Python Regex

Steven Wang

August 17, 2016

```python
# Author : Steven Wang  Date: 20160816

import re

# r"string" : "\" means just a backslash, no "escape"

if re.search("ape","The ape was at the apex"):
    print("There is an ape")

allApes = re.findall("ape.","The ape was at the apex")
# '.' will replace any single chararter or single space
for i in allApes:
    print(i)

theStr = "The ape was at the apex"

for i in re.finditer("ape.", theStr):
    locTuple = i.span()
# return a Tuple that contain its location in the string
    print(locTuple)
# >>> (4, 8)
# >>> (19, 23)
    print(theStr[locTuple[0]:locTuple[1]])
# >>> ape
# >>> apex

animalStr = "Cat rat mat pat"
allAnimals = re.findall("[crmfp]at",animalStr)
# match certain character
for i in allAnimals:
    print(i)

# >>>rat
# >>>mat
# >>>pat

```

```
someAnimals = re.findall("[c-mC-M]at", animalStr)
# match a series of range
for i in someAnimals:
    print(i)
# >>>Cat
# >>>mat

someAnimals = re.findall("[^Cr]at", animalStr)
# match any character BUT 'C' and 'r'
for i in someAnimals:
    print(i)
# >>>mat
# >>>pat

# Replace strings
owlFood = "rat cat mat pat"
regex = re.compile("[cr]at")
# create a object that contains a certain pattern
owlFood = regex.sub("owl",owlFood)
print(owlFood)
# >>>owl owl mat pat

# Backslash problem
randStr = "Here is \\stuff"
print("Find \\stuff :", re.search("\\stuff", randStr))
# >>>Find \stuff : None
print("Find \\stuff :", re.search("\\\\stuff", randStr))
# >>>Find \stuff : <_sre.SRE_Match object; span=(8, 14), match
   ='\\stuff'>
print("Find \\stuff :", re.search(r"\\stuff", randStr))
# "rawstrings", this would not treat '\' as special character

# Dot problem
randStr = "F.B.I. I.R.S. CIA"
print("Matches:", len(re.findall(".\..\..\.", randStr)))
# >>>Matches: 2

# Whitespace problem
randStr = '''This is a long
string that goes
on for many lines
'''
print(randStr)
regex = re.compile("\n")
randStr = regex.sub(" ", randStr)
print(randStr)

```

```
# >>>This is a long string that goes on for many lines

# \b : Backspace
# \f : Form Feed
# \r : Carriage Return
# \t : Tab
# \v : Vertical Tab

# \r\n : This would show up a lot in windows and html

# Matching numbers
# \d : [0-9]    These 2 are the same, will match any number
# \D : [^0-9]   These 2 are the same, will match any character
   but a number

randStr = "12345"
print("Matches :", len(re.findall("\d", randStr)))
# >>>Matches : 5
print("Matches :", len(re.findall("\d{5}", randStr)))
# match a 5 digit number
# >>>Matches : 1
print("Matches :", len(re.findall("\d{2}", randStr)))
# >>>Matches : 2
# 1st : 12 ; 2nd : 34

# Matching within a range between 5 and 5 digits
numStr = "123 12345 123456 1234567"
print("Matches :", len(re.findall("\d{5,7}", numStr)))
# either 5, 6 or 7 digits
# >>>Matches : 3

# Matching any single letter or  mber
# \w : [a-zA-Z0-9_]
# \W : [^a-zA-Z0-9_]

phNum = "857-555-3158"
if re.search("\w{3}-\w{3}-\w{4}", phNum):
    print("It is a phone number")
# >>>It is a phone number

# \s : [\f\n\r\t\v]
# will match all kinds of space
# \S : [^\f\n\r\t\v]

if re.search("\w{2,20}\s\w{2,20}", "Steven Wang"):
    print("It is valid")

