Contents

Using Python to Access Web Data

University of Michigan
Charles Severance

March 31, 2016

Part I

Regular Expresion

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

`Import re`	import library *re*
`re.search(patt, text)`	see if a pattern matches is found in text (return TRUE/FALSE)
`re.findall(patt, text)`	extract pattern if found in text (return a list with expression matching pattern)

Regular expression is a concise and flexible means for matching string in a text.

0.1 *re* library and functions

0.1.1 Examples

```
if re.search('^From:', line)
        print line
```

If the expression 'From:' is found at the beginning of a line (^), return **TRUE** and print the line.

```
if re.search('^From:', line)
        print line
% TRUE if 'From' is found at the begining of the line '^'
```

```
if re.search('^X.*:', line)
        print line
% TRUE if 'X' followed by any number of char (.) finishing with ':'
```

```
x = 'My 2 favorite numbers are 19 and 42'
y = re.findall('[0-9]+', x) % any number of digits
%Return ['2', '19', '42']
```

```
x = 'My 2 favorite numbers are 19 and 42'
y = re.findall('[AEIOU]+', x)  % 1 or more capital voyel
% Return []
```

0.1.2 Greedy Matching

The repeats (* and +) tried to match the largest possible string (greedy).

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

For *Non-greedy* match:

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

0.1.3 Fine Tuning string extracting

```
x = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall('\S+@\S+', x)
print y
%Return ['stephen.marquard@uct.ac.za']
```

- Find a ligne starting with 'From', followed by a blank and match what's in '( )'

```
x = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
y = re.findall(^From (\S+@\S+)', x)
print y
%Return ['stephen.marquard@uct.ac.za']
```

- Find domain name

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

'$\hat{}$' in [ ] means NOT ('[$\hat{}$] not white space character
' * ' match many no white space character

- If you want a special regular expression character to behave normally, use '\':

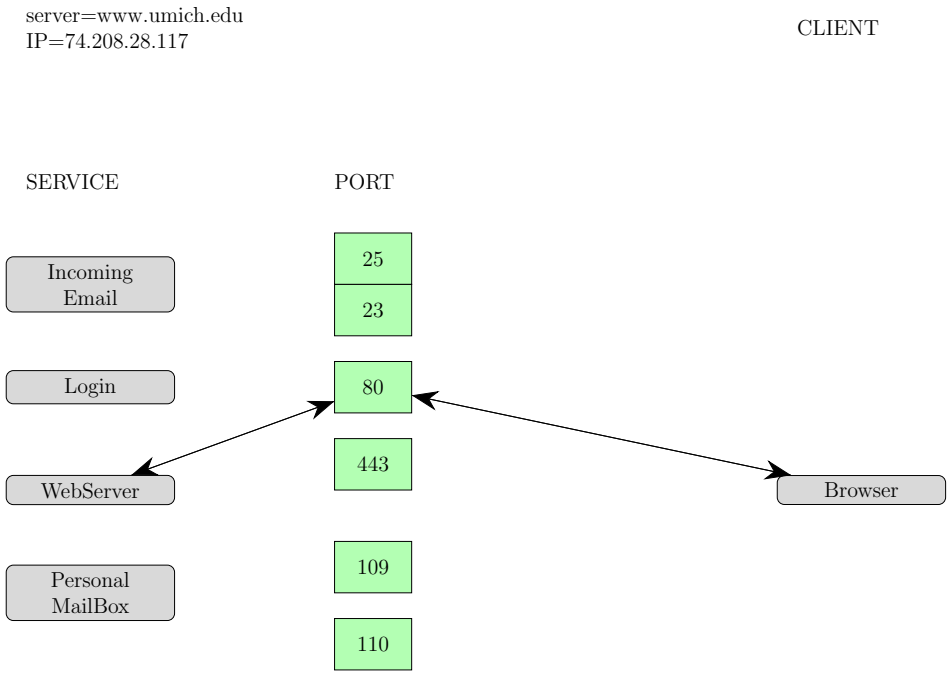
```
\$[0-9.]+  % \$=real dollar
           % a digit or a '.' at least one or more
```

Part II

Networked Program

0.2 Sockets in Python

Python built-in support for TCP sockets




```
import socket
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) %create a socket
mysock.connect( ('www.py4inf.com', 80) ) %connect socket to server
% HOST=www.py4inf.com and PORT=80
```

