Regex

Wednesday, September 19, 2018 4:15 PM

https://www.guru99.com/python-regular-expressions-complete-tutorial.html

Write code that would extract hashtags from the following tweet:

```
tweet = "@nltk Text analysis is awesome! #regex #pandas #python"
extract = [t for t in tweet.split() if t.startswith("#")]
print(extract)
```

```
['#regex', '#pandas', '#python']
```

- Callouts are more than just tokens beginning with '@'

  @UN_Spokesperson @katyperry @coursera

- Match *something* after '@'
  - Alphabets
  - Numbers
  - Special symbols like '_'

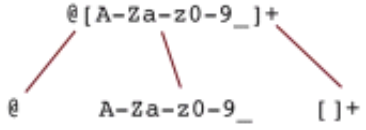
@[A-Za-z0-9_]+

```
>>> text10 = '"Ethics are built right into the ideals and objectives of the
United Nations" #UNSG @ NY Society for Ethical Culture bit.ly/2guVelr @UN
@UN_Women'
>>> text11 = text10.split(' ')
>>> [w for w in text11 if w.startswith('@')]
['@', '@UN', '@UN_Women']
```

Import regular expressions first!

```
>>> import re
>>> [w for w in text11 if re.search('@[A-Za-z0-9_]+', w)]
```

Parsing the callout regular expression



- starts with @
- followed by any alphabet (upper or lower case), digit, or underscore
- that repeats at least once, but any number of times

Meta-characters: Character matches

. : wildcard, matches a single character
^ : start of a string
$: end of a string
[ ] : matches one of the set of characters within [ ]
[a-z] : matches one of the range of characters a, b, ..., z
[^abc] : matches a character that is not a, b, or, c
a|b : matches either a or b, where a and b are strings
( ) : Scoping for operators
\ : Escape character for special characters (\t, \n, \b)

```
.       - Any Character Except New Line
\d      - Digit (0-9)
\D      - Not a Digit (0-9)
\w      - Word Character (a-z, A-Z, 0-9, _)
\W      - Not a Word Character
\s      - Whitespace (space, tab, newline)
\S      - Not Whitespace (space, tab, newline)
```

Meta-characters: Character symbols

\b : Matches word boundary
\d : Any digit, equivalent to [0-9]
\D : Any non-digit, equivalent to [^0-9]
\s : Any whitespace, equivalent to [\t\n\r\f\v]
\S : Any non-whitespace, equivalent to [^ \t\n\r\f\v]
\w : Alphanumeric character, equivalent to [a-zA-Z0-9_]
\W : Non-alphanumeric, equivalent to [^a-zA-Z0-9_]

Meta-characters: Repetitions

* : matches zero or more occurrences
\+ : matches one or more occurrences
? : matches zero or one occurrences
{n} : exactly n repetitions, n≥ 0
{n,} : at least n repetitions
{,n} : at most n repetitions
{m,n} : at least m and at most n repetitions

```
>>> [w for w in text11 if re.search('@[A-Za-z0-9_]+', w)]
['@UN', '@UN_Women']

>>> [w for w in text11 if re.search('@\w+', w)]
['@UN', '@UN_Women']
```

- **Finding specific characters**

```
>>> text12 = 'ouagadougou'

>>> re.findall(r'[aeiou]', text12)
['o', 'u', 'a', 'a', 'o', 'u', 'o', 'u']

>>> re.findall(r'[^aeiou]', text12)
['g', 'd', 'g']
```

Regular Expression for Dates (contd.)

```
>>> dateStr = '23-10-2002\n23/10/2002\n23/10/02\n10/23/2002\n23 Oct 2002\n23
October 2002\nOct 23, 2002\nOctober 23, 2002\n'

>>> re.findall(r'\d{2}[/-]\d{2}[/-]\d{4}', dateStr)
['23-10-2002', '23/10/2002', '10/23/2002']

>>> re.findall(r'\d{2}[/-]\d{2}[/-]\d{2,4}', dateStr)
['23-10-2002', '23/10/2002', '23/10/02', '10/23/2002']

>>> re.findall(r'\d{1,2}[/-]\d{1,2}[/-]\d{2,4}', dateStr)
['23-10-2002', '23/10/2002', '23/10/02', '10/23/2002']
```

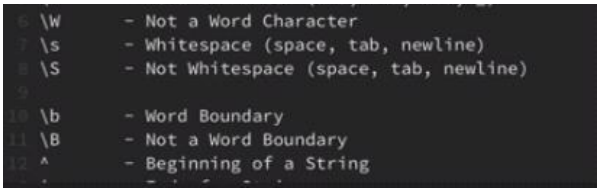
23-10-2002
23/10/2002
23/10/02
10/23/2002

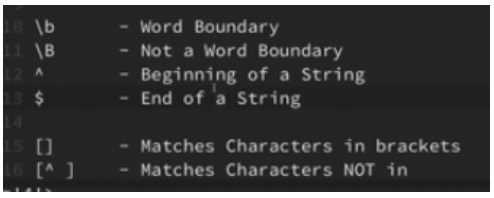
Regex for Dates (contd.)

