문자열?

그런 달달한 것이 남아있긴 한가?

<<발표를 하게 된 계기>>

A : "어, 이 파일 UTF-8로 저장되어 있나봐요! 어쩌요?"
B : 그럼 유니코드로 읽어보면 될거야!

“In the beginning was the **Word**, and the **Word** was with…” - John 1:1

“태초에 말씀이 계시니라 이 말씀이…”

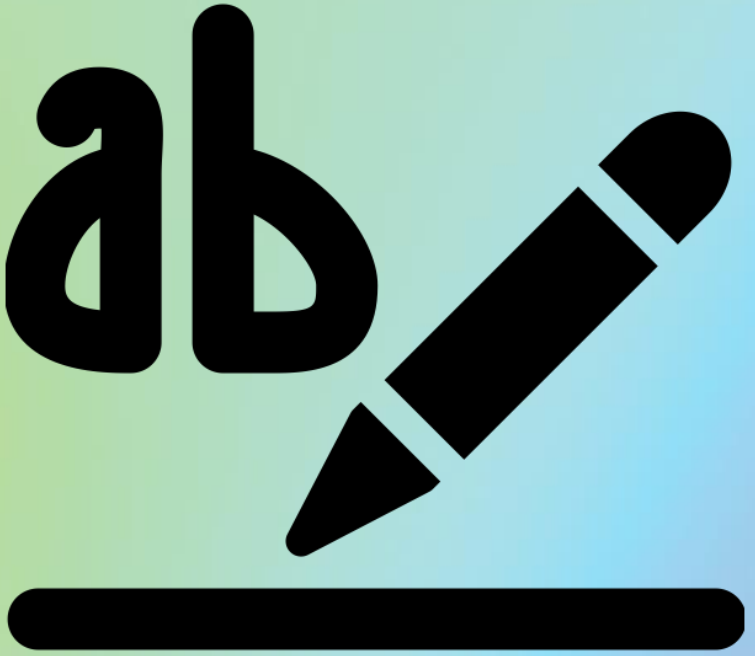
문자열에 대한 우리들의 이미지?

Created by Sharon Showalter
from Noun Project

"char*" or "String"

일반적으론 "문자(alphabet)의 집합(series)"

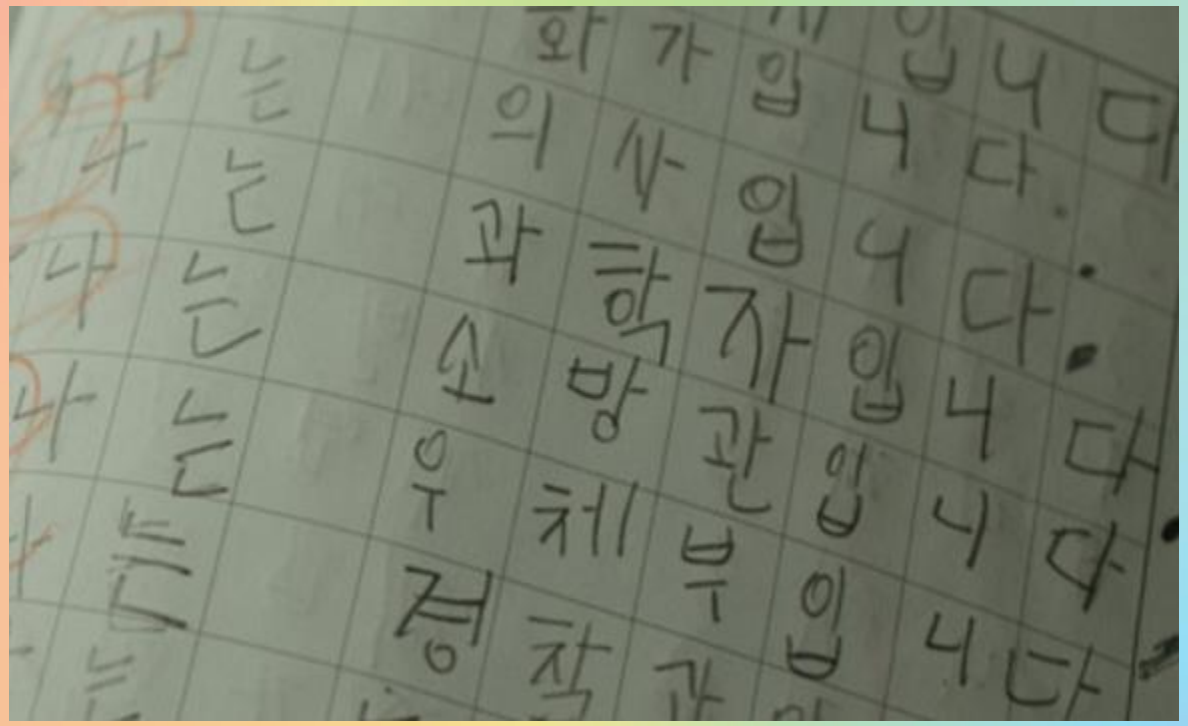
Created by Alfredo Hernandez
from Noun Project



