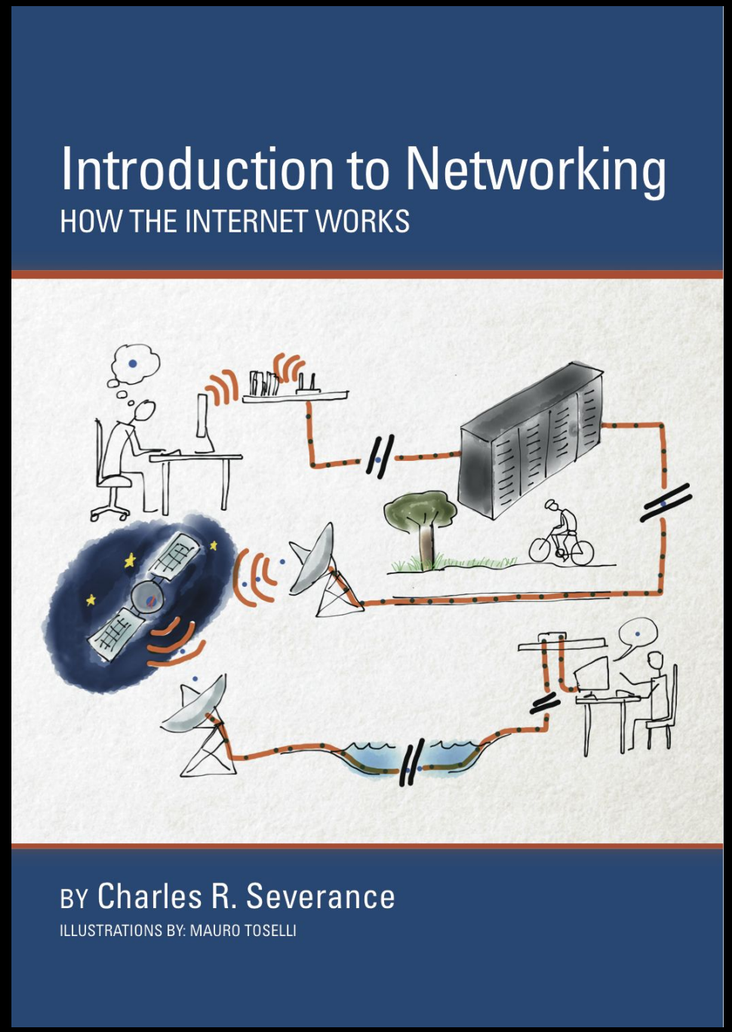
네트워크에 연결된 프로그램

제12장

Python for Everybody
www.py4e.com

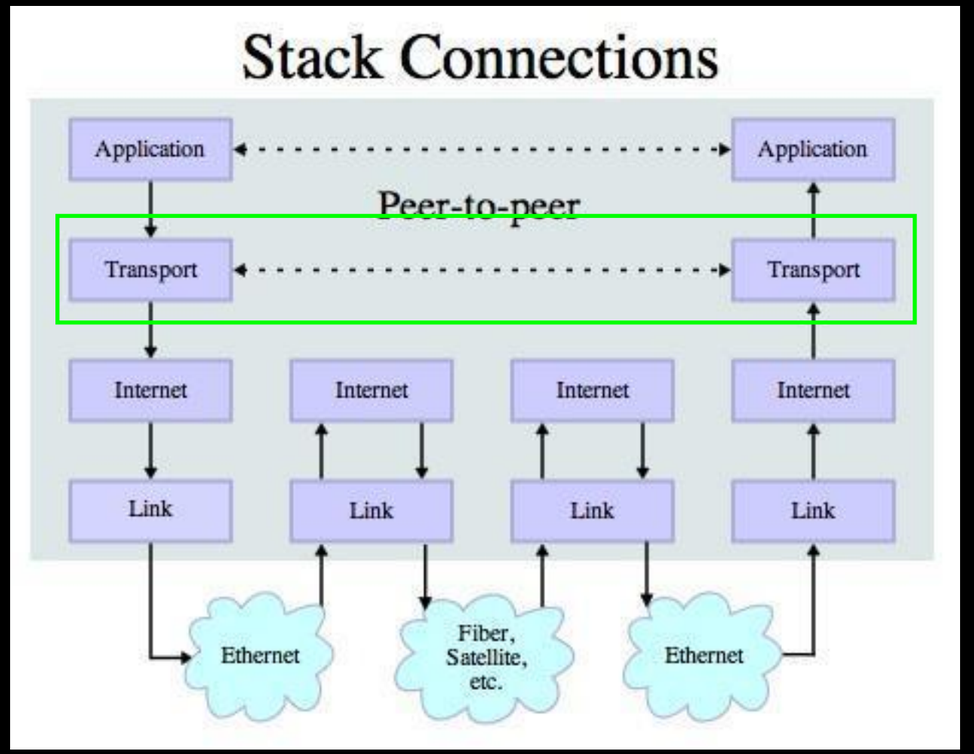
책 소개

- 만약 네트워크에 대해 좀 더 알아보고 싶다면 아래를 참고
- www.net-intro.com



전송 제어 프로토콜 (TCP)

- IP(인터넷 프로토콜) 위에서 구성
- IP는 데이터를 잃어버릴 수 있음 - 데이터를 저장하고 있다가 손실이 일어난 것으로 추정되면 재전송
- 전송 윈도우를 통해 흐름 제어을 조절
- 믿을만한 pipe 역할을 제공



