

Networked Programs

Chapter 12

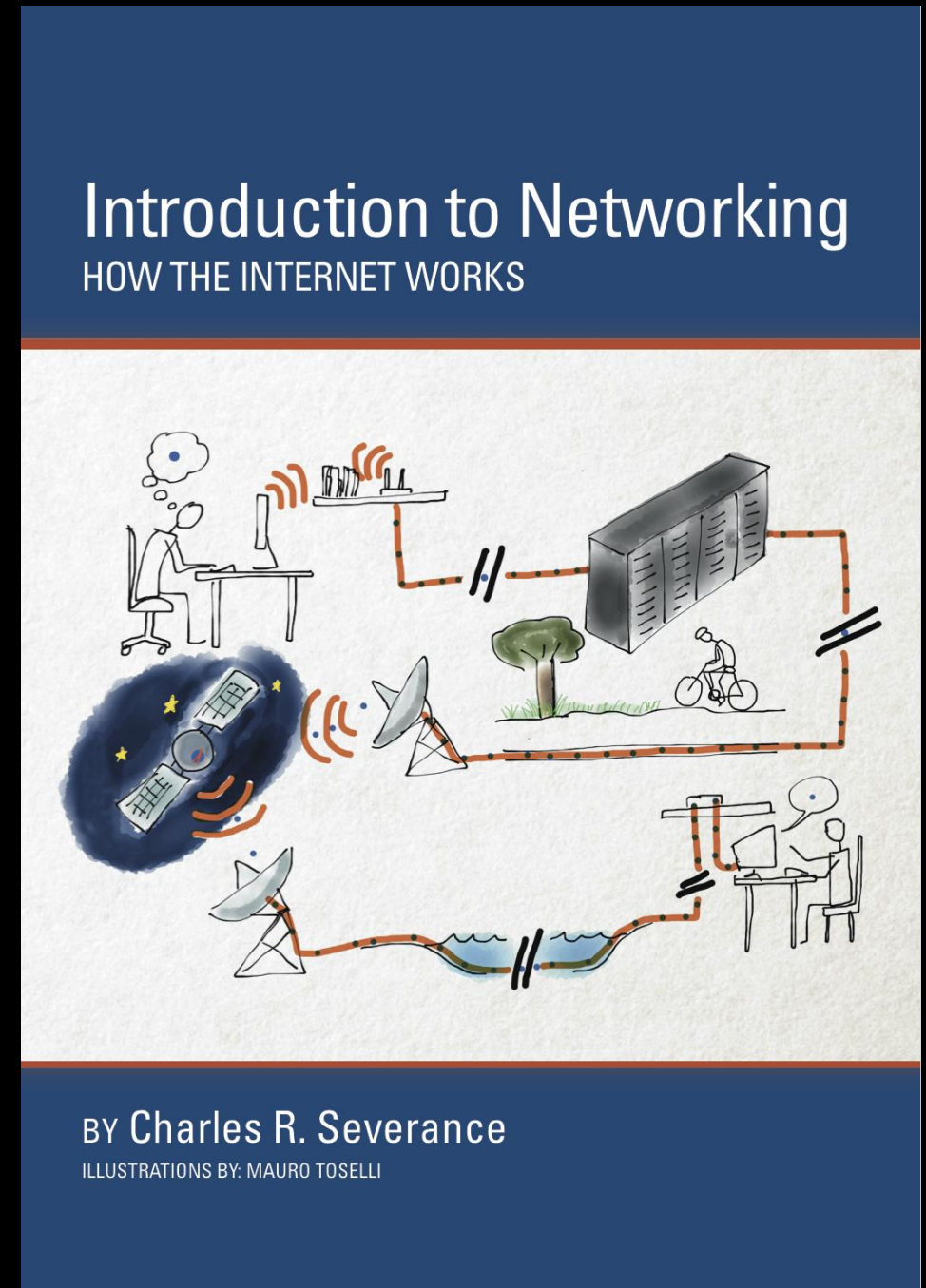


Python for Everybody
www.py4e.com



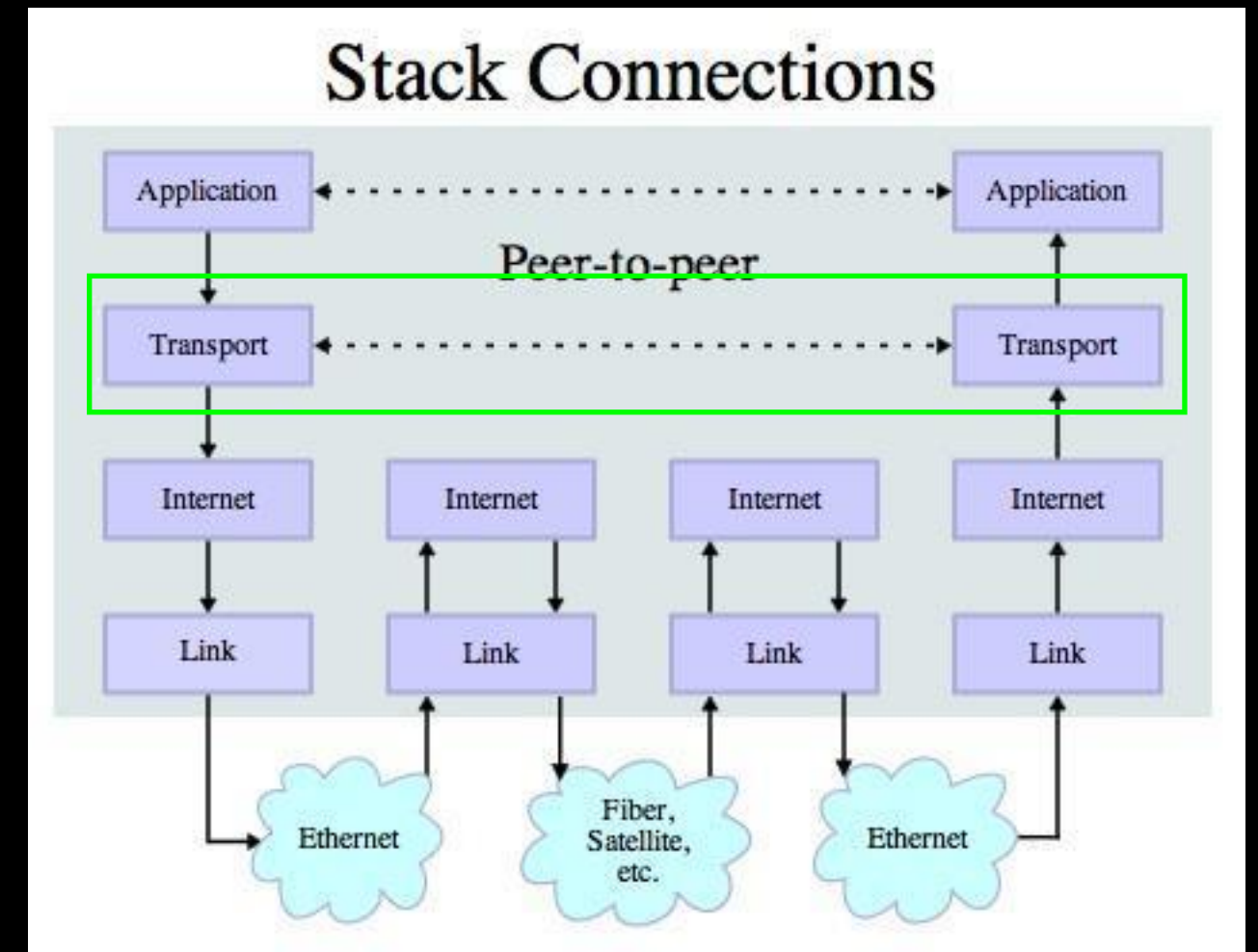
A Free Book on Network Architecture

- If you find this topic area interesting and/or need more detail
- www.net-intro.com