Created by iconsmind.com
from Noun Project

우리에겐 "이런 느낌?!"

H	e	l	l	o
[0]	[1]	[2]	[3]	[4]



컴퓨터에겐 " {0,1} 집합(set)"

$$\{0,\ 1\}$$

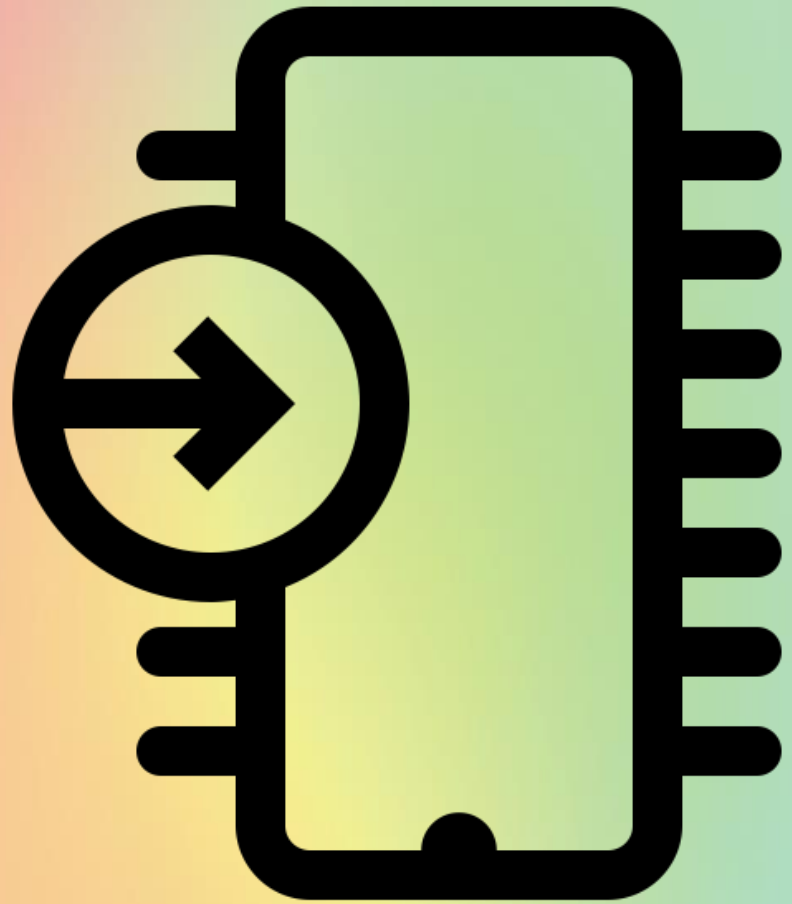
{0,1}을 '비트(bit)'라 호칭!

bit

그러나 컴퓨터는 '바이트Byte'를 선호함!