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

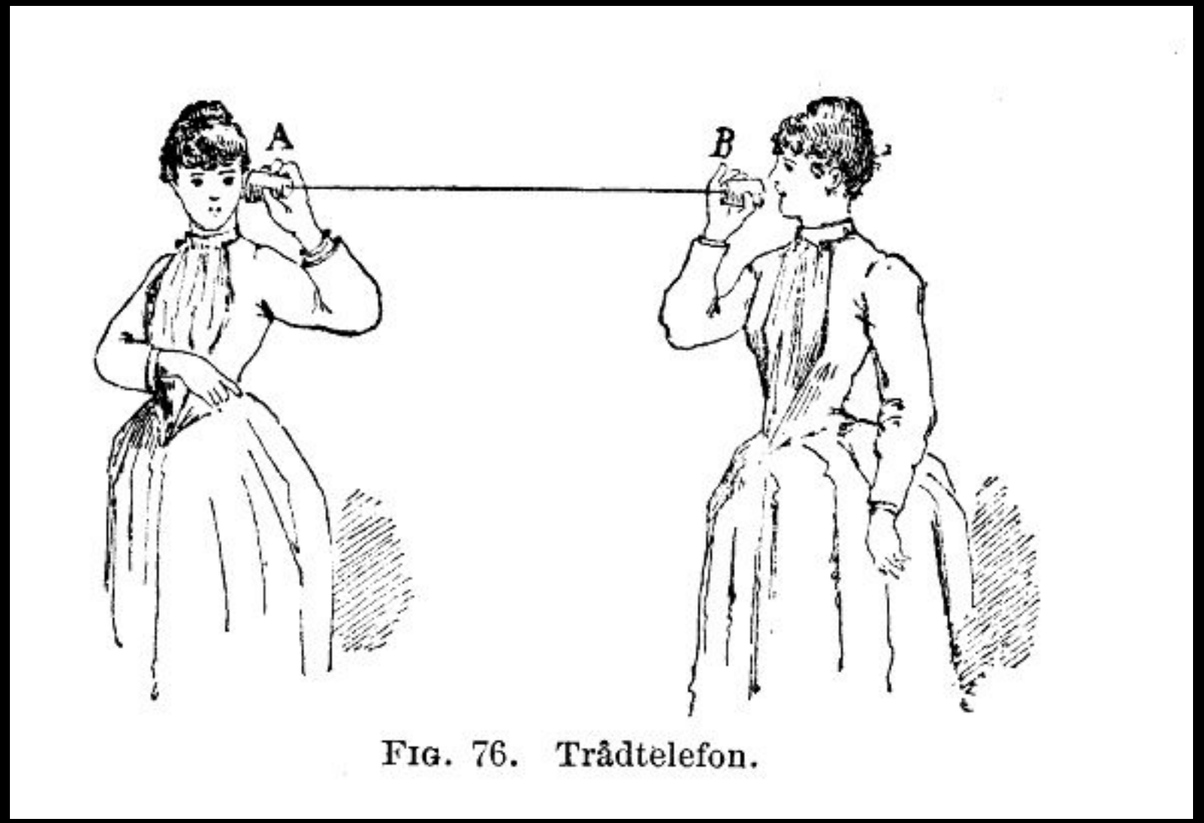


Fig. 76. Trådtelefon.

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

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

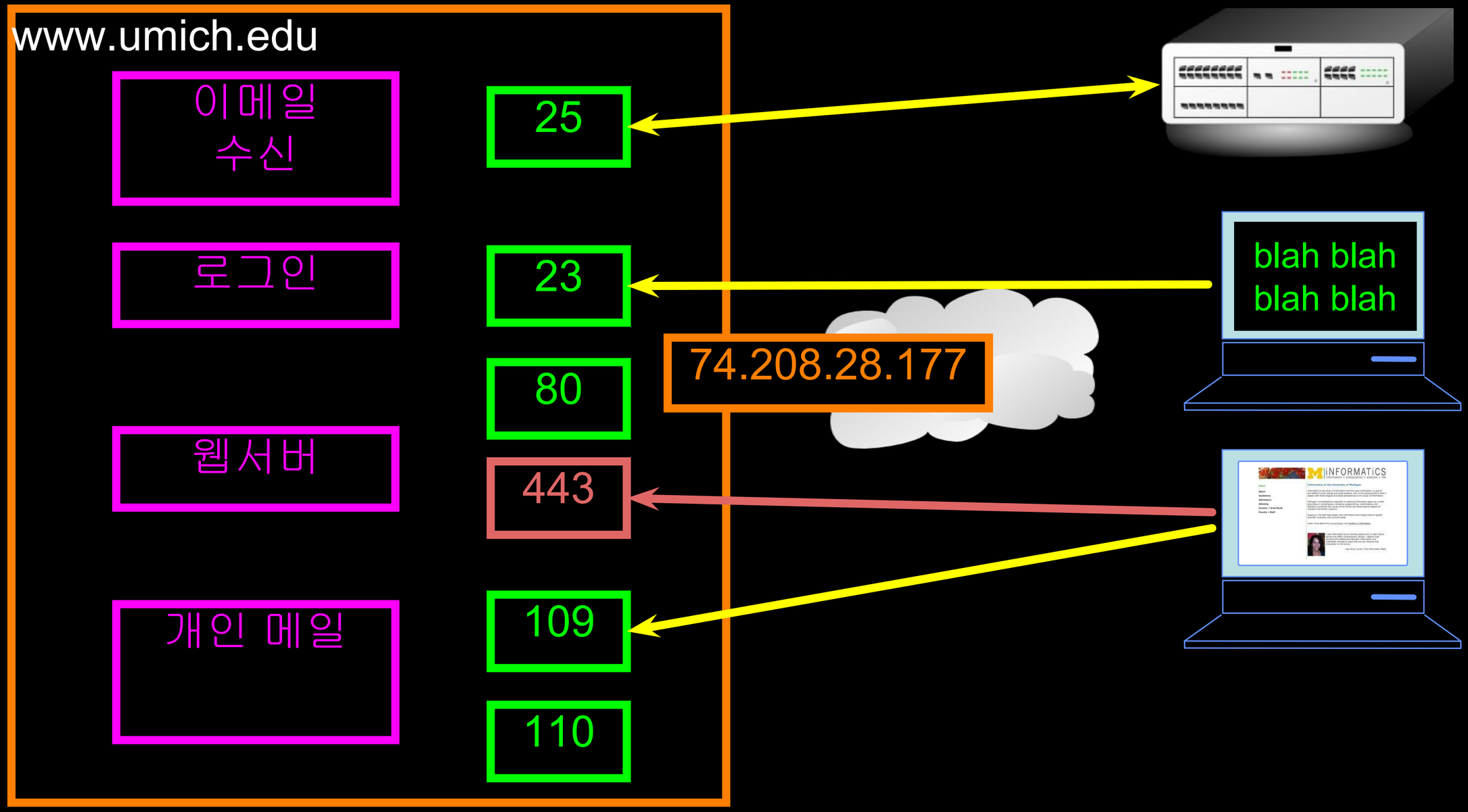
TCP 연결 / 소켓

"컴퓨터 네트워킹에서 인터넷 소켓, 또는 네트워크 소켓은
인터넷 프로토콜을 기반으로 한 인터넷 등의
컴퓨터 네트워킹에서 양방향 커뮤니케이션의 끝점입니다"

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

TCP 포트 번호

- 포트는 애플리케이션에 대응되거나 프로세스에 대응되는 소프트웨어 커뮤니케이션의 말단
- 한 서버에 여러 네트워크 애플리케이션이 존재할 수 있게 해줌
- 잘 알려진 포트 번호는 다음을 참고
  http://en.wikipedia.org/wiki/TCP_and_UDP_port

