

# 2023학년도 2학기 언어데이터과학

## 제9강 문자 인코딩과 유니코드

박수민

서울대학교 인문대학 언어학과

2023년 10월 4일 수요일

## 지난 시간에 배운 것

- 1 Token frequency, type frequency, document frequency

## 오늘의 목표

- 1 문자 인코딩이 무엇인지 설명할 수 있다.
- 2 한글 인코딩에서 겪었던 문제가 무엇인지 말할 수 있다.
- 3 유니코드의 특징을 말할 수 있다.

# 부호화-복호화 모형 (Encoding-decoding model)

## 주요 개념

**부호화** 정보를 다른 형태로 변환하는 처리 (Encoding)

**복호화** 부호화된 형태를 원래 정보로 복원하는 처리 (Decoding)

**부호** 정보 변환 규칙 체계 (Code)

**코덱** 부호화·복호화를 수행하는 기계나 알고리즘 (Codec)

## 목적

표준화, 보안, 압축 등

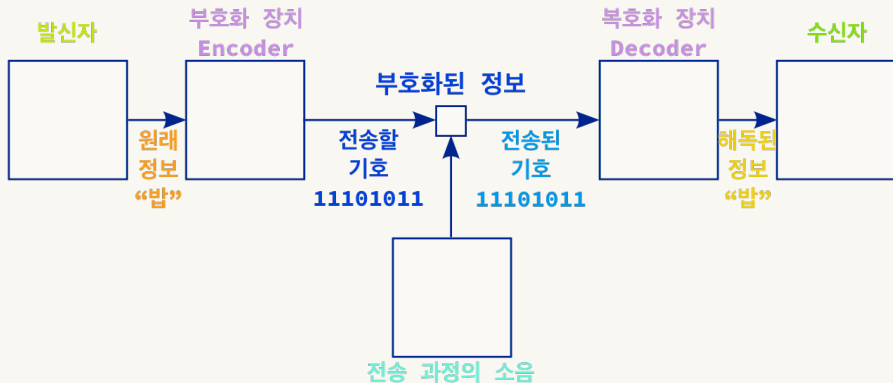
## 부호화 예시

(주로 인간용) → (주로 기계용)

- 알파벳 → 모스 부호
- 집 → 주소
- 음악 → 악보
- 이미지 → 픽셀
- 평문 → 암호문

# 부호화-복호화 모형(Encoding-decoding model)

통신 이론



Shannon. (1948). "The Mathematical Theory of Communication". The Bell System Technical Journal 27, 379-423.

[https://commons.wikimedia.org/wiki/File:Shannon\\_communication\\_system.svg](https://commons.wikimedia.org/wiki/File:Shannon_communication_system.svg)

<http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf>

## 오늘의 부호

문자 코딩: 문자  $\rightarrow$   $N$ 바이트 이진수

**예시** '가'  $\rightarrow$  11101010 10110000 10000000

### 1바이트

- = 8비트
- = 2진수 여덟 자리
- = 2진수 네 자리 두 개
- = 16진수 두 개

### 표현 예시

**10진수** 111

**2진수** 1101111

**1바이트** 0110 1111

**16진수** 6 F

### 16진수

10 A

11 B

12 C

13 D

14 E

15 F

## 표현 가능한 정보의 가짓수

**1바이트**  $2^8 = 256$

**2바이트**  $2^{16} = 65536$

**3바이트**  $2^{24} = 16777216$

## 문자 개수

**숫자** 10

**로마자**  $26 \times 2 = 52$

**한글**  $19 \times 21 \times 28 = 11172$

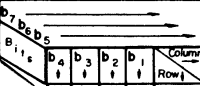
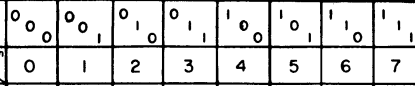
**한자** 106230 (異體字字典)

## ASCII(American Standard Code for Information Interchange)

## 아스키 코드

- 7비트( $2^7 = 128$ 가지)로
  - 영어 대소문자(52개),
  - 숫자(10개),
  - 특수 문자(32개),
  - 공백 문자(1개)

