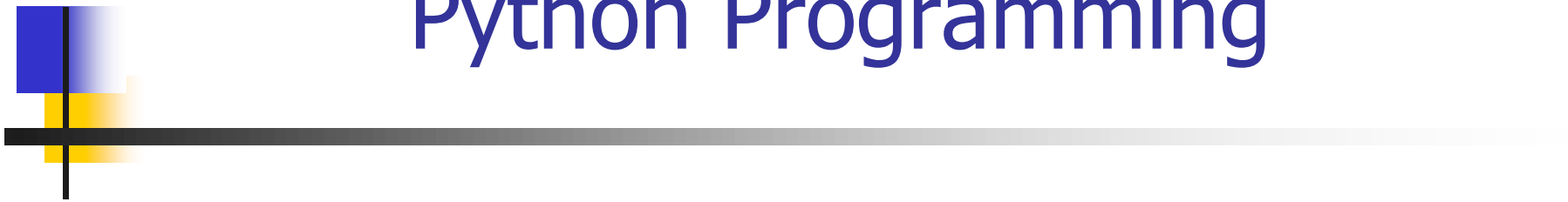


Python Programming

IOE 373 Lecture 16

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

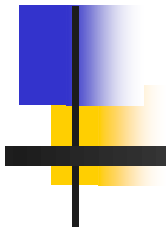
Regular Expressions

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 understand 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)
`?`	Repeats a character zero or one time
`[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.findall() to 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)
```

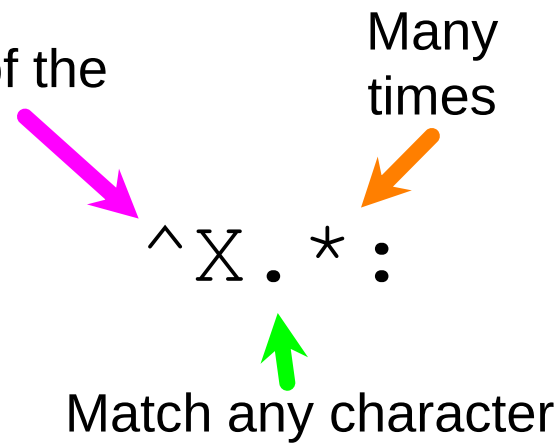
We fine-tune what is matched by adding special characters to the string

Wild-Card Characters

The dot character matches any character

If you add the asterisk character, the character is “any number of times”

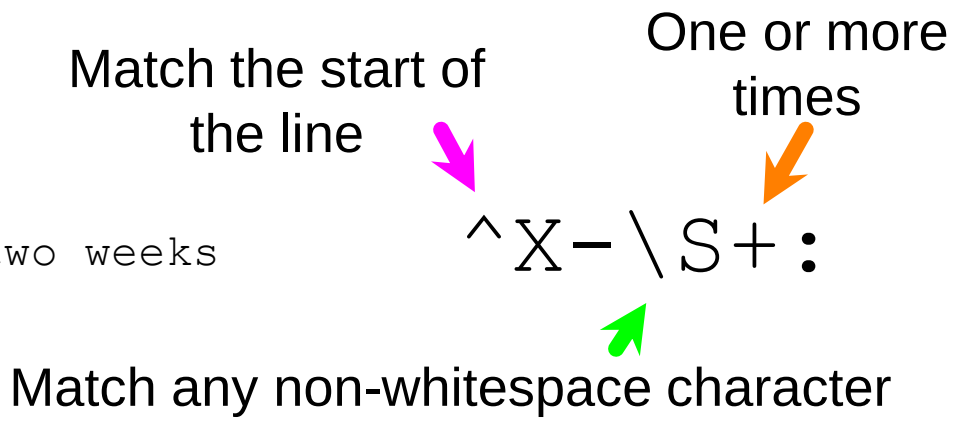
```
X-Sieve: CMU Sieve 2.3
X- DSPAM-Result: Innocent
XDSPAM-Confidence: 0.8475
X  Content-Type-Message-Body: text/plain
```



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
X-: Very Short
X-Plane is behind schedule: two weeks
```



