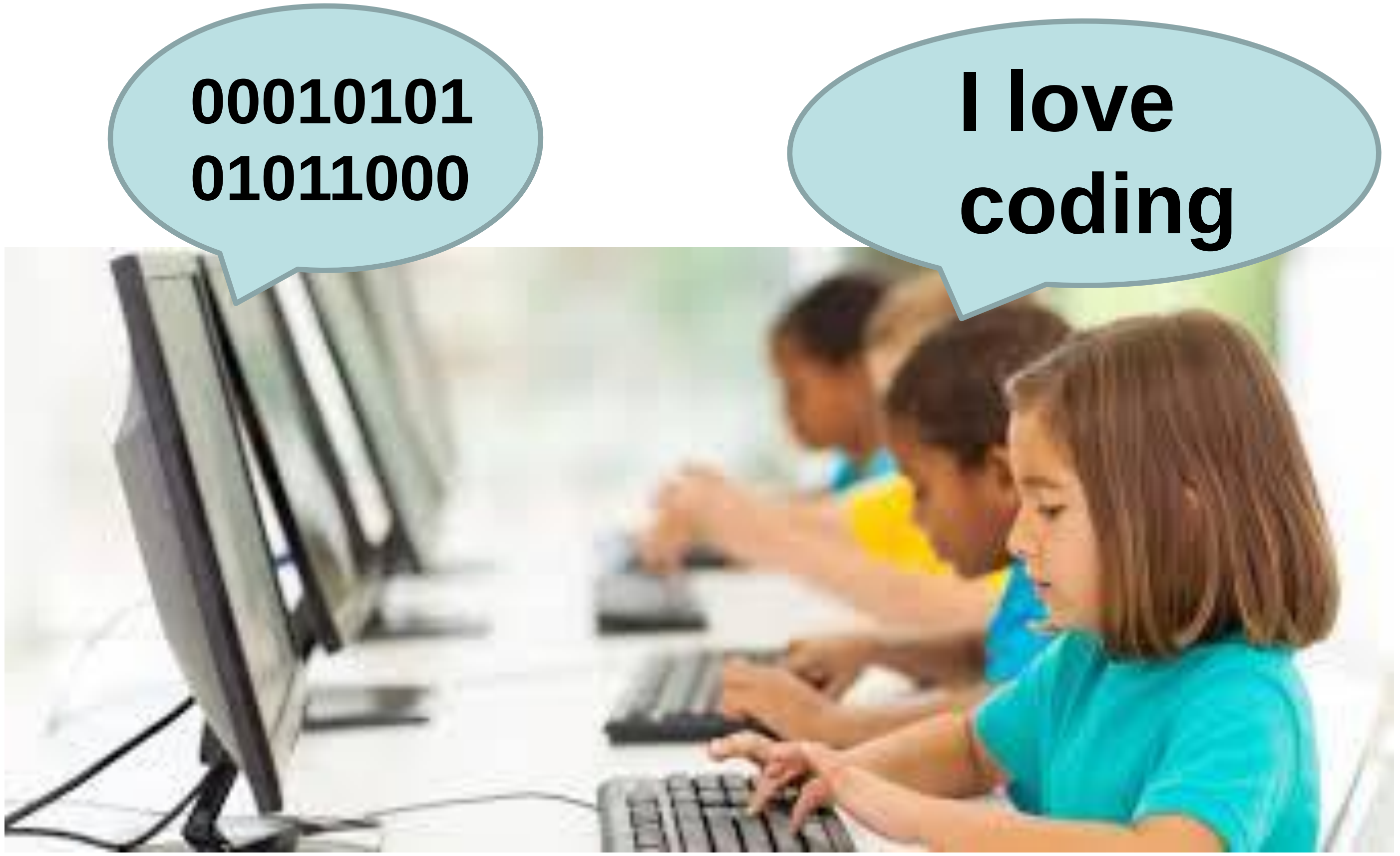
Encoding of Character Sets

Humans and Computers



Only the Binary Numbers Can Be Used in Computers

- What are the data in CPU registers ?
  - Binary numbers
- What are the data in Memory ?
  - Binary numbers
- What are the data in Files ?
  - Binary numbers
- What are the data in I/O devices ?
  - Binary numbers