## 를 표현하는 인코딩 방식

												
b4	b3	b2	b1	Column Row	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0	0	0	0	0	NUL	DLE	SP	0	@	P	\	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

[https://commons.wikimedia.org/wiki/File:USASCII\\_code\\_chart.png](https://commons.wikimedia.org/wiki/File:USASCII_code_chart.png)

# ASCII(American Standard Code for Information Interchange)

## 아스키 코드에서 A에 대응하는 7비트 값 찾기

1000001 (가로 0001 → 오른쪽 네 자리; 세로 100 → 왼쪽 세 자리)

b7 b6 b5					b4 b3 b2 b1					Column				
					Row	0	1	2	3	4				
					0	0	0	0	0	NUL	DLE	SP	0	@
					0	0	0	1	1	SOH	DC1	!	1	A

## 의문

8비트를 채워야 1바이트가 되는데 나머지 1비트는 어디에 있는가?



# ASCII(American Standard Code for Information Interchange)

## 아스키 코드로 1바이트(=8비트) 완성하기

7비트에 패리티 비트(parity bit)를 추가한다.

**A** 1000001 → **0**1000001

**B** 1000010 → **0**1000010

**C** 1000011 → **1**1000011

**D** 1000100 → **0**1000100

**E** 1000101 → **1**1000101

## 패리티 비트의 값 정하기

**0** 7비트 중 1의 개수가 짝수인 경우

**1** 7비트 중 1의 개수가 홀수인 경우

## 예시

**11101100**처럼 패리티 비트의 값이 맞지 않으면 전송 과정에서 오류가 생겼음을 알 수 있다.

# Extended ASCII

영어 이외의 유럽어를 위한 문자 인코딩

## 확장 아스키 코드

- 8비트( $2^8 = 256$  가지)를 모두 사용하여 1문자를 1바이트로 표현하는 인코딩 방식
  - 기존 아스키 코드의 문자
  - 수학 기호( $\times, \geq, \pi$ )
  - 확장 로마자( $\acute{e}, \zeta$  등)

## 지역별 예시

**ISO 8859-1** 서유럽

**ISO 8859-2** 동유럽

**ISO 8859-3** 남유럽

**ISO 8859-4** 북유럽

**ISO 8859-5** 키릴 문자

... ..

## 영어 이외의 유럽어를 위한 문자 인코딩

$$\tilde{N} \rightarrow 1101 \ 0001$$
[illegible]

## ISO 8859-1

$$\dot{N} \rightarrow 1101 \ 0001$$

				b <sub>0</sub>	0	0	0	0	0	0	0	0	1	1	1	1	1	1
				b <sub>1</sub>	0	0	0	0	1	1	1	1	0	0	0	0	1	1
				b <sub>2</sub>	0	0	1	1	0	0	1	1	0	0	1	1	0	0
				b <sub>3</sub>	0	1	0	1	0	1	0	1	0	1	0	1	0	0
				00	01	02	03	04	05	06	07	08	09	10	11	12	13	
b <sub>4</sub>	b <sub>5</sub>	b <sub>6</sub>	b <sub>7</sub>	00			SP	0	@	P	`	p			NSBP	°	Í	Đ
0	0	0	0	01			!	1	A	Q	a	q			À	à	Á	Ñ

## ISO 8859-2

<http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-094.pdf>

# Extended ASCII

영어 이외의 유럽어를 위한 문자 인코딩

## 파이썬에서 인코딩하기

```
>>> from unicodedata import lookup
>>> n1 = lookup('LATIN CAPITAL LETTER N WITH TILDE')
>>> n2 = lookup('LATIN CAPITAL LETTER N WITH ACUTE')
>>> print(n1, n2)
Ñ  Ñ
>>> n1.encode('ISO 8859-1')
b'\xd1'
>>> n2.encode('ISO 8859-2')
b'\xd1'
```

# Extended ASCII

영어 이외의 유럽어를 위한 문자 인코딩

## 파이썬에서 디코딩하기

```
>>> bytes = b'\xd1'  
>>> bytes.decode('ISO 8859-1')  
'Ñ'  
>>> bytes.decode('ISO 8859-2')  
'Ń'
```

## 한글 인코딩의 문제

- 한글(옛한글 제외) 글자 수는 11172개다.
  - ⇒  $2^8 = 256$ 보다 많고  $2^{16} = 65536$ 보다 적다.
  - ⇒ 1바이트로 표현할 수 없다. 최소한 2바이트가 필요하다.

