Internet Programming in Python - Week 2

Application layer Protocols, mostly HTTP

http://xkcd.com/269/

Brian Dorsey
brian@dorseys.org

A moment to reflect

Main course web page & updates

http://briandorsey.info/uwpython/

http://uwipip.tumblr.com/

class mailing list

if you're not on it,
talk to Ted

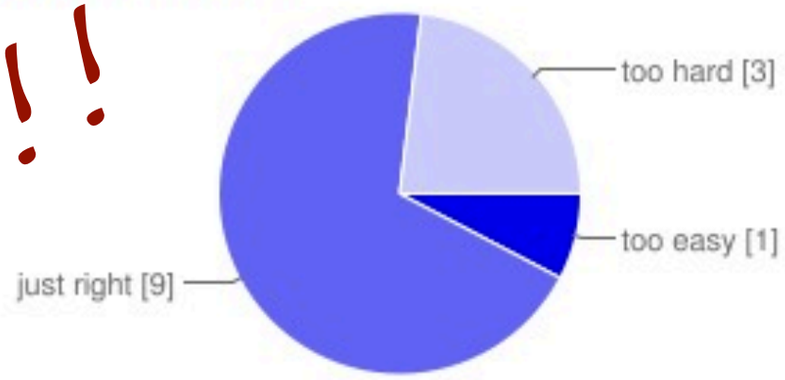
guest speakers

You rock!

You all rock. Great feedback, lots of activity on the list, worked hard on a poorly defined assignment

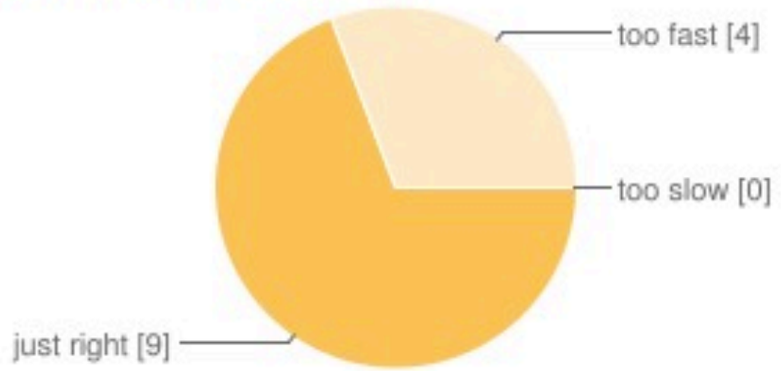
Thanks!!!

Difficulty of the material?



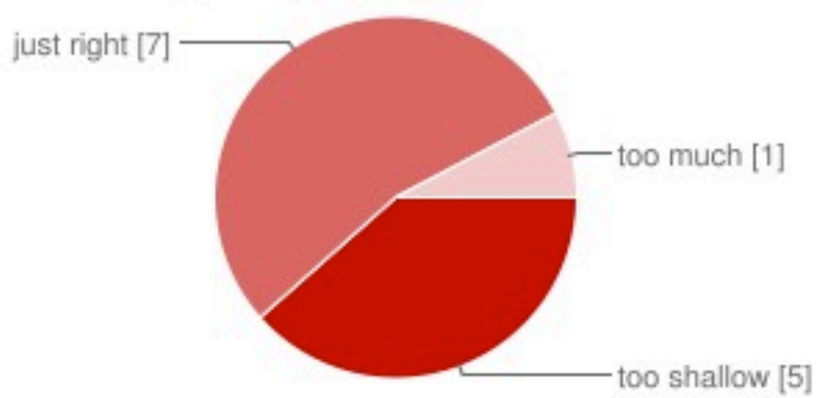
too easy	1	8%
just right	9	69%
too hard	3	23%

Pace of presenting the material?



too slow	0	0%
just right	9	69%
too fast	4	31%

Depth of covering the topics?



too shallow	5	38%
just right	7	54%
too much	1	8%

Huge thanks for filling out the poll! I think some folks were left out though, and this might be showing things a bit more balanced than they are.

"It was a shock to my system diving in like that."

