Regular Expressions

re module

python

by

Jedsada Luengaramsuk

Date: 4 February 2021
End:7 February 2021

Table of content

Table of Contents

What is regular expression?

- module that allow you to specify and replace a pattern of text to search for.

Quotes

“Knowing [regular expressions] can mean the difference between solving a problem in 3 steps and solving it in 3,000 steps. When you’re a nerd, you forget that the problems you solve with a couple keystrokes can take other people days of tedious, error-prone work to slog through." **Cory Doctorow**

Regular expression matching process

1. import re
2. Create a Regex object with regexObject = re.compile(r’{character class}’)

- Use raw string
- Use regex special symbol for optional matching

3. use regex object to search/match

- match_object/mo = regexObject.search()

Example

```
#Jedsada Luengaramsuk (jedsada-io)
#27 January 2021

#import re module to start using regular expression
import re

#Step 1 create a regex object

#Passing string representation of regular expression to re.compile()
#This is the string representation for 3nums+dash+3nums+dash+4nums
phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')



#Step 2 Matching Regex Objects
#search() method searches the string that match with regex object
mo = phineNumRegex.search('My number is 415-555-4242')
print('Phone number found: ' + mo.group())

#Output
#Phone number found: 415-555-4242
```

Regex character classes

Table 7-1: Shorthand Codes for Common Character Classes

Shorthand character class	Represents
\d	Any numeric digit from 0 to 9.
\D	Any character that is *not* a numeric digit from 0 to 9.
\w	Any letter, numeric digit, or the underscore character. (Think of this as matching "word" characters.)
\W	Any character that is *not* a letter, numeric digit, or the underscore character.
\s	Any space, tab, or newline character. (Think of this as matching "space" characters.)
\S	Any character that is *not* a space, tab, or newline.

\d = num

\D != num

\w = letter, num, _ and word

\W != letter,num, _

\s = space,tab,newline (space)

\S != space,tab,newline (space)

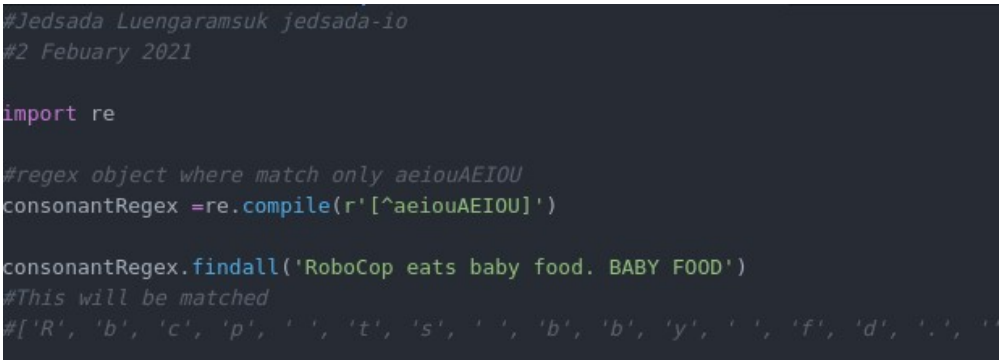
Character classes example

```python
#Jedsada Luengaramsuk jedsada-io
#2 Febuary 2021

import re

#Regex object of one or more num , space then one or more word
xmasRegex = re.compile(r'\d+\s\w+')


xmasRegex.findall('12 drummers, 11 pipers, 10 lords, 9 ladies, 8 m

#This will show all because it's fit the condition above
```

xmasRegex.py

Making Your Own Character Classes

- when you want to match character but those default digit are too board
- using brackets []
- [aeiouAEIOU] < this will match only this
- [a-zA-Z0-9] is also work



```
#Jedsada Luengaramsuk jedsada-io
#2 Febuary 2021

import re

#regex object where match only aeiouAEIOU
consonantRegex =re.compile(r'[^aeiouAEIOU]')

consonantRegex.findall('RoboCop eats baby food. BABY FOOD')
#This will be matched
#['R', 'b', 'c', 'p', ' ', 't', 's', ' ', 'b', 'b', 'y', ' ', 'f', 'd', '.', '
```