Data Are Not Always (Binary) Integers

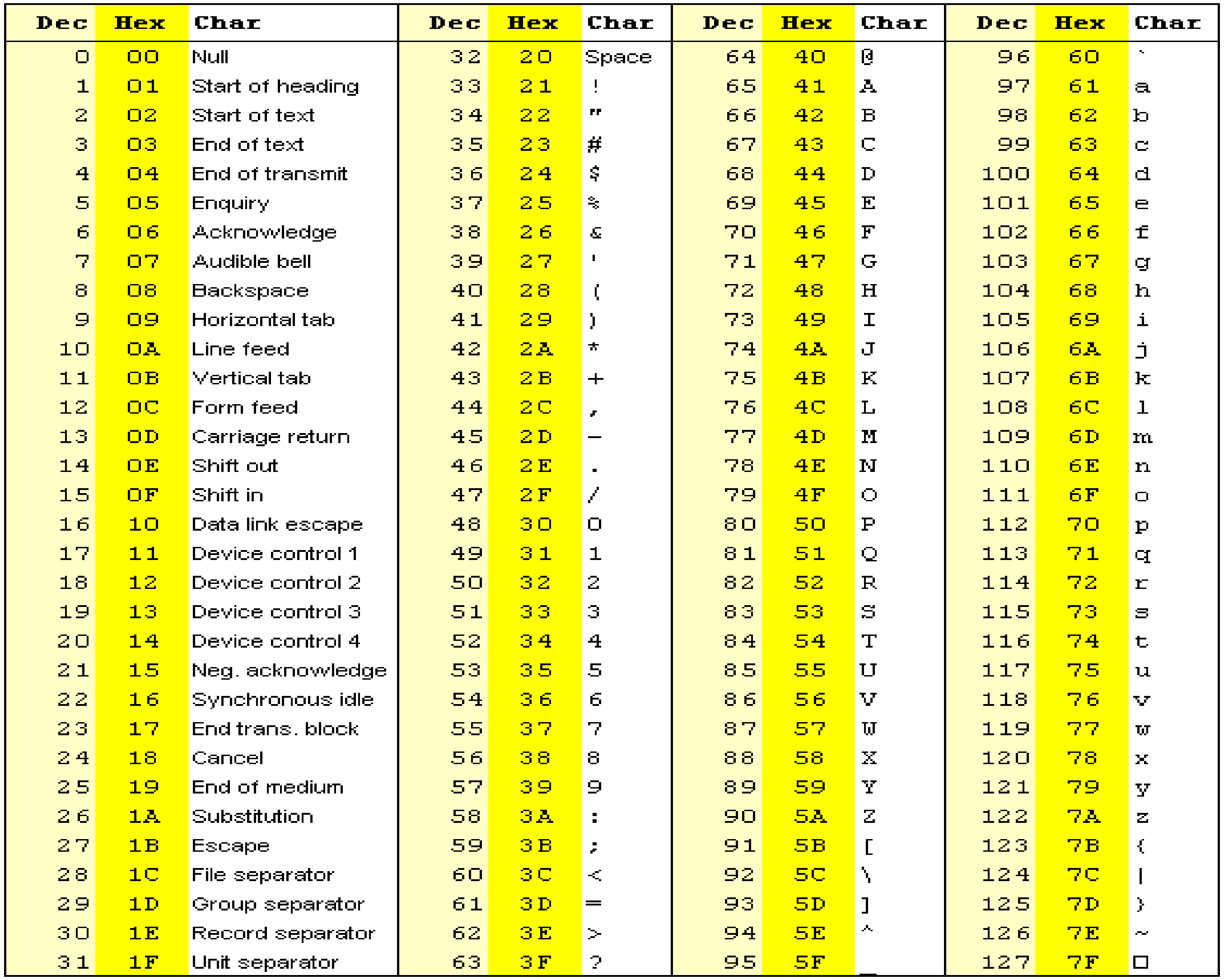
- Numbers, Characters, Colors, Geometry (rectangle, triangle, …), …
- We need encoding, meaning
- We need conversion between **data set** and **(binary) integers**.
- Characters are no different, need conversion between **character set** and **(binary) integers**.