"It would be good to get the slides a couple of days before the class .. say Sunday night"

"REALLY keep the presentations to 5 minutes. :^)"

"In the first class, most of us wrote in IDLE, so it seems we need to go over the basics of some command line stuff."

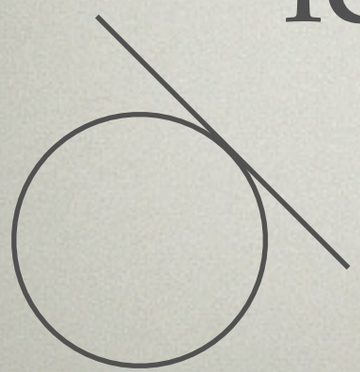
"Can you start into django as soon as possible, thanks"

"These are topics that are easy to assume people know, but most people probably don't know and they only take a minute to go over."

Virtual Machine (VM)

Wikipedia:

"a software implementation of a programmable machine, where the software implementation is constrained within another computer at a higher or lower level of symbolic abstraction."



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

bleh

for this class:

Virtual Machine (VM):
any operating system
running on top of software
instead of hardware

why do we care?

why do we have to use it?

"Hands-on work in class is very helpful but I think it would be better to have several simple demos to try out before class and a more challenging one to start before class and discuss and complete during class."

planned

|---- |---- |----

better?

--|-- --|-- --|--

thank you

for diving in and fighting all the logistics stuff

it really is necessary.

the order could
have been better

:/

Live on the internet - letting other people run your code. (hosting options, virtual machines, copying files, git, a bit of unix, and just enough Apache configuration)

added this topic for week four, moved the previous 4 &5 back a week

safe place to fail

experiment, break things on accident and then on purpose

Questions and review (20)

some thoughts from the assignments

two numbers

```
newString = receivedString.split(',')
finalString = int(newString[0]) + int(newString[1])
```

```
mid = data.find(',')
answer = int(data[:mid]) + int(data[mid+1:])
client.send(str(answer))
```

```
l = data.split(',')
r = int(l[0].strip()) + int(l[1].strip())
client.send(str(r))
```

list of numbers

```
l = data.split(' ')
val = 0
for i in l:
    val = val + int(i)
client.send(str(val))
```

```
operands = infixString.split("+")
operands = [float(x.replace(" ","")) for x in operands]
reply = str(sum(operands))
```

```
sdata = data.split(',')
intsdata = [int(x) for x in sdata]
adddata = sum(intsdata)
```

interactive

```
conn, addr = s.accept()
print 'Connected by', addr
data = 'Enter a Number:'
conn.send(data)

number1 = conn.recv(1024)
data = 'Enter another Number:'
conn.send(data)

number2 = conn.recv(1024)
total = int(number1) + int(number2)
data = "Total = " + str(total)
conn.send(data)
conn.close()
```

```
s.connect((HOST, PORT))

data = s.recv(1024)
print repr(data)
number = raw_input()
s.send(number)

data = s.recv(1024)
print repr(data)
number = raw_input()
s.send(number)

data = s.recv(1024)
print repr(data)
s.close()
```

VARIATION

- protocol (lines, split character, etc)
- error handling
- functions, or inline
- request/response vs. interactive
- client side user input

binary?