```

```
# + : will match one or more characters
print("Matches :", len(re.findall("a+", "a as ape bug")))
# >>>Matches : 3

# Matching an email addresss
email_list = "ap@apple.com, mail@gmail.com, w@b.com ce@.com"
print("Email Matches :", len(re.findall("[\w._%+-]{2,20}@[\w
    .-]{2,20}\.[A-Za-z]{2,3}", email_list)))
# no need for '\' when in "[]"

# summary
# Did you find a match
# if re.search("REGEX", yourString)

# Get list of matches
# print("Matches :", len(re.findall("REGEX", yourString)))

# Get a pattern object
# regex = re.compile("REGEX")

# Substitute the match
# yourString = regex.sub("substitution", yourString)

# [ ]    : Match what is in the brackets
# [^ ]   : Match anything not in the brackets
# .      : Match any 1 character or space
# +      : Match 1 or more of what proceeds
# \n     : Newline
# \d     : Any 1 number
# \D     : Anything but a number
# \w     : Same as [a-zA-Z0-9_]
# \W     : Same as [^a-zA-Z0-9_]
# \s     : Same as [\f\n\r\t\v]
# \S     : Same as [^\f\n\r\t\v]
# {5}    : Match 5 of what proceeds the curly brackets
# {5,7}  : Match values that are between 5 and 7 in length

randStr = "cat cats"
regex = re.compile("[cat]+s?")
# "+" will match one or more trailing characters
matches = re.findall(regex, randStr)
# The question mark is called a quantifier
# colou?r matches both colour and color
# Nov(ember)? matches Nov and November
for i in matches:
    print(i)

```

```
randStr = "cat cats cat's"
regex = re.compile("[cat]+['s]*")
# will match all three

# eg.
regex = re.compile("[\w\s]+[\r]?\n")
# \s space
# find a line

# greedy and lazy matching
randStr = "<name>Steven</name><name>Wang</name>"
# we want to get things between the tags
`regex = re.compile("<name>.*</name>")
# However, "*" is what we called "greedy" and it search for the
    largest scope; It grab the biggest match possible

`regex = re.compile("<name>.*?</name>")
# add "?" to make it lazy
# +?
# {5}?

# greedy : grab the biggest match possible
# lazy : grab the smallest match possible

# Word boundaries
# \b
randStr = "ape at the apex"
# we want only the first match
regex = re.compile(r"\bape\b")

# String boundaries
# ^ : Beginning of the string
# $ : End of the string
randStr = "Match everying up to @"
regex = re.compile(r"^.*[^@]")
matches = re.findall(regex, randStr)

randStr = "@ Get this string"
regex = re.compile(e"[^\s].*$")

randStr = '''Ape is big
Turtle is slow
Steven is good
'''
# multiline string

regex = re.compile(r"(?m)^.*?\s")
```

```
# (?m) target each line in the multiline strings

# Substrings
randStr = "My anumer is 888-000-1111"
regex = re.compile((r"888-(.*)")
# only get 000-1111

regex = re.compile(r"888-(.*)-(.*)")
# 555 1111

# Summary
# Did you find a match
# if re.search("REGEX", yourString)

# Get list of matches
# print("Matches :", len(re.findall("REGEX", yourString)))

# Get a pattern object
# regex = re.compile("REGEX")

# Substitute the match
# yourString = regex.sub("substitution", yourString)

# [ ]      : Match what is in the brackets
# [^ ]     : Match anything not in the brackets
# ( )      : Return surrounded submatch
# .          : Match any 1 character or space
# +          : Match 1 or more of what proceeds
# ?          : Match 0 or 1
# *          : Match 0 or More
# *?         : Lazy match the smallest match
# \ b        : Word boundary
# ^          : Beginning of String
#  $         : End of String
# \ n        : Newline
# \ d        : Any 1 number
# \ D        : Anything but a number
# \ w        : Same as [a-zA-Z0-9_]
# \ W        : Same as [^a-zA-Z0-9_]
# \ s        : Same as [\f\n\r\t\v]
# \ S        : Same as [^\f\n\r\t\v]
# {5}      : Match 5 of what proceeds the curly brackets
# {5,7} : Match values that are between 5 and 7 in length
# ($m)     : Allow ^ on multiline string

# ---------- Back References ----------
# A back reference allows you to to reuse the expression
```

```
# that proceeds it

randStr = "<a href='#'><b>the ling</b></a>"

# eg.
regex = re.compile(r"<b>(.*?)</b>")
newStr = re.sub(regex, r"\1", randStr)
# "\1" back reference
print(newStr)

# eg 2.
randStr = "123-456-7890"
regex = re.compile(r"([\d]{3})-([\d]{3}-[\d]{4})")
newStr = re.sub(regex, r"(\1)\2", randStr)
# \1 : ([\d]{3})
# \2 : ([\d]{3}-[\d]{4})
print(randStr)

# eg 3.
randStr = "https://www.youtube.com http://www.google.com"
# goal
# <a href='https://www.youtube.com'>www.youtube.com</a>
# <a href='http://www.google.com'>www.google.com</a>
regex = re.compile(r"(https?://[\w.]+"))
# ? : "s" may or may not be there
newStr = re.sub(regex, r"<a href='\1'>\2</a>", randStr)

# Look ahead
# (?=expression)
randStr = "one two three four"
regex = re.compile(r"\w+(?=\b)")
# (?=\b) go not return word boundaries
# eg. (?=,); (?=.)

# Look behind
# (?<=expression)
randStr = "1. Bread 2. Apples 3. Lettuce"
regex = re.compile(r"(?<=\d.\s)\w+")
# (?<=\d.\s) : look for it but do not want return it

randStr"<h1>This is the title</h1> <h1>Another title</h1>"
regex = re.compile(r"(?<=<h1>).+?(?=</h1>)")
# .+? : lazy

# negative look head/behind
# look for text that does not match the pattern