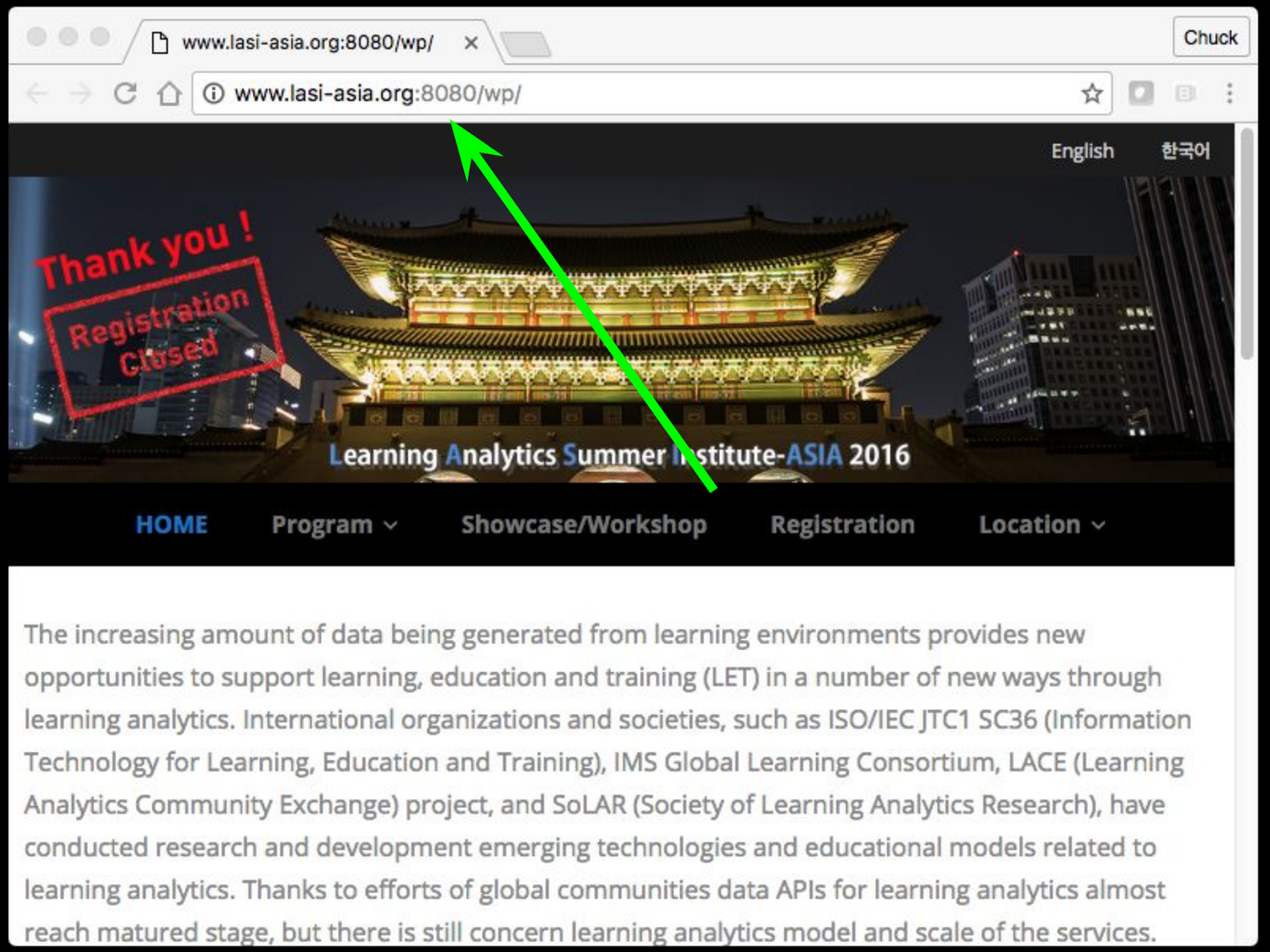


Clipart: http://www.clker.com/search/networksym/1

공통 TCP 포트

- 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

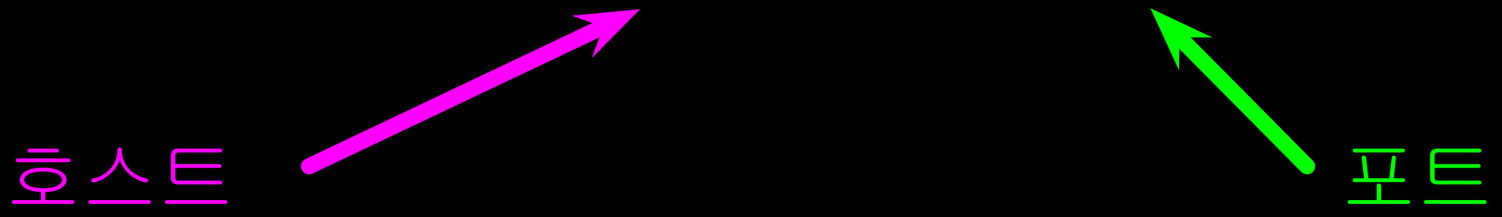


URL에 포트 번호가 있는 경우가 있는데, 이는 웹 서버가 '관례적으로 정해진' 포트에서 돌지 않는 경우

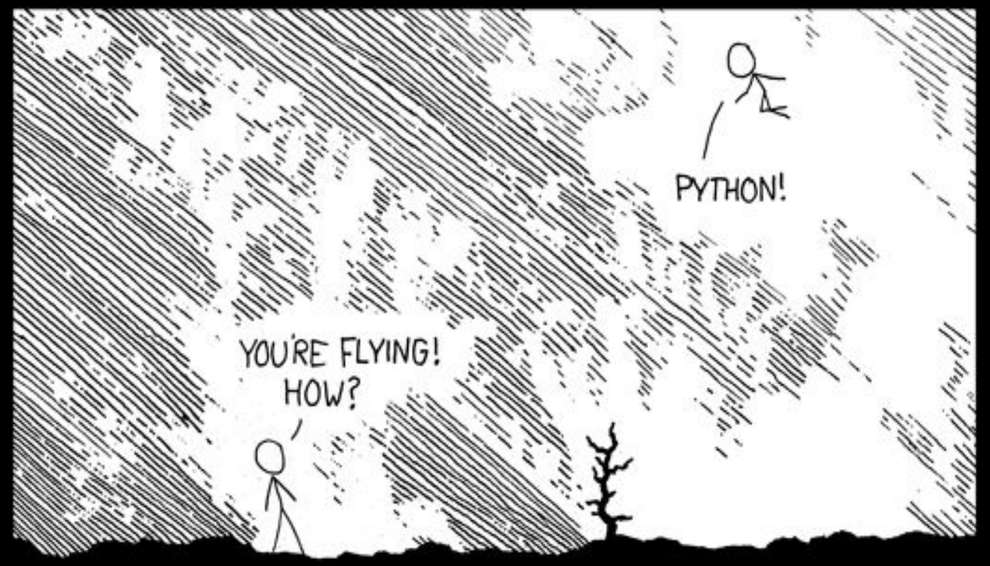
파이썬에서의 소켓

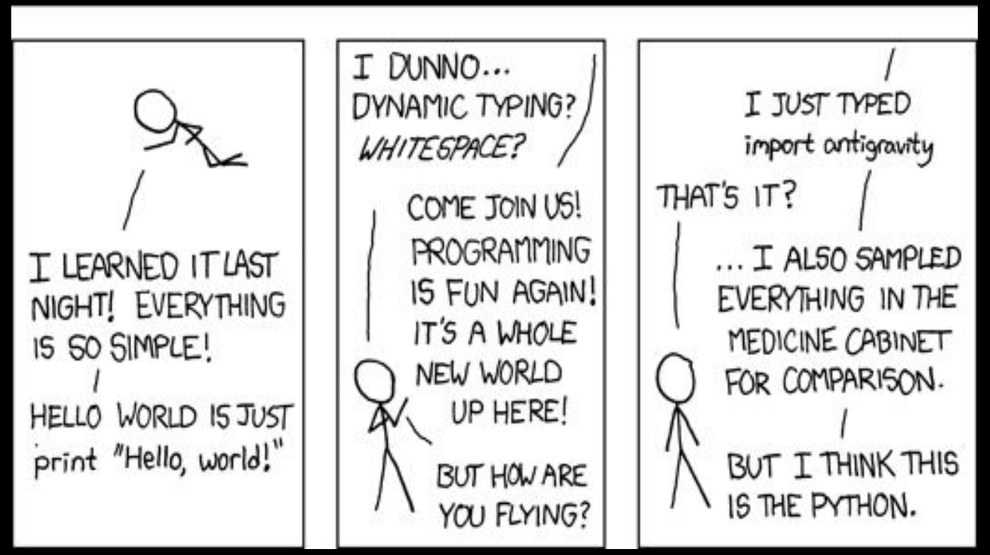
파이썬은 내부적으로 TCP 소켓을 지원

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



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



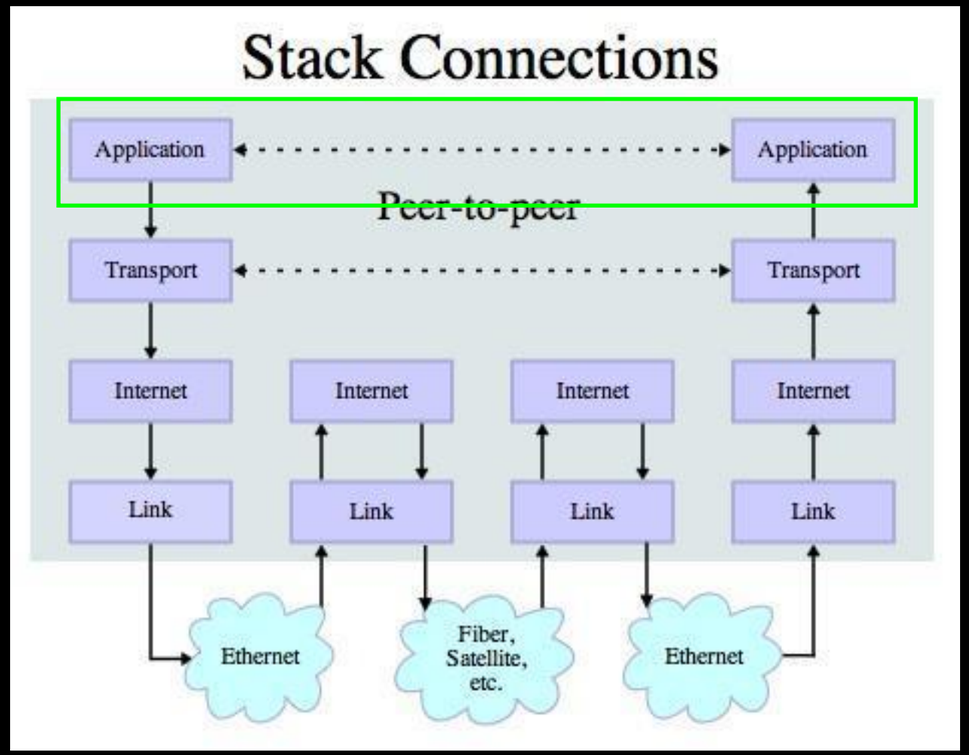


http://xkcd.com/353/

애플리케이션 프로토콜

애플리케이션 프로토콜

- TCP(와 파이썬)는 믿을만한 소켓을 제공. 소켓으로 어떤 일을 할 수 있고 어떤 문제를 해결?
- 애플리케이션 프로토콜
  - 메일
  - 월드 와이드 웹(WWW)



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

HTTP - 하이퍼텍스트 전송 프로토콜

- 인터넷, 애플리케이션 레이어에서 가장 많이 사용되는 프로토콜
- 웹을 위해 개발 - HTML, 이미지, 문서 등을 가져옴
- 문서 외에 다양한 데이터에도 확장하여 사용 가능
  - RSS, 웹 서비스 등
