Using Web Services

Chapter 13

Python for Informatics: Exploring Information
www.py4inf.com

Data on the Web

- With the HTTP Request/Response well understood and well supported there was a natural move toward exchanging data between programs using these protocols

XML

Marking up data to send across the network...

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

eXtensible Markup Language

- Primary purpose is to help information systems share structured data
- It started as a simplified subset of the Standard Generalized Markup Language (SGML), and is designed to be relatively human-legible

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

XML Basics

- Start Tag
- End Tag
- Text Content
- Attribute
- Self Closing Tag

```
<person>
  <name>Chuck</name>
  <phone type="intl">
    +1 734 303 4456
  </phone>
  <email hide="yes" />
</person>
```

White Space

```
<person>
  <name>Chuck</name>
  <phone type="intl">
    +1 734 303 4456
   </phone>
   <email hide="yes" />
</person>
```

Line ends do not matter. White space is generally discarded on text elements. We indent only to be readable.

```
<person>
  <name>Chuck</name>
  <phone type="intl">+1 734 303 4456</phone>
   <email hide="yes" />
</person>
```

Some XML...

```
<recipe name="bread" prep_time="5 mins" cook_time="3 hours">
  <title>Basic bread</title>
  <ingredient amount="8" unit="dL">Flour</ingredient>
  <ingredient amount="10" unit="grams">Yeast</ingredient>
  <ingredient amount="4" unit="dL" state="warm">Water</ingredient>
  <ingredient amount="1" unit="teaspoon">Salt</ingredient>
  <instructions>
    <step>Mix all ingredients together.</step>
    <step>Knead thoroughly.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Knead again.</step>
    <step>Place in a bread baking tin.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Bake in the oven at 180(degrees)C for 30 minutes.</step>
  </instructions>
</recipe>
```

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

XML Terminology

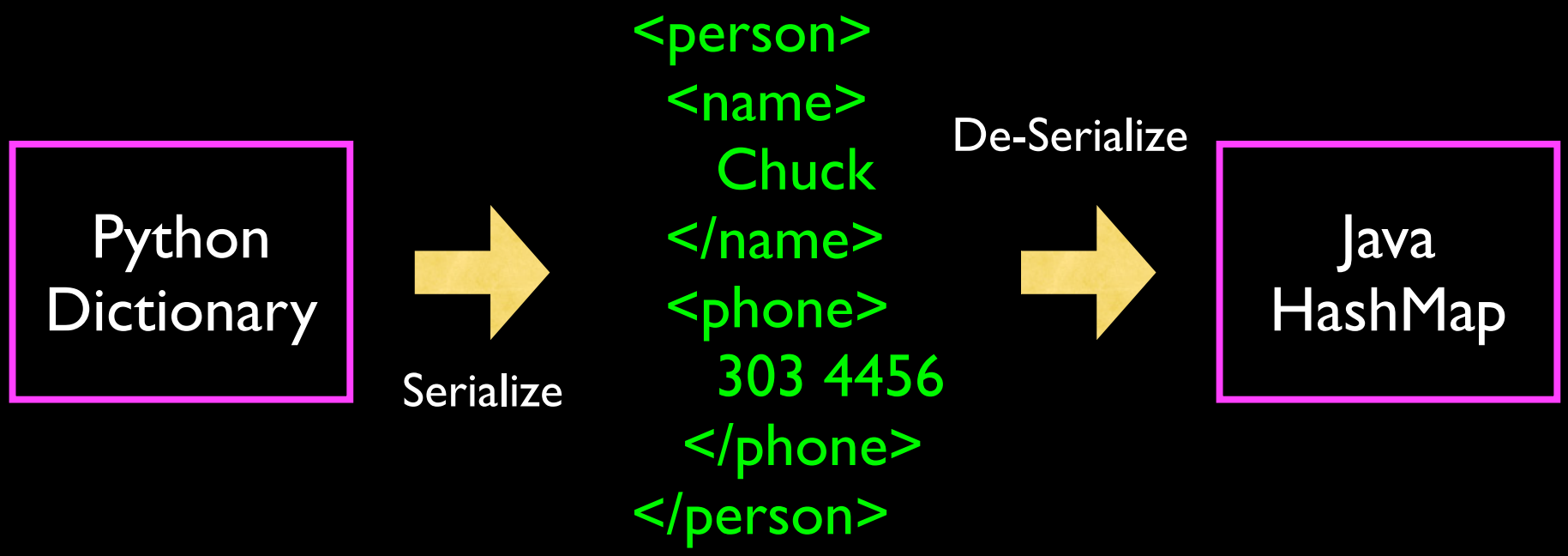
- Tags indicate the beginning and ending of elements
- Attributes - Keyword/value pairs on the opening tag of XML
- Serialize / De-Serialize - Convert data in one program into a common format that can be stored and/or transmitted between systems in a programming language independent manner

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

Sending Data across the "Net"

a.k.a. "Wire Protocol" - What we send on the "wire"

Agreeing on a "Wire Format"



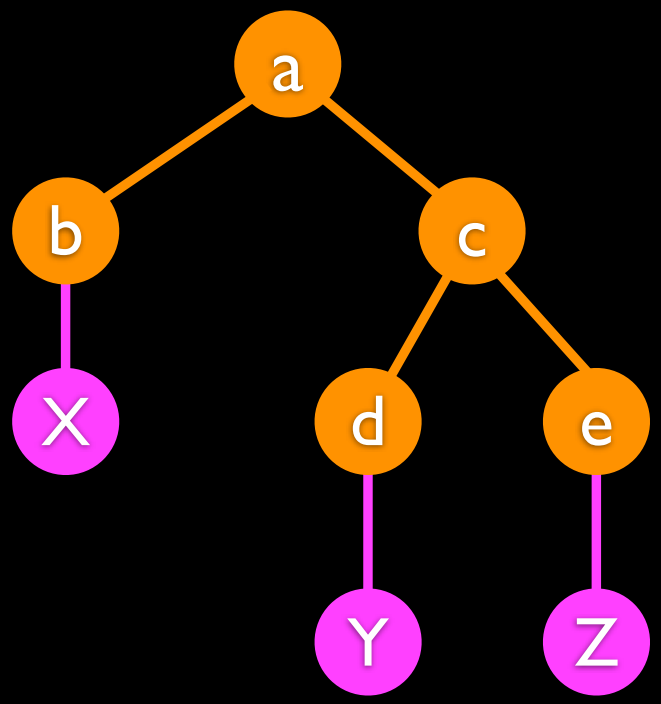
XML “Elements” (or Nodes)