Matching and Extracting Data

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

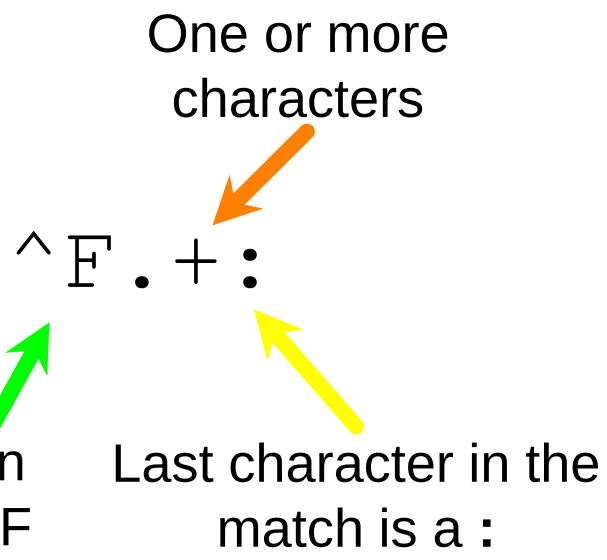
When we use re.findall(), it returns a list of zero or more sub-strings that match the regular expression

```
>>> 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 in both directions (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:' ?

```
x = 'From: Using the From: character'
y = re.findall('From:', x)
print(y)
```

```
['From:', 'From:']
```

Non-Greedy Matching

Not all regular expression repeat codes are greedy!
If you add a ? character, the + and * chill out a bit...

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

One or more characters but not greedy

```
^F.+?:
```

First character in the match is an F

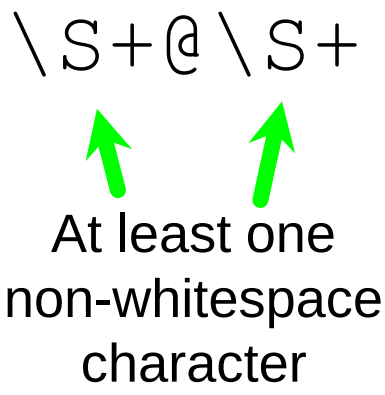
Last character in the match is a :

Fine-Tuning String Extraction

You can refine the match for re.findall() and separately determine which portion of the match is to be extracted by using parentheses

```
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']
```



Fine-Tuning String Extraction

Parentheses 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']
```

```
^From (\S+@\S+)
```


```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print(atpos)
21
>>> sppos = data.find(' ',atpos)
>>> print(sppos)
31
>>> host = data[atpos+1 : sppos]
>>> print(host)
uct.ac.za
```

Extracting a host name - using find and string slicing

The Double Split Pattern

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('@([^ ]*)',lin)
print(y)

['uct.ac.za']
```

```
'@([^ ]*)'
```

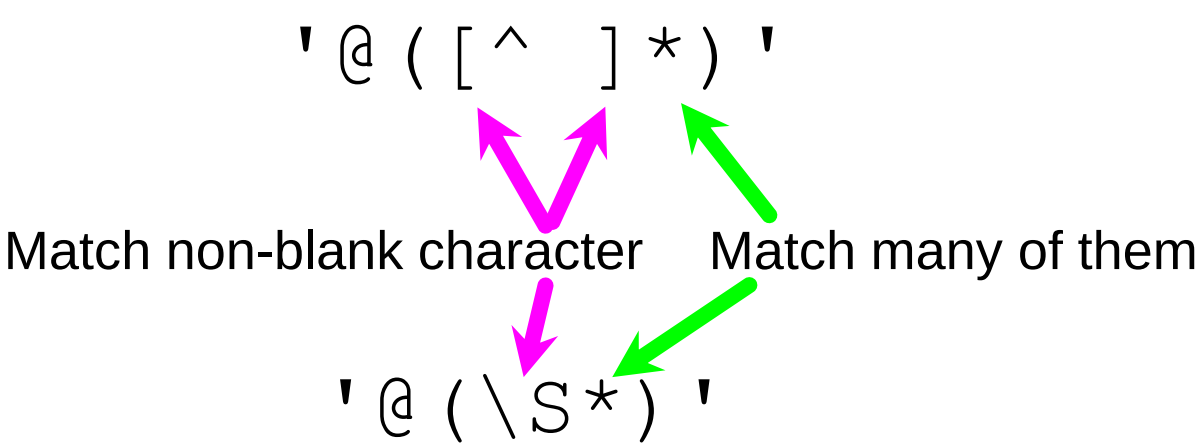
Look through the string until you find an at sign

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('@([^ ]*)',lin)
print(y)

['uct.ac.za']
```