"0101 1001"

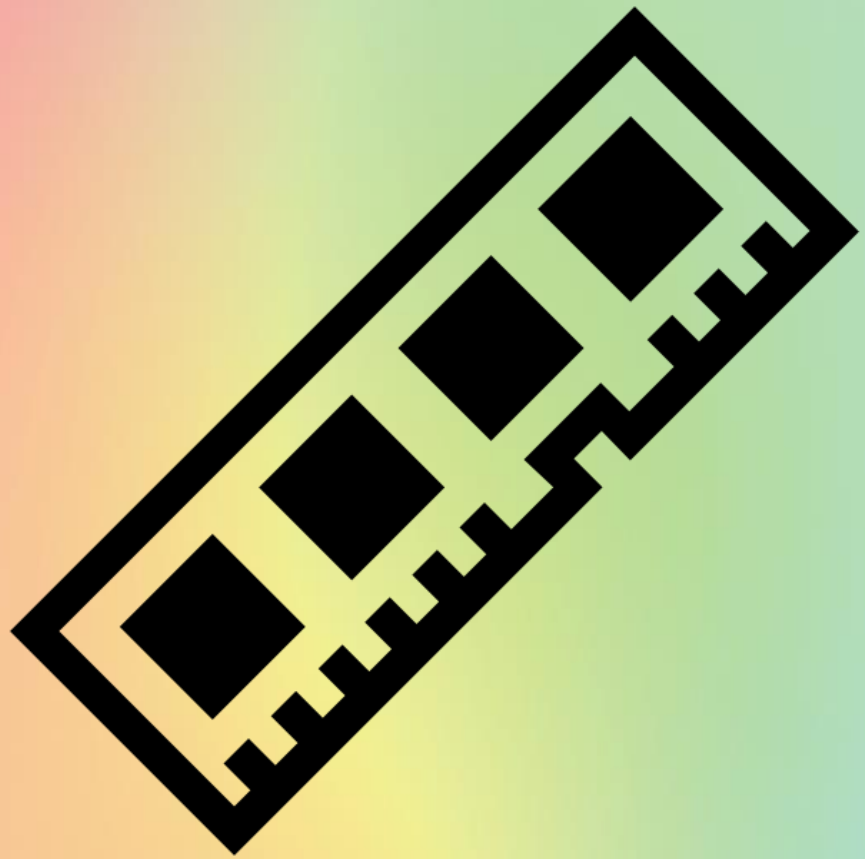
바이트가 모여있는 그 곳 "메모리"



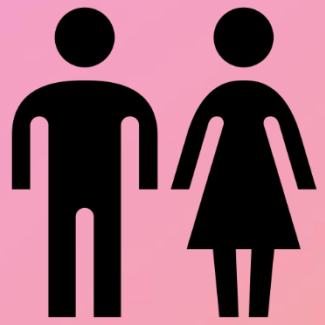
Created by Arthur Shlain
from Noun Project

노이만의 '속박(束縛)'

즉, 실행은 이 곳에서'만' 합니다.



Created by Creative Stall
from Noun Project



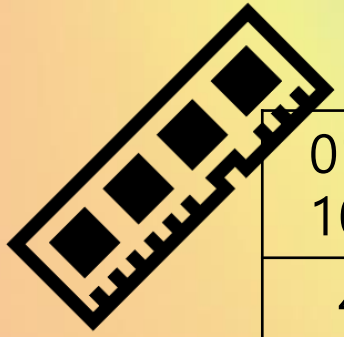
Created by David García
from Noun Project

"Hello"

Created by Gregor Črešnar
from Noun Project

H	E	L	L	O
0	1	2	3	4

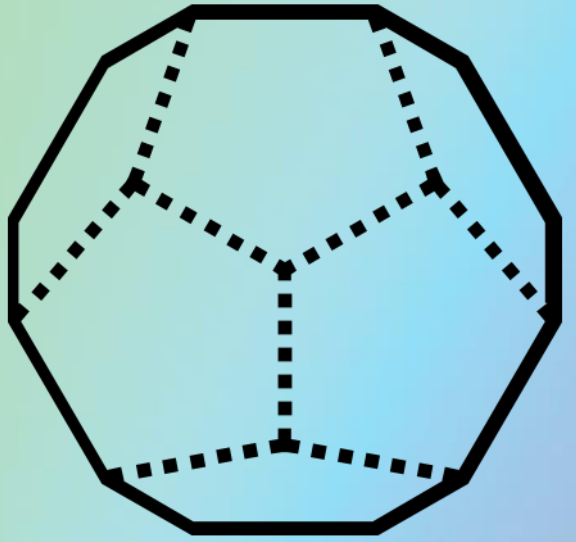
Created by Edward Boatman
from Noun Project



