How (and Why) We Speak in Unicode

Devon Peticolas

```
UnicodeDecodeError: 'ascii' codec can't
decode byte 0xc3 in position 6: ordinal not
in range(128)
```

Morse Code

1863

A	.-	M	--	Y	-.--	6	-....
B	-...	N	-.	Z	--..	7	--...
C	-.-.	O	---	Ä	.-.-	8	---..
D	-..	P	.--.	Ö	---.	9	----.
E	.	Q	--.-	Ü	..--	.	.-.-.-
F	..-.	R	.-.	Ch	----	,	--..--
G	--.	S	...	0	-----	?	..--..
H	....	T	-	1	.----	!	..--.
I	..	U	..-	2	..---	:	---...
J	.---	V	...-	3	...--	“	.-..-.
K	-.-	W	.--	4	....-	‘	.----.
L	.-..	X	-..-	5	.....	=	-...-

Character Set

A list of characters recognized by hardware

Encoding

Is a system of rules that converts a character set to and from binary

```python
MORSE_ENCODE = {
    'A': '.-',   'B': '-...', 'C': '-.-.',
    'D': '-..',  'E': '.',    'F': '..-.',
    'G': '--.',  'H': '....', 'I': '..',
    'J': '.---', 'K': '-.-',  'L': '.-..',
    'M': '--',   'N': '-.',   'O': '---',
    'P': '.--.', 'Q': '--.-', 'R': '.-.',
    'S': '...',  'T': '-',    'U': '..-',
    'V': '...-', 'W': '.--',  'X': '-..-',
    'Y': '-.--', 'Z': '--..',
}

def encode(s):
    encoded = []
    for letter in s:
        encoded.append(MORSE_ENCODE[letter])
    return ' '.join(encoded)
```

```
>>> encode('BURRITO')
'-... ..- .-. .-. .. - ---'
```

```
MORSE_DECODE = {
    '.-': 'A',   '-...': 'B', '-.-.': 'C',
    '-..': 'D',  '.': 'E',    '..-.': 'F',
    '--.': 'G',  '....': 'H', '..': 'I',
    '.---': 'J', '-.-': 'K',  '.-..': 'L',
    '--': 'M',   '-.': 'N',   '---': 'O',
    '.--.': 'P', '--.-': 'Q', '.-.': 'R',
    '...': 'S',  '-': 'T',    '..-': 'U',
    '...-': 'V', '.--': 'W',  '-..-': 'X',
    '-.--': 'Y', '--..': 'Z',
}

def decode(s):
    decoded = []
    for code in s.split(' '):
        decoded.append(MORSE_DECODE[code])
    return ''.join(decoded)
```

```
>>> decode('-... ..- .-. .-. .. - ---')
'BURRITO'
```

Baudot Code

1874

(No Model.)

11 Sheets—Sheet 6.

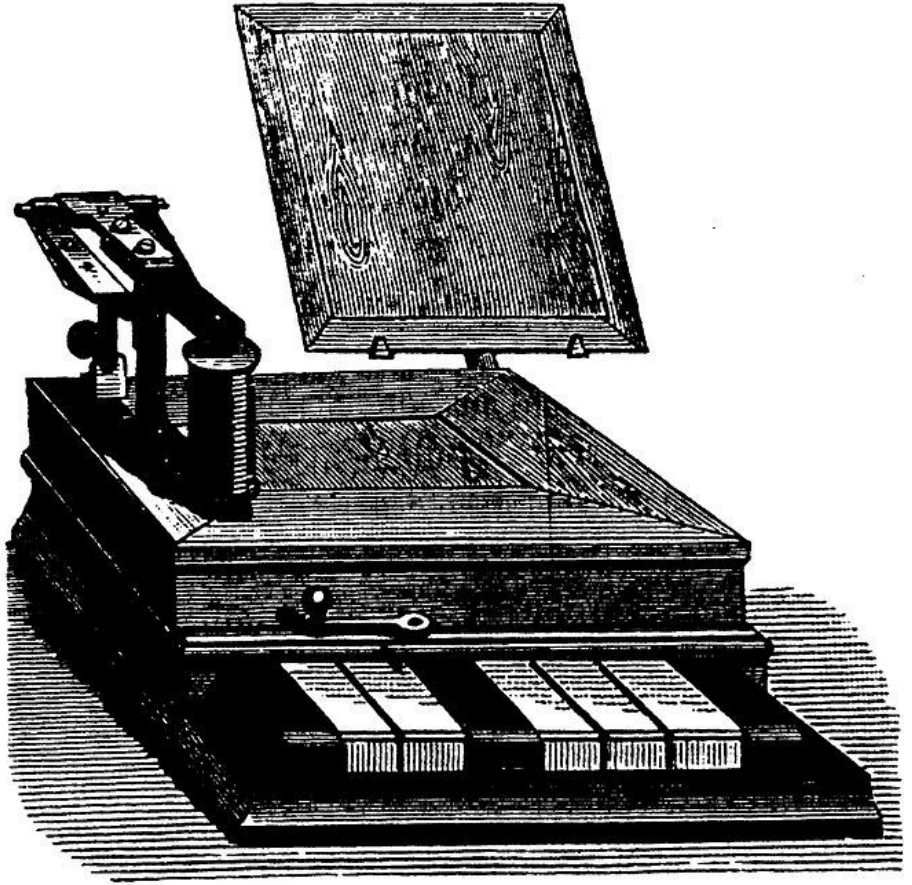
J. M. E. BAUDOT.