➔ **CHARACTER ENCODING**

Character Sets

- US Alphabets only
  - ASCII
- Western European Character Set
  - ISO 8859-1
- Korean Hangul Character Set
  - CP949, etc, etc…
- All Characters Used on Earth
  - UNICODE (includes Korean Hangul)

ASCII Code – US Alphabets



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	–	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	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	Neg. 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	End 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	Substitution	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	□

유니코드, 한글 문자 Set (Unicode Hangul Syllables)

- http://en.wikipedia.org/wiki/Hangul_Syllables
- **Code point, U+AC00 ~ U+D7AF**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+AC0x	가	각	갂	갃	간	갅	갆	갇	갈	갉	갊	갋	갌	갍	갎	갏
U+AC1x	감	갑	값	갓	갔	강	갖	갗	갘	같	갚	갛	개	객	갞	갟
U+AC2x	갠	갡	갢	갣	갤	갥	갦	갧	갨	갩	갪	갫	갬	갭	갮	갯
U+AC3x	갰	갱	갲	갳	갴	갵	갶	갷	갸	갹	갺	갻	갼	갽	갾	갿
U+AC4x	걀	걁	걂	걃	걄	걅	걆	걇	걈	걉	걊	걋	걌	걍	걎	걏
………	………															
U+D7Ax	힠	힡	힢	힣												

Code Point is NOT Encoding

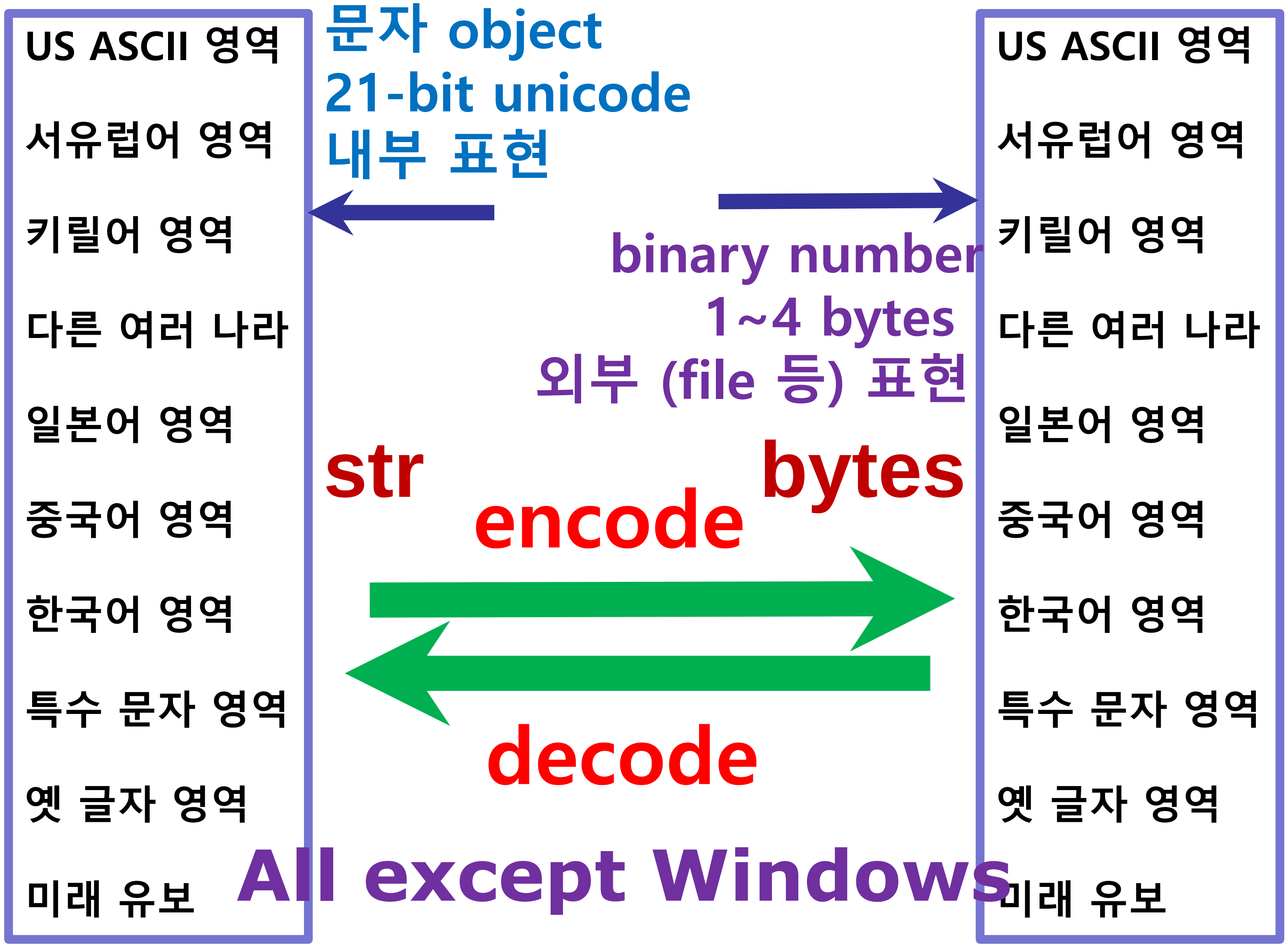
- Code Point is a **Representation** of a character. (example: a, ɑ, U+0061)

```
>>> 'a'
'a'
>>> '\u0061'
'a'
>>> '\uac00'
'가'
```