Created by Creative Stall
from Noun Project

0100 1000	0110 0101	0110 1100	0110 1100	0110 1111
48	65	6C	6C	6F

'인식 (認識, realization)'



Created by Daan Dirk
from Noun Project

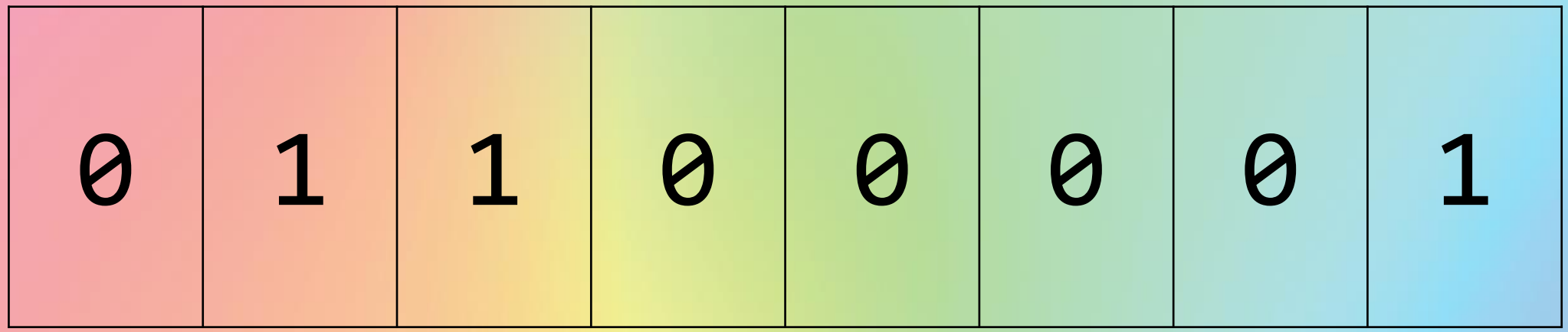
인식론에 대한 궁금증은

"In the memory was the bit, and the bit was with..."

"메모리에 비트가 존재했고, 그 비트의 집합인 '바이트'를 우리는 문자열로 인식하게 된다."

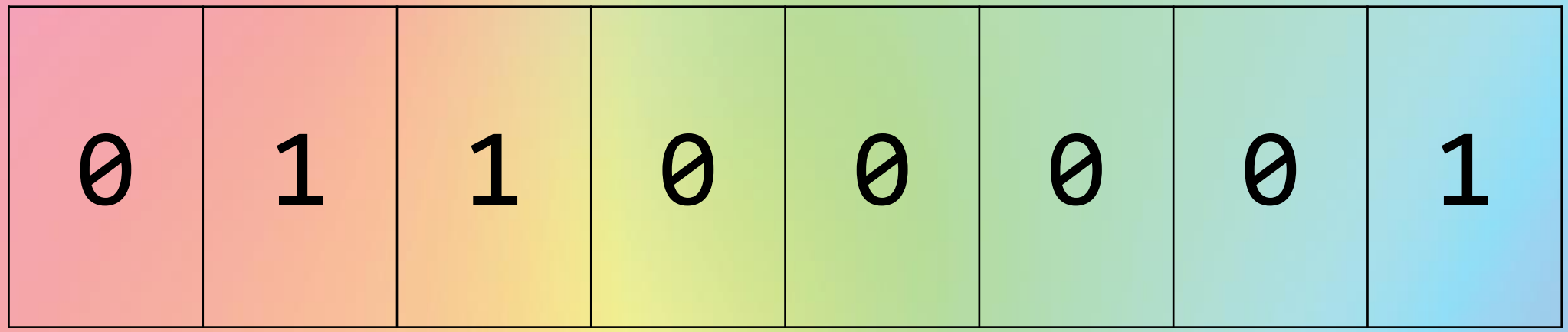
君子不器(군자불기) - 공자

군자는 그릇이 아니다


0 1 1 0 0 0 0 1

君子不器(군자불기)

'61'인가요? 'a'인가요?