Regex object optional symbol

Regex Symbols summary

- The ? matches zero or one of the preceding group.
- The * matches zero or more of the preceding group.
- The + matches one or more of the preceding group.
- The {n} matches exactly n of the preceding group.
- The {n,} matches n or more of the preceding group.
- The {,m} matches 0 to m of the preceding group.
- The {n,m} matches at least n and at most m of the preceding group.
- {n,m}? or *? or +? performs a nongreedy match of the preceding group.
- ^spam means the string must begin with spam.
- spam$ means the string must end with spam.

- The . matches any character, except newline characters.
- \d , \w , and \s match a digit, word, or space character, respectively.
- \D , \W , and \S match anything except a digit, word, or space character, respectively.
- [abc] matches any character between the brackets (such as a, b, or c).
- [^abc] matches any character that isn’t between the brackets.

Pipe | matching multiple groups

| = pipe

| can be use anywhere you want to match.

r'Batman|Tina Fey' = match either 'Batman' or 'Tina Fey'

Pipe example

BatmanorTinafey.py

```python
#Jedsada Luengaramsuk (jedsada.io)
#Date: 28 January 2021

#Regex object match either Batman or Tina Fey
heroRegex = re.compile(r'Batman|Tina Fey)

#matching object 1
mo1 = heroRegex.search('Batman and Tina Fey.')

mo.group()
#This will show 'Batman' because it see Batman before Tina BatmanorTinafey

mo2 = heroRegex.search('Tina Fey and Batman.')
mo2.group()
#This will show Tina Fey because it come before Batman
```

? > Optional Matching

- ? flags the group.

- (text)? means that the pattern inside is an optional group

? Example

This regex matches both 'Batwoman' and 'Batman'

bat(wo)?man.py

```
#Jedsada Luengaramsuk (jedsada.io)
#Date: 28 January 2021

#Regex object match either Batman or Tina Fey
heroRegex = re.compile(r'Batman|Tina Fey)

#matching object 1
mo1 = heroRegex.search('Batman and Tina Fey.')

mo.group()
#This will show 'Batman' because it see Batman before Tina BatmanorTinafey

mo2 = heroRegex.search('Tina Fey and Batman.')
mo2.group()
#This will show Tina Fey because it come before Batman
```

* > matching Zero or More(Star)

- * (star or asterisk)

- match zero or more

- (text)* group that precedes the star can occur any number of times in the text.

* Example

StarbatRegex.py

```python
#Jedsada LUengaramsuk jedsada-io
#29 January 2021

import re

#* mean match zero or more
batRegex = re.compile(r'Bat(wo)*man')

mo1 = batRegex.search('The Adventures of Batman')
mo1.group()
#Batman

mo2 = batRegex.search('The Adventures of Batwoman')
mo2.group()
#Batwoman

mo3 = batRegex.search('The Adventures of Batwowowowoman')
#Batwowowwowoman
#This will match because of * match one or more.
```