  - 기본 컨셉: 연결 - 문서 요청 - 문서 수신 - 연결 종료

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

HTTP

HyperText Transfer Protocol을 줄인 말이며,
브라우저가 서버로부터 인터넷을 통해 웹 문서를
받는 경우의 규칙을 정한 것

프로토콜이란?

- 규칙의 모음. 모두가 따르므로 서로가 서로의 행동을 예측 가능
- 서로 충돌하지 않아야 함
  - 미국의 이차선 도로에서는 오른쪽 도로로 달려야 함
  - 영국의 이차선 도로에서는 왼쪽 도로로 달려야 함

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

프로토콜 호스트 문서

http://www.youtube.com/watch?v=x2GylLq59rI

1:17 - 2:19

서버로부터 데이터 받기

- 사용자가 ‘href=값’을 가지고 있는 앵커 태그를 클릭해 새로운 페이지로 이동할 때마다 브라우저는 웹 서버와 연결을 만들고 GET 요청을 실행해 페이지 URL에 나타난 값을 수신
- 서버는 문서를 포맷팅하고 유저에게 보여주는 HTML 문서를 리턴

웹 서버

80

하이퍼링크
└ 이 페이지에서 클릭하면
다른 페이지로 간다는 표시

브라우저

웹 서버

80

브라우저

클릭

요청

웹 서버

80

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

브라우저

클릭

요청

웹 서버

80

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

브라우저

Click

요청

웹 서버

80

응답

<h1>The Second Page</h1> If you like, you can switch back to the First Page.

