Using python for web access

Notes

####Chapter 11####

##Regular Expressions##

##string matching and parsing

single character, compact language

^ matches the beginning of line

$ matches the end of line

. any character

\s whitespace

\S 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

( the string extraction is to start

) the string extraction is to end

##before using, type “import re”

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): ##return TRUE of FALSE##

print line

if line.startswith(‘From:’)

print line

if re.search (‘^From:’, line)

print line

### dot character matches any character

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

eg. ^X.\*: (match the start of the line, match any character, many times)

^X- \S+: (\S any non-whitespace character, one or more times)

## re.findall()

eg. [0-9] + # one digit between 0 and 9, one or more digits

import re

x = ‘My 2 favorite numbers are 19 and 42’

y = re.findall(‘[0-9]+’, x)

print y # gives out a python list

y = re.findall(‘[AEIOU]+’, x)

## Greedy matching as default, always go with the longer result

To switch to non-greedy matching ^F.+?:

## y = re.findall(‘\S+@\S+’, x)

\S+ at least one non-whitespace character

() # the start and end of what to extract

y = re.findall(‘^From (\S+@\S+)’, x)

y = re.findall (‘@([^ ]\* ‘, lin) ##[^ ] means match non-blank character, \* many of them

‘ ^From [.\*@([^](mailto:.*@([%5e) ]\*)’

eg. stuff = re.findall (‘^X-DSPAM-Condifence: ([0-9.]+)’, line) #0 through 9 or .#

if len(stuff) != 1 : continue # skip if no match

num = float (stuff[0])

numlist.append (num)

print ‘Maximum:’, max(numlist)

####Chapter 12####

###Networks and sockets###

Transport layer from browsers to servers

TCP port numbers

#import socket #(socket library in python)

Mysock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

#library.method, make a socket#

mysock.connet ((‘www.py4inf.com’, 80)) #Host, Port, start connection between 2 things#

web browsering to a web surver(HTTP)

#rules to allow browsers to retrieve web documents from servers over the internet#

URL: protocol #http# + host + document

Request-response-cycle port 80, web

## Making and HTTP request

Request a document (or the default document)

GET <http://www.dr.chuck.com/page1.htm>

##”Hacking” HTTP

$telnet [www.dr.chuck.com](http://www.dr.chuck.com) 80

#makes a connect, and send across socket#

GET [http://www.dr-chuc.com/page1.htm HTTP/1.0](http://www.dr-chuc.com/page1.htm%20HTTP/1.0) #gives out a IP number

##Write a web browser

Eg. An HTTP request in python

Import socket

Mysock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

Mysock.connect ((‘www.py4inf.com’, 80))

Mysock.send (‘GET <http://www.py4inf.com/code/romeo.txt> HTTP/1.0\n\n’)

## \n\n = press enter twice, same as the telnet#

While True ##reading loop#

data = mysock.recv(512) #give up to 512

if (len(data)<1): #if get nothing, end of file, give back -1, break out of the loop

break

print data

mysock.close()

##Making HTTP easier with urllib

Import urllib

Fhand = urllib.urlopen(‘http://www.py4inf.com/code/romeo.txt’) #one parameter

#identical to open url and read, do the rest as working on a file handle

For line in fhand:

Print line.strip() #loop through line by line in url ## only gets the context

Counts = dict ()

for line in fhand:

Words = line.split()

for word in words:

counts[word] = counts.get(word,0) + 1

print counts

####Week 4####

#Programs that surf the web #

Web scraping

Beautiful soup #software [www.crummy.com/software/BeautifulSoup/](http://www.crummy.com/software/BeautifulSoup/)

Place the BeautifulSoup.py file in the same folder as Python code

Import urllib

From BeautifulSoup import \*

url = raw\_input(‘Enter –‘) #prompt for input#

html = urllib.urlopen(url).read() #read method, read all lines##html is a string of entile page#

soup = BeautifulSoup(html) #soup object, the parsed html data, can ask soup questions#

#Retrieve a list of the anchor tags

#Each tag is like a dictionary of HTML attributes

tags = soup(‘a’) #to look for things like <a……></a>##just give out a tags themselves##get anchor tags#

#smart look up##give out all links in the same page#

for tag in tags: #loop through all ‘a’ tags in the document#

print tag.get(‘href’, None)

####Chapter 13####

##Web services##

#XML/JSON

The wire protocol python dictionary vs Java Hashmap

Serialize/de-serialize

XML = extensible markup language

Doing documents, array

XML elements: simple elements, complex elements

Eg. <name> Chuck </name>

Start tag, end tag, attributes (key value pairs), self closing tag (< />)

Tags indicate the beginning and ending of elements

XML as a tree

XML schema

#contract

Schema XSD

MinOccurs, MaxOccurs

#Parsing XML in python

XML is built into python

Triple quoted strings ‘’’

Import xml.etree.ElementTree as ET

Tree = ET.fromstring(data) #parsing or deserialization

Print ‘Name:’, tree.find(‘name’).text

Print ‘Attr:’, tree.find(‘email’).get(‘hide’)

####Chapter 13####

##JSON and the REST architecture

#JSON represents data as nested “lists” and “dictionaries”

import json

data = ‘’’{ }’’’

info = json.loads(data) #de-serialize structure

print ‘Name: ‘, info[“name”]

print ‘Hide:’, info[“email’][“hide”]

need to de-serialize

two objects separated by a comma, sub 0 and sub 1

info = json.loads(input)

print ‘User count: ‘, len(info)

for item in info:

print ‘Name’, item[‘name’]

print ‘Id’, item[‘id’]

service oriented approaches

API: application program interface

Google geo-coding API

Print json.dumps(js,indent=4)

Lat = js[“results”][0][“geometry”][“location”][“lat”] #go into sub-objects layer down layer

API key

Twitter API

Hidden.py, consumer key

Import oauth #library

Import hidden #library

Def augment (url, parameters)