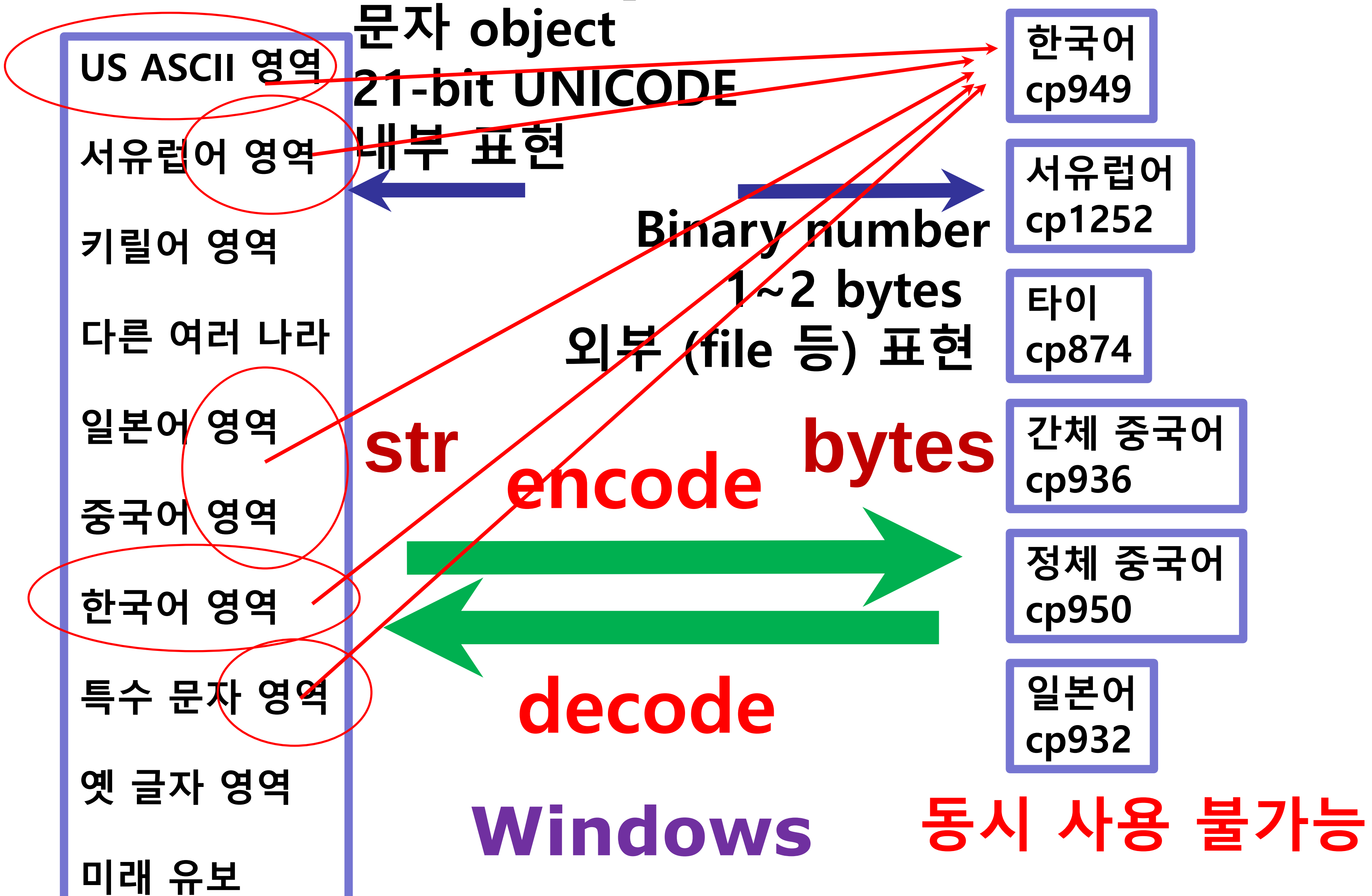
'a' can be written only if English input method is supported.