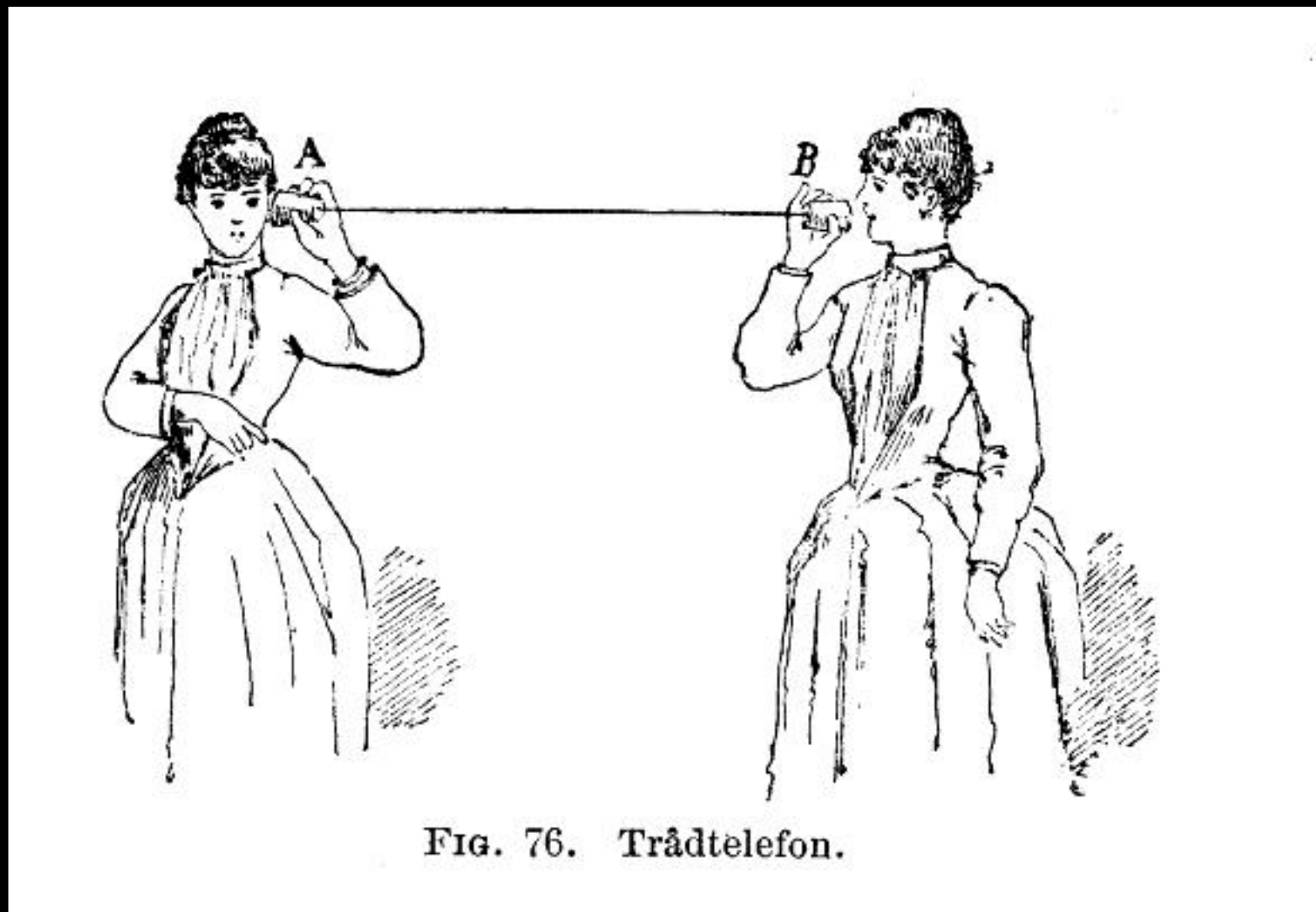


Transport Control Protocol (TCP)

- Built on top of IP (Internet Protocol)
- Assumes IP might lose some data
 - stores and retransmits data if it seems to be lost
- Handles “flow control” using a transmit window
- Provides a nice reliable **pipe**



Source: http://en.wikipedia.org/wiki/Internet_Protocol_Suite



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

<http://www.flickr.com/photos/kitcowan/2103850699/>

TCP Connections / Sockets

“In computer networking, an Internet **socket** or network **socket** is an endpoint of a bidirectional **inter-process** communication flow across an **Internet** Protocol-based computer network, such as the **Internet**.”

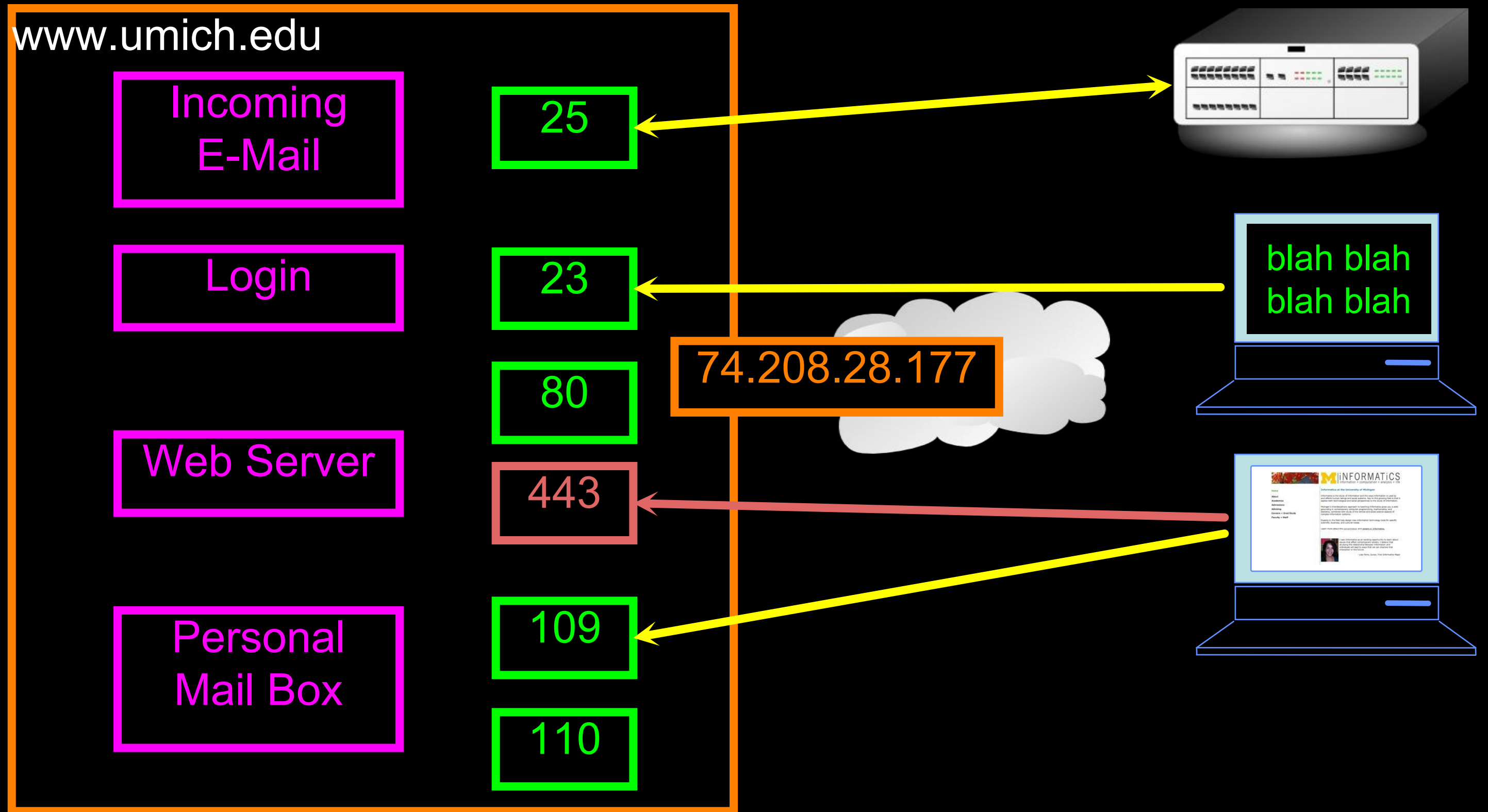


http://en.wikipedia.org/wiki/Internet_socket

TCP Port Numbers

- A port is an **application-specific** or process-specific software communications endpoint
- It allows multiple networked applications to coexist on the same server
- There is a list of well-known TCP port numbers