```

```
# (?!expression) : Negative Look Ahead
# (?<!expression) : Negative Look Behind

randStr = "8 Apples $3, 1 Bread $1, 1 Orange $2"
regex = re.compile(r"(?<!\$)\d+")
matches = re.findall(regex, randStr)
print(len(matches))

##################################################
# summary
# [ ]      : Match what is in the brackets
# [^ ]     : Match anything not in the brackets
# ( )      : Return surrounded submatch
# .          : Match any 1 character or space
# +          : Match 1 or more of what proceeds
# ?          : Match 0 or 1
# *          : Match 0 or More
# *?         : Lazy match the smallest match
# \ b        : Word boundary
# ^          : Beginning of String
#  $         : End of String
# \ n        : Newline
# \ d        : Any 1 number
# \ D        : Anything but a number
# \ w        : Same as [a-zA-Z0-9_]
# \ W        : Same as [^a-zA-Z0-9_]
# \ s        : Same as [\f\n\r\t\v]
# \ S        : Same as [^\f\n\r\t\v]
# {5}      : Match 5 of what proceeds the curly brackets
# {5,7} : Match values that are between 5 and 7 in length
# ($m)     : Allow ^ on multiline string

# Use a back reference to substitute what is between the
# bold tags and eliminate the bold tags
# re.sub(r"<b>(.*?)</b>", r"\1", randStr)

# Use a look ahead to find all characters of 1 or more
# with a word boundary, but don't return the word
# boundary
# re.findall(r"\w+(?=\b)", randStr)

# Use a look behind to find words starting with a number,
# period and space, but only return the word that follows
# re.findall(r"(?<=\d.\s)\w+", randStr)

# Use a negative look behind to only return numbers without
# a $ in front of them
```

```
# re.findall(r"(?<!\$)\d+", randStr)

#########################################################
# | : or
randStr = "1. Dog 2. Cat 3. Turtle"
regex = re.compile(r"\d\.\s(Dog|Cat)")
matches = re.findall(regex, randStr)
for i in matches:
    print(i)

# Group

birthday = input("Enter your birthday(mm-dd-yyyy):")
bdRegex = re.search(r"(\d{1,2})-(\d{1,2})-(\d{4})", birthday)
print("You were born on", bdRegex.group())
print("Month", bdRegex.group(1))
print("Day", bdRegex.group(2))
print("Year", bdRegex.group(3))

# match object
match = re.search(r"\d{2}", "The chicken is 13 months old.")
print("Match : ", match.group())
print("Span : ", match.span())
print("Start : ", match.start())
print("End : ", match.end())

# Name groups
randStr = "December 21 1999"
regex = r"^(?P<month>\w+)\s(?P<day>\d+)\s(?P<year>\d+)"
print("Span : ", match.group('month'))
print("Start : ", match.group('day'))
print("End : ", match.group('year'))
```

python regex