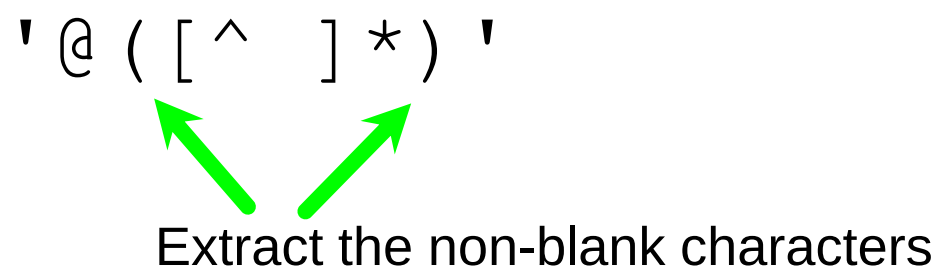


The Regex Version

```
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('@([^ ]*)',lin)
print(y)

['uct.ac.za']
```

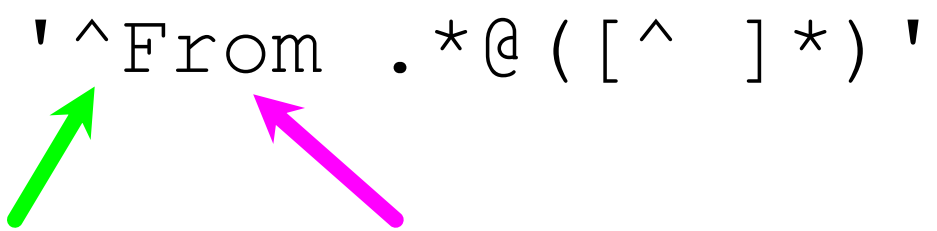


Even Cooler Regex Version

```
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']
```



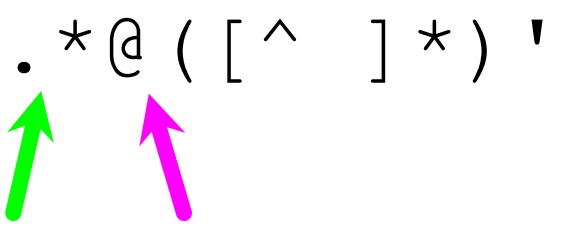
Starting at the beginning of the line, look for the string 'From '

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 .*@([^ ]*)',lin)
print(y)

['uct.ac.za']
```



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

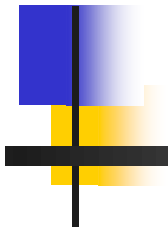
```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
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

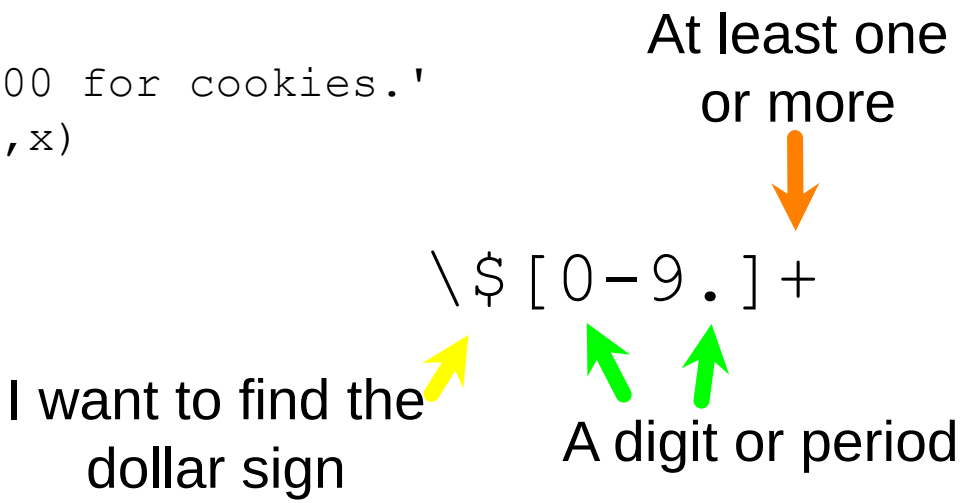
```
import re
hand = open('mbox-short.txt')
numlist = list()
for line in hand:
    line = line.rstrip()
    stuff = re.findall('^X-DSPAM-Confidence: ([0-9.]+)', line)
    if len(stuff) != 1 :  continue
    print(stuff)
    num = float(stuff[0])
    numlist.append(num)
print('Maximum:', max(numlist))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

Maximum: 0.9907

Escape Character

If you want to find a special character that is also a regular expression character (e.g. $) prefix it with '\'

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



If you use $ in a regular expression that means match the end of line…

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