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

브라우저

클릭

요청

웹 서버

80

응답

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

<h1>The Second Page</h1> If you like, you can switch back to the First Page.

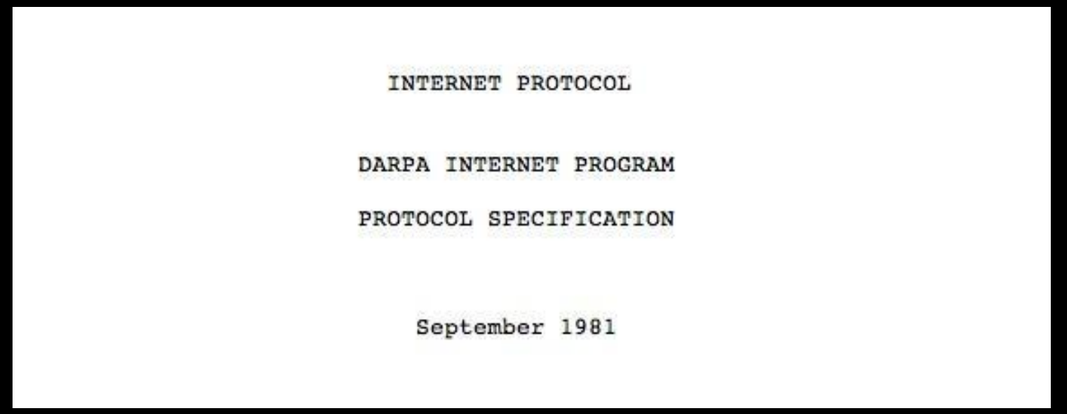
브라우저

클릭

파싱/
렌더링

인터넷 표준

- 모든 인터넷 프로토콜 기준은 한 기관에 의해 개발
- Internet Engineering Task Force (IETF)
- www.ietf.org
- 기준은 “RFCs”라고 부름 - “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

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

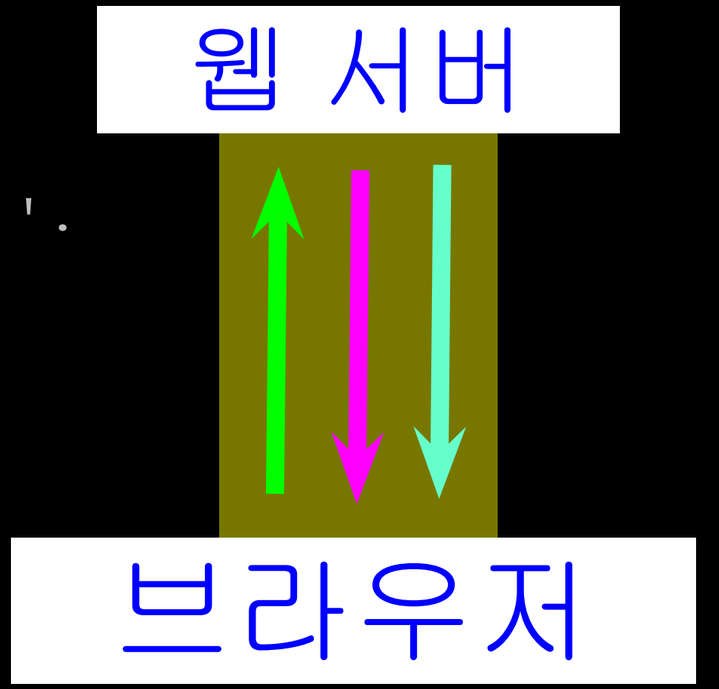
The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

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

HTTP 요청을 만드는 법

- 서버에 연결하고 “www.dr-chuck.com"
- 문서를 요청 (또는 기본 문서 요청)
    - 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

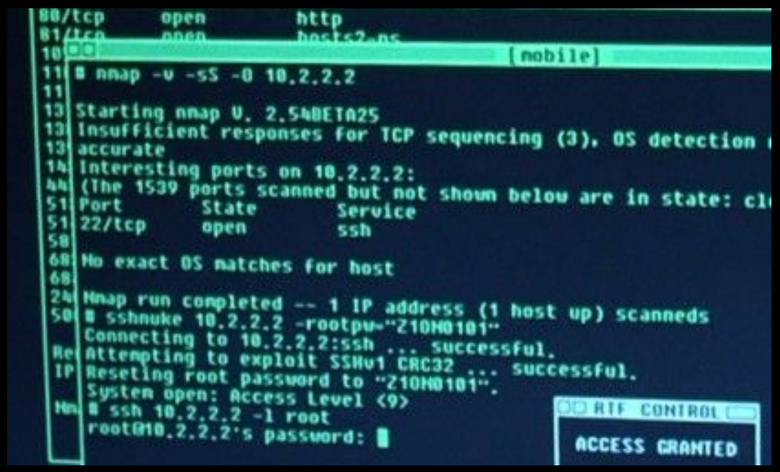
```
$ telnet www.dr-chuck.com 80
Trying 74.208.28.177...
Connected to www.dr-chuck.com.Escape character is '^]'.
GET http://www.dr-chuck.com/page1.htm HTTP/1.0

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.
```



영화에 나오는 해킹

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

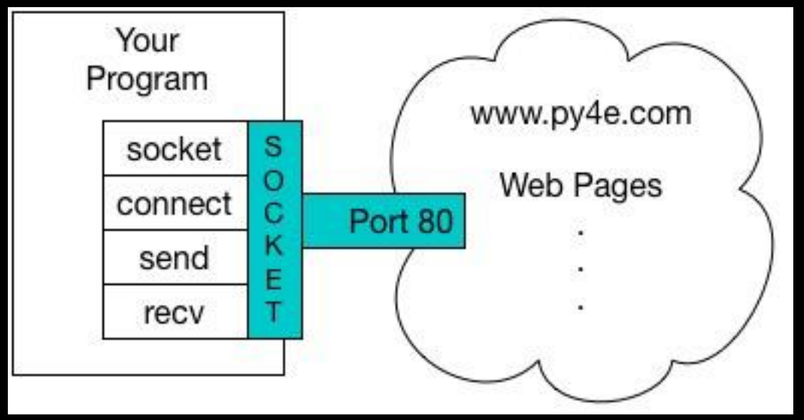
http://nmap.org/movies.html



웹 브라우저 만들기!

