
$  env C:\\Users\\Mohd.Uzair\\AppData\\Local\\Programs\\Python\\Python38-32\\python.exe c:\\Users\\Mohd.Uzair\\
pythonFiles\\lib\\python\\debugpy\\launcher 50015 -- "g:\\python_practice\\regular expression example 6.py"
['Lakshyadeep', 'is', 'one', 'of', 'the', 'beautiful', 'union', 'territory']
```

- You can control the number of occurrences by specifying the **maxsplit** parameter:

```
import re

txt = "The rain in Spain"
x = re.split("\s", txt, 1)
print(x)
```

- here in code it will occure one time when it will match first time, as we have given here parameter 1 as occurrence.

META CHARACTERS

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"
()	Capture and group	

Special Sequences

A set is a set of characters inside a pair of square brackets `[]` with a special meaning:

A special sequence is a `\` followed by one of the characters in the list below, and has a special meaning:

Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"

\b	Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\Bain" r"ain\B"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"

Sets

A set is a set of characters inside a pair of square brackets `[]` with a special meaning:

Set	Description
[arn]	Returns a match where one of the specified characters (`a`, `r`, or `n`) are present
[a-n]	Returns a match for any lower case character, alphabetically between `a` and `n`
[^arn]	Returns a match for any character EXCEPT `a`, `r`, and `n`
[0123]	Returns a match where any of the specified digits (`0`, `1`, `2`, or `3`) are present
[0-9]	Returns a match for any digit between `0` and `9`
[0-5][0-9]	Returns a match for any two-digit numbers from `00` and `59`
[a-zA-Z]	Returns a match for any character alphabetically between `a` and `z`, lower case OR upper case
[+]	In sets, `+`, `*`, `|`, `()`, `$`, `{}` , dot ,has no special meaning, so `[+]` means: return a match for any `+` character in the string

CODE EXERCISE

Validate the email address submit by user.

```
'''character class'''
#suppose we want to verify that email typed is in correct format or not
#it should have @ and .com included in it

import re

regex = '^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}$'
# for custom mails use: '^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w+$'
# plus + means one or more occurence
# question mark means zero or one occurence
# star means zero or more occurence

def check(email):

    if(re.search(regex,email)):
        print("Valid Email")

    else:
        print("Invalid Email")

email = input('enter your email address: ')
check(email)
```

EXAMPLE: CARRAT AND DOLLAR META CHARACTER

```
'''caret ^ and dollar $ meta characters'''
# caret signifies that starting of string
# dollar signifies the ending of string

import re
pattern = r'^gee.y$'
string='geeky'
if re.search(pattern,string):
    print('match found')
else:
    print('match not found')
```

TESTING IN PYTHON

https://www.geeksforgeeks.org/unit-testing-python-unittest/