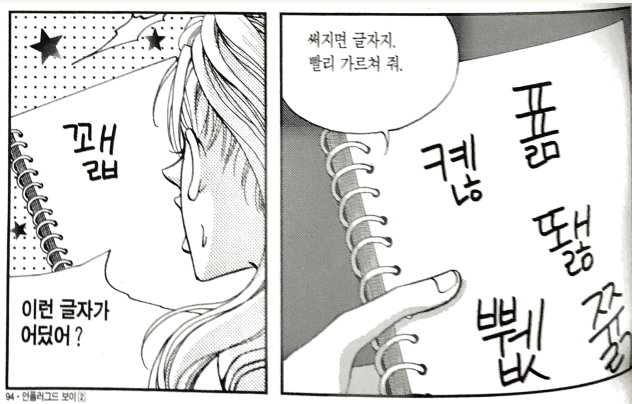
## 한글 인코딩 방법

- 역사** N바이트 조합형, 3바이트, 7비트 완성형, 2바이트 조합형, 2바이트 완성형, 확장 완성형, ...
- 표준** 유니코드(UTF-8, UTF-16, UTF-32)
  - 모든 문자를 부호화·표현·처리하는 산업 표준

완성형 (KSC 5601)

한글 2,350자만 표현 가능

- 똥방각하 (×)
- 펄시콜라 (×)
- 설미 (×)
- 뽕, 켄, 품, ... (×)



천계영. 언플러그드 보이 2. 서울문화사. 1997.

# 다양한 한글 코드

완성형 (KSC 5601)

‘김설믹’라는 이름이 입력되지 않는 이유는 은행이나 통신사, 대학 등 민간에서 사용하는 대형 전산시스템의 한국산업표준(KS)이 ‘한글조합형코드’가 아닌 ‘한글완성형코드’(EUC-KR)인 탓이다. 완성형코드는 국제표준과 충돌이 적다는 장점이 있지만, 미리 조합되어 있는 글자 외의 문자는 인식할 수 없다는 단점이 있다. ‘믹’, ‘켁’ 같이 빈도수가 낮은 문자들은 코드에 등록하지 않아 문자로 보지 못하는 셈이다. 이 완성형코드는 한글 초·중·종성으로 조합 가능한 한글 문자 1만1172자 중 2350자만 표현할 수 있다.

[http://www.hani.co.kr/arti/society/society\\_general/864914.html](http://www.hani.co.kr/arti/society/society_general/864914.html)



# 다양한 한글 코드

완성형 (KSC 5601)

## 인코딩 예시

```
>>> '뭇'.encode('euc-kr')  
b'\xb9\xcb'  
>>> '미'.encode('euc-kr')  
b'\xb9\xcc'
```

‘뭇’과 ‘미’ 사이에 나오는 ‘믹’에 대응하는 기호가 없다!

# 다양한 한글 코드

확장 완성형 (Microsoft Unified Hangul)

## 의의

2,350자 외에도 코드가 배당되었다.

## 문제

코드 순서가 가나다순이 아니다.

- 가나다순 정렬이 불가능하다.
- 자소 분해가 불가능하다.

## 인코딩 예시

```
>>> '뭇'.encode('cp949')  
b'\xb9\xcb'  
>>> '믹'.encode('cp949')  
b'\x92\xde'  
>>> '미'.encode('cp949')  
b'\xb9\xcc'
```

‘믹’가 있지만 ‘뭇’과 ‘미’ 사이가 아니다!