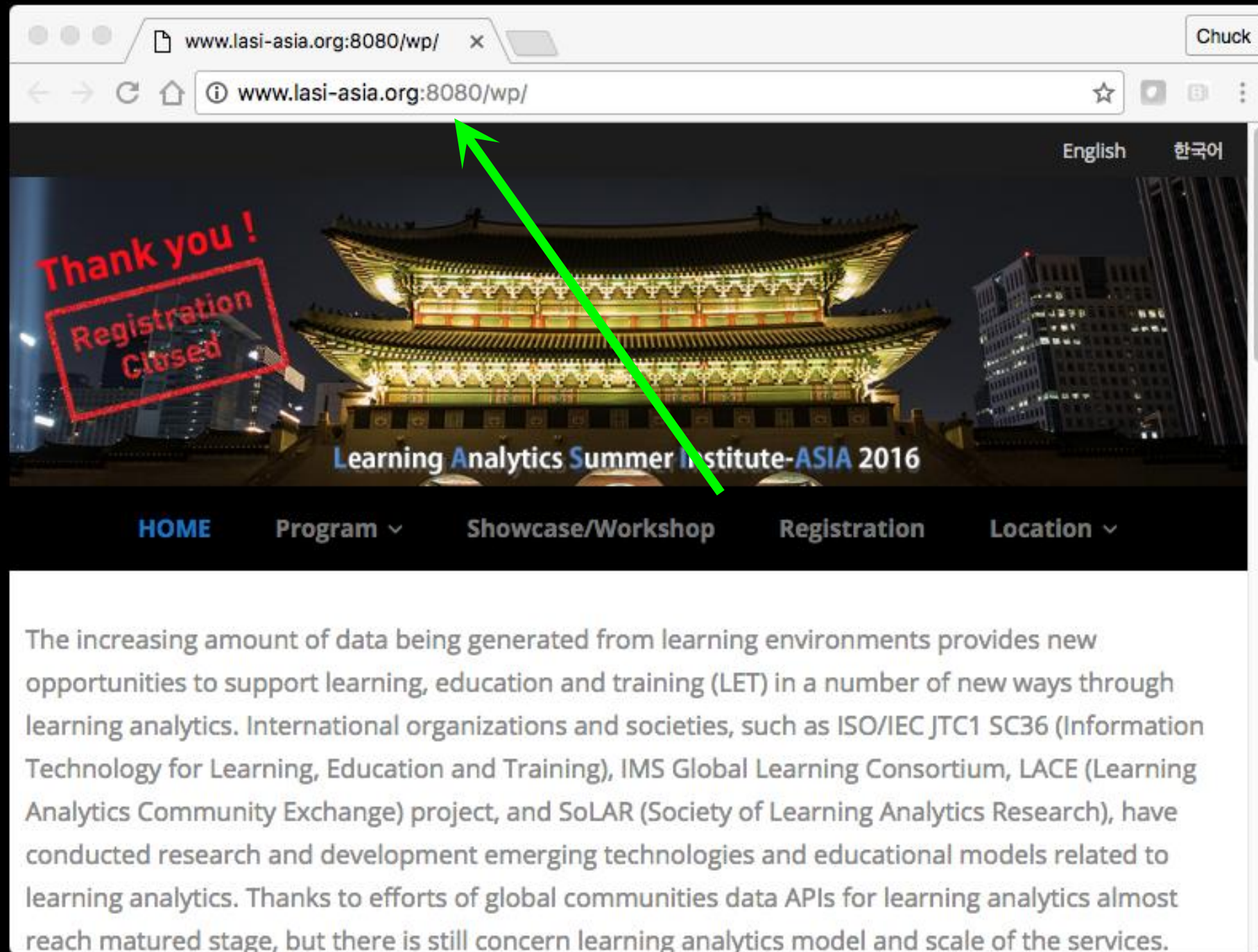
http://en.wikipedia.org/wiki/TCP_and_UDP_port



Common TCP Ports

- Telnet (23) - Login
- SSH (22) - Secure Login
- HTTP (80)
- HTTPS (443) - Secure
- SMTP (25) (Mail)
- IMAP (143/220/993) - Mail Retrieval
- POP (109/110) - Mail Retrieval
- DNS (53) - Domain Name
- FTP (21) - File Transfer

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



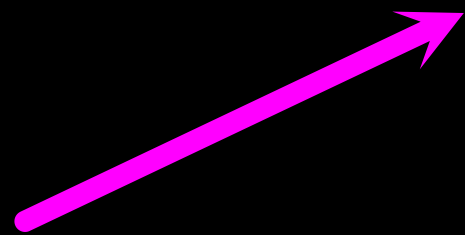
Sometimes we see the port number in the URL if the web server is running on a “non-standard” port.

Sockets in Python

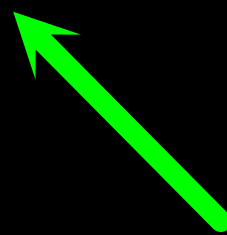
Python has built-in support for TCP Sockets

```
import socket
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect( ('data.pr4e.org', 80) )
```