+ (Plus) matching one or more

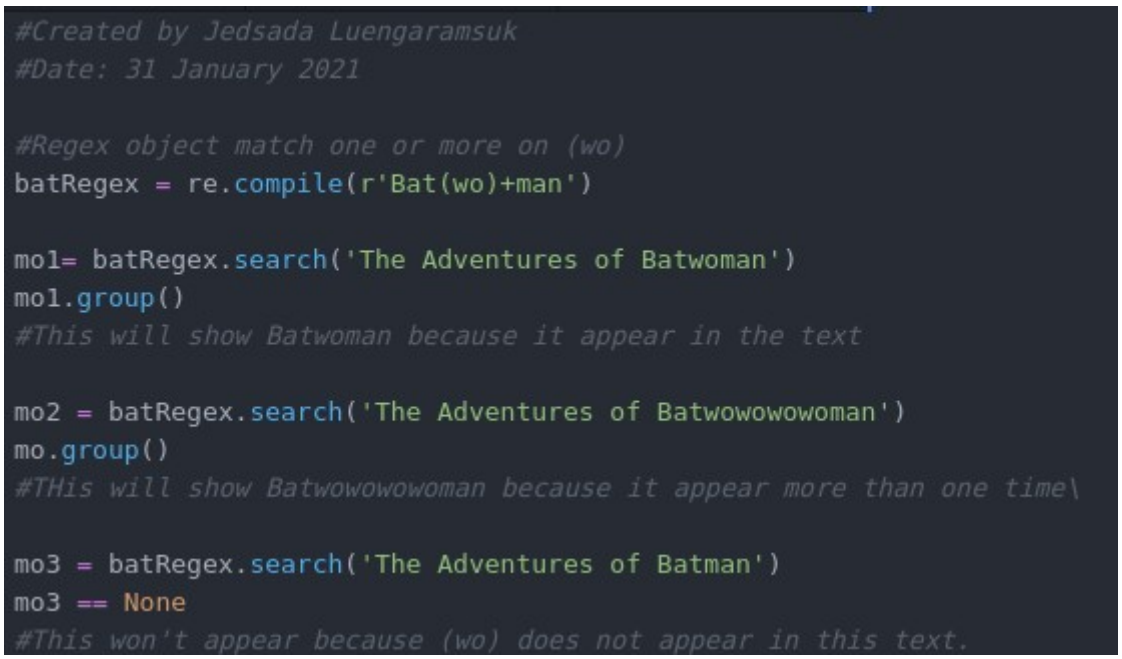
- +

- match one or more

- (text)+The group need to appear in matching string

+ Example

(wo)+.py



```
#Created by Jedsada Luengaramsuk
#Date: 31 January 2021

#Regex object match one or more on (wo)
batRegex = re.compile(r'Bat(wo)+man')

mo1= batRegex.search('The Adventures of Batwoman')
mo1.group()
#This will show Batwoman because it appear in the text

mo2 = batRegex.search('The Adventures of Batwowowowoman')
mo.group()
#THis will show Batwowowowoman because it appear more than one time\

mo3 = batRegex.search('The Adventures of Batman')
mo3 == None
#This won't appear because (wo) does not appear in this text.
```

{} Matching Specific Repetitions with Curly Brackets

- (optional object){number of time it need to appear}

{} Curly brackets exmaple

haha.py

```python
#Jedsada Luengaramsuk jedsada-io
import re


#Create regex object. Search for hahaha
haRegex = re.compile(r'(ha){3}')

mo1=haRegex.search('HaHaHa')
mo1.group()
#Show HaHaHa (Perfect matched)

mo2 = haRegex.search('Ha')
mo2 = None
#Return True

#Summary
#()is for the text {} is for the number of times it must appear
```

? Greedy and Nongreedy Matching

- ? declear nongreedy match

- Greedy match : match the most of that optional regex object (default)

Greedy vs Non-greedy matching example

greedy.py

```python
#Jedsada Luengaramsuk jedsada-io
#1 Febuary 2021

#Regex object that find 3Ha or 5Ha
greedyHaRegex = re.compile(r'(Ha){3,5}'')
mo1 = greedyHaRegex.search('HaHaHaHaHa')
mo1.group()
#This will show HaHaHaHaHa because it proitize 5 instead of 3



#Search for nongreedy(Smaller) instead
mo2 = nongreedyHaRegex.search('HaHaHaHaHa')
mo2.group()
#HaHaHa This one prioritize the smaller one
```

^ and $ The Caret and Dollar Sign Characters

- ^ caret at the start off a regex
  - a match must occur at the beginning of the searched text
- $ dollar sign at the end of regex to incicate the string must end with this regex pattern
- ^ and $ together to indicate the start and the end