## 북한의 표준 문자 코드(KPS 9566)

## ■ 한글 특수문자 존재

행\렬	72	73	74	75	76	77	78	79	80	81
1										
2										
3	h	i	j	k	l	m	n	o	p	q
4	김	일	성	김	정	일				
5	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
6	VIII	IX	X							i
7	<sup>8</sup>	<sup>9</sup>	1/2	1/3	2/3	1/4	3/4		0	1
8	Pa	kPa	MPa	GPa	ℓ	μℓ	ml	dl	kl	gal

박진호. 국어 정보화의 방향: 문자 코드를 중심으로. 새국어생활 25-2. 2015.

# 컴퓨터에서 한글이 깨지는 이유

부호화 방법과 복호화 방법이 일치하는 경우: 깨지지 않음

```
>>> '고기'.encode('utf-8').decode('utf-8')  
'고기'  
>>> '고기'.encode('euc-kr').decode('euc-kr')  
'고기'
```

# 컴퓨터에서 한글이 깨지는 이유

## 부호화 방법과 복호화 방법이 일치하지 않는 경우: 깨짐

```
>>> '고기'.encode('utf-8').decode('euc-kr', 'ignore')  
'怨媛'  
>>> '고기'.encode('euc-kr').decode('utf-8', 'replace')  
'⚔⚔⚔⚔'
```

‘占쏙옷’을 아십니까?

# 유니코드

모든 문자를 부호화·표현·처리하는 산업 표준

Unicode Consortium(<https://unicode.org>)

## 특징

- 한글 11,172자가 가나다순으로 배열
- 옛한글 자모 포함
- 아스키 코드에 존재하는 문자는 아스키 코드와 같은 포인트에 대응

## 인코딩 방식

UTF-8, UTF-16, UTF-32 등

## 차트 구성

## A960

### Hangul Jamo Extended-A

**A97F**

	A96	A97
0	𠂇 A960	𠂈 A970
1	𠂉 A961	𠂊 A971
2	𠂋 A962	𠂌 A972
3	𠂍 A963	𠂎 A973

### Old initial consonants

A960	ㅓ	HANGUL CHOSEONG TIKEUT-MIEUM
A961	ㅕ	HANGUL CHOSEONG TIKEUT-PIEUP
A962	ㅖ	HANGUL CHOSEONG TIKEUT-SIOS
A963	ㅗ	HANGUL CHOSEONG TIKEUT-CIEUC
A964	ㅛ	HANGUL CHOSEONG RIEUL-KIYEOK
A965	ㅜ	HANGUL CHOSEONG RIEUL-SSANGKIYEOK
A966	ㅠ	HANGUL CHOSEONG RIEUL-TIKEUT
A967	ㅡ	HANGUL CHOSEONG RIEUL-SSANGTIKEUT
A968	ㅟ	HANGUL CHOSEONG RIEUL-MIEUM
A969	ㅡ	HANGUL CHOSEONG RIEUL-PIEUP
A96A	ㅢ	HANGUL CHOSEONG RIEUL-SSANGPIEUP
A96B	ㅣ	HANGUL CHOSEONG RIEUL-KAPYEOUNPIEUP
A96C	ㅤ	HANGUL CHOSEONG RIEUL-SIOS
A96D	ㅥ	HANGUL CHOSEONG RIEUL-CIEUC
A96E	ㅦ	HANGUL CHOSEONG RIEUL-KHIEUKH