PRINTING TELEGRAPH.

No. 388,244. Patented Aug. 21, 1888.

Fig. 24.

	1	2	3	4	5
A	+	−	−	−	−
B	−	−	+	+	−
C	+	−	+	+	−
D	+	+	+	+	−
E	+	+	−	−	−
F	+	+	−	−	−
G	−	+	−	+	−
H	−	+	−	+	−
I	+	+	−	+	−
J	−	+	+	−	−
K	+	−	−	+	−
L	+	−	−	+	+
M	+	+	−	+	+
N	−	+	−	+	+
O	+	+	+	+	−
P	+	+	+	−	+
Q	+	−	+	+	+
R	−	−	+	+	+
S	+	−	+	−	+
T	+	−	+	−	+
U	+	+	+	−	−
V	−	+	+	−	+
W	−	+	−	−	+
X	−	−	+	−	−
Y	+	+	−	−	−
Z	+	−	−	−	+
&	−	−	−	+	+
∴	−	−	−	+	−
	−	−	−	−	−

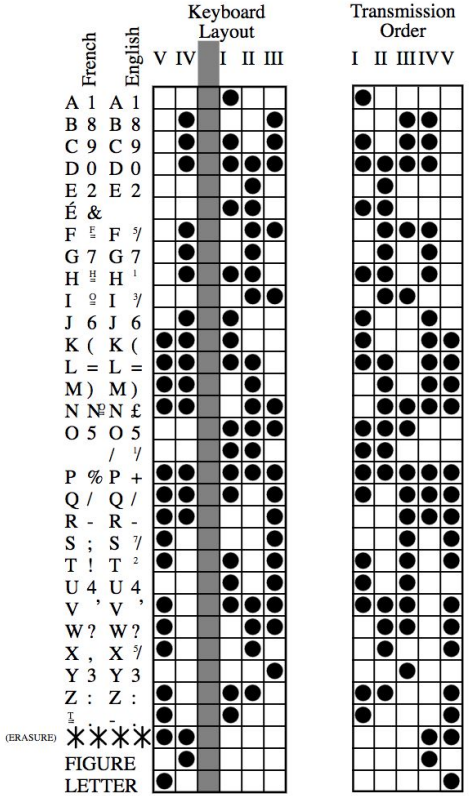


INVENTOR:

Jean Maurice Émile Baudot,

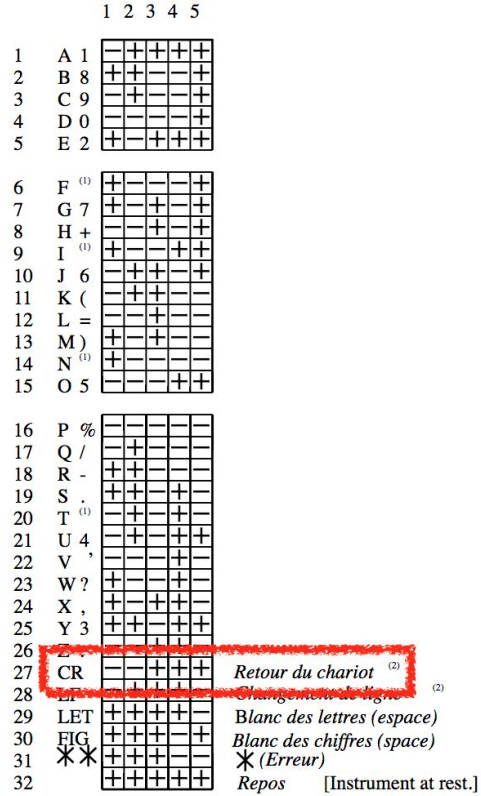
Baudôt Code

Alphabetic Presentation



Data from Pendry, H. W. *The Baudôt Printing Telegraph System*. 2ed. (London: Sir Isaac Pitman & Sons, Ltd., 1919): 43-44. and *Telegraph Regulations annexed to the International Telecommunication Convention; Final Protocol to the Telegraph Regulations (Madrid, 1932)*. (London: His Majesty's Stationery Office, 1933): Chapter IX, Article 35, p. 34.
Drawing 2010 www.CircuitousRoot.com; Public Domain

Alphabet télégraphique international no. 1 (ITA-1)



(1) *A la disposition de chaque administration pour son service interieur.*

(2) *Pour l'impremeur sur pages*

ASCII

1963

USASCII code chart

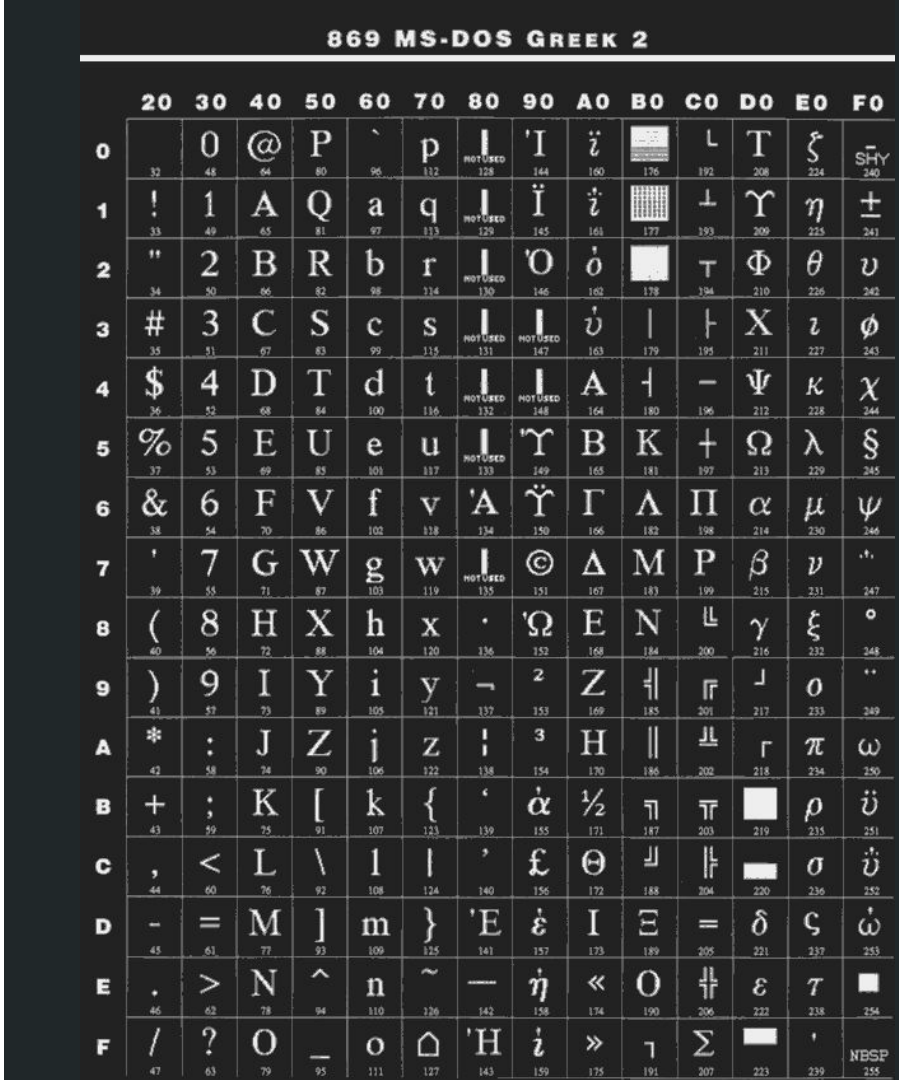
Bits b_7 b_6 b_5 →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	Row ↓ Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	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