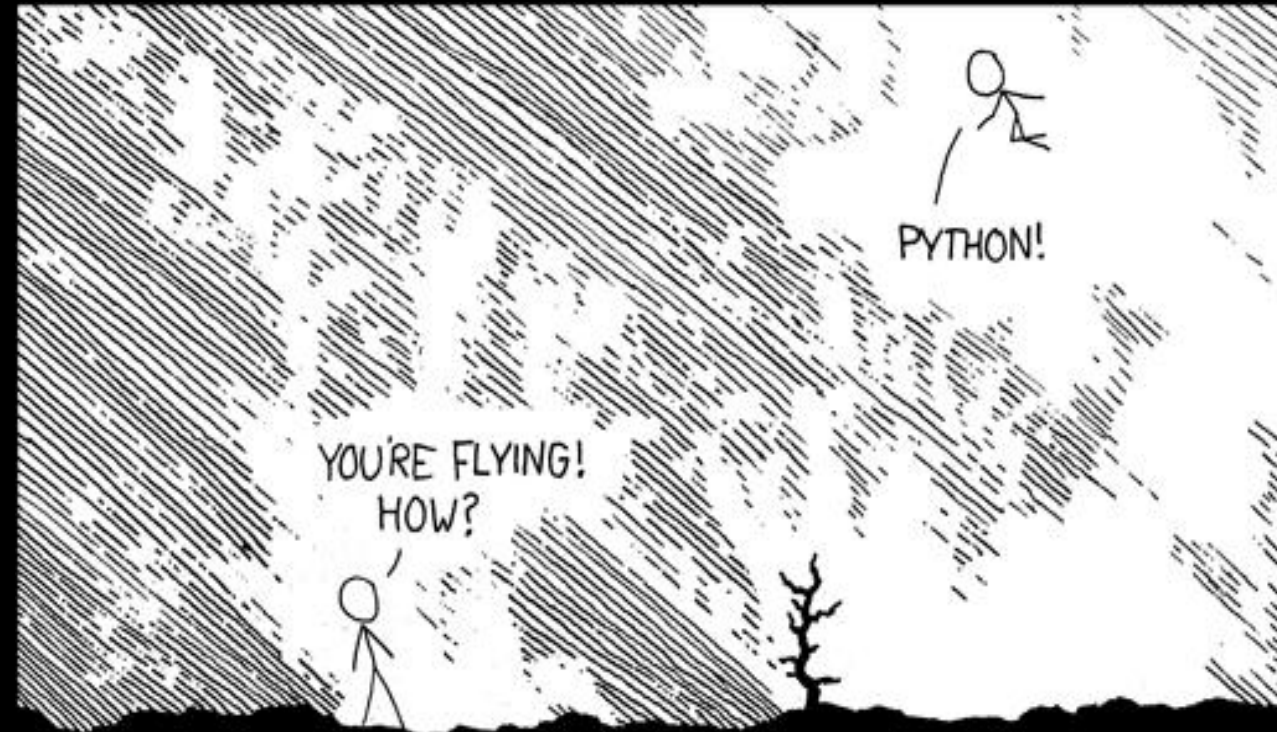
Host



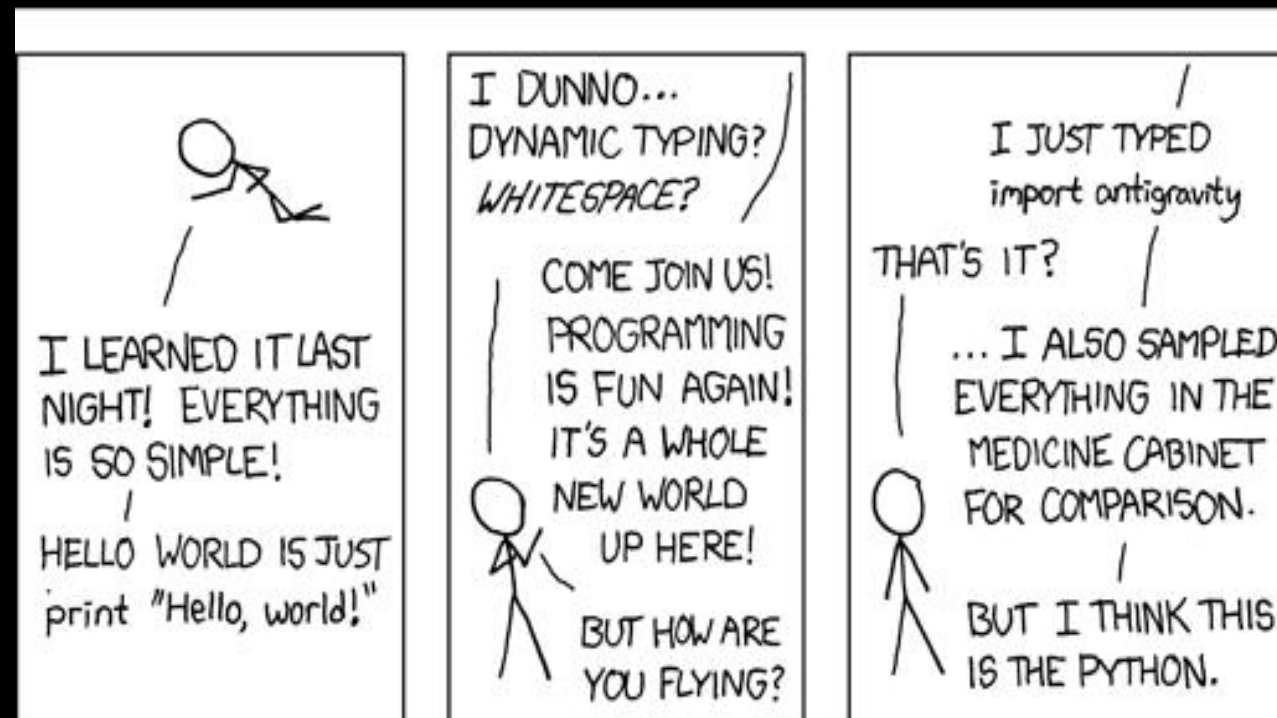
Port



<http://docs.python.org/library/socket.html>



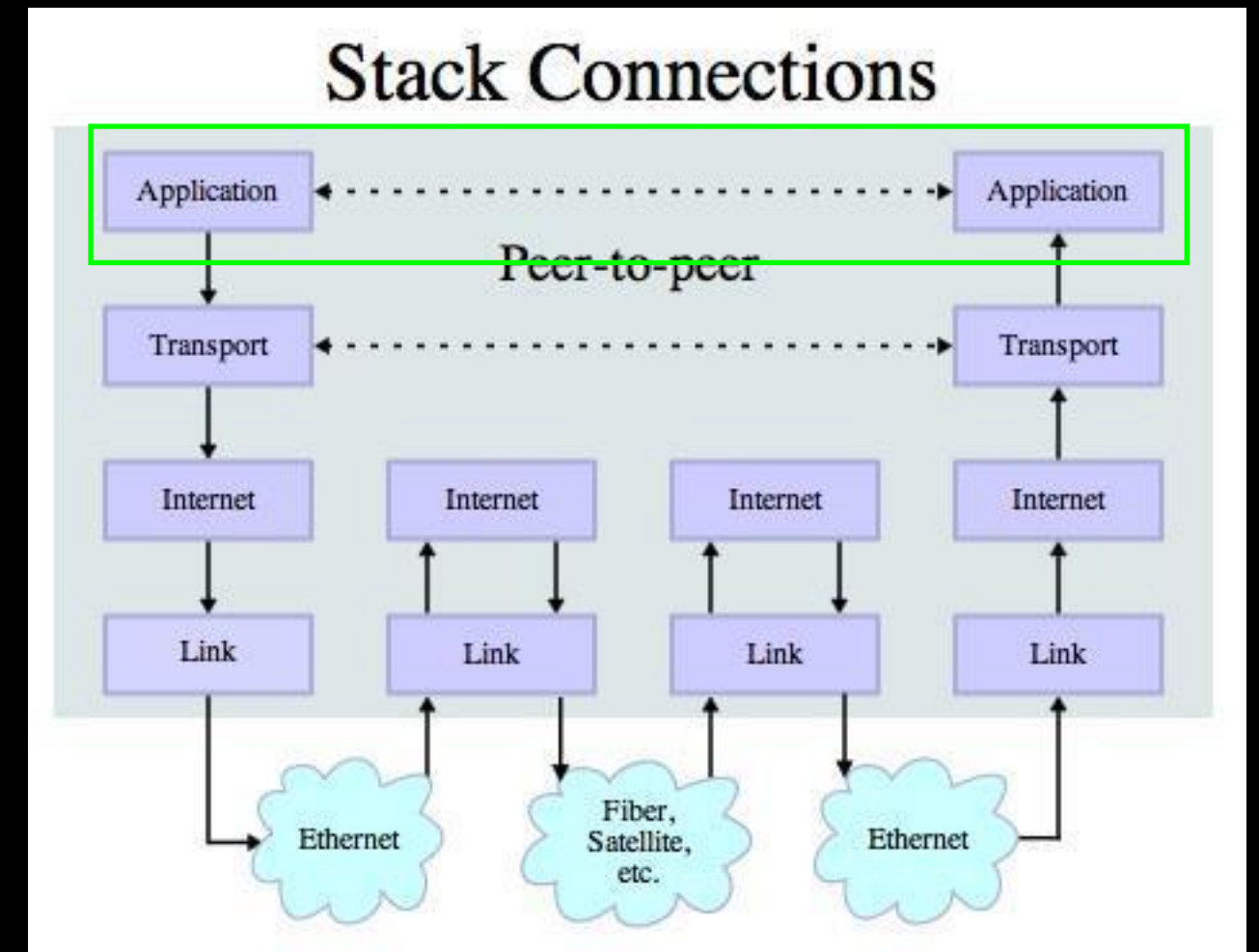
<http://xkcd.com/353/>



Application Protocols

Application Protocol

- Since TCP (and Python) gives us a reliable **socket**, what do we want to do with the **socket**? What problem do we want to solve?
- Application Protocols
 - Mail
 - World Wide Web



Source: http://en.wikipedia.org/wiki/Internet_Protocol_Suite

HTTP - Hypertext Transfer Protocol

- The dominant Application Layer Protocol on the Internet
- Invented for the Web - to Retrieve HTML, Images, Documents, etc.
- Extended to retrieve data in addition to documents - RSS, Web Services, etc. Basic Concept - Make a Connection - Request a document - Retrieve the Document - Close the Connection

<http://en.wikipedia.org/wiki/Http>

HTTP

The HyperText Transfer Protocol is the set of rules to allow browsers to retrieve web documents from servers over the Internet

What is a Protocol?

- A set of rules that all parties follow so we can predict each other's behavior
- And not bump into each other
 - On two-way roads in USA, drive on the right-hand side of the road
 - On two-way roads in the UK, drive on the left-hand side of the road



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

protocol

host

document

<http://www.youtube.com/watch?v=x2GyILq59rI>

1:17 - 2:19



Getting Data From The Server

- Each time the user clicks on an anchor tag with an href= value to switch to a new page, the browser makes a connection to the web server and issues a “GET” request - to GET the content of the page at the specified URL
- The server returns the HTML document to the browser, which formats and displays the document to the user

Web Server

80



Browser



Web Server

80



Browser

Click



Request

Web Server

80

GET http://www.dr-chuck.com/page2.htm

Browser

Click



Request

Web Server

80

GET http://www.dr-chuck.com/page2.htm

Browser

Click



Request

Web Server

80

Response

```
<h1>The Second  
Page</h1><p>If you like, you  
can switch back to the <a  
href="page1.htm">First  
Page</a>.</p>
```

GET http://www.dr-chuck.com/page2.htm

Browser

Click



Request

Web Server

80

Response

```
<h1>The Second  
Page</h1><p>If you like, you  
can switch back to the <a  
href="page1.htm">First  
Page</a>.</p>
```

GET http://www.dr-chuck.com/page2.htm

Browser

Click

Parse/
Render



Internet Standards

- The standards for all of the Internet protocols (inner workings) are developed by an organization
- Internet Engineering Task Force (IETF)
- www.ietf.org
- Standards are called “RFCs” - “Request for Comments”

INTERNET PROTOCOL

DARPA INTERNET PROGRAM

PROTOCOL SPECIFICATION

September 1981

The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise).