URL htpp:// www.dr-chuck.com/ page.htm
what protocol to use host document

0.3 getting data from the server

0.3.1 Making a HTTP Request: the old way

- Connect to the server (www.dr-chuck.com)
- request a document: GET http://www.dr-chuck.com/page1.html

```
import socket
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) %create a socket
mysock.connect( ('www.py4inf.com', 80) ) %connect socket to server
mysock.send('GET http://www.py4inf.com/code/romeo.txt HTTP/1.0\n\n'
while True:
        data = mysock.recv(512) %512 chars
        if( len(data) < 1 ): %end of file
                break
        print data
mysock.close()
```

0.3.2 Making a HTTP Request: *urllib*

```
import urllib
fhand = urllib.urlopen( 'http://www.py4inf.com/code/romeo.txt' )
for line in fhand:
        print line.strip()
```

0.3.3 BeautifulSoup

BeautifulSoup is a library that enables efficient way of conducting web scraping/crawling.

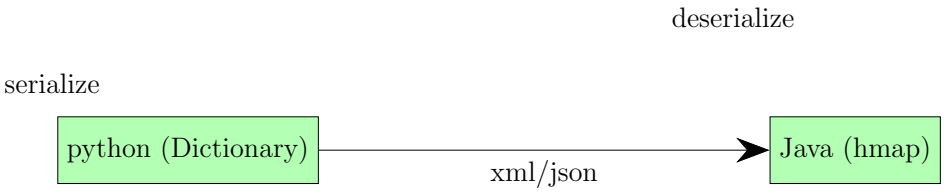
```
import urllib
from BeautifulSoup import *
url = raw.input('Enter— ') %user to enter url
htm = urllib.urlopen( url ).read() %read all
soup = BeautifulSoup(html) %the entire webpage becomes an object
%retrieve a list of the anchor tags
%Each tag is like a dictionary of HTML attributes
tags = soup( 'a' )
for tag in tags:
        print tag.get( 'href', None)
```

Part III

Data on the Web

They are 2 wire formats use for application to exchange data:

- json (javascript object notation)
- xml (extensible Markup language)



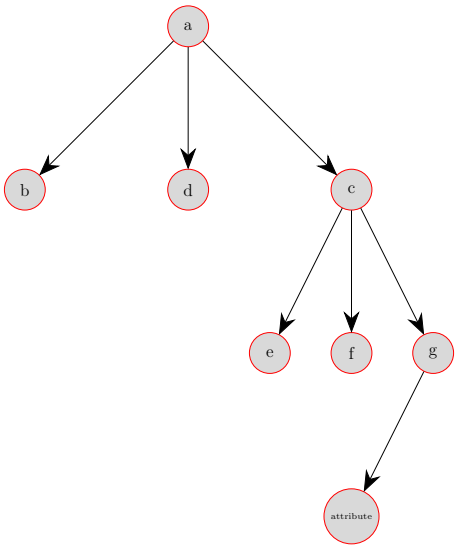
0.4 XML: extensible Markup

XML elements or nodes:

```
<people>
        <person>
                <name>Chuck</name>
                <phone>303 4456</phone>
        </person>
        <person>
                <name>Noah</name>
                <phone>303 4456</phone>
        </person>
</people>
```

- start tag: <name>
- single element : <name>Chuck</name>
- Complex element: <person>.....</person>
- attribute: <phone **type=”intl”**>1 734 303 4456 </phone>
- self closing tag: <email hide =”yes”/>
- end tag: <name>

0.4.1 XML as a Tree



0.4.2 Parsing XML

```
import xml.etree.ElementTree as ET
data = '''
        <people>
                <name>Chuck</name>
                <phone type="intl"> 1 734 303 4456</phone>
                <email hide="yes"/>
        </person>  '''

tree = ET.fromstring(data)
print 'Name:', tree.find('name').text %Chuck
print 'Attr:', tree.find('email').get('hide') %yes
```

```
import xml.etree.ElementTree as ET
data = '''
<stuff>
        <users>
                <user x="2">
                        <id> 001 </id>
                        <name>Chuck</name>
                </user>
                <user x="7">
                        <id> 009 </id>
                        <name> Brent </name>
                </user>
        <users>
</stuff> '''

stuff = ET.fromstring(data)
lst = stuff.findall( 'users/user' )
print 'User count:', len(lst)
for item in lst
        print 'Name', item.find('name').text
        print 'Id', item.find('id').text
        print 'Attribute', item.get('x').text
```

0.5 JSON

Syntax: JSON represents data as nested lists and dictionaries.