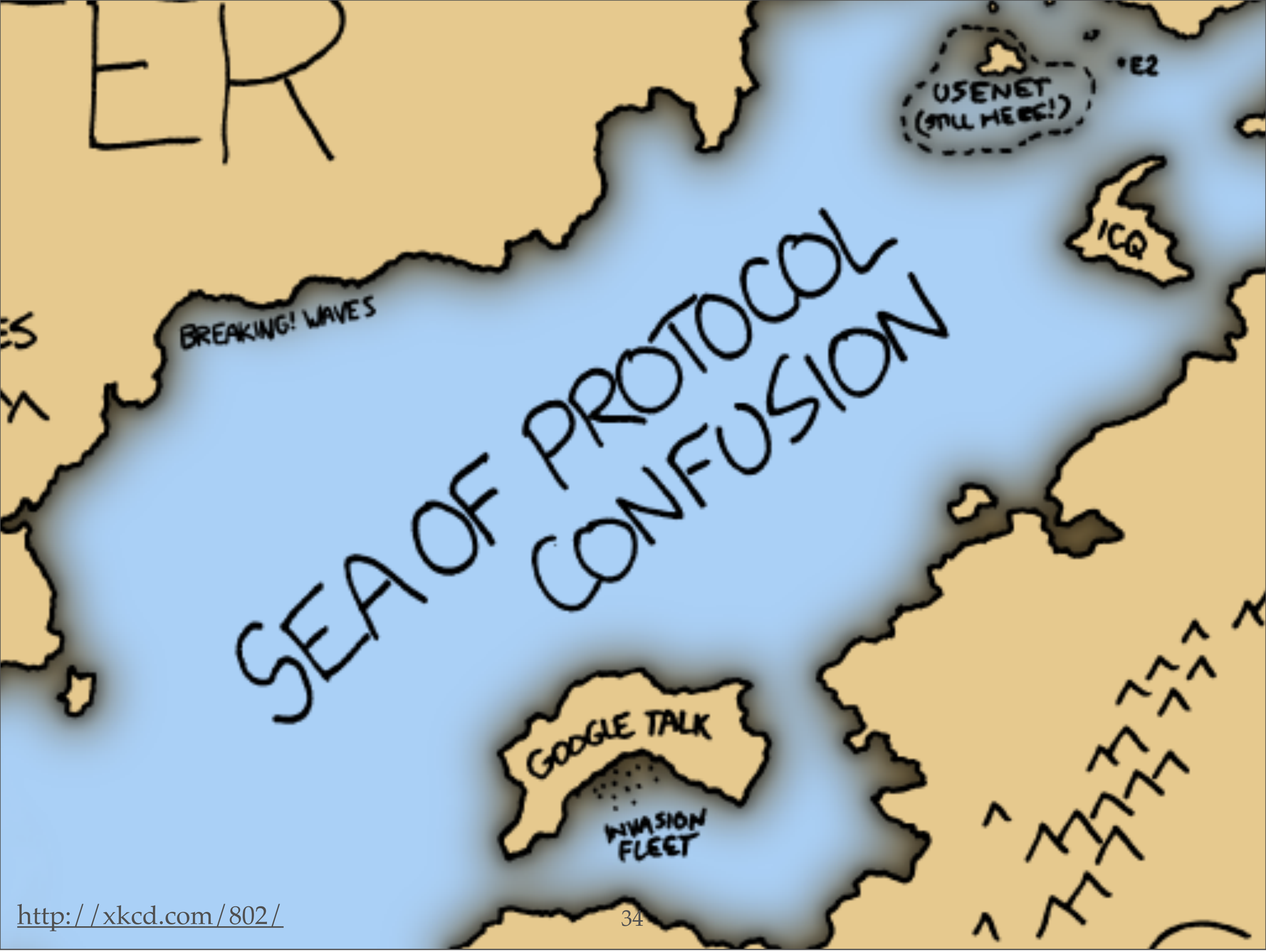
python 2.x strings

struct module

questions?

Lecture A (20)

more protocols



http://xkcd.com/802/

questions?

SMTP

```
S: 220 foo.com Simple Mail Transfer Service Ready
C: EHLO bar.com
S: 250-foo.com greets bar.com
S: 250-8BITMIME
S: 250-SIZE
S: 250-DSN
S: 250 HELP
C: MAIL FROM:<Smith@bar.com>
S: 250 OK
C: RCPT TO:<Jones@foo.com>
S: 250 OK
C: RCPT TO:<Green@foo.com>
S: 550 No such user here
C: RCPT TO:<Brown@foo.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: Blah blah blah...
C: ...etc. etc. etc.
C: .
S: 250 OK
C: QUIT
S: 221 foo.com Service closing transmission channel
```

