

Regular Expressions

Chapter II



Python for Informatics: Exploring Information
www.py4inf.com



open.michigan

Unless otherwise noted, the content of this course material is licensed under a Creative Commons Attribution 3.0 License.
<http://creativecommons.org/licenses/by/3.0/>.

Copyright 2011, Charles Severance



Regular Expressions

In computing, a regular expression, also referred to as "regex" or "regexp", provides a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters. A regular expression is written in a formal language that can be interpreted by a regular expression processor.

http://en.wikipedia.org/wiki/Regular_expression

Regular Expressions

Really clever "wild card" expressions for matching and parsing strings.

http://en.wikipedia.org/wiki/Regular_expression



Really smart "Find" or "Search"

Understanding Regular Expressions

- Very powerful and quite cryptic
- Fun once you get to use them
- Regular expressions are a language unto themselves
- A language of "marker characters" - programming with characters
- It is kind of an "old school" language - compact

Regular Expression Quick Guide

^	Matches the beginning of a line
\$	Matches the end of the line
.	Matches any character
\s	Matches whitespace
\S	Matches any non-whitespace character
*	Repeats a character zero or more times
*?	Repeats a character zero or more times (non-greedy)
+	Repeats a character one or more times
+?	Repeats a character one or more times (non-greedy)
[aeiou]	Matches a single character in the listed set
[^XYZ]	Matches a single character not in the listed set
[a-z0-9]	The set of characters can include a range
(Indicates where string extraction is to start
)	Indicates where string extraction is to end

The Regular Expression Module

- Before you can use regular expressions in your program, you must import the library using "import re"
- You can use re.search() to see if a string matches a regular expression similar to using the find() method for strings
- You can use re.match() extract portions of a string that match your regular expression similar to a combination of find() and slicing:
var[5:10]

Using re.search() like find()

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if line.find('From:') >= 0:
        print line
```

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('From:', line) :
        print line
```

Using re.search() like startswith()

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if line.startswith('From:') :
        print line
```

```
import re
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('^From:', line) :
        print line
```

We fine-tune what is matched by adding special characters to the string

Wild-Card Characters

- The dot character matches any character
- If you add the asterisk character, the character is "any number of times"

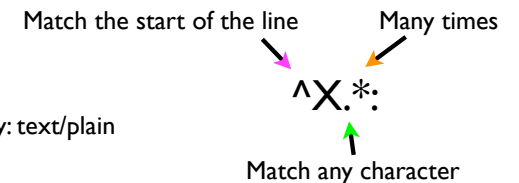
```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```

`^X.*:`

Wild-Card Characters

- The dot character matches any character
- If you add the asterisk character, the character is "any number of times"

```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```



Wild-Card Characters

- The dot character matches any character
- If you add the asterisk character, the character is "any number of times"

X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain

Match the start of the line Many times

`^X.*:`

Match any character

Fine-Tuning Your Match

- Depending on how "clean" your data is and the purpose of your application, you may want to narrow your match down a bit

X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent

XPlane is behind schedule: two weeks

Match the start of the line Many times

`^X.*:`

Match any character

Fine-Tuning Your Match

- Depending on how "clean" your data is and the purpose of your application, you may want to narrow your match down a bit

X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent

XPlane is behind schedule: two weeks

Match the start of the line One or more times

`^X-|S+:`

Match any non-whitespace character

Matching and Extracting Data

- The `re.search()` returns a True/False depending on whether the string matches the regular expression
- If we actually want the matching strings to be extracted, we use `re.findall()`

`[0-9]+`

One or more digits

```
>>> import re
>>> x = 'My 2 favorite numbers are 19 and 42'
>>> y = re.findall('[0-9]+',x)
>>> print y
['2', '19', '42']
```

Matching and Extracting Data

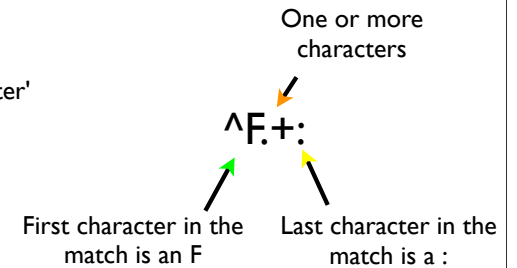
- When we use `re.findall()` it returns a list of zero or more sub-strings that match the regular expression

```
>>> import re
>>> x = 'My 2 favorite numbers are 19 and 42'
>>> y = re.findall('[0-9]+',x)
>>> print y
['2', '19', '42']
>>> y = re.findall('[AEIOU]+',x)
>>> print y
[]
```

Warning: Greedy Matching

- The repeat characters (`*` and `+`) push outward (greedy) to match the largest possible string

```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+:', x)
>>> print y
['From: Using the :']
```



Why not 'From:'?

Fine Tuning String Extraction

- You can refine the match for `re.findall()` and separately determine which portion of the match that is to be extracted using parenthesis

From `stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008`