파이썬에서의 HTTP 요청

```
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 헤더

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

HTTP 바디

문자와 문자열에 대해...

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	0000000	NUL	32	0x20	040	0100000	space	64	0x40	100	1000000	@	96	0x60	140	1100000	`
1	0x01	001	0000001	SOH	33	0x21	041	0100001	!	65	0x41	101	1000001	A	97	0x61	141	1100001	a
2	0x02	002	0000010	STX	34	0x22	042	0100010	"	66	0x42	102	1000010	B	98	0x62	142	1100010	b
3	0x03	003	0000011	ETX	35	0x23	043	0100011	#	67	0x43	103	1000011	C	99	0x63	143	1100011	c
4	0x04	004	0000100	EOT	36	0x24	044	0100100	$	68	0x44	104	1000100	D	100	0x64	144	1100100	d
5	0x05	005	0000101	ENQ	37	0x25	045	0100101	%	69	0x45	105	1000101	E	101	0x65	145	1100101	e
6	0x06	006	0000110	ACK	38	0x26	046	0100110	&	70	0x46	106	1000110	F	102	0x66	146	1100110	f
7	0x07	007	0000111	BEL	39	0x27	047	0100111	'	71	0x47	107	1000111	G	103	0x67	147	1100111	g
8	0x08	010	0001000	BS	40	0x28	050	0101000	(	72	0x48	110	1001000	H	104	0x68	150	1101000	h
9	0x09	011	0001001	TAB	41	0x29	051	0101001	)	73	0x49	111	1001001	I	105	0x69	151	1101001	i
10	0x0A	012	0001010	LF	42	0x2A	052	0101010	*	74	0x4A	112	1001010	J	106	0x6A	152	1101010	j
11	0x0B	013	0001011	VT	43	0x2B	053	0101011	+	75	0x4B	113	1001011	K	107	0x6B	153	1101011	k
12	0x0C	014	0001100	FF	44	0x2C	054	0101100	,	76	0x4C	114	1001100	L	108	0x6C	154	1101100	l
13	0x0D	015	0001101	CR	45	0x2D	055	0101101	-	77	0x4D	115	1001101	M	109	0x6D	155	1101101	m
14	0x0E	016	0001110	SO	46	0x2E	056	0101110	.	78	0x4E	116	1001110	N	110	0x6E	156	1101110	n
15	0x0F	017	0001111	SI	47	0x2F	057	0101111	/	79	0x4F	117	1001111	O	111	0x6F	157	1101111	o
16	0x10	020	0010000	DLE	48	0x30	060	0110000	0	80	0x50	120	1010000	P	112	0x70	160	1110000	p
17	0x11	021	0010001	DC1	49	0x31	061	0110001	1	81	0x51	121	1010001	Q	113	0x71	161	1110001	q
18	0x12	022	0010010	DC2	50	0x32	062	0110010	2	82	0x52	122	1010010	R	114	0x72	162	1110010	r
19	0x13	023	0010011	DC3	51	0x33	063	0110011	3	83	0x53	123	1010011	S	115	0x73	163	1110011	s
20	0x14	024	0010100	DC4	52	0x34	064	0110100	4	84	0x54	124	1010100	T	116	0x74	164	1110100	t
21	0x15	025	0010101	NAK	53	0x35	065	0110101	5	85	0x55	125	1010101	U	117	0x75	165	1110101	u
22	0x16	026	0010110	SYN	54	0x36	066	0110110	6	86	0x56	126	1010110	V	118	0x76	166	1110110	v
23	0x17	027	0010111	ETB	55	0x37	067	0110111	7	87	0x57	127	1010111	W	119	0x77	167	1110111	w
24	0x18	030	0011000	CAN	56	0x38	070	0111000	8	88	0x58	130	1011000	X	120	0x78	170	1111000	x
25	0x19	031	0011001	EM	57	0x39	071	0111001	9	89	0x59	131	1011001	Y	121	0x79	171	1111001	y
26	0x1A	032	0011010	SUB	58	0x3A	072	0111010	:	90	0x5A	132	1011010	Z	122	0x7A	172	1111010	z
27	0x1B	033	0011011	ESC	59	0x3B	073	0111011	;	91	0x5B	133	1011011	[	123	0x7B	173	1111011	{
28	0x1C	034	0011100	FS	60	0x3C	074	0111100	<	92	0x5C	134	1011100	\	124	0x7C	174	1111100	|
29	0x1D	035	0011101	GS	61	0x3D	075	0111101	=	93	0x5D	135	1011101	]	125	0x7D	175	1111101	}
30	0x1E	036	0011110	RS	62	0x3E	076	0111110	>	94	0x5E	136	1011110	^	126	0x7E	176	1111110	~
31	0x1F	037	0011111	US	63	0x3F	077	0111111	?	95	0x5F	137	1011111	_	127	0x7F	177	1111111	DEL

https://en.wikipedia.org/wiki/ASCII
http://www.catonmat.net/download/ascii-cheat-sheet.png

간단한 문자열 표현방법

- 각 문자는 0~256 사이의 숫자로 대응되어 저장되며, 이는 메모리에서 8비트를 차지
- 8비트를 메모리에서 "byte"로 정함 (예: “내 USB는 8기가바이트짜리야”)
- ord() ASCII 문자에 대응되는 숫자를 리턴