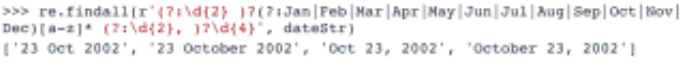
23 October 2002
Oct 23, 2002
October 23, 2002

```
>>> re.findall(r'\d{2} (Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)
\d{4}', dateStr)
['Oct']
```


```
>>> re.findall(r'\d{2} (Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)
\d{4}', dateStr)
['Oct']
```







. - Any Character Except New Line

https://www.youtube.com/watch?v=K8L6KVGG-7o

```
import os
os.chdir('D:/Data Science/POC/Regex')
import re

print('\tHello')
print(r'\tHello')

'''
print('\tHello')
      Hello

print(r'\tHello')
\tHello
'''

text_to_search = '''
abcdefghijklmnopqurtuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
Ha HaHa
MetaCharacters (Need to be escaped):
. ^ $ * + ? { } [ ] \ | ( )
coreyms.com
321-555-4321
123.555.1234
123*555*1234
800-555-1234
900-555-1234
Mr. Schafer
Mr Smith
Ms Davis
Mrs. Robinson
Mr. T
cat
mat
pat
bat
'''

sentence = 'Start a sentence and then bring it to an end'

pattern = re.compile(r'abc')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)
##<_sre.SRE_Match object; span=(1, 4), match='abc'>

###IMPORTANT  =====>MetaCharacters (Need to be
escaped):
##. ^ $ * + ? { } [ ] \ | ( )
## Ex... \.   \$  if we are trying to search these characters in
the data
pattern = re.compile(r'coreyms\.com')
pattern = re.compile(r'.')
## all digits
```

\d - Digit (0-9)

\D - Not a Digit (0-9)

\w - Word Character (a-z, A-Z, 0-9, _)

\W - Not a Word Character

\s - Whitespace (space, tab, newline)

\S - Not Whitespace (space, tab, newline)

\b - Word Boundary

\B - Not a Word Boundary

^ - Beginning of a String

$ - End of a String

[] - Matches Characters in brackets

[^] - Matches Characters NOT in brackets

| - Either Or

( ) - Group

Quantifiers:

* - 0 or More

+ - 1 or More

? - 0 or One

{3} - Exact Number

{3,4} - Range of Numbers (Minimum, Maximum)

Sample Regexs

[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+

From <https://github.com/CoreyMSchafer/code_snippets/blob/master/Python -Regular-Expressions/snippets.txt>

```
pattern = re.compile('\d')
##excluding all digits
pattern = re.compile('\D')

##include words a-z A-Z 0-9 and _
pattern = re.compile('\w')

##space /tab and newline
pattern = re.compile('\s')

##Not space /tab and newline

pattern = re.compile('\S')


##word boundary
pattern = re.compile(r'\bHa')

##without word boundary
pattern = re.compile(r'\BHa')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)


##Beginning of string ^

pattern = re.compile(r'^Start')
matches= pattern.finditer(sentence)
for match in matches:
   print(match)

##End of string
pattern = re.compile(r'end$')
matches= pattern.finditer(sentence)
for match in matches:
   print(match)

##Phone no matching
pattern = re.compile(r'\d\d\d.\d\d\d.\d\d\d')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)

###Read data file and do regex for phone no identification
pattern = re.compile(r'\d\d\d.\d\d\d.\d\d\d')
with open('data.txt','r',encoding ='utf-8') as f:
   contents = f.read()
   matches = pattern.finditer(contents)
   for match in matches:
      print(match)


##Include character sets by using brackets.
pattern = re.compile(r'\d\d\d[-.]\d\d\d[-.]\d\d\d')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)

##any number with 8 or 9 with 00
pattern = re.compile(r'[89]00[-.]\d\d\d[-.]\d\d\d')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)



##find all characters ending with at except for bat and pat
pattern = re.compile(r'[^bp]at')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)

##add a quantifier to check telephone
```

```
pattern = re.compile(r'\d{3}[-.]\d{3}[-.]\d{4}')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)


##optional check  use ? for ex Mr and Mr. if we want to
identify both
pattern = re.compile(r'Mr\.?')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)

##few more combo

pattern = re.compile(r'Mr\.?\s[A-Z]\w+')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)


##or check  use () and usi |
pattern = re.compile(r'M(r|s|rs)\.?\s[A-Z]\w+')
matches= pattern.finditer(text_to_search)
for match in matches:
   print(match)

###EMAIL matching
 emails = '''
anshumanmah@gmail.com
rajdps@hotmail.com
indie@gov.in
axy.abc@gmail.com
corey-anderson@gmail.com
corey123@gmail.com
corey_Anderson@gmail.com
'''
pattern = re.compile(r'[A-Za-z.]+@[a-zA-Z]+\.com')
matches= pattern.finditer(emails)
for match in matches:
   print(match)

pattern = re.compile(r'[A-Za-z.0-9-_]+@[a-zA-Z]+\.(com|in)')
matches= pattern.finditer(emails)
for match in matches:
   print(match)


##for all type of emails
pattern = re.compile(r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-
zA-Z0-9-.]+')
matches= pattern.finditer(emails)
for match in matches:
   print(match)

##() means group
###urls
urls = '''
https://www.google.com
http://coreyms.com
https://youtube.com
https://www.nasa.gov
'''

pattern = re.compile(r'https?://(www\.)?(\w+)(\.\w+)')

subbed_urls = pattern.sub(r'\2\3', urls)

print(subbed_urls)



##findall only returns strings no other info
```

```
##################
# matches = pattern.finditer(urls)

# for match in matches:
#     print(match.group(3))

sentence = 'Start a sentence and then bring it to an end'

pattern = re.compile(r'start', re.I)

matches = pattern.search(sentence)

print(matches)
```