'가' can be written only if Korean input method is supported.

unicode vs. utf-8



unicode vs. cp949 (Windows-949)



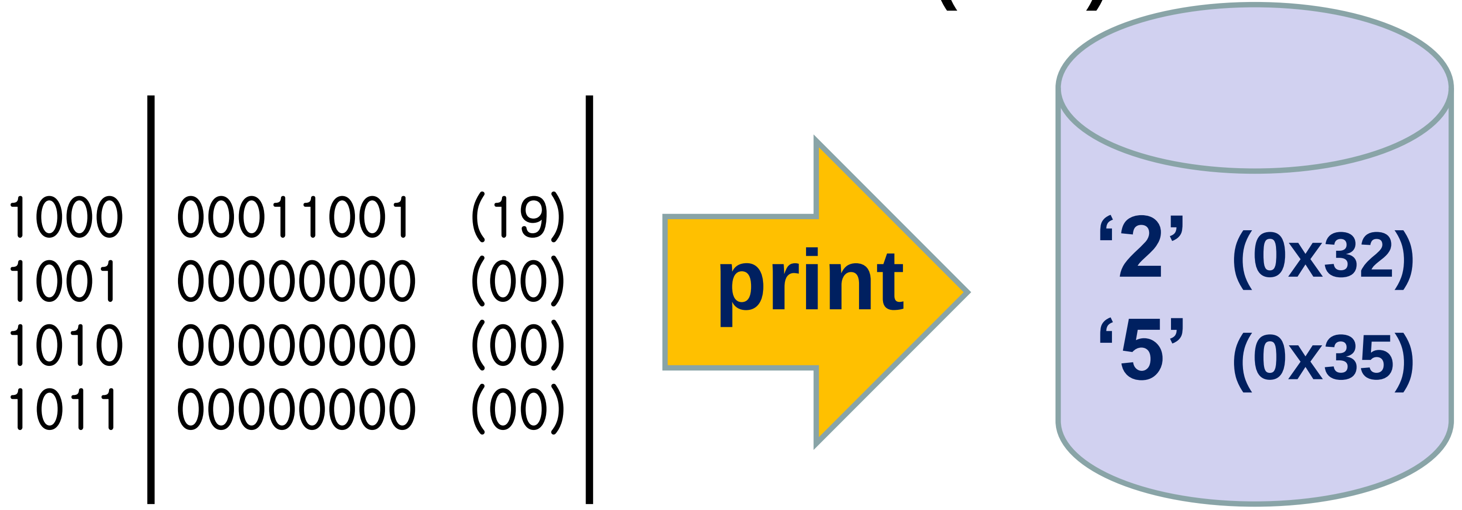
Encode & Decode

```
>> ‘가’.encode(‘utf-8’)
b’₩xea₩xb0₩x80’
>> b’₩xea₩xb0₩x80’.decode(‘utf-8’)
‘가’
>> ‘가’.encode(‘cp949’)
b’₩xb0₩xa1’
>> b’₩xb0₩xa1’.decode(‘cp949’)
‘가’
```

Text **File**

```
int  number = 25
fprintf(fp, "%d", number);
```

25 = 000…..00000011001 (binary)
= 00000019 (hex)



convert binary numbers to characters

Character Encoding of a FILE

- ‘cp949’ for Windows-Korean (10~20%)
- ‘UTF-8’ for all the others including macOS, iOS, GNU/Linux, Android, etc. (80+%)
- **We use ‘UTF-8’ on every platform for all program and data files at all the platforms including Windows.**
  - ❖ cp949 is legacy, and utf-8 is present and future of character encoding.
  - ❖ for cross-platform compatibility

Creator of Steve Jobs, Linus Torvalds, & Bill Gates

with Brian Kernighan and Ken Thompson