```
>>> 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
>>>
```

1960~70년대에는
1바이트를 한 문자로
사용

Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char
0	0x00	000	0000000	NUL	32	0x20	040	0100000	space	64	0x40	100	1000000	@	96	0x60	140	1100000	`
1	0x01	001	0000001	SOH	33	0x21	041	0100001	!	65	0x41	101	1000001	A	97	0x61	141	1100001	a
2	0x02	002	0000010	STX	34	0x22	042	0100010	"	66	0x42	102	1000010	B	98	0x62	142	1100010	b
3	0x03	003	0000011	ETX	35	0x23	043	0100011	#	67	0x43	103	1000011	C	99	0x63	143	1100011	c
4	0x04	004	0000100	EOT	36	0x24	044	0100100	$	68	0x44	104	1000100	D	100	0x64	144	1100100	d
5	0x05	005	0000101	ENQ	37	0x25	045	0100101	%	69	0x45	105	1000101	E	101	0x65	145	1100101	e
6	0x06	006	0000110	ACK	38	0x26	046	0100110	&	70	0x46	106	1000110	F	102	0x66	146	1100110	f
7	0x07	007	0000111	BEL	39	0x27	047	0100111	'	71	0x47	107	1000111	G	103	0x67	147	1100111	g
8	0x08	010	0001000	BS	40	0x28	050	0101000	(	72	0x48	110	1001000	H	104	0x68	150	1101000	h
9	0x09	011	0001001	TAB	41	0x29	051	0101001	)	73	0x49	111	1001001	I	105	0x69	151	1101001	i
10	0x0A	012	0001010	LF	42	0x2A	052	0101010	*	74	0x4A	112	1001010	J	106	0x6A	152	1101010	j
11	0x0B	013	0001011	VT	43	0x2B	053	0101011	+	75	0x4B	113	1001011	K	107	0x6B	153	1101011	k
12	0x0C	014	0001100	FF	44	0x2C	054	0101100	,	76	0x4C	114	1001100	L	108	0x6C	154	1101100	l
13	0x0D	015	0001101	CR	45	0x2D	055	0101101	-	77	0x4D	115	1001101	M	109	0x6D	155	1101101	m
14	0x0E	016	0001110	SO	46	0x2E	056	0101110	.	78	0x4E	116	1001110	N	110	0x6E	156	1101110	n
15	0x0F	017	0001111	SI	47	0x2F	057	0101111	/	79	0x4F	117	1001111	O	111	0x6F	157	1101111	o
16	0x10	020	0010000	DLE	48	0x30	060	0110000	0	80	0x50	120	1010000	P	112	0x70	160	1110000	p
17	0x11	021	0010001	DC1	49	0x31	061	0110001	1	81	0x51	121	1010001	Q	113	0x71	161	1110001	q
18	0x12	022	0010010	DC2	50	0x32	062	0110010	2	82	0x52	122	1010010	R	114	0x72	162	1110010	r
19	0x13	023	0010011	DC3	51	0x33	063	0110011	3	83	0x53	123	1010011	S	115	0x73	163	1110011	s
20	0x14	024	0010100	DC4	52	0x34	064	0110100	4	84	0x54	124	1010100	T	116	0x74	164	1110100	t
21	0x15	025	0010101	NAK	53	0x35	065	0110101	5	85	0x55	125	1010101	U	117	0x75	165	1110101	u
22	0x16	026	0010110	SYN	54	0x36	066	0110110	6	86	0x56	126	1010110	V	118	0x76	166	1110110	v
23	0x17	027	0010111	ETB	55	0x37	067	0110111	7	87	0x57	127	1010111	W	119	0x77	167	1110111	w
24	0x18	030	0011000	CAN	56	0x38	070	0111000	8	88	0x58	130	1011000	X	120	0x78	170	1111000	x
25	0x19	031	0011001	EM	57	0x39	071	0111001	9	89	0x59	131	1011001	Y	121	0x79	171	1111001	y
26	0x1A	032	0011010	SUB	58	0x3A	072	0111010	:	90	0x5A	132	1011010	Z	122	0x7A	172	1111010	z
27	0x1B	033	0011011	ESC	59	0x3B	073	0111011	;	91	0x5B	133	1011011	[	123	0x7B	173	1111011	{
28	0x1C	034	0011100	FS	60	0x3C	074	0111100	<	92	0x5C	134	1011100	\	124	0x7C	174	1111100	|
29	0x1D	035	0011101	GS	61	0x3D	075	0111101	=	93	0x5D	135	1011101	]	125	0x7D	175	1111101	}
30	0x1E	036	0011110	RS	62	0x3E	076	0111110	>	94	0x5E	136	1011110	^	126	0x7E	176	1111110	~
31	0x1F	037	0011111	US	63	0x3F	077	0111111	?	95	0x5F	137	1011111	_	127	0x7F	177	1111111	DEL

Code Charts

Home | Site Map | Search

Unicode 9.0 Character Code Charts

SCRIPTS | SYMBOLS | NOTES

http://unicode.org/charts/

Find chart by hex code: Go Related links: Name index Help & links

Scripts

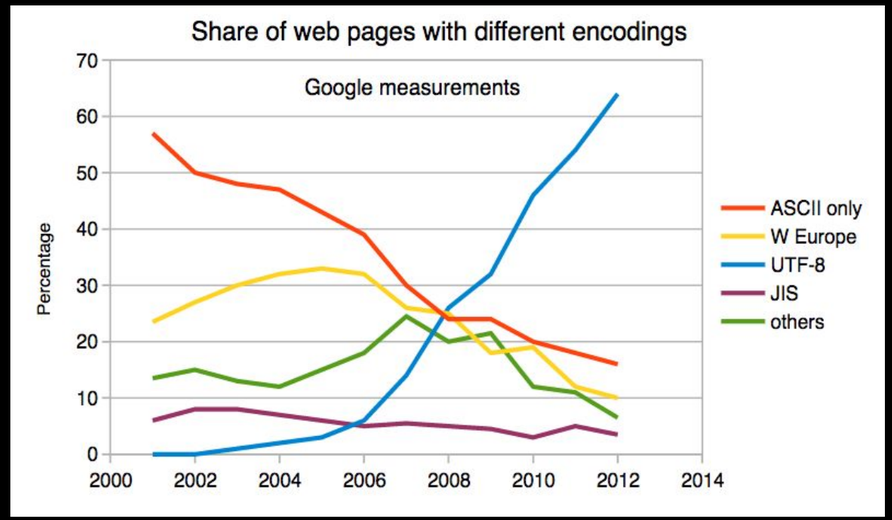
European Scripts	African Scripts	South Asian Scripts	Indonesia & Oceania Scripts
Armenian	*Adlam*	*Ahom*	*Balinese*
Armenian Ligatures	*Bamum*	*Bengali and Assamese*	*Batak*
Caucasian Albanian	Bamum Supplement	*Bhaiksuki*	*Buginese*
Cypriot Syllabary	*Bassa Vah*	*Brahmi*	*Buhid*
Cyrillic	*Coptic*	*Chakma*	*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*

여러 바이트로 된 문자

보다 다양한 문자를 나타내기 위해서는 더 많은 바이트를 쓸 필요가 있음

- UTF-16 – 길이 고정됨 - 2 바이트
- UTF-32 – 길이 고정됨 - 4 바이트
- UTF-8 – 1-4 bytes
  - ASCII를 포함하며, 호환
  - ASCII를 자동으로 감지 가능
  - UTF-8 은 시스템 간에 데이터를 교환할 때 가장 실용적으로 추천되는 인코딩 형식입니다

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



파이썬 내 문자열의 종류

```
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'>
>>>
```

파이썬3에서 모든 문자열은 유니코드임

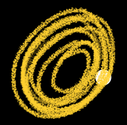
파이썬2 vs 파이썬3

```
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'>
```

파이썬3과 유니코드

- 파이썬3에서 모든 문자열은 유니코드 형식
- 그러므로 파일에서 데이터를 가져와 파이썬에서 작업하는 경우 거의 대부분 “그냥 작동”합니다



그러나 소켓을 통해 네트워크로 데이터를 전송하거나 DB와 연결하는 경우 데이터를 인코딩/디코딩해야 함 (UTF-8이 많이 쓰임)