- Simple Element
- Complex Element

```
<people>
  <person>
    <name>Chuck</name>
    <phone>303 4456</phone>
  </person>
  <person>
    <name>Noah</name>
    <phone>622 7421</phone>
  </person>
</people>
```

XML as a Tree

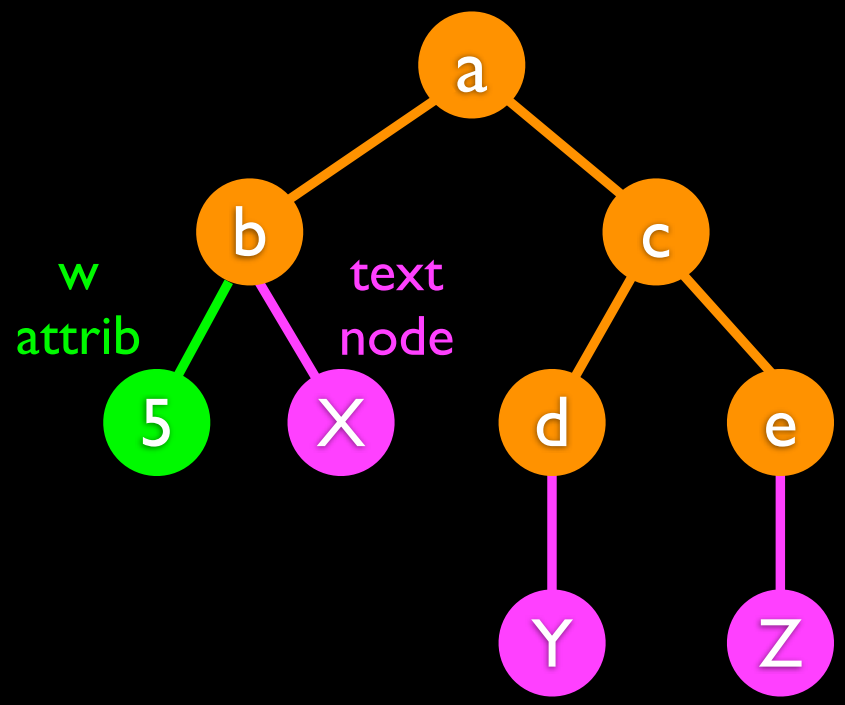
```
<a>
    <b>X</b>
    <c>
        <d>Y</d>
        <e>Z</e>
    </c>
</a>
```



Elements Text

XML Text and Attributes

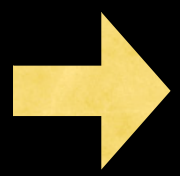
```
<a>
    <b w="5">X</b>
    <c>
        <d>Y</d>
        <e>Z</e>
    </c>
</a>
```



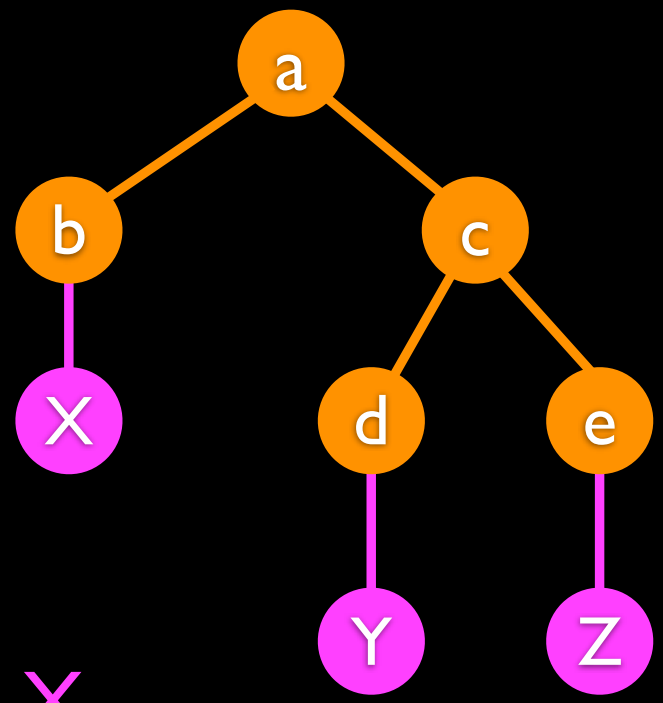
Elements Text

XML as Paths

```
<a>
    <b>X</b>
    <c>
        <d>Y</d>
        <e>Z</e>
    </c>
</a>
```



/a/b	X
/a/c/d	Y
/a/c/e	Z



Elements Text

XML Schema

Describing a "contract" as to what is acceptable XML.

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

http://en.wikibooks.org/wiki/XML_Schema

XML Schema

- Description of the legal format of an XML document
- Expressed in terms of constraints on the structure and content of documents
- Often used to specify a "contract" between systems - "My system will only accept XML that conforms to this particular Schema."
- If a particular piece of XML meets the specification of the Schema - it is said to "validate"

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

XML Validation

XML Document

XML Schema Contract

XML Validation

XML Document

```
<person>
  <lastname>Severance</lastname>
  <age>17</age>
  <dateborn>2001-04-17</dateborn>
</person>
```

XML Schema Contract

```
<xs:complexType name="person">
  <xs:sequence>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="age" type="xs:integer"/>
    <xs:element name="dateborn" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```

Many XML Schema Languages

- Document Type Definition (DTD)
  - http://en.wikipedia.org/wiki/Document_Type_Definition
- Standard Generalized Markup Language (ISO 8879:1986 SGML)
  - http://en.wikipedia.org/wiki/SGML
- XML Schema from W3C - (XSD)
  - http://en.wikipedia.org/wiki/XML_Schema_(W3C)

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

XSD XML Schema (W3C spec)

- We will focus on the World Wide Web Consortium (W3C) version
- It is often called "W3C Schema" because "Schema" is considered generic
- More commonly it is called XSD because the file names end in .xsd

http://www.w3.org/XML/Schema

http://en.wikipedia.org/wiki/XML_Schema_(W3C)

XSD Structure