The internet protocol uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

Source: <http://tools.ietf.org/html/rfc791>

Network Working Group
Request for Comments: 2616
Obsoletes: 2068
Category: Standards Track

R. Fielding
UC Irvine
J. Gettys
Compaq/W3C
J. Mogul
Compaq
H. Frystyk
W3C/MIT
L. Masinter
Xerox
P. Leach
Microsoft
T. Berners-Lee
W3C/MIT
June 1999

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information

<http://www.w3.org/Protocols/rfc2616/rfc2616.txt>

5 Request

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

```
Request      = Request-Line           ; Section 5.1
               *(( general-header      ; Section 4.5
                  | request-header     ; Section 5.3
                  | entity-header ) CRLF) ; Section 7.1
               CRLF
               [ message-body ]       ; Section 4.3
```

5.1 Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

```
Request-Line  = Method SP Request-URI SP HTTP-Version CRLF
```

Making an HTTP request

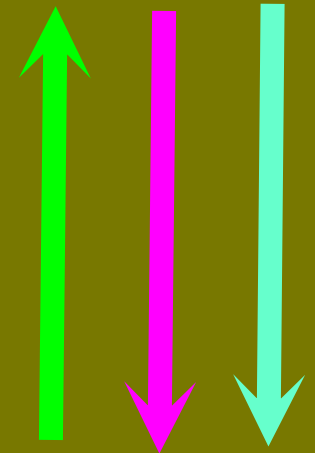
- Connect to the server like www.dr-chuck.com"
- Request a document (or the default document)
 - GET <http://www.dr-chuck.com/page1.htm> HTTP/1.0
 - GET <http://www.mlive.com/ann-arbor/> HTTP/1.0
 - GET <http://www.facebook.com> HTTP/1.0

```
$ telnet www.dr-chuck.com 80
Trying 74.208.28.177...
Connected to www.dr-chuck.com.Escape character is '^]'.
GET http://www.dr-chuck.com/page1.htm HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 08 Jan 2015 01:57:52 GMT
Last-Modified: Sun, 19 Jan 2014 14:25:43 GMT
Connection: close
Content-Type: text/html

<h1>The First Page</h1>
<p>If you like, you can switch to
the <a href="http://www.dr-chuck.com/page2.htm">Second
Page</a>.</p>
Connection closed by foreign host.
```

Web Server



Browser

Accurate Hacking in the Movies

- Matrix Reloaded
- Bourne Ultimatum
- Die Hard 4
- ...

<http://nmap.org/movies.html>



```
80/tcp    open      http
81/tcp    open      hosts2-ns
10.0.0.1  [mobile]
11 # nmap -u -ss -O 10.2.2.2
11
13 Starting nmap V. 2.54BETA25
13 Insufficient responses for TCP sequencing (3). OS detection i
13 accurate
14 Interesting ports on 10.2.2.2:
44 (The 1539 ports scanned but not shown below are in state: cl
51 Port      State      Service
51 22/tcp     open      ssh
58
68 No exact OS matches for host
68
24 Nmap run completed -- 1 IP address (1 host up) scanned
50 # sshnuke 10.2.2.2 -rootpw="Z10H0101"
Connecting to 10.2.2.2:ssh ... successful.
Re Attempting to exploit SSHv1 CRC32 ... successful.
IP Resetting root password to "Z10H0101".
System open: Access level (9)
Hn # ssh 10.2.2.2 -l root
root@10.2.2.2's password:
ACCESS GRANTED
```

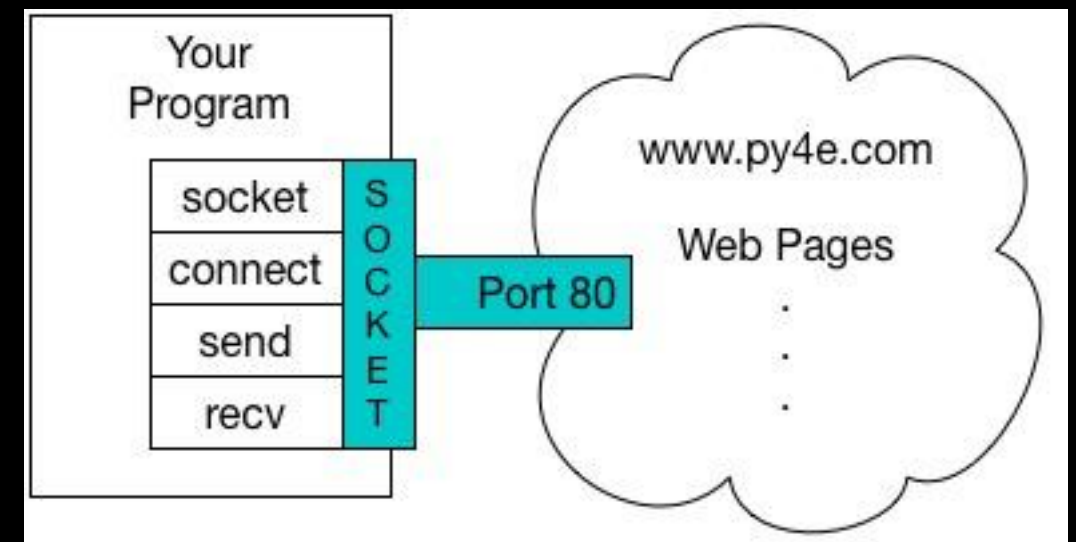
Let's Write a Web Browser!

An HTTP Request in Python

```
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if (len(data) < 1):
        break
    print(data.decode(), end='')
mysock.close()
```



```
HTTP/1.1 200 OK
Date: Sun, 14 Mar 2010 23:52:41 GMT
Server: Apache
Last-Modified: Tue, 29 Dec 2009 01:31:22 GMT
ETag: "143c1b33-a7-4b395bea"
Accept-Ranges: bytes
Content-Length: 167
Connection: close
Content-Type: text/plain
```

```
But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief
```

HTTP Header

```
while True:
    data = mysock.recv(512)
    if ( len(data) < 1 ) :
        break
    print(data.decode())
```

HTTP Body

About Characters and Strings...