http://tools.ietf.org/html/rfc5321#appendix-D

http://tools.ietf.org/html/rfc5321#appendix-D

POP3

```
C: <client connects to service port 110>
S: +OK POP3 server ready <1896.6971@mailgate.dobbs.org>
C: USER bob
S: +OK bob
C: PASS redqueen
S: +OK bob's maildrop has 2 messages (320 octets)
C: STAT
S: +OK 2 320
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends the text of message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <the POP3 server sends the text of message 2>
S: .
C: DELE 2
S: +OK message 2 deleted
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
C: <client hangs up>
```

http://www.faqs.org/docs/artu/ch05s03.html

IMAP

```
C: <client connects to service port 143>
S: * OK example.com IMAP4rev1 v12.264 server ready
C: A0001 USER "frobozz" "xyzzy"
S: * OK User frobozz authenticated
C: A0002 SELECT INBOX
S: * 1 EXISTS
S: * 1 RECENT
S: * FLAGS (\Answered \Flagged \Deleted \Draft \Seen)
S: * OK [UNSEEN 1] first unseen message in /var/spool/mail/esr
S: A0002 OK [READ-WRITE] SELECT completed
C: A0003 FETCH 1 RFC822.SIZE                    Get message sizes
S: * 1 FETCH (RFC822.SIZE 2545)
S: A0003 OK FETCH completed
C: A0004 FETCH 1 BODY[HEADER]                   Get first message header
S: * 1 FETCH (RFC822.HEADER {1425}
<server sends 1425 octets of message payload>
S: )
S: A0004 OK FETCH completed
C: A0005 FETCH 1 BODY[TEXT]                     Get first message body
S: * 1 FETCH (BODY[TEXT] {1120}
<server sends 1120 octets of message payload>
S: )
S: * 1 FETCH (FLAGS (\Recent \Seen))
S: A0005 OK FETCH completed
C: A0006 LOGOUT
S: * BYE example.com IMAP4rev1 server terminating connection
S: A0006 OK LOGOUT completed
C: <client hangs up>
```

http://www.faqs.org/docs/artu/ch05s03.html

IMAP (reordering)

```
C: <client connects to service port 143>
S: * OK example.com IMAP4rev1 v12.264 server ready
C: A0001 USER "frobozz" "xyzzy"
S: * OK User frobozz authenticated
C: A0002 SELECT INBOX
S: * 1 EXISTS
S: * 1 RECENT
S: * FLAGS (\Answered \Flagged \Deleted \Draft \Seen)
S: * OK [UNSEEN 1] first unseen message in /var/spool/mail/esr
S: A0002 OK [READ-WRITE] SELECT completed
C: A0003 FETCH 1 RFC822.SIZE                    Get message sizes
C: A0004 FETCH 1 BODY[HEADER]                   Get first message header
C: A0005 FETCH 1 BODY[TEXT]                     Get first message body
S: * 1 FETCH (RFC822.SIZE 2545)
S: A0003 OK FETCH completed
S: * 1 FETCH (RFC822.HEADER {1425}
<server sends 1425 octets of message payload>
S: )
S: A0004 OK FETCH completed
S: * 1 FETCH (BODY[TEXT] {1120}
<server sends 1120 octets of message payload>
S: )
S: * 1 FETCH (FLAGS (\Recent \Seen))
S: A0005 OK FETCH completed
C: A0006 LOGOUT
S: * BYE example.com IMAP4rev1 server terminating connection
S: A0006 OK LOGOUT completed
C: <client hangs up>
```

http://www.faqs.org/docs/artu/ch05s03.html

GNU Go Text Protocol (GTP)

```
1 boardsize 7
=1

2 clear_board
=2

3 play black D5
=3

4 genmove white
=4 C3

5 play black C3
?5 illegal move

6 play black E3
=6

7 showboard
=7
   A B C D E F G
 7 . . . . . . . 7
 6 . . . . . . . 6
 5 . . + X + . . 5
 4 . . . + . . . 4
 3 . . O . X . . 3
 2 . . . . . . . 2     WHITE (O) has captured 0 stones
 1 . . . . . . . 1     BLACK (X) has captured 0 stones
   A B C D E F G

8 quit
=8
```

http://www.gnu.org/software/gnugo/gnugo_19.html

HTTP

```
GET /index.html HTTP/1.1
Host: www.example.com
```

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8

<data>
```

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

many others only run
or optionally run
over HTTP

now, in Python

smtplib