```
<person>
  <lastname>Severance</lastname>
  <age>17</age>
  <dateborn>2001-04-17</dateborn>
</person>
```

- xs:element
- xs:sequence
- xs:complexType

```
<xs:complexType name="person">
  <xs:sequence>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="age" type="xs:integer"/>
    <xs:element name="dateborn" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```

XSD Constraints

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"
            minOccurs="1" maxOccurs="1" />
      <xs:element name="child_name" type="xs:string"
             minOccurs="0" maxOccurs="10" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<person>
  <full_name>Tove Refsnes</full_name>
  <child_name>Hege</child_name>
  <child_name>Stale</child_name>
  <child_name>Jim</child_name>
  <child_name>Borge</child_name>
</person>
```

http://www.w3schools.com/Schema/schema_complex_indicators.asp

XSD Data Types

```
<xs:element name="customer" type="xs:string"/>
<xs:element name="start" type="xs:date"/>
<xs:element name="startdate" type="xs:dateTime"/>
<xs:element name="prize" type="xs:decimal"/>
<xs:element name="weeks" type="xs:integer"/>
```

```
<customer>John Smith</customer>
<start>2002-09-24</start>
<startdate>2002-05-30T09:30:10Z</startdate>
<prize>999.50</prize>
<weeks>30</weeks>
```

It is common to represent time in UTC/GMT given that servers are often scattered around the world.

http://www.w3schools.com/Schema/schema_dtypes_numeric.asp

ISO 8601 Data/Time Format

2002-05-30T09:30:10Z

Year-month-day

Time of day

Time-zone - typically specificed in UTC / GMT rather than local time zone.

http://en.wikipedia.org/wiki/ISO_8601

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

```xml
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Address">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Recipient" type="xs:string" />
        <xs:element name="House" type="xs:string" />
        <xs:element name="Street" type="xs:string" />
        <xs:element name="Town" type="xs:string" />
        <xs:element minOccurs="0" name="County" type="xs:string" />
        <xs:element name="PostCode" type="xs:string" />
        <xs:element name="Country">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="FR" />
              <xs:enumeration value="DE" />
              <xs:enumeration value="ES" />
              <xs:enumeration value="UK" />
              <xs:enumeration value="US" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<Address
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
   xsi:noNamespaceSchemaLocation="SimpleAddress.xsd">
  <Recipient>Mr. Walter C. Brown</Recipient>
  <House>49</House>
  <Street>Featherstone Street</Street>
  <Town>LONDON</Town>
  <PostCode>EC1Y 8SY</PostCode>
  <Country>UK</Country>
</Address>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shiporder">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="orderperson" type="xs:string"/>
   <xs:element name="shipto">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="address" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="country" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="item" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="note" type="xs:string" minOccurs="0"/>
      <xs:element name="quantity" type="xs:positiveInteger"/>
      <xs:element name="price" type="xs:decimal"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="orderid" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<shiporder orderid="889923"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="shiporder.xsd">
 <orderperson>John Smith</orderperson>
 <shipto>
  <name>Ola Nordmann</name>
  <address>Langgt 23</address>
  <city>4000 Stavanger</city>
  <country>Norway</country>
 </shipto>
 <item>
  <title>Empire Burlesque</title>
  <note>Special Edition</note>
  <quantity>1</quantity>
  <price>10.90</price>
 </item>
 <item>
  <title>Hide your heart</title>
  <quantity>1</quantity>
  <price>9.90</price>
 </item>
</shiporder>
```

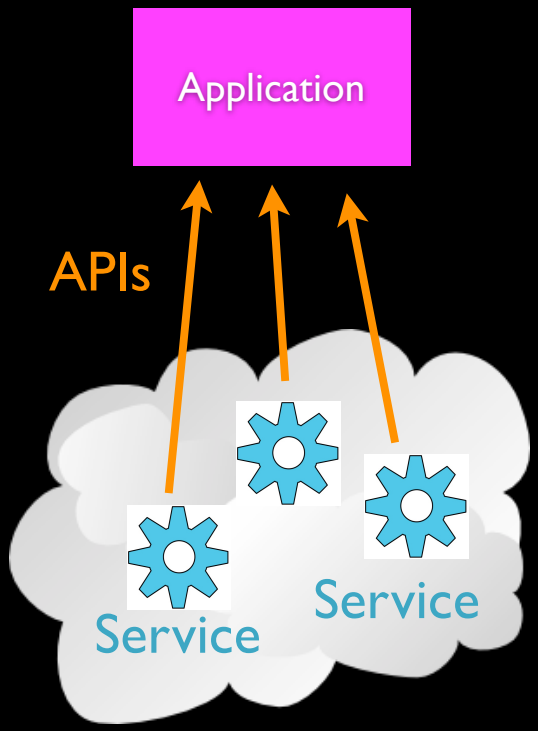
http://www.w3schools.com/Schema/schema_example.asp

Service Oriented Approach

http://en.wikipedia.org/wiki/Service-oriented_architecture

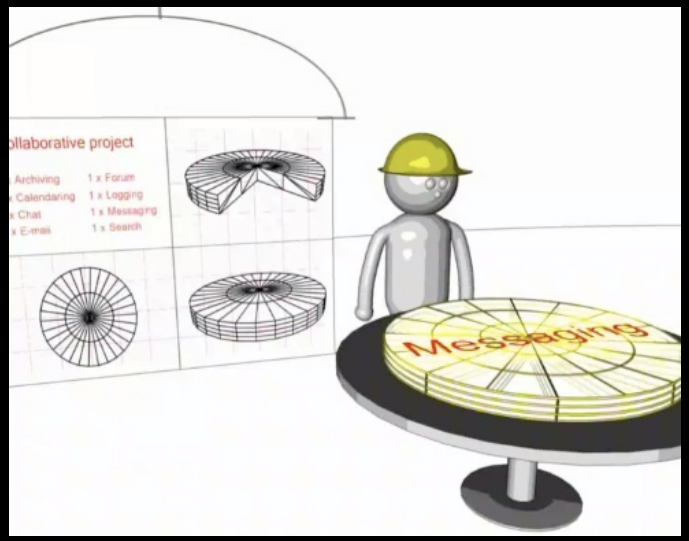
Service Oriented Approach

- Most non-trivial web applications use services
- They use services from other applications
  - Credit Card Charge
  - Hotel Reservation systems
- Services publish the "rules" applications must follow to make use of the service (API)



Multiple Systems

- Initially - two systems cooperate and split the problem
- As the data/service becomes useful - multiple applications want to use the information / application



http://www.vimeo.com/7591954

5:15

Web Services

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

Web Service Technologies

- SOAP - Simple Object Access Protocol (software)
  - Remote programs/code which we use over the network
  - Note: Dr. Chuck does not like SOAP because it is overly complex
- REST - Representational State Transfer (resource focused)
  - Remote resources which we create, read, update and delete remotely

http://en.wikipedia.org/wiki/SOAP_(protocol)
http://en.wikipedia.org/wiki/REST

Twitter API - a REST Example

twitter

The Twitter API

Biz Stone (Founder of Twitter): The API has been arguably the most important, or maybe even inarguably, the most important thing we've done with Twitter. It has allowed us, first of all, to keep the service very simple and create a simple API so that developers can build on top of our infrastructure and come up with ideas that are way better than our ideas, and build things like Twitterrific, which is just a beautiful elegant way to use Twitter that we wouldn't have been able to get to, being a very small team. So, the API which has easily 10 times more traffic than the website, has been really very important to us.

http://readwritetalk.com/2007/09/05/biz-stone-co-founder-twitter/

Application Program Interface

The API itself is largely abstract in that it specifies an interface and controls the behavior of the objects specified in that interface. The software that provides the functionality described by an API is said to be an "implementation" of the API. An API is typically defined in terms of the programming language used to build an application.

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

Twitter REST API

- A series of URLs which you retrieve which return data
- Much like the information on twitter.com
- Returns XML data in the HTTP Document

https://dev.twitter.com/docs/api

twitter developers | API Status | Documentation | Discussions | Your apps | Sign out | Search

GET statuses/public_timeline

Home › Documentation › Timeline Resources › statuses/public_timeline

Returns the 20 most recent statuses, including retweets if they exist, from non-protected users.

The public timeline is cached for 60 seconds. Requesting more frequently than that will not return any more data, and will count against your rate limit usage.

URL

http://api.twitter.com/*version*/statuses/public_timeline.*format*

Supported formats

json, xml, rss, atom

Supported request methods

GET

Requires Authentication

false About authentication »

Timeline resources

- statuses/public_timeline
- statuses/home_timeline
- statuses/friends_timeline
- statuses/user_timeline
- statuses/mentions
- statuses/retweeted_by_me
- statuses/retweeted_to_me
- statuses/retweets_of_me

Tweets resources

User resources

Trends resources

Local Trends resources

List resources

https://dev.twitter.com/doc/get/statuses/public_timeline