```
Python 3.5.1
>>> x = b'abc'
>>> type(x)
<class 'bytes'>
>>> x = '이광춘'
>>> type(x)
<class 'str'>
>>> x = u'이광춘'
>>> type(x)
<class 'str'>
```

파이썬 문자열에서 Byte로

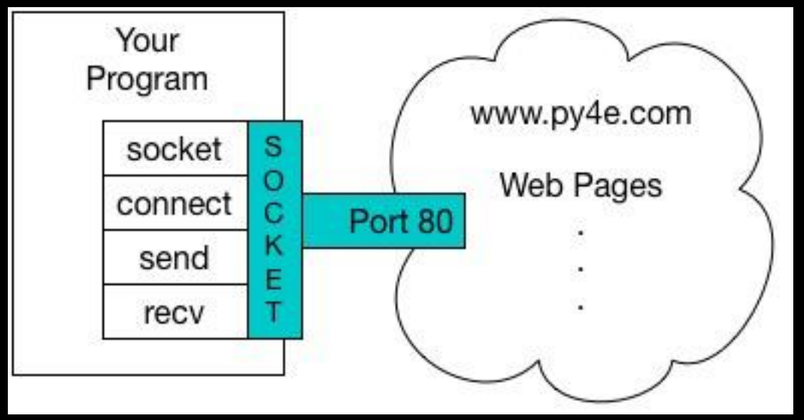
- 네트워크 소켓 등 외부 자원과 통신하는 경우, 문자열이 아니라 Byte 형식을 사용해야 함. 따라서 파이썬 3에서는 문자열을 Byte로 인코딩 필요.

- 외부에서 데이터를 가져오는 경우 해당 문자셋에 대하여 디코딩을 해야 파이썬3에서 정상적인 문자열으로 사용할 수 있다

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

파이썬에서의 HTTP 요청

```
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"`*`)`

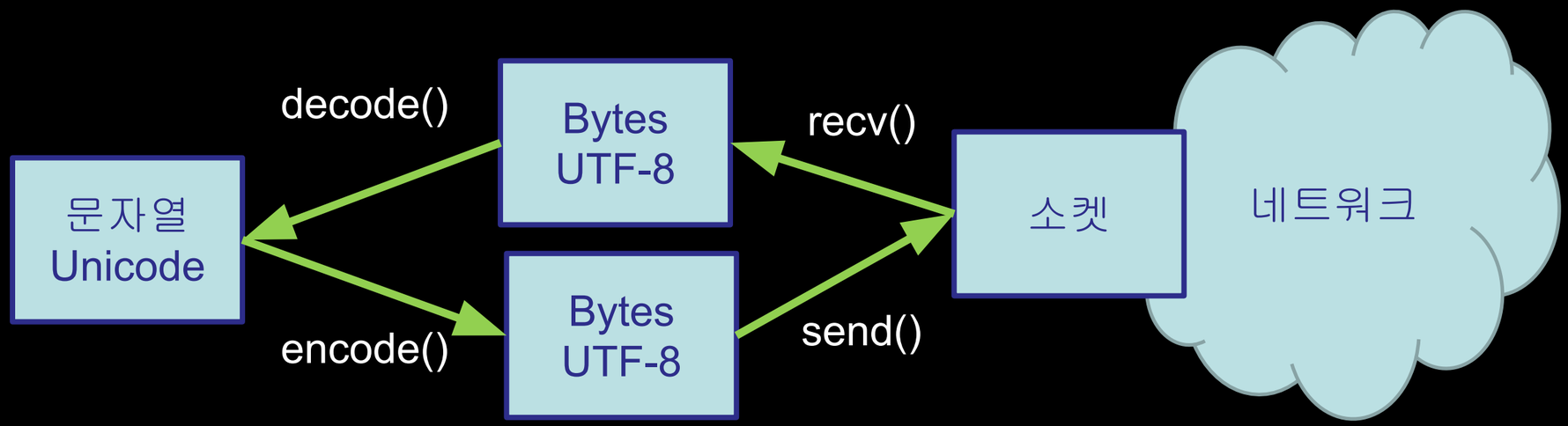
`bytearray.`**`decode`**`(`*`encoding="utf-8", errors="strict"`*`)`

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()
```

urllib로 HTTP 요청 간소화

파이썬에서 urllib 사용

HTTP는 굉장히 많이 쓰이기 때문에 소켓을 다루고 웹 페이지를 불러오는 라이브러리가 있음

```
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

파일처럼...

```
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

웹 페이지 읽기

```
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

링크 따라가기

```
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

Google 의 코드 첫 줄

```
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())
```

HTML 파싱
(웹 스크래핑라고도 함)

웹 스크래핑이란?

- 프로그램이나 스크립트가 브라우저처럼 행동하며 페이지를 살펴보고 정보를 추출하고 조사하는 것을 지칭

- 검색엔진은 웹 페이지를 스크래핑함
  - 이걸 스파이더링 또는 크롤링이라고도 함

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

왜 스크래핑 하나?

- 데이터를 가져오기 - 특히 소셜 데이터 - 누가 연결돼 있는지
- 외부로 내보내는 기능이 없는 시스템에서 데이터 가져오기
- 사이트를 모니터링하며 새로운 정보 감지
- 검색엔진의 데이터베이스를 구축하기 위한 스크래핑

웹 페이지 스크래핑

- 웹 페이지 스크래핑은 웹 페이지 내용을 마음대로 빼간다는 점에서 논란의 여지가 있음
- copyright된 정보를 다시 출판하는 것은 허용되지 않음
- 이용약관을 위배하지 않도록 유의

쉬운 방법 - BeautifulSoup

- 문자열 탐색으로 어렵게 접근하는 것도 가능하긴 함
- 무료 소프트웨어 라이브러리 BeautifulSoup 을 사용하는 방법도 있음 (www.crummy.com)

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

BeautifulSoup 설치

```
# 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 urllinks.py
Enter - http://www.dr-chuck.com/page1.htm
http://www.dr-chuck.com/page2.htm
```

요약

- TCP/IP는 애플리케이션 사이에 파이프/소켓을 구축
- 애플리케이션 프로토콜로 이 파이프를 사용
- HyperText Transfer Protocol(HTTP)는 간단하지만 굉장히 강력한 프로토콜
- 파이썬은 소켓, HTTP, and HTML 파싱을 충실히 지원

Acknowledgements / Contributions

...

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

Contributor:

- Seung-June Lee (plusjune@gmail.com)
- Connect Foundation

Translation:

- Yang Incheol (inchyangv@gmail.com)
- Jeungmin Oh (tangza@gmail.com)