<https://unicode.org/charts/PDF/UA960.pdf>

## UTF-8로 한글 인코딩하기

형식: 3바이트 1110XXXX 10XXXXXX 10XXXXXX

- 빈칸 16자리 → 4자리 2진수(=1자리 16진수) 네 개로 표현 가능
- 범위: AC00-D7A3

	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	ACA	ACB	ACC	ACD	ACE	ACF
0	가 AC00	감 AC10	감 AC20	갇 AC30	갈 AC40	각 AC50	갸 AC60	거 AC70	검 AC80	겐 AC90	갻 ACA0	결 ACB0	격 ACC0	겻 ACD0	고 ACE0	곰 ACF0
1	각 AC01	갑 AC11	갸 AC21	갹 AC31	갼 AC41	갽 AC51	갾 AC61	걱 AC71	겁 AC81	겻 AC91	겹 ACA1	겪 ACB1	결 ACC1	겼 ACD1	곡 ACE1	굽 ACF1
2	갹 AC02	갺 AC12	갻 AC22	갼 AC32	갽 AC42	갾 AC52	갿 AC62	겨 AC72	겻 AC82	겹 AC92	갺 ACA2	겪 ACB2	곶 ACC2	겼 ACD2	곡 ACE2	곶 ACF2
3	갻 AC03	갼 AC13	갽 AC23	갿 AC33	갺 AC43	갻 AC53	갼 AC63	갽 AC73	갿 AC83	갺 AC93	갻 ACA3	곶 ACB3	곶 ACC3	곶 ACD3	곶 ACE3	곶 ACF3



## 1 '가'에 대응하는 16진수 AC00

- $AC00_{(16)} = 1010\ 1100\ 0000\ 0000_{(2)}$

- 1110**1010** 10**110000** 10**0000000**

<https://unicode.org/charts/PDF/UAC00.pdf>

	AC0	AC1	AC2
0	가 AC00	감 AC10	감 AC20
1	각 AC01	갑 AC11	갇 AC21
2	깁 AC02	값 AC12	깁 AC22

# 파이썬에서 활용하기

## 문자 코드 포인트

```
>>> ord('가')
44032
>>> chr(ord('가'))
'가'
>>> chr(44032)
'가'
>>> hex(44032)
'0xac00'
```

## 문자 이름

```
>>> from unicodedata import name, lookup
>>> name('가')
'HANGUL SYLLABLE GA'
>>> lookup(name('가'))
'가'
>>> lookup('HANGUL SYLLABLE GA')
'가'
```

## 파이썬에서 활용하기

# 현대 한글 자모

**초성(19개)** ㄱ ㅋ ㆁ ㄷ ㅌ ㄴ ㄹ ㅁ ㅂ ㅅ ㅆ ㅈ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ ㅊ

**중성(21개)**    ㅏ   ㅑ   ㅓ   ㅕ   ㅗ   ㅛ   ㅜ   ㅠ   ㅡ   ㅣ   ㅚ   ㅝ   ㅞ   ㅟ   ㅠ   ㅡ   ㅢ   ㅣ   ㅤ   ㅥ   ㅦ

**종성(28개)** ㅇ ㄱ ㅋ ㆁ ㄷ ㅌ ㄴ ㄹ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅁ ㅂ ㅅ ㅆ ㅈ ㅊ ㅋ ㆅ ㆇ ㆉ ㆋ ㆍ ㆏ ㆑ ㆓ ㆕ ㆗ ㆙

## 다음 중성과의 거리

```
>>> ord('개') - ord('가')
28
>>> ord('월') - ord('울')
28
```

## 다음 초성과의 거리

```
>>> ord('파') - ord('가')
588
>>> ord('줄') - ord('울')
588
```

# 요약

## 문자 인코딩

문자를  $N$ 바이트 이진수로 변환하는 규칙 체계

## 바람직한 한글 인코딩의 요건

- 현대 한글 11,172자를 모두 포함
- 자모순으로 배열

## 유니코드

세계의 모든 문자를 컴퓨터에서 통합된 체계로 표현하기 위한 표준

**코드 포인트** `ord()`  $\leftrightarrow$  `chr()`

**문자 이름** `unicodedata.name()`  $\leftrightarrow$  `unicodedata.lookup()`

## 다음 시간에 배울 것

동아시아 고전 텍스트 데이터를 다룰 때 고려해야 할 문제

- 1 옛한글
- 2 한자

## 숙제 제출 일정

### 숙제06 8주차 발표 논문 선정

■ 2023-10-06 1:00 PM

### 숙제02 AWK 연습

■ 2023-10-10 1:00 PM

### 숙제05 Word Cloud

■ 2023-10-10 1:00 PM

### 숙제07 모두의 말뭉치 사이트 가입 인증 (<https://corpus.korean.go.kr>)

■ 2023-10-13 1:00 PM