ASCII

American
Standard Code
for Information
Interchange

Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char
0	0x00	000	00000000	NUL	32	0x20	040	01000000	space	64	0x40	100	10000000	@	96	0x60	140	11000000	`
1	0x01	001	00000001	SOH	33	0x21	041	01000001	!	65	0x41	101	10000001	A	97	0x61	141	11000001	a
2	0x02	002	00000010	STX	34	0x22	042	01000010	"	66	0x42	102	10000010	B	98	0x62	142	11000010	b
3	0x03	003	00000011	ETX	35	0x23	043	01000011	#	67	0x43	103	10000011	C	99	0x63	143	11000011	c
4	0x04	004	00000100	EOT	36	0x24	044	01000100	\$	68	0x44	104	10000100	D	100	0x64	144	11000100	d
5	0x05	005	00000101	ENQ	37	0x25	045	01000101	%	69	0x45	105	10000101	E	101	0x65	145	11000101	e
6	0x06	006	00000110	ACK	38	0x26	046	01000110	&	70	0x46	106	10000110	F	102	0x66	146	11000110	f
7	0x07	007	00000111	BEL	39	0x27	047	01000111	'	71	0x47	107	10000111	G	103	0x67	147	11000111	g
8	0x08	010	00010000	BS	40	0x28	050	01010000	(72	0x48	110	10010000	H	104	0x68	150	11010000	h
9	0x09	011	00010001	TAB	41	0x29	051	01010001)	73	0x49	111	10010001	I	105	0x69	151	11010001	i
10	0x0A	012	00010010	LF	42	0x2A	052	01010010	*	74	0x4A	112	10010010	J	106	0x6A	152	11010010	j
11	0x0B	013	00010011	VT	43	0x2B	053	01010011	+	75	0x4B	113	10010011	K	107	0x6B	153	11010011	k
12	0x0C	014	00010100	FF	44	0x2C	054	01010100	,	76	0x4C	114	10010100	L	108	0x6C	154	11010100	l
13	0x0D	015	00010101	CR	45	0x2D	055	01010101	-	77	0x4D	115	10010101	M	109	0x6D	155	11010101	m
14	0x0E	016	00010110	SO	46	0x2E	056	01010110	.	78	0x4E	116	10010110	N	110	0x6E	156	11010110	n
15	0x0F	017	00010111	SI	47	0x2F	057	01010111	/	79	0x4F	117	10010111	O	111	0x6F	157	11010111	o
16	0x10	020	00100000	DLE	48	0x30	060	01100000	0	80	0x50	120	10100000	P	112	0x70	160	11100000	p
17	0x11	021	00100001	DC1	49	0x31	061	01100001	1	81	0x51	121	10100001	Q	113	0x71	161	11100001	q
18	0x12	022	00100010	DC2	50	0x32	062	01100010	2	82	0x52	122	10100010	R	114	0x72	162	11100010	r
19	0x13	023	00100011	DC3	51	0x33	063	01100011	3	83	0x53	123	10100011	S	115	0x73	163	11100011	s
20	0x14	024	00100100	DC4	52	0x34	064	01100100	4	84	0x54	124	10100100	T	116	0x74	164	11100100	t
21	0x15	025	00100101	NAK	53	0x35	065	01100101	5	85	0x55	125	10100101	U	117	0x75	165	11100101	u
22	0x16	026	00100110	SYN	54	0x36	066	01100110	6	86	0x56	126	10100110	V	118	0x76	166	11100110	v
23	0x17	027	00100111	ETB	55	0x37	067	01100111	7	87	0x57	127	10100111	W	119	0x77	167	11100111	w
24	0x18	030	00110000	CAN	56	0x38	070	01110000	8	88	0x58	130	10110000	X	120	0x78	170	11110000	x
25	0x19	031	00110001	EM	57	0x39	071	01110001	9	89	0x59	131	10110001	Y	121	0x79	171	11110001	y
26	0x1A	032	00110010	SUB	58	0x3A	072	01110010	:	90	0x5A	132	10110010	Z	122	0x7A	172	11110010	z
27	0x1B	033	00110011	ESC	59	0x3B	073	01110011	;	91	0x5B	133	10110011	[123	0x7B	173	11110011	{
28	0x1C	034	00110100	FS	60	0x3C	074	01110100	<	92	0x5C	134	10110100	\	124	0x7C	174	11110100	
29	0x1D	035	00110101	GS	61	0x3D	075	01110101	=	93	0x5D	135	10110101]	125	0x7D	175	11110101	}
30	0x1E	036	00110110	RS	62	0x3E	076	01110110	>	94	0x5E	136	10110110	^	126	0x7E	176	11110110	~
31	0x1F	037	00110111	US	63	0x3F	077	01110111	?	95	0x5F	137	10110111	_	127	0x7F	177	11110111	DEL

<https://en.wikipedia.org/wiki/ASCII>

<http://www.catonmat.net/download/ascii-cheat-sheet.png>

Representing Simple Strings

- Each character is represented by a number between 0 and 256 stored in 8 bits of memory
- We refer to "8 bits of memory as a **"byte"** of memory – (i.e. my disk drive contains 3 Terabytes of memory)
- The **ord()** function tells us the numeric value of a simple ASCII character

```
>>> print(ord('H'))
72
>>> print(ord('e'))
101
>>> print(ord('\n'))
10
>>>
```


ASCII

```
>>> print(ord('H'))  
72  
>>> print(ord('e'))  
101  
>>> print(ord('\n'))  
10  
>>>
```

In the 1960s and 1970s,
we just assumed that
one byte was one
character

Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char
0	0x00	000	00000000	NUL	32	0x20	040	01000000	space	64	0x40	100	10000000	@	96	0x60	140	11000000	`
1	0x01	001	00000001	SOH	33	0x21	041	01000001	!	65	0x41	101	10000001	A	97	0x61	141	11000001	a
2	0x02	002	00000010	STX	34	0x22	042	01000010	"	66	0x42	102	10000010	B	98	0x62	142	11000010	b
3	0x03	003	00000011	ETX	35	0x23	043	01000011	#	67	0x43	103	10000011	C	99	0x63	143	11000011	c
4	0x04	004	00000100	EOT	36	0x24	044	01000100	\$	68	0x44	104	10000100	D	100	0x64	144	11000100	d
5	0x05	005	00000101	ENQ	37	0x25	045	01000101	%	69	0x45	105	10000101	E	101	0x65	145	11000101	e
6	0x06	006	00000110	ACK	38	0x26	046	01000110	&	70	0x46	106	10000110	F	102	0x66	146	11000110	f
7	0x07	007	00000111	BEL	39	0x27	047	01000111	'	71	0x47	107	10000111	G	103	0x67	147	11000111	g
8	0x08	010	00010000	BS	40	0x28	050	01010000	(72	0x48	110	10010000	H	104	0x68	150	11010000	h
9	0x09	011	00010001	TAB	41	0x29	051	01010001)	73	0x49	111	10010001	I	105	0x69	151	11010001	i
10	0x0A	012	00010010	LF	42	0x2A	052	01010010	*	74	0x4A	112	10010010	J	106	0x6A	152	11010010	j
11	0x0B	013	00010011	VT	43	0x2B	053	01010011	+	75	0x4B	113	10010011	K	107	0x6B	153	11010011	k
12	0x0C	014	00010100	FF	44	0x2C	054	01010100	,	76	0x4C	114	10010100	L	108	0x6C	154	11010100	l
13	0x0D	015	00010101	CR	45	0x2D	055	01010101	-	77	0x4D	115	10010101	M	109	0x6D	155	11010101	m
14	0x0E	016	00010110	SO	46	0x2E	056	01010110	.	78	0x4E	116	10010110	N	110	0x6E	156	11010110	n
15	0x0F	017	00010111	SI	47	0x2F	057	01010111	/	79	0x4F	117	10010111	O	111	0x6F	157	11010111	o
16	0x10	020	00100000	DLE	48	0x30	060	01100000	0	80	0x50	120	10100000	P	112	0x70	160	11100000	p
17	0x11	021	00100001	DC1	49	0x31	061	01100001	1	81	0x51	121	10100001	Q	113	0x71	161	11100001	q
18	0x12	022	00100010	DC2	50	0x32	062	01100010	2	82	0x52	122	10100010	R	114	0x72	162	11100010	r
19	0x13	023	00100011	DC3	51	0x33	063	01100011	3	83	0x53	123	10100011	S	115	0x73	163	11100011	s
20	0x14	024	00100100	DC4	52	0x34	064	01100100	4	84	0x54	124	10100100	T	116	0x74	164	11100100	t
21	0x15	025	00100101	NAK	53	0x35	065	01100101	5	85	0x55	125	10100101	U	117	0x75	165	11100101	u
22	0x16	026	00100110	SYN	54	0x36	066	01100110	6	86	0x56	126	10100110	V	118	0x76	166	11100110	v
23	0x17	027	00100111	ETB	55	0x37	067	01100111	7	87	0x57	127	10100111	W	119	0x77	167	11100111	w
24	0x18	030	00110000	CAN	56	0x38	070	01110000	8	88	0x58	130	10110000	X	120	0x78	170	11110000	x
25	0x19	031	00110001	EM	57	0x39	071	01110001	9	89	0x59	131	10110001	Y	121	0x79	171	11110001	y
26	0x1A	032	00110010	SUB	58	0x3A	072	01110010	:	90	0x5A	132	10110010	Z	122	0x7A	172	11110010	z
27	0x1B	033	00110011	ESC	59	0x3B	073	01110011	;	91	0x5B	133	10110011	[123	0x7B	173	11110011	{
28	0x1C	034	00110100	FS	60	0x3C	074	01110100	<	92	0x5C	134	10110100	\	124	0x7C	174	11110100	
29	0x1D	035	00110101	GS	61	0x3D	075	01110101	=	93	0x5D	135	10110101]	125	0x7D	175	11110101	}
30	0x1E	036	00110110	RS	62	0x3E	076	01110110	>	94	0x5E	136	10110110	^	126	0x7E	176	11110110	~
31	0x1F	037	00110111	US	63	0x3F	077	01110111	?	95	0x5F	137	10110111	_	127	0x7F	177	11110111	DEL