```
import smtplib

template = """From: %s
To: %s
Subject: %s

"""

from_addr = 'Darth Vader <darth@deathstar.com>'
to_addrs = 'briandorsey@gmail.com'
subject = "I'm your father."
message = 'message body'
headers = template % (from_addr, to_addrs, subject)

s = smtplib.SMTP('mail.blueboxgrid.com')
s.ehlo()
s.sendmail(from_addr, to_addrs, headers + message)
s.close()
```

poplib

```
import poplib
import string, random

SERVER = "pop.spam.egg"
USER  = "mulder"
PASSWORD = "trustno1"

# connect to server
server = poplib.POP3(SERVER)

# login
server.user(USER)
server.pass_(PASSWORD)

# list items on server
resp, items, octets = server.list()

# download a random message
id, size = string.split(random.choice(items))
resp, text, octets = server.retr(id)
```

http://effbot.org/librarybook/poplib.htm

imaplib

```
import imaplib
import string, random

SERVER = "imap.spam.egg"
USER  = "mulder"
PASSWORD = "trustno1"

# connect to server
server = imaplib.IMAP4(SERVER)

# login
server.login(USER, PASSWORD)
server.select()

# list items on server
resp, items = server.search(None, "ALL")
items = string.split(items[0])

# fetch a random item
id = random.choice(items)
resp, data = server.fetch(id, "(RFC822)")
text = data[0][1]
```

http://effbot.org/librarybook/imaplib.htm

httplib

```
import httplib

conn = httplib.HTTPConnection("www.python.org")
conn.request("GET", "/index.html")
r1 = conn.getresponse()
print r1.status, r1.reason
data1 = r1.read()

conn.close()
```

http://docs.python.org/release/2.6.5/library/httplib.html

urllib2

```
import urllib2

f = urllib2.urlopen('http://www.python.org/')
print f.read()
```

http://docs.python.org/release/2.6.5/library/urllib2.html#examples

what's included in the standard library?

non-stdlib libraries?

a library exists for
pretty much every protocol

for SSH / SFTP

Paramikio

http://www.lag.net/paramiko/

Windows: http://www.voidspace.org.uk/python/modules.shtml#pycrypto
http://stackoverflow.com/questions/432385/sftp-in-python-platform-independent

Lab A (20)

groups of 4ish if possible, pairs if not

Lab A

- use urllib2 to download and save a url
  ex: http://briandorsey.info/uwpython/week01/email_vm.py
- write a program which reads a text file with a single URL on each line and attempts to save each to a file.

Break
(10)

API talk today?
Let's chat

photos?

API Talks (20)

API

Application Programming Interface

Web APIs

HTTP + data

talks

http://ProgrammableWeb.com

schedule the rest

I love the interpreter.

Lecture B (20)

HTTP

HTTP Made Really Easy

did it work?

```
GET /path/to/index.html HTTP/1.0
```

HTTP Requests

HTTP Responses

```
GET /path/to/index.html HTTP/1.0
```

resources

(nouns)

required first line
optional headers
“blank” line (CRLF)
data

```
GET /path/to/index.html HTTP/1.0
```

HTTP Requests

```
GET /path/to/index.html HTTP/1.0
```

methods
(verbs)

```
GET /path/to/index.html HTTP/1.0
```

GET

POST

PUT

DELETE

```
GET /path/to/index.html HTTP/1.0
```

POST = Create

GET = Read

PUT = Update

DELETE = Delete

```
GET /path/to/index.html HTTP/1.0
```

safe

side effects

GET

POST
PUT
DELETE