0.5.1 Example I

```
import json

data = ''' {  %data is a disctionary here '{'
                        "name": "Chuck",  % key=name, value = string
                        "phone": {
                                "type": "intl",
                                "number": "1+ 734 303 4456",
                        },
                        "email": {
                                "hide": "yes",
                        }
                } '''

info = json.loads(data) % deserialize/parsing (info is a disctionary)
print "Name:", info["name"]  % Return Chuck
print "Hide:", info["email"]["hide"]  % Return yes
```

0.5.2 Example II

```
import json

input = ''' [  %data is an array '['
                        { "id": "001", %this is a dictionary
                                "x": "2",
                                "name": "Chuck",
                        },
                        { "id": "009",
                                "x": "7",
                                "name": "Chuck",
                        },
                ]'''

info = json.loads(input) % deserialize/parsing (info is a disctionary)
print "User count:", len(info)
for item in info:
        print 'Name:', item['name']
        print 'Id:', item['id']
        print 'Attribute:', item['x']
```

Part IV

Accessing APIs in python

API defined set of rules to interface with an application (Google Geocoding API, Twitter API)

0.6 Google GeoCoding API: get long/lat of a location

We start with a url:
http://maps.googleapis.com/maps/api/geocode/json?sensor=false&address=Ann+Arbor%2C+MI

The actual adress	Ann+Arbor%2C+MI
+	=space
2%C	= comma

```
import urllib
import json

serviceurl = 'http://maps.googleapis.com/maps/api/geocode/json?'

while true:
        address = raw_input('Enter location: ') %Enter location
        if len(address) < 1: break

        url = serviceurl
        + urllibe.urlencode({'sensor':'false', 'address':adress})
        %encode the user input as compatible with API
        print 'Retrieving', url
        uh = urllib.urlopen(url)
        data = uh.read() %read json
        print 'Retrieved', len(data), 'characters'

        try: js = json.loads(str(data))
        except: js = None
        if 'status' not in js or js['status'] != ' OK':
                %reading the status = 'OK'
                print '=== Failure To Retrieve ==='
                print data
                continue

        print json.dumps(js, indent=4) %make the js readable with indentation

        lat = js["results"][0]["geometry"]["location"]["lat"]
        lng = js["results"][0]["geometry"]["location"]["lng"]
        print 'lat', lat, 'lng',lng
        location = js['results'][0]['formatted address']
        print location
```

0.7 Twitter API

Twitter requires to get access token to use API for an application :

```
%hidden.py
def oauth() :
        return {
                "consumer_key" : "h7Lu....Ng",
                "consumer_secret" : "dNKen....7Q",
                "token_key" : "10013322....NPAR",
                "token_secret" : "H0yc....qoIp",
        }
```

```
%twurl.py=
import urllib
import oauth
import hidden
        def augment(url,parameters) :
        secrets = hidden.oauth()
        consumer = oauth.OAuthconsumer(secrets['consumer_key'],
        secrets['consumer_secret'])
        token = oauth.OauthToken(secrets['token_key'],
        secrets['token_secret']
        oauth_request = oauth.OAuthrequest.from_consumer_and_token(consumer,
        token=token, http_method='GET', http_url=url, parameters=parameters)
        oauth_request.sign_request(oauth.OAUthSignatureMethod_HMAC_SHA1(),
        consumer, token)
        return oauth_request.to_url()
```

Return: http://api.twitter.com/1.1/statuses/user_timeline.json?count=2 &oauth_version=1.0&oauth_token=.....oauth_consumer..etc

```
%twtest.py=
import urllib
from twurl import augment

print '* Calling Twitter ...'
url = augment('https://api.twitter/1.1/statuses/user_timeline.json',
 {'screen_name': 'drchuck', 'count': '2'} )
% the first part of the url is our request for 2 user_timelines of dr—chuck
%the 2nd part is the signature
print url
connection = urllib.urlopen(url)
data = connection.read()
print data
headers = connection.info().dict
print headers
```