Unicode 9.0 Character Code Charts

[SCRIPTS](#) | [SYMBOLS](#) | [NOTES](#)<http://unicode.org/charts/>Find chart by hex code: Related links: [Name index](#) [Help & links](#)

Scripts

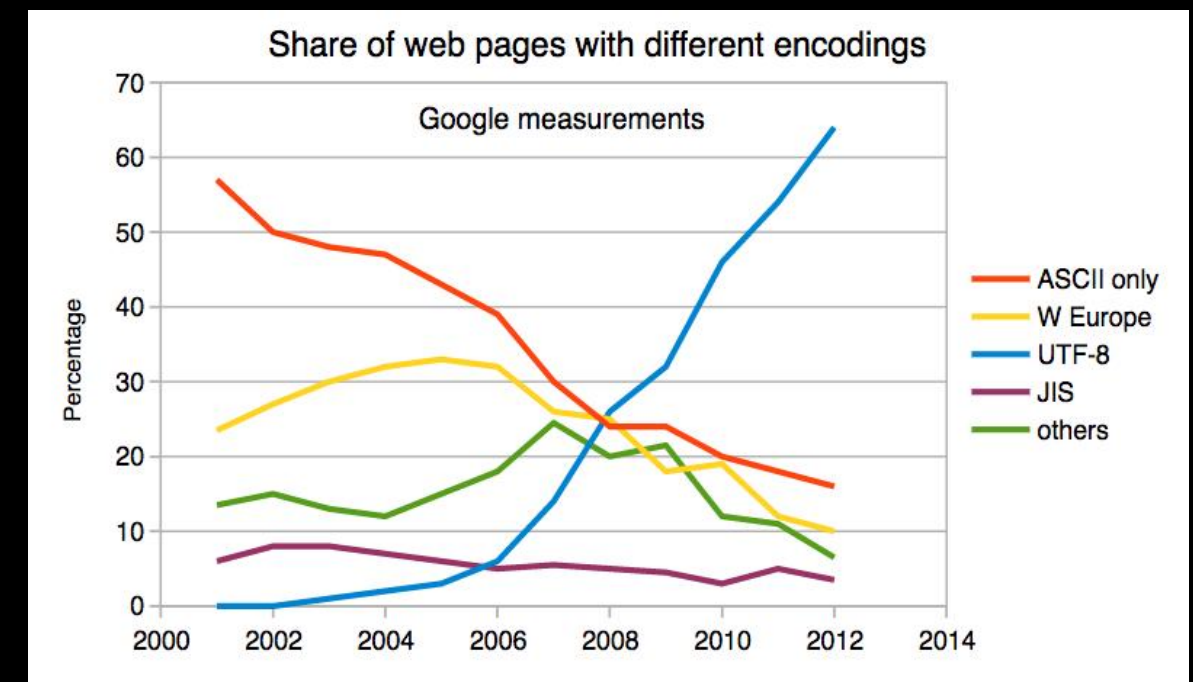
European Scripts	African Scripts	South Asian Scripts	Indonesia & Oceania Scripts
Armenian	Adlam	Ahom	Balinese
Armenian Ligatures	Bamum	Bengali and Assamese	Batak
Caucasian Albanian	Bamum Supplement	Bhaiksuki	Buginese
Cypriot Syllabary	Bassa Vah	Brahmi	Buhid
Cyrillic	Coptic	Chakma	Hanunoo
Cyrillic Supplement	Coptic in Greek block	Devanagari	Javanese
Cyrillic Extended-A	Coptic Epact Numbers	Devanagari Extended	Rejang
Cyrillic Extended-B	Egyptian Hieroglyphs (1MB)	Grantha	Sundanese
Cyrillic Extended-C	Ethiopic	Gujarati	Sundanese Supplement
Elbasan	Ethiopic Supplement	Gurmukhi	Tagalog
Georgian	Ethiopic Extended	Kaithi	Tagbanwa
Georgian Supplement	Ethiopic Extended-A	Kannada	East Asian Scripts
Glagolitic	Mende Kikakui	Kharoshthi	Bopomofo
Glagolitic Supplement	Meroitic	Khojki	Bopomofo Extended
Gothic	Meroitic Cursive	Khudawadi	CJK Unified Ideographs (Han) (35MB)
Greek	Meroitic Hieroglyphs	Lepcha	CJK Extension-A (6MB)
Greek Extended	N'Ko	Limbu	CJK Extension B (40MB)
Ancient Greek Numbers	Osmanya	Mahajani	CJK Extension C (3MB)
Latin	Tifinagh	Malayalam	CJK Extension D
Basic Latin (ASCII)	Vai	Meetei Mayek	CJK Extension E (3.5MB)
Latin-1 Supplement	Middle Eastern Scripts	Meetei Mayek Extensions	(see also UniHan Database)
Latin Extended-A	Anatolian Hieroglyphs	Modi	CJK Compatibility Ideographs

Multi-Byte Characters

To represent the wide range of characters computers must handle we represent characters with more than one byte

- UTF-16 – Fixed length - Two bytes
- UTF-32 – Fixed Length - Four Bytes
- **UTF-8** – 1-4 bytes
 - Upwards compatible with ASCII
 - Automatic detection between ASCII and UTF-8
 - **UTF-8 is recommended practice for encoding data to be exchanged between systems**

<https://en.wikipedia.org/wiki/UTF-8>



Two Kinds of Strings in Python

Python 2.7.10

```
>>> x = '이광춘'
>>> type(x)
<type 'str'>
>>> x = u'이광춘'
>>> type(x)
<type 'unicode'>
>>>
```

Python 3.5.1

```
>>> x = '이광춘'
>>> type(x)
<class 'str'>
>>> x = u'이광춘'
>>> type(x)
<class 'str'>
>>>
```

In Python 3, all strings are Unicode

Python 2 versus Python 3

Python 2.7.10

```
>>> x = b'abc'
```

```
>>> type(x)
```

```
<type 'str'>
```

```
>>> x = '이광춘'
```

```
>>> type(x)
```

```
<type 'str'>
```

```
>>> x = u'이광춘'
```

```
>>> type(x)
```

```
<type 'unicode'>
```

Python 3.5.1

```
>>> x = b'abc'
```

```
>>> type(x)
```

```
<class 'bytes'>
```

```
>>> x = '이광춘'
```

```
>>> type(x)
```

```
<class 'str'>
```

```
>>> x = u'이광춘'
```

```
>>> type(x)
```

```
<class 'str'>
```

Python 3 and Unicode

- In Python 3, all strings internally are UNICODE
- Working with string variables in Python programs and reading data from files usually "just works"
- When we talk to a network resource using sockets or talk to a database we have to encode and decode data (usually to UTF-8)

Python 3.5.1

```
>>> x = b'abc'
```

```
>>> type(x)
```

```
<class 'bytes'>
```

```
>>> x = '이광춘'
```

```
>>> type(x)
```

```
<class 'str'>
```

```
>>> x = u'이광춘'
```

```
>>> type(x)
```

```
<class 'str'>
```


Python Strings to Bytes

- When we talk to an external resource like a network socket we send bytes, so we need to encode Python 3 strings into a given character encoding
- When we read data from an external resource, we must decode it based on the character set so it is properly represented in Python 3 as a string

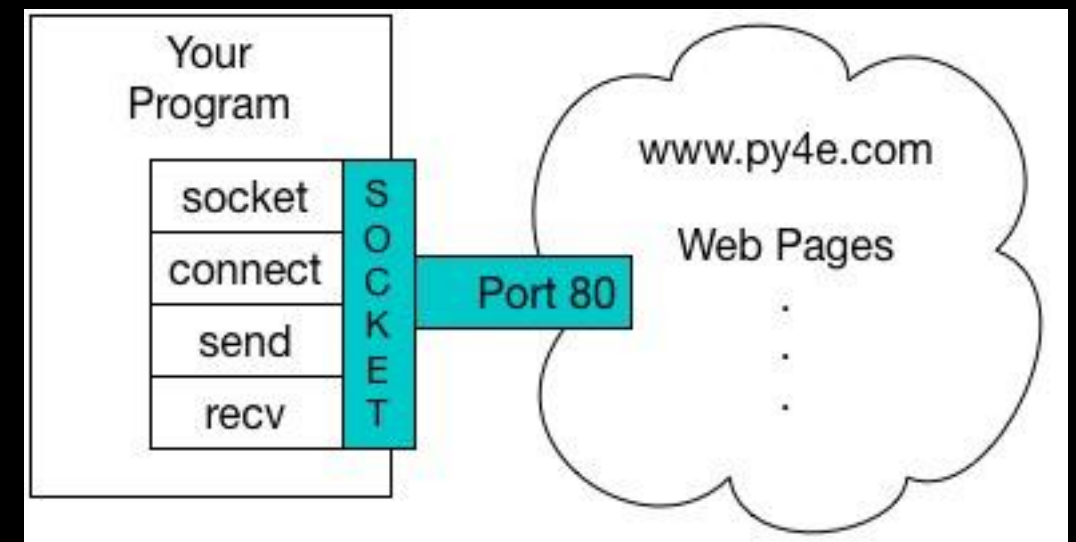
```
while True:
    data = mysock.recv(512)
    if ( len(data) < 1 ) :
        break
    mystring = data.decode()
    print(mystring)
```