```
GET /path/to/index.html HTTP/1.0
```

idempotent

GET
PUT
DELETE

changes
after each
request

POST

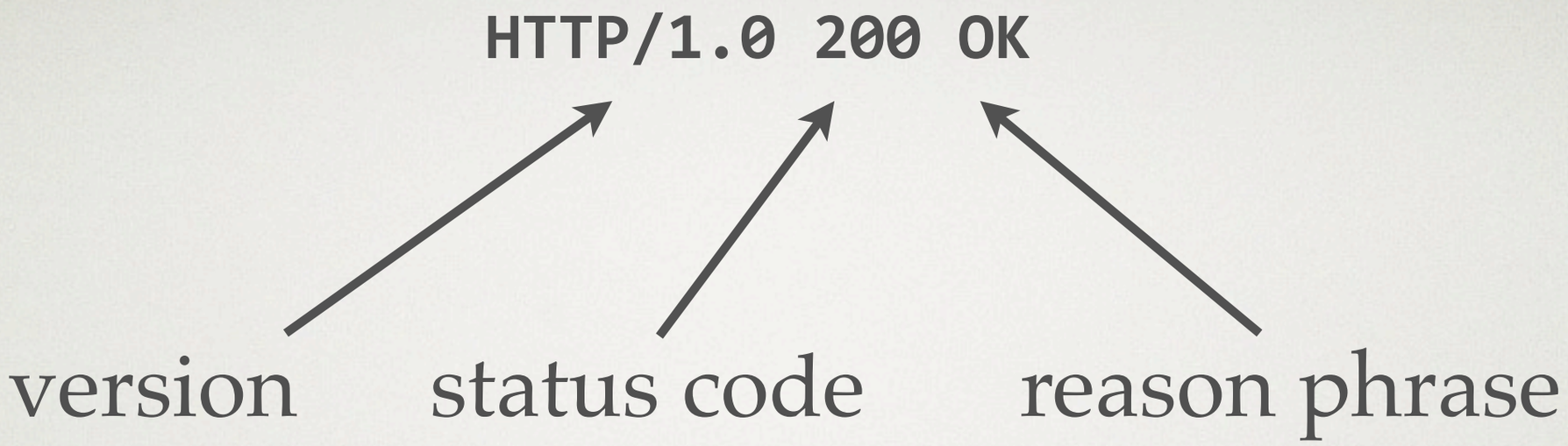
idempotent - multiple requests with same values should result in the same state

```
GET /path/to/index.html HTTP/1.0
```

resources != files

```
HTTP/1.0 200 OK
```

HTTP Responses


HTTP/1.0 200 OK
version
status code
reason phrase

```
HTTP/1.0 200 OK
```

- 1xx indicates an informational message
- 2xx indicates success of some kind
- 3xx redirects the client to another URL
- 4xx indicates an error on the client's part
- 5xx indicates an error on the server's part

`HTTP/1.0 200 OK`

- 200 OK
- 404 Not Found
- 301 Moved Permanently
- 302 Moved Temporarily
- 303 See Other (HTTP 1.1 only)
- 500 Server Error

http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

```
GET /path/file.html HTTP/1.0
User-Agent: HTTPTool/1.0
[blank line here]
```

request

---

response

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354

<html>
<body>
<h1>Happy New Millennium!</h1>
(more file contents)
  . . .
</body>
</html>
```

DEBUGGING TOOLS

- windows: http://www.fiddler2.com/fiddler2/
- firefox: http://getfirebug.com/
- safari: built in
- IE: built in

questions?

Break

(10)

Guest Lecture

(20 +10)

The Thirty Minute Web Server

(presented in thirty minutes)

code

git clone git://github.com/briandorsey/uwpython_web.git

or in our Dropbox folder

Lab B

(20)

Lab B

- run thirty_minute_webserver.py
- point your web browser at it, experiment. Break code, add prints, etc
- add /time url which returns HTML with the actual current time

review

```
python -m SimpleHTTPServer 8000
```

```
url = 'http://www.python.org/'
webbrowser.open_new_tab(url)
```

WRAPUP

Assignment

- Update thirty_minute_webserver.py
- Instead of showing the content of .py files, run the script and return its' output instead. (the stdlib subprocess module will be useful)
- test with script which outputs the time (print_time.py)
- update the print_time.py script to output html
- You've now written a dynamic webserver!
- Turn in: http://bit.ly/upipip_week2

Info

Office hours:
Sunday 2-5pm

Top Pot Donuts
2124 5th Ave
Seattle, WA 98121
206-728-1966