```xml
<?xml version="1.0" encoding="UTF-8"?>
<statuses type="array">
<status>
  <created_at>Thu Jul 15 23:24:33 +0000 2010</created_at>
  <id>18639350000</id>
  <text>se fuder</text>
  <source>web</source>
  <truncated>false</truncated>
  <in_reply_to_status_id></in_reply_to_status_id>
  <in_reply_to_user_id></in_reply_to_user_id>
  <favorited>false</favorited>
  <in_reply_to_screen_name></in_reply_to_screen_name>
  <user>
    <id>61949587</id>
    <name>leonor</name>
    <screen_name>leonor_</screen_name>
    <location></location>
    <description></description>
    <profile_image_url>http://a1.twimg.com/profile_images/1015735169/Foto0133_normal.jpg</profi
    <url></url>
    <protected>false</protected>
    <followers_count>91</followers_count>
    <profile_background_color>ffffff</profile_background_color>
    <profile_text_color>f745b9</profile_text_color>
    <profile_link_color>f00c95</profile_link_color>
    <profile_sidebar_fill_color></profile_sidebar_fill_color>
    <profile_sidebar_border_color>969090</profile_sidebar_border_color>
    <friends_count>197</friends_count>
```


https://dev.twitter.com/doc/get/statuses/public_timeline

```
<?xml version="1.0" encoding="UTF-8"?>
<users type="array">
<user>
  <id>14870169</id>
  <name>gbhatnag</name>
  <screen_name>gbhatnag</screen_name>
  <location>iPhone: 42.284775,-83.732422</location>
  <profile_image_url>http://s3.amazonaws.com/twitter_production/profile_images/
54535105/profile_normal.jpg</profile_image_url>
  <followers_count>29</followers_count>
  <status>
    <created_at>Sun Mar 15 17:52:44 +0000 2009</created_at>
    <id>1332217519</id>
    <text>to add to @aatorres: projects that may fall into pervasive computing,
situated technologies, distributed media, would be interesting #sxsw</text>
  </status>
</user>
<user>
  <id>928961</id>
  <name>Rasmus Lerdorf</name>
   ……..
</user>
</users>
```


https://api.twitter.com/1/statuses/friends/drchuck.xml

Retrieving Twitter Data in Python

http://www.py4inf.com/code/twitter1.py

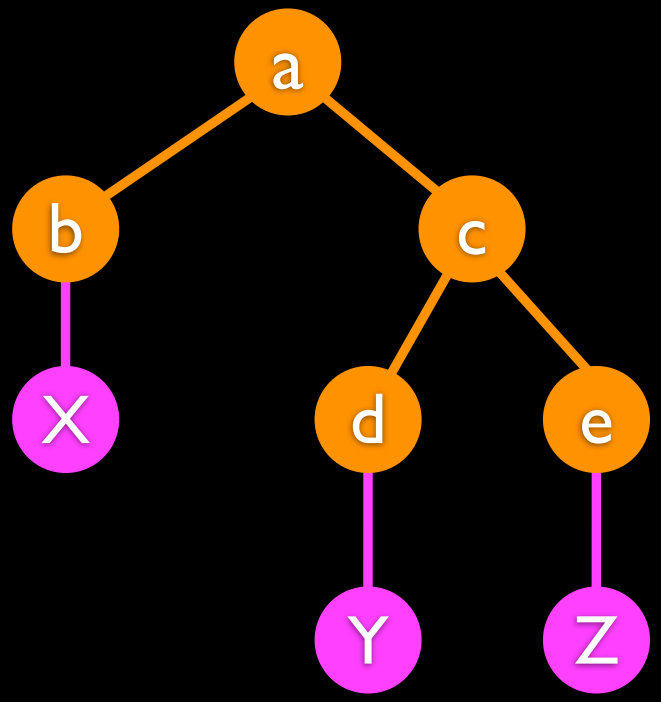
```
import urllib