문자집합(charset, Character Set)

Created by lastspark
from Noun Project

주도권 쟁탈전

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	A	97	61	a
2	2	*[START OF TEXT]*	34	22	"	66	42	B	98	62	b
3	3	*[END OF TEXT]*	35	23	#	67	43	C	99	63	c
4	4	*[END OF TRANSMISSION]*	36	24	$	68	44	D	100	64	d
5	5	*[ENQUIRY]*	37	25	%	69	45	E	101	65	e
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	F	102	66	f
7	7	*[BELL]*	39	27	'	71	47	G	103	67	g
8	8	*[BACKSPACE]*	40	28	(	72	48	H	104	68	h
9	9	*[HORIZONTAL TAB]*	41	29	)	73	49	I	105	69	i
10	A	*[LINE FEED]*	42	2A	*	74	4A	J	106	6A	j
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	K	107	6B	k
12	C	*[FORM FEED]*	44	2C	,	76	4C	L	108	6C	l
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	M	109	6D	m
14	E	*[SHIFT OUT]*	46	2E	.	78	4E	N	110	6E	n
15	F	*[SHIFT IN]*	47	2F	/	79	4F	O	111	6F	o
16	10	*[DATA LINK ESCAPE]*	48	30	0	80	50	P	112	70	p
17	11	*[DEVICE CONTROL 1]*	49	31	1	81	51	Q	113	71	q
18	12	*[DEVICE CONTROL 2]*	50	32	2	82	52	R	114	72	r
19	13	*[DEVICE CONTROL 3]*	51	33	3	83	53	S	115	73	s
20	14	*[DEVICE CONTROL 4]*	52	34	4	84	54	T	116	74	t
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	5	85	55	U	117	75	u
22	16	*[SYNCHRONOUS IDLE]*	54	36	6	86	56	V	118	76	v
23	17	*[ENG OF TRANS. BLOCK]*	55	37	7	87	57	W	119	77	w
24	18	*[CANCEL]*	56	38	8	88	58	X	120	78	x
25	19	*[END OF MEDIUM]*	57	39	9	89	59	Y	121	79	y
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	Z	122	7A	z
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
29	1D	*[GROUP SEPARATOR]*	61	3D	=	93	5D	]	125	7D	}
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