```
>>> y = re.findall('\S+@\S+',x)
>>> print y
['stephen.marquard@uct.ac.za']
>>> y = re.findall('^From:.*? (\S+@\S+)',x)
>>> print y
['stephen.marquard@uct.ac.za']
```

At least one non-whitespace character

Fine Tuning String Extraction

- Parenthesis are not part of the match - but they tell where to start and stop what string to extract

From `stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008`

```
>>> y = re.findall('\S+@\S+',x)
>>> print y
['stephen.marquard@uct.ac.za']
>>> y = re.findall('^From:.*? (\S+@\S+)',x)
>>> print y
['stephen.marquard@uct.ac.za']
```

At least one non-whitespace character

The Double Split

- Sometimes we split a line one way and then grab one of the pieces of the line and split that piece again

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
'uct.ac.za'

The Double Split

- Sometimes we split a line one way and then grab one of the pieces of the line and split that piece again

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
'uct.ac.za'

The Regex Version

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('@([^\s]*)', lin)
print y
['uct.ac.za']
```

'@([^\s]*)'

Look through the string until you find an at-sign

The Regex Version

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('@([^\s]*)', lin)
print y
['uct.ac.za']
```

'@([^\s]*)'

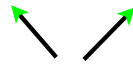
Match non-blank character Match many of them

The Regex Version

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('@([^\s]*)',lin)
print y
['uct.ac.za']
```

'@([^\s]*)'



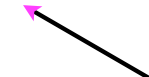
Extract the non-blank characters

Even Cooler Regex Version

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('^From .*@([^\s]*)',lin)
print y
['uct.ac.za']
```

'^From .*@([^\s]*)'



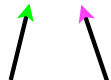
Starting at the beginning of the line, look for the string 'From'

Even Cooler Regex Version

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('^From .*@([^\s]*)',lin)
print y
['uct.ac.za']
```

'^From .*@([^\s]*)'



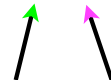
Skip a bunch of characters, looking for an at-sign

Even Cooler Regex Version

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('^From .*@([^\s]*)',lin)
print y
['uct.ac.za']
```

'^From .*@([^\s]*)'




Skip a bunch of characters, looking for an at-sign

Even Cooler Regex Version

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('^From .*@([ ^ ]*)',lin)
print y
['uct.ac.za']
```

'^From .*@([^]*)'


Start 'extracting'

Even Cooler Regex Version

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('^From .*@([ ^ ]*)',lin)
print y
['uct.ac.za']
```

'^From .*@([^]*)'



Match non-blank character Match many of them

Even Cooler Regex Version

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('^From .*@([ ^ ]*)',lin)
print y
['uct.ac.za']
```

'^From .*@([^]*)'


Stop 'extracting'

Spam Confidence

```
import re
hand = open('mbox-short.txt')
numlist = list()
for line in hand:
    line = line.rstrip()
    stuff = re.findall('^X-DSPAM-Confidence: ([0-9.]*)', line)
    if len(stuff) != 1 : continue
    num = float(stuff[0])
    numlist.append(num)

print 'Maximum:', max(numlist)
```

python ds.py
Maximum: 0.9907

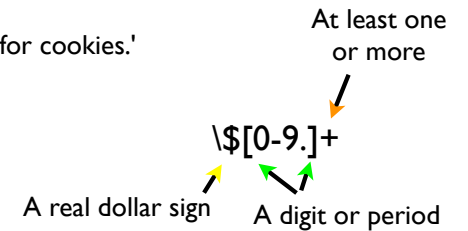
Regular Expression Quick Guide

<code>^</code>	Matches the beginning of a line
<code>\$</code>	Matches the end of the line
<code>.</code>	Matches any character
<code>\s</code>	Matches whitespace
<code>\S</code>	Matches any non-whitespace character
<code>*</code>	Repeats a character zero or more times
<code>*?</code>	Repeats a character zero or more times (non-greedy)
<code>+</code>	Repeats a character one or more times
<code>+?</code>	Repeats a character one or more times (non-greedy)
<code>[aeiou]</code>	Matches a single character in the listed set
<code>[^XYZ]</code>	Matches a single character not in the listed set
<code>[a-z0-9]</code>	The set of characters can include a range
<code>(</code>	Indicates where string extraction is to start
<code>)</code>	Indicates where string extraction is to end

Escape Character

- If you want a special regular expression character to just behave normally (most of the time) you prefix it with `'\'`

```
>>> import re
>>> x = 'We just received $10.00 for cookies.'
>>> y = re.findall('\$[0-9.]+',x)
>>> print y
['$10.00']
```



Summary

- Regular expressions are a cryptic but powerful language for matching strings and extracting elements from those strings
- Regular expressions have special characters that indicate intent