TWITTER_URL = 'https://api.twitter.com/1/statuses/friends/ACCT.xml'

while True:
      print ''
      acct = raw_input('Enter Twitter Account:')
      if ( len(acct) < 1 ) : break
      url = TWITTER_URL.replace('ACCT', acct)
      document = urllib.urlopen (url).read()
      print document[:250]
```

Viewing XML as a Tree

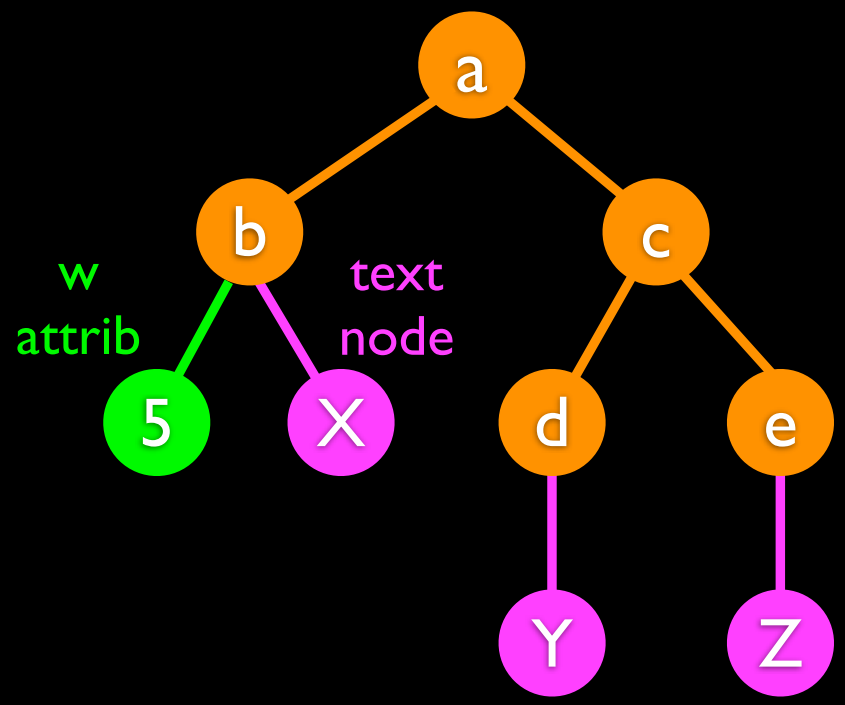
```
<a>
    <b>X</b>
    <c>
        <d>Y</d>
        <e>Z</e>
    </c>
</a>
```



Elements Text

XML Text and Attributes

```
<a>
    <b w="5">X</b>
    <c>
        <d>Y</d>
        <e>Z</e>
    </c>
</a>
```



Elements Text

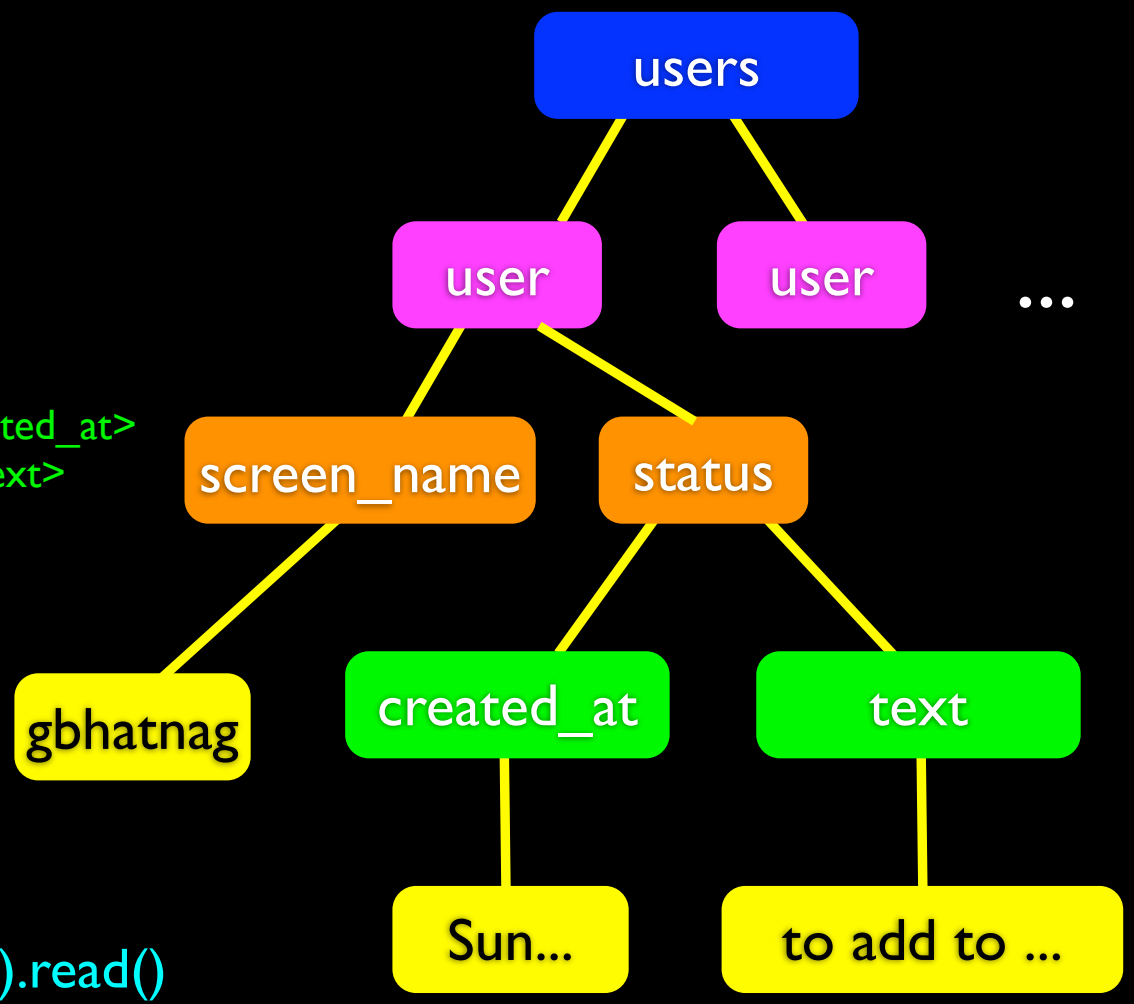
The ElementTree Library

- The ElementTree Library in Python reads XML from a file or string and creates a tree of nodes that we can then look through and extract data from