The 8th Bit

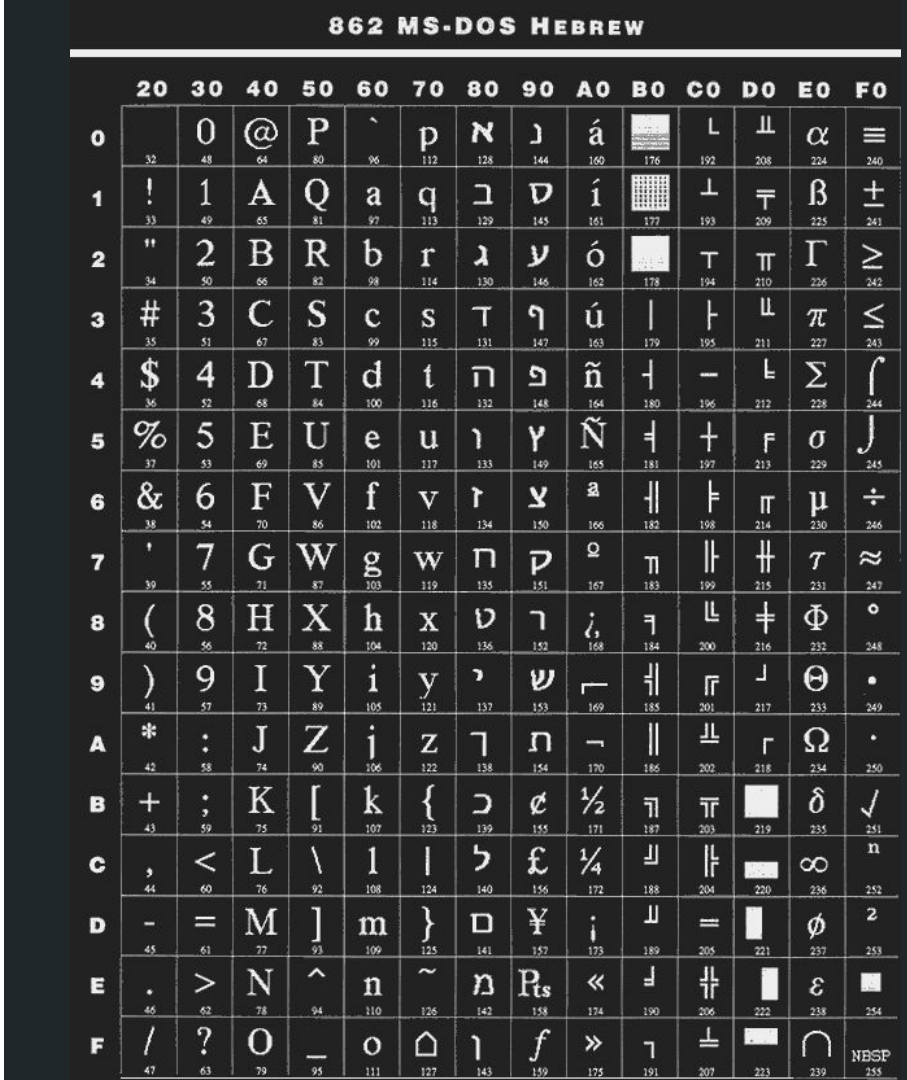
Latin-1


1252 WINDOWS LATIN 1 (ANSI)
20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0
0 1 2 3 4 5 6 7 8 9 A B C D E F
NOT USED
NBSP
SHY