Caret and Dollor sign example

CaretandDollar.py

```python
#Jedsada Luengaramsuk jedsada-io
#2 Febuary 2021

import re

#regex object start with Hello
beginsWithHello = re.compile(r'^Hello')
beginsWithHello.search('Hello world!')
#This will matched the string

#However this will not matched
beginsWithHello.search('He said hello.') == None




#This is the regex object that end with number
endsWithNumber = re.compile(r'\d$')

#This will matched
endsWithNumber.search('Your number is 42')

endsWithNumber.search('Your number is forty two.') == None
#This will return true


#Regex that both begin and end with one or more num Characters
#With one or more num chars
wholeStringIsNum = re.compile(r'^\d+$')
wholeStringIsNum.search('1234567890')
#This one will match

#This one won't because it's contain chars
wholeStringIsNum.search('12345xyz67890') == None

#This one won't because it contain space
wholeStringIsNum.search('12 34567890') == None
```

The Wildcard Character

- . (or dot)

- called wildcard

- matched any character except for a newline.

```
#Jedsada LUENGARAMSUK jedsada-io
#Date: 2 Febuary 2021

import re

atRegex = re.compile(r'.at')
atRegex.findall('The cat in the hat sat on the flat mat.')
#All matched because it's not newline chars
```

.* Matching everything with Dot-Star

-(.*)

- anything

- anything except a newline

```
#Jedsada Luengaramsuk jedsada-io
#2 Feb 2021
import re

#Regex object where matched First Name then everything except new line. Also the name as  last name
nameRegex = re.compile(r'First Name: (.*) Last Name: (.*)'')

mo = nameRegex.search('First Name: Jedsada Last Name: LUENGARAMSUK')

mo.group(1)
#Jedsada

mo.group(2)
#LUENGARAMSUK
```

.*? Match everything in a non-greedy way

everythingGreedyvsNongreedy.py

```python
#Jedsada LUEGNARAMSUK jedsada-io
#2 Febuary 2021

import re

#Match everything in a non greedy way
nongreedyRegex = re,compile(r'<.*?>'')

mo = nongreedyRegex.search('<To serve man> for dinner.>')

mo.group()
#'<To serve man> for dinner.>'
```

.* ,re.DOTALLMatching Newlines with the Dot Character

- match all character including newline

newlineRegex.py

```python
#Jedsada LUengaramsuk jedsada-io
#Date: 2 Febuary 2021

import re


#This is everything excluding newline
noNewLineRegex = re.compile('.*')
noNewLineRegex.search('Serve the public trust. \nProtect the innocent. \mUphold the law.').group()

#'Serve the public trust'




#This is everything including newline
#re.DOTALL
noNewLineRegex = re,compile('.*', re.DOTALL)
newlineRegex.search('Serve the public trust.\nProtect the innocent.
\nUphold the law.').group()

#'Serve the public trust.\nProtect the innocent.\nUphold the law.'
```

Regex search/sub method

Pattern matching methods

regexObject.search()

- search one string that match with regex object

```python
#Jedsada Luengaramsuk (jedsada-io)
#Date: 28 January 2021

import re

#Create regex objectg
phoneNumRegex = re.compile(r'(\d\d\d)-(\d\d\d-\d\d\d\d)')
mo = phoneNumRegex.search('My number is 415-555-4242')


mo.group(1)
#This will show group one (\d\d\d)

mo.group(2)
#This will show group 2 (\d\d\d\d)

mo.group(0)
#This will show no group(\d\d\d)-(\d\d\d-\d\d\d\d)

mo.group()
#This will show no group the same as mo.group(0)

mo.groups()
#This will retrive all the groups ('group1','group2')


areaCode, mainNumber = mo.groups()
#This will name areaCode group1, name mainNumber group2
print(areaCode)
#This will print group1
print(mainNumber)
#This will print group2
```

regexObject.group()

- grab matching text from just one group.