```
import urllib
import xml.etree.ElementTree as ET

    document = urllib.urlopen (url).read()
    print 'Retrieved', len(document), 'characters.'
    tree = ET.fromstring(document)
```

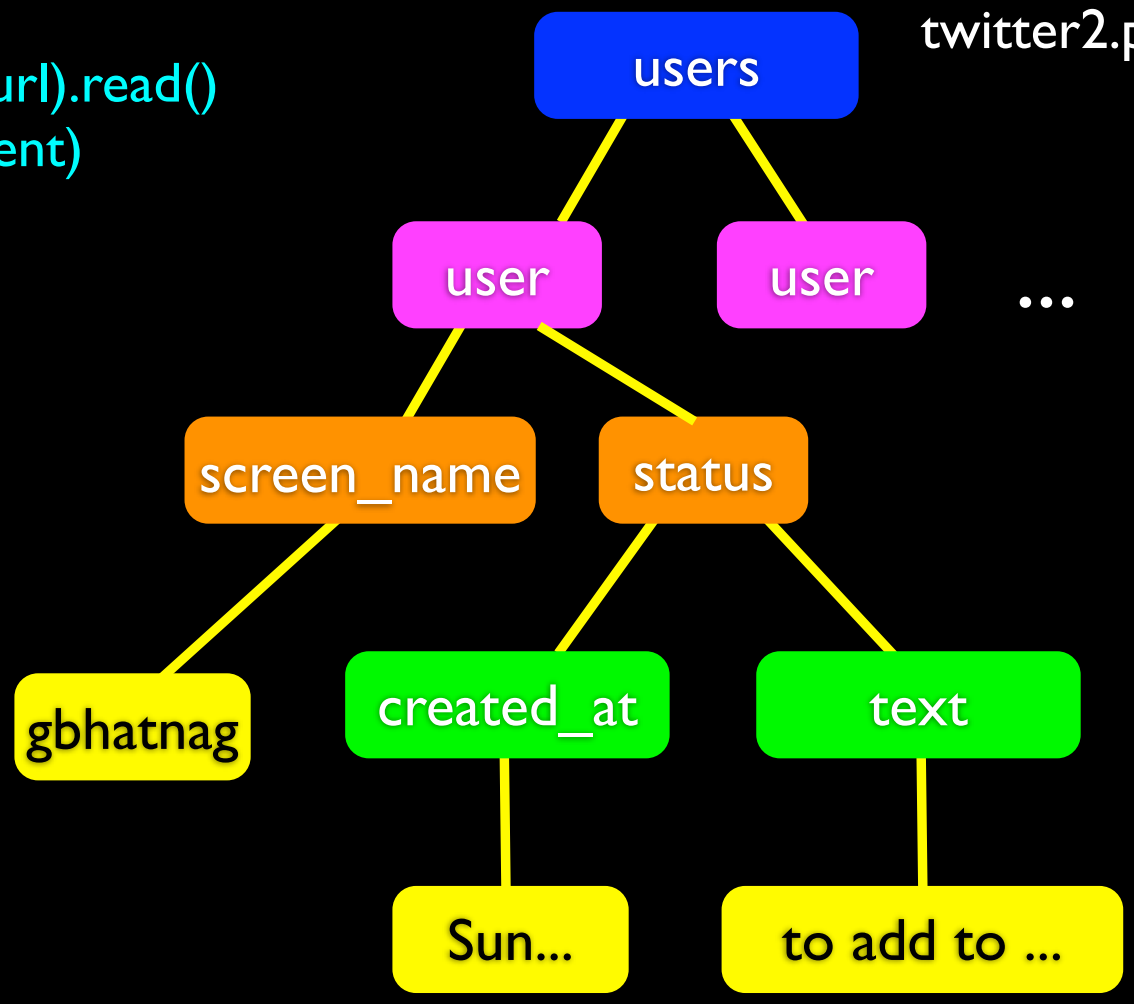
```
<?xml version="1.0" encoding="UTF-8"?>
<users type="array">
<user>
  <id>14870169</id>
  <name>Gaurav Bhatnagar</name>
  <screen_name>gbhatnag</screen_name>
  <location>42.28,-83.74</location>
  <status>
    <created_at>Sun Mar 15 17:52:44</created_at>
    <text>to add to @aatorres: projects</text>
  </status>
</user>
<user>
  <id>928961</id>
  <name>Rasmus Lerdorf</name>
   ……..
</user>
</users>
```




