Regular Expressions

A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are widely used in UNIX world.

The module re provides full support for Perl-like regular expressions in Python. The re module raises the exception re.error if an error occurs while compiling or using a regular expression.

Match, Search, Findall, Substitute

`re.match(pattern, string, flags=0)` This function attempts to match RE pattern to string with optional flags.

`re.search`(*pattern*, *string*, *flags=0*) Scan through *string* looking for the first location where the regular expression *pattern* produces a match, and return a corresponding *match object*. Return `None` if no position in the string matches the pattern; note that this is different from finding a zero-length match at some point in the string.

`re.findall`(*pattern*, *string*, *flags=0*) Return all non-overlapping matches of *pattern* in *string*, as a list of strings. The *string* is scanned left-to-right, and matches are returned in the order found.

`re.sub`(*pattern*, *repl*, *string*, *count=0*, *flags=0*) Return the string obtained by replacing the leftmost non-overlapping occurrences of *pattern* in *string* by the replacement *repl*. If the pattern isn't found, *string* is returned unchanged. *repl* can be a string or a function.

Exercise 1

```
import re
line = "My Dogs are smarter than your Cats"
results = re.findall('[dc]\S+s', line, re.M|re.I)
if results:
    print ("results: ", results)
else:
    print ("Nothing found")
```

Exercise 2

```
import re
phone = "(631) 615-4215 # School Nova phone number"
num = re.sub('#.*$', '', phone)
print ("phone num: ", num)
num = re.sub('\D', '', phone)
print ("phone num: ", num)
```

regularexpressions

Anchors

^	Start of string
\A	Start of string
$	End of string
\Z	End of string
\b	Word boundary
\B	Not word boundary
\<	Start of word
\>	End of word

Character Classes

\c	Control character
\s	White space
\S	Not white space
\d	Digit
\D	Not digit
\w	Word
\W	Not word
\x	Hexadecimal digit
\O	Octal digit

POSIX

[:upper:]	Upper case letters
[:lower:]	Lower case letters
[:alpha:]	All letters
[:alnum:]	Digits and letters
[:digit:]	Digits
[:xdigit:]	Hexadecimal digits
[:punct:]	Punctuation
[:blank:]	Space and tab
[:space:]	Blank characters
[:cntrl:]	Control characters
[:graph:]	Printed characters
[:print:]	Printed characters and spaces
[:word:]	Digits, letters and underscore

Assertions

?=	Lookahead assertion
?!	Negative lookahead
?<=	Lookbehind assertion
?!= or ?<!	Negative lookbehind
?>	Once-only Subexpression
?()	Condition [if then]
?()|	Condition [if then else]
?#	Comment

Available free from
www.ILoveJackDaniels.com

Quantifiers

*	0 or more
+	1 or more
?	0 or 1
{3}	Exactly 3
{3,}	3 or more
{3,5}	3, 4 or 5

Quantifier Modifiers

"x" below represents a quantifier

x?	Ungreedy version of "x"

Escape Character

\	Escape Character

Metacharacters (must be escaped)

^	[	.
$	{	*
(	\	+
)	|	?
<	>	

Special Characters

\n	New line
\r	Carriage return
\t	Tab
\v	Vertical tab
\f	Form feed
\xxx	Octal character xxx
\xhh	Hex character hh

Groups and Ranges

.	Any character except new line (\n)
(a|b)	a or b
(...)	Group
(?:...)	Passive Group
[abc]	Range (a or b or c)
[^abc]	Not a or b or c
[a-q]	Letter between a and q
[A-Q]	Upper case letter between A and Q
[0-7]	Digit between 0 and 7
\n	nth group/subpattern

Note: Ranges are inclusive.

Pattern Modifiers

g	Global match
i	Case-insensitive
m	Multiple lines
s	Treat string as single line
x	Allow comments and white space in pattern
e	Evaluate replacement
U	Ungreedy pattern

String Replacement (Backreferences)

$n	nth non-passive group
$2	"xyz" in /^(abc(xyz))$/
$1	"xyz" in /^(?:abc)(xyz)$/
$`	Before matched string
$'	After matched string
$+	Last matched string
$&	Entire matched string

Sample Patterns

Pattern	*Will Match*
([A-Za-z0-9-]+)	Letters, numbers and hyphens
(\d{1,2}\/\d{1,2}\/\d{4})	Date (e.g. 21/3/2006)
([^\s]+(?=\.(jpg|gif|png))\.\2)	jpg, gif or png image
(^[1-9]{1}$\|^[1-4]{1}[0-9]{1}$|^50$)	Any number from 1 to 50 inclusive
(#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})?)	Valid hexadecimal colour code
((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,15})	String with at least one upper case letter, one lower case letter, and one digit (useful for passwords).
(\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})	Email addresses
(\<(/?[^\>]+)\>)	HTML Tags

Note: These patterns are intended for reference purposes and have not been extensively tested. Please use with caution and test thoroughly before use.

Homework

Search through your hard drive for email addresses using the sample program below. Adjust the program to fit your environment (root directory, file extension etc.):

```
import os, fnmatch, re, sys
found = set()

pattern = re.compile(r'\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]
{2,4}\b',re.I)

for root, dir, files in os.walk("/Users/serge/projects"):
    #print (root)

    for file in fnmatch.filter(files, "*.txt"):
        file = os.path.join(root, file)
        #print(file)
        if os.path.isfile(file) and os.access(file, os.R_OK):

            try:

                  for line in open(file, 'r'):

                        found.update(pattern.findall(line))

            except UnicodeDecodeError:

                  print ("Program could not open file ", file)

for myMatch in found:

    print (myMatch)
```

Additional Documentation can be found at https://docs.python.org/3/library/re.html