- adding () will create grups in the regex

- e.g. (\d\d\d)-(\d\d\d-\d\d\d\d)

```python
#Jedsada Luengaramsuk (jedsada-io)
#27 January 2021

#import re module to start using regular expression
import re

#Step 1 create a regex object

#Passing string representation of regular expression to re.compile()
#This is the string representation for 3nums+dash+3nums+dash+4nums
phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')



#Step 2 Matching Regex Objects
#search() method searches the string that match with regex object
mo = phineNumRegex.search('My number is 415-555-4242')
print('Phone number found: ' + mo.group())

#Output
#Phone number found: 415-555-4242
```

regexObject.findall()

- findall() method

- return every match in the searched strin

- search() only return one (first match)

findall example

```python
#Jedsada Luengaramsuk jedsada-io
#1 Febuary 2021

phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
mo = phoneNumRegex.search('Cell: 415-555-9999 Work: 212-555-0000')
mo.group()
#'415-555-9999' will be shown because it's a first matched


#This is when I use findall instead
phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
phoneNumRegex.findall('Cell: 415-555-9999 Work: 212-555-0000')
#['415-555-9999', '212-555-0000'] This will post all the searches
```

regex.SUB() Substituting Strings with the sub() Method

- sub() method

- passed two arguments

- first argument > a string to replace any matches

- second argument > a string for the regular expression

.sub() example

nameRegexsub.py

```python
#Jedsada Luengaramsuk jedsada-io
#2 Febuary 2021

import re

namesRegex = re.compile(r'Agent \w+')
namesRegex.sub('CENSORED', 'Agent Alice gave the secret documents to Agent Bob.')
#'CENSORED gave the secret documents to CENSORED.
#Change Alice to CENSORED
```

Second argument of re.compile()

re.VERBOSE -Managine Complex Regexes

- compilcated text patterns require long , regex.

- use verbose mode instead

- re.VERBOSE as a second argument to re.compile()

Normal re.compile()

```
phoneRegex = re.compile(r'((\d{3}|\(\d{3}\))?(\s|-|\.)?\d{3}(\s|-|\.)\d{4} (\s*(ext|x|ext.)\s*\
d{2,5})?)')
```

re.compile()+ re.VERBOSE

```
phoneRegex = re.compile(r'''(
  (\d{3}|\(\d{3}\))? # area code
  (\s|-|\.)?  # separator
  \d{3}      # first 3 digits
  (\s|-|\.)   # separator
  \d{4}       # last 4 digits
  (\s*(ext|x|ext.)\s*\d{2,5})?  # extension
  )''', re.VERBOSE)
```

- Allow you to have a new comment with multiple newline

re.I , re.IGNORECASE : Case-Insensitive example

robocop.py

```python
#Jedsada Luengaramsuk jedsada-io
#2 Febuary 2021

import re

#Match all case

robocop = re.compile(r'robocop',re.I)
robocop.search('RoboCop is part man, part machine, all cop.').group()
#RoboCop

robocop.search('ROBOCOP protects the innocent.').group()
#ROBOCOP

robocop.search('Al, why does your programming book talk about robocop so much?').group()
#robocop
```

re.dotall Matching Newlines

- match all character including newline

```python
#Jedsada LUengaramsuk jedsada-io
#Date: 2 Febuary 2021

import re


#This is everything excluding newline
noNewLineRegex = re.compile('.*')
noNewLineRegex.search('Serve the public trust. \nProtect the innocent. \mUphold the law.').group()

#'Serve the public trust'




#This is everything including newline
#re.DOTALL
noNewLineRegex = re,compile('.*', re.DOTALL)
newlineRegex.search('Serve the public trust.\nProtect the innocent.
\nUphold the law.').group()

#'Serve the public trust.\nProtect the innocent.\nUphold the law.'
```