```
document = urllib.urlopen (url).read()
tree = ET.fromstring(document)
```

twitter2.py

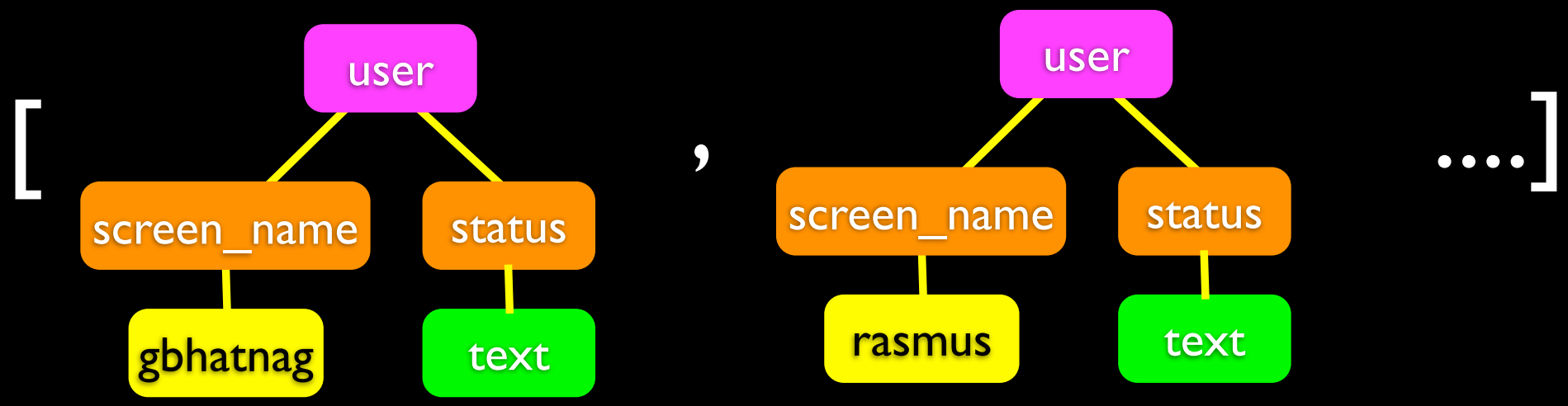
```
document = urllib.urlopen (url).read()
tree = ET.fromstring(document)
```



http://www.py4inf.com/code/twitter2.py

twitter2.py

```
document = urllib.urlopen (url).read()
tree = ET.fromstring(document)
for usr in tree.findall('user'):
```

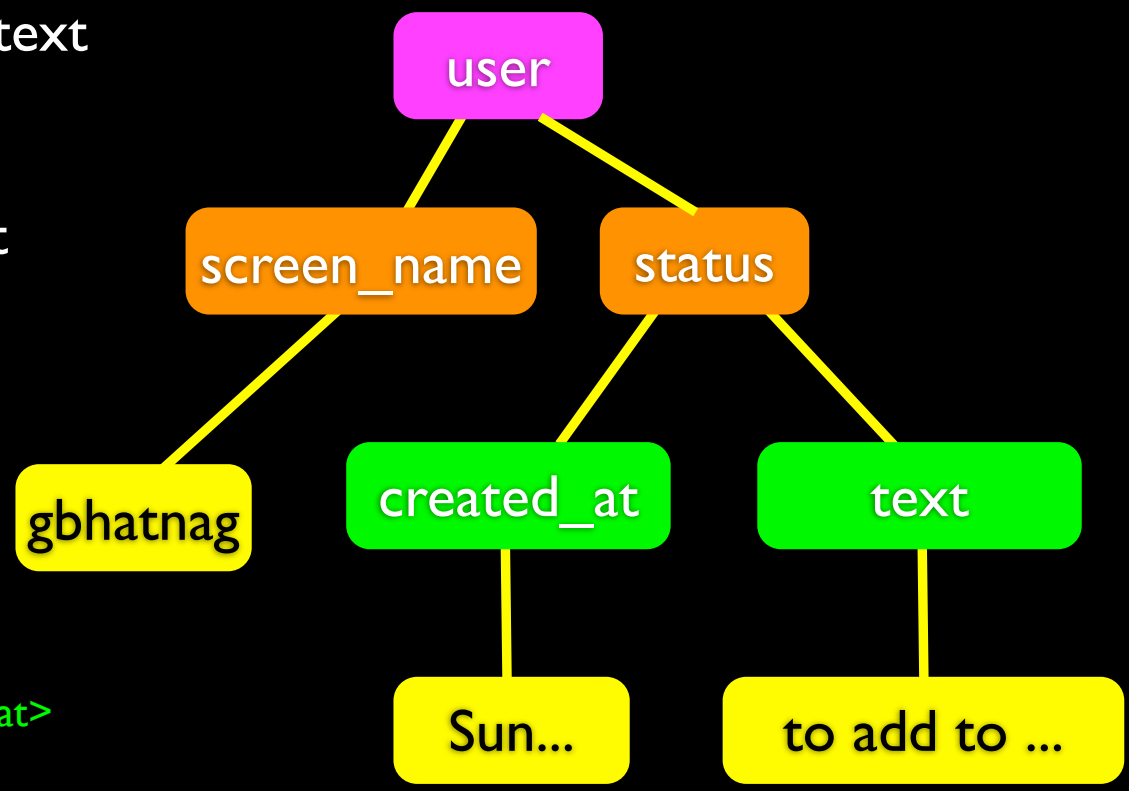


findall pulls out a Python List of user 'nodes' / sub-trees.

twitter2.py

```
for usr in tree.findall('user'):
    count = count + 1
    if count > 4 : break
    print usr.find('screen_name').text
    status =  usr.find('status')
    if status is not None :
        txt = status.find('text').text
        print '  ',txt[:50]
```