고통의 시작


CJK
stands for
China, Japan, and Korea
ABBREVIATIONS
Abbreviations.com

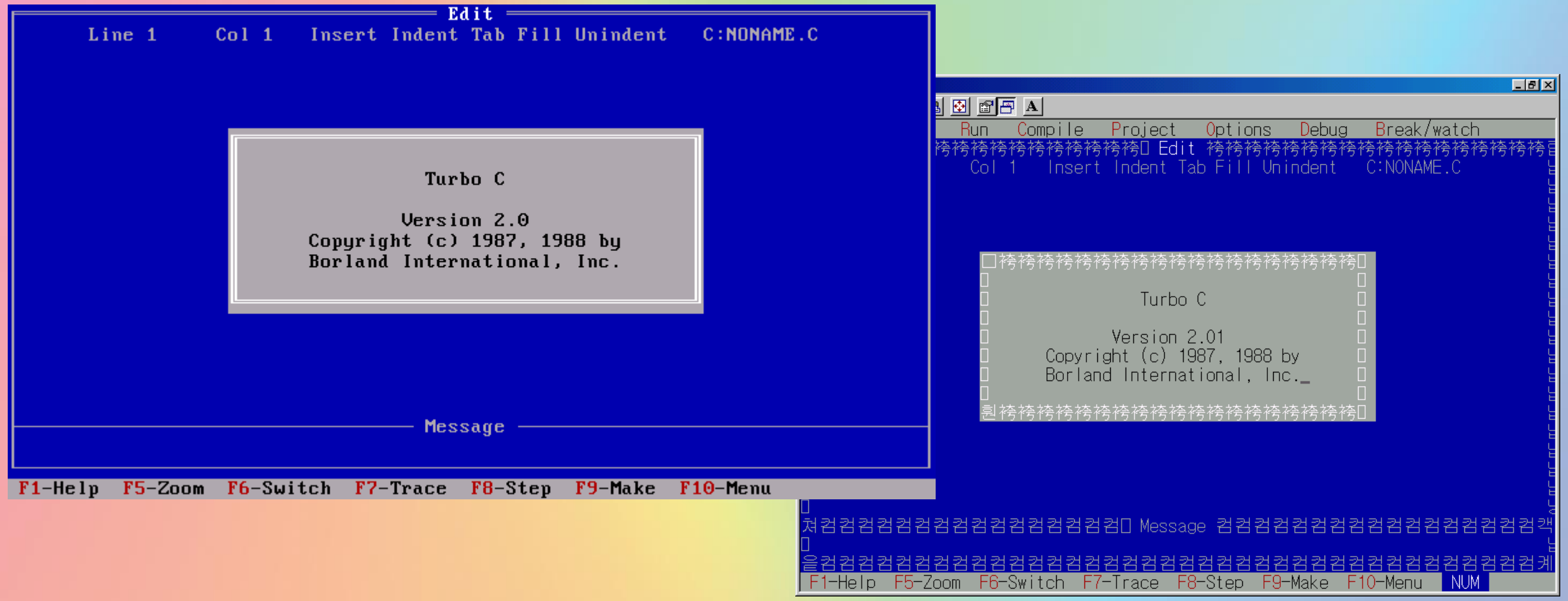
정해진 틀 혹은 그릇

"1Byte 크기의 문자를 정의하였고
인간이 인식하는 최초의 방식은
'ASCII'였다"