An HTTP Request in Python

```
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\n\n'.encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if (len(data) < 1):
        break
    print(data.decode())
mysock.close()
```



```
bytes.decode(encoding="utf-8", errors="strict")
```

```
bytearray.decode(encoding="utf-8", errors="strict")
```

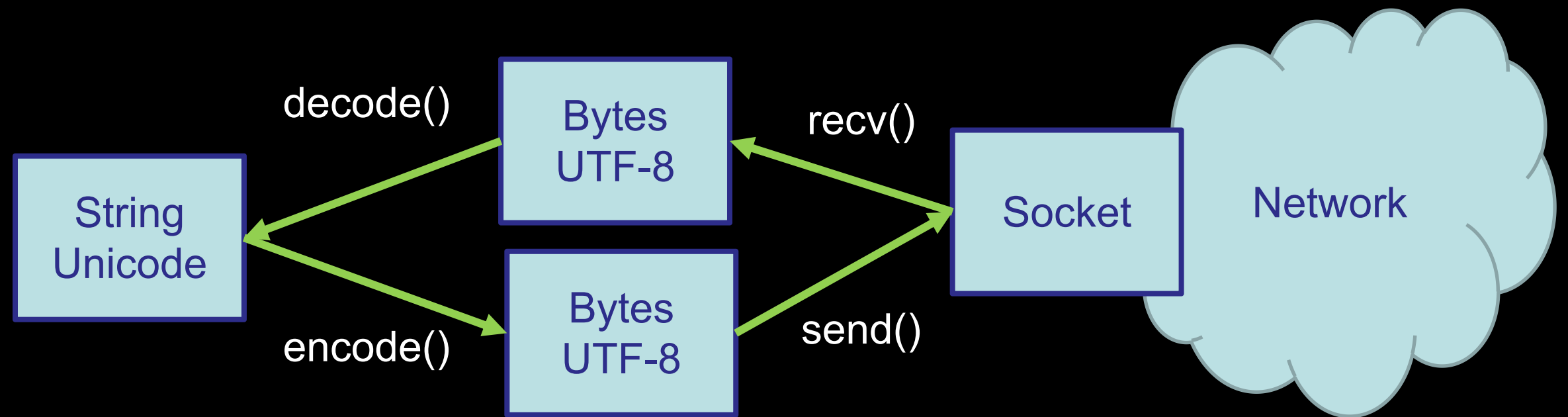
Return a string decoded from the given bytes. Default encoding is 'utf-8'. *errors* may be given to set a different error handling scheme. The default for *errors* is 'strict', meaning that encoding errors raise a `UnicodeError`. Other possible values are 'ignore', 'replace' and any other name registered via `codecs.register_error()`, see section [Error Handlers](#). For a list of possible encodings, see section [Standard Encodings](#).

```
str.encode(encoding="utf-8", errors="strict")
```

Return an encoded version of the string as a bytes object. Default encoding is 'utf-8'. *errors* may be given to set a different error handling scheme. The default for *errors* is 'strict', meaning that encoding errors raise a `UnicodeError`. Other possible values are 'ignore', 'replace', 'xmlcharrefreplace', 'backslashreplace' and any other name registered via `codecs.register_error()`, see section [Error Handlers](#). For a list of possible encodings, see section [Standard Encodings](#).

<https://docs.python.org/3/library/stdtypes.html#bytes.decode>

<https://docs.python.org/3/library/stdtypes.html#str.encode>



```
import socket
```

```
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\n\n'.encode()
mysock.send(cmd)
```

```
while True:
    data = mysock.recv(512)
    if (len(data) < 1):
        break
    print(data.decode())
mysock.close()
```

Making HTTP Easier With urllib

Using `urllib` in Python

Since HTTP is so common, we have a library that does all the socket work for us and makes web pages look like a file

```
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
for line in fhand:
    print(line.decode().strip())
```

`urllib1.py`


```
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
for line in fhand:
    print(line.decode().strip())
```

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

urllib1.py

Like a File...

```
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')

counts = dict()
for line in fhand:
    words = line.decode().split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1
print(counts)
```

urlwords.py

Reading Web Pages

```
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://www.dr-chuck.com/page1.htm')
for line in fhand:
    print(line.decode().strip())
```

```
<h1>The First Page</h1>
<p>If you like, you can switch to the <a
href="http://www.dr-chuck.com/page2.htm">Second
Page</a>.
</p>
```

urllib2.py

Following Links

```
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://www.dr-chuck.com/page1.htm')
for line in fhand:
    print(line.decode().strip())
```

```
<h1>The First Page</h1>
<p>If you like, you can switch to the <a
href="http://www.dr-chuck.com/page2.htm">Second
Page</a>.
</p>
```

urllib2.py

The First Lines of Code @ Google?

```
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://www.dr-chuck.com/page1.htm')
for line in fhand:
    print(line.decode().strip())
```

Parsing HTML (a.k.a. Web Scraping)

What is Web Scrapping?

- When a program or script pretends to be a browser and retrieves web pages, looks at those web pages, extracts information, and then looks at more web pages
- Search engines scrape web pages - we call this “spidering the web” or “web crawling”

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

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

Why Scrape?

- Pull data - particularly social data - who links to who?
- Get your own data back out of some system that has no “export capability”
- Monitor a site for new information
- Spider the web to make a database for a search engine

Scraping Web Pages

- There is some controversy about web page scraping and some sites are a bit snippy about it.
- Republishing copyrighted information is not allowed
- Violating terms of service is not allowed

The Easy Way - Beautiful Soup

- You could do string searches the hard way
- Or use the free software library called **BeautifulSoup** from www.crummy.com

You didn't write that awful page. You're just trying to get some data out of it. BeautifulSoup is here to help. Since 2004, it's been saving programmers hours or days of work on quick-turnaround screen scraping projects.

Beautiful Soup

"A tremendous boon." -- Python411 Podcast

[[Download](#) | [Documentation](#) | [Hall of Fame](#) | [Source](#) | [Discussion group](#)]

If BeautifulSoup has saved you a lot of time and money, the best way to pay me back is to check out [Constellation Games](#), my sci-fi novel about alien video games.

You can [read the first two chapters for free](#), and the full novel starts at 5 USD. Thanks!

If you have questions, send them to [the discussion group](#). If you find a bug, [file it](#).



<https://www.crummy.com/software/BeautifulSoup/>

BeautifulSoup Installation

```
# To run this, you can install BeautifulSoup
# https://pypi.python.org/pypi/beautifulsoup4

# Or download the file
# http://www.py4e.com/code3/bs4.zip
# and unzip it in the same directory as this file

import urllib.request, urllib.parse, urllib.error
from bs4 import BeautifulSoup

...
```

urllinks.py

```
import urllib.request, urllib.parse,
urllib.error
from bs4 import BeautifulSoup

url = input('Enter - ')
html = urllib.request.urlopen(url).read()
soup = BeautifulSoup(html, 'html.parser')

# Retrieve all of the anchor tags
tags = soup('a')
for tag in tags:
    print(tag.get('href', None))
```

python urlinks.py

Enter - **<http://www.dr-chuck.com/page1.htm>**

<http://www.dr-chuck.com/page2.htm>

Summary

- The TCP/IP gives us pipes / sockets between applications
- We designed application protocols to make use of these pipes
- HyperText Transfer Protocol (HTTP) is a simple yet powerful protocol
- Python has good support for sockets, HTTP, and HTML parsing



Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and open.umich.edu and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

... Insert new Contributors here