```
<user>
  <id>14870169</id>
  <name>Gaurav Bhatnagar</name>
  <screen_name>gbhatnag</screen_name>
  <location>42.28,-83.74</location>
  <status>
    <created_at>Sun Mar 15 17:52:44</created_at>
    <text>to add to @aatorres: projects</text>
  </status>
</user>
```

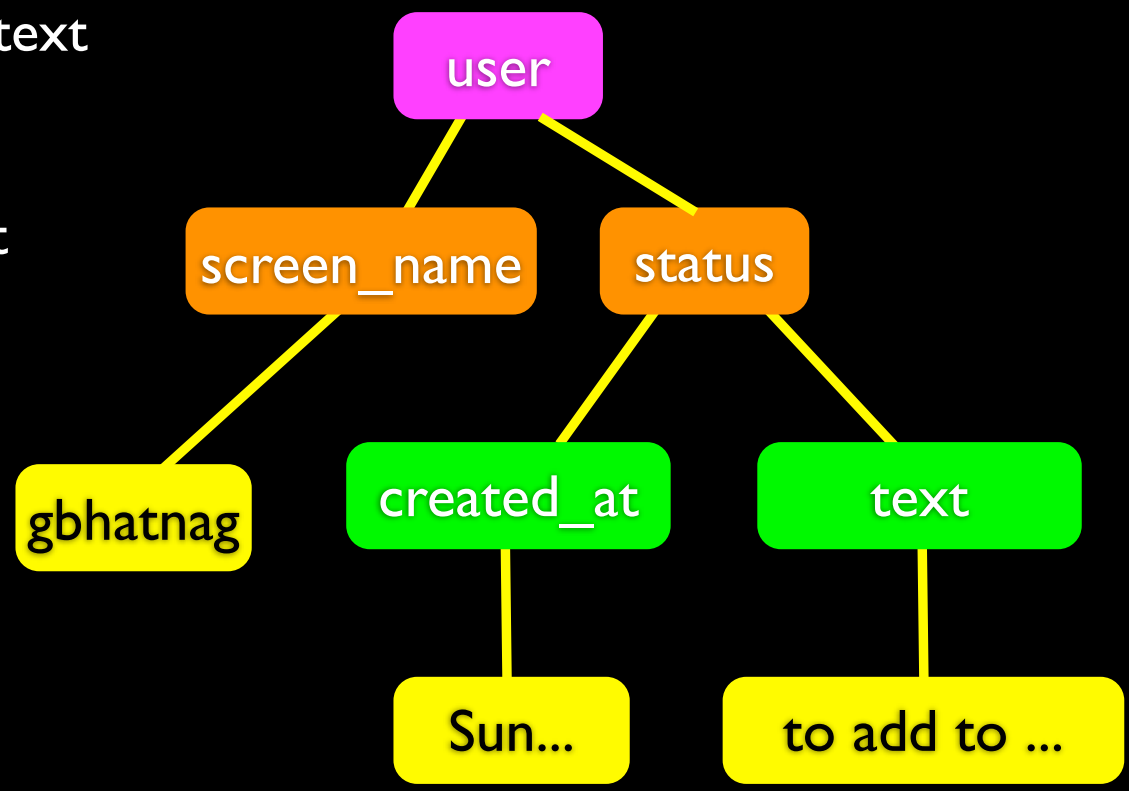


twitter2.py

```
for usr in tree.findall('user'):
    count = count + 1
    if count > 4 : break
    print usr.find('screen_name').text
    status =  usr.find('status')
    if status is not None :
        txt = status.find('text').text
        print '  ',txt[:50]
```

```
$ python twitter2.py

Enter Twitter Account: drchuck
gbhatnag
  to add to @aatorres: projects that
rasmus
  @nine_L Which shop is that?
```



```
$ python twitter2.py

Enter Twitter Account: drchuck
bnmnetp
   @wilw @TUAW I wish I hadn't thrown mine away 10 ye
fielding
   I still remember when the Web was an open source p
kcblot
   RT @mattmaurer: NEWS: @Tulane picks @Blackboard ov
RichardDreyfuss
   A+ RT @cliveatkinson: @RichardDreyfuss Your gonna
```

```
<user>
  <id>14870169</id>
  <name>Gaurav Bhatnagar</name>
  <screen_name>gbhatnag</screen_name>
  <location>42.28,-83.74</location>
  <status>
    <created_at>Sun Mar 15 17:52:44</created_at>
    <text>to add to @aatorres: projects</text>
  </status>
</user>
```

```
python twitter3.py
Enter Twitter Account:drchuck
gbhatnag
  42.28,-83.74
  to add to @aatorres: projects that may fall into p
rasmus
  Sunnyvale, California
  Grr..  #lazyweb, how do I tell Thunderbird to use
```

JavaScript Object Notation

JavaScript Object Notation

- Douglas Crockford - "Discovered" JSON
- Object literal notation in JavaScript

https://vimeo.com/38054451
http://www.youtube.com/watch?v=-C-JoyNuQJs

JSON

http://www.json.org/

Introducing JSON

العربية Български 中文 Český Nederlandse Dansk English Esperanto Française Deutsch Ελληνικά עברית Magyar Indonesia
Italiano 日本 한국어 فارسی Polski Português Română Русский Српски Slovenščina Español Svenska Türkçe Tiếng Việt

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

An *object* is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right

object
  {}
  { *members* **}**
members
  pair
  pair **,** *members*
pair
  string **:** *value*
array
  []
  [*elements* **]**
elements
  value
  value **,** *elements*
value
  string
  number
  object

```
[
  {
    "id": 721273,
    "location": "Luther College",
    "name": "Brad Miller",
    "statuses_count": 765,
    "screen_name": "bnmnetp",
    "url": "http:\/\/reputablejournal.com",
    "followers_count": 169,
    "status": {
      "id_str": "273593172989452290",
      "in_reply_to_screen_name": "wilw",
      "text": "@wilw @TUAW I wish I hadn\u2019t thrown mine away 10 years ago!",
      "created_at": "Wed Nov 28 01:04:30 +0000 2012",
    },
    "description": "Professor of Computer Science"
  },
  {
    "id": 9081272,
    "location": "Tustin, CA, USA",
    "name": "Roy T. Fielding",
```


https://api.twitter.com/1/statuses/friends/drchuck.json

Summary

- Service Oriented Architecture - allows an application to be broken into parts and distributed across a network
- An Application Program Interface (API) is a contract for interaction
- Web Services provide infrastructure for applications cooperating (an API) over a network - SOAP and REST are two styles of web services
- XML and JSON are serialization formats