"너는 벌써 자기의 이마에다 인의(仁義)라는 자자(刺字)를 해 버렸다" - 장자

CJK의 역습

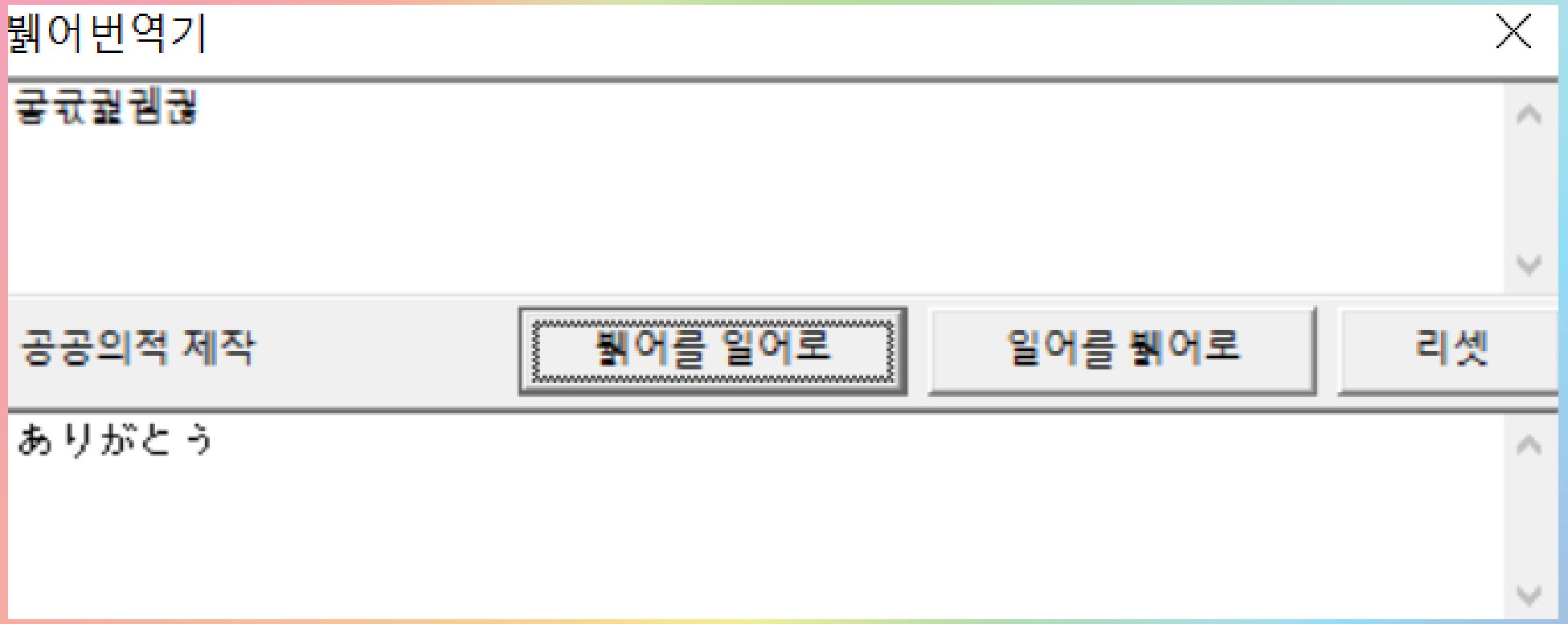
愛案以衣位囲胃印英栄塩億加果貨課芽改械害
街各覚完官管関観願希季紀喜旗器機議求泣救
給挙漁共協鏡競極訓軍郡径型景芸欠結建健験
固功好候航康告差菜最材昨札刷殺察参産散残
士氏史司試児治辞失借種周祝順初松笑唱焼象
照賞臣信成省清静席積折節説浅戦選然争倉巣
束側続卒孫帯隊達単置仲貯兆腸低底停的典伝
徒努灯堂働特得毒熱念敗梅博飯飛費必票標不
夫付府副粉兵別辺変便包法望牧末満未脈民無
約勇要養浴利陸良料量輪類令冷例歴連老労録


Edit
Line 1 Col 1 Insert Indent Tab Fill Unindent C:NONAME.C
Turbo C
Version 2.0
Copyright (c) 1987, 1988 by
Borland International, Inc.
Message
F1-Help F5-Zoom F6-Switch F7-Trace F8-Step F9-Make F10-Menu
Run Compile Project Options Debug Break/watch
Edit
Col 1 Insert Indent Tab Fill Unindent C:NONAME.C
Turbo C
Version 2.01
Copyright (c) 1987, 1988 by
Borland International, Inc.
Message
F1-Help F5-Zoom F6-Switch F7-Trace F8-Step F9-Make F10-Menu NUM

위대함은 알지만 뭐가 위대한지 몰랐던...

종류	자 소
초성	ㄱ, ㄴ, ㄷ, ㄹ, ㅁ, ㅂ, ㅅ, ㅇ, ㅈ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎ, ㄲ, ㄸ, ㅃ, ㅆ, ㅉ
중성	ㅏ, ㅑ, ㅓ, ㅕ, ㅗ, ㅛ, ㅜ, ㅠ, ㅡ, ㅣ, ㅐ, ㅒ, ㅖ, ㅘ, ㅝ, ㅚ, ㅟ, ㅙ, ㅞ, ㅢ
종성	ㄱ, ㄴ, ㄷ, ㄹ, ㅁ, ㅂ, ㅅ, ㅇ, ㅈ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎ, ㄲ, ㅆ, ㄳ, ㄵ, ㄶ, ㄺ, ㄻ, ㄼ, ㄽ, ㄾ, ㄿ, ㅀ, ㅄ

MBCS(멀티바이트 문자 집합)



인코딩_최종_final_final!

메모리에 담긴 바이트를 어떤 순서대로 인식(decoding)하고 기록(encoding) 할 것인가?

Byte order a.ka. Endianness

바이트를 쓰거나 읽는 순서

FE FF(Big) v.s. FF FE(little)

자세한 설명인 이 곳

인코딩_최종_final_final_마지막.zip

메모리에 담긴 바이트를 정해진 순서(Endianness, Byte Order Mark)에 따라 인식(decoding)하고 기록(encoding) 할 것인가?

인코딩!

EUC-KR, EUC-JP, Shift-JIS, BIG5, GB2312

EUC-KR?

KSC5601 + KSC5636 = EUC-KR

아
아
아
아


모두의 도움이
필요하다!
하늘로 손을
올려줘!
빨리… !!
모, 모든
지구인들아!
부탁한다!!
제발 부탁이니
파워를 나눠줘!!

UN같은 'UNICODE' 등장!

{UNICODE, Character Set, CodePoint(U+OOOO)}

“…premature optimization is the root of all evil” - Donald Knuth

We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil - Donald Knuth

4Byte는 너무 큰 것 같은데?!

인코딩	영어	한글
EUC-KR	1 byte	2 byte
UCS-2	2 byte	2 byte
UTF-8	1 byte	3 byte
UTF-16	2 byte	2 byte
UTF-32	4 byte	4 byte

7차 교육과정의 핵심
" 선택과 집중 "

UTF-8

- 하위호환 보장
  - ASCII와 호환
- 가변 바이트
  - len() 연산 오버헤드 발생

{Python 3, 'UNICODE', 'UTF-8'}

Python's Unicode Support

Now that you've learned the rudiments of Unicode, we can look at Python's Unicode featu

The String Type

Since Python 3.0, the language features a `str` type that contain Unicode characters, mea
as Unicode.

The default encoding for Python source code is UTF-8, so you can simply include a Unic

파이썬 3는 "유니코드 문자 셋"을 사용하고
해당 문자 셋의 기본적인 "인코딩/디코딩"은
"UTF-8"을 사용한다.

Think Different - Apple ad

{Python3, str, Unicode}

{Python3, str, Real/Just Unicode}


```
>>> _text = open("utf8.txt")
>>> _text
<_io.TextIOWrapper name='utf8.txt' mode='r' encoding='UTF-8'>
>>> print(_text.read())
```

바이트 배열은 바이트 배열 입니다.

```
>>> _str = _text.read()
>>> _str.encode("utf8")
b'\xf0\x9f\xa7\x80\n'
>>> _str.encode("utf16")
b'\xff\xfe>\xd8\xc0\xdd\n\x00'
>>> _str.encode("utf32")
b'\xff\xfe\x00\x00\xc0\xf9\x01\x00\n\x00\x00\x00'
>>> type(_str.encode("utf8"))
<class 'bytes'>
>>> type(_str.encode("utf16"))
<class 'bytes'>
>>> type(_str.encode("utf32"))
<class 'bytes'>
>>>
```

Byte와 Str은 서로 다른 '타입'

```
>>> _str = "Hello World!"
>>> _byte = b"Hello Pycon APAC 2016"
>>> _str_with_byte = _str + _byte
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'bytes' object to str implicitly
>>>
```

결론

- 파이썬3에서 사용하는 문자셋은 유니코드
- 유니코드를 저장/읽는 기본적인 방법은 "UTF-8"
- 유니코드와 UTF-8은 같지만 다른 영역에서 존재

이 질문은...

- UTF-OO에서 UTF-XX로 변경하는 방법
  - "바이트 ➔ STR ➔ 바이트" 형태인 '유니코드 샌드위치'를 사용
  - 유니코드 샌드위치는 http://nedbatchelder.com/text/unipain.html 참고!
- 윈도우에서 UTF-OO에 처리는 *NIX와 다를 수 있으니 스팀 머신은 각별한 주의가 필요!
  - thx! CP949
- Pycon APAC 2016 행사장에 상주하고 있으니 질문 사항은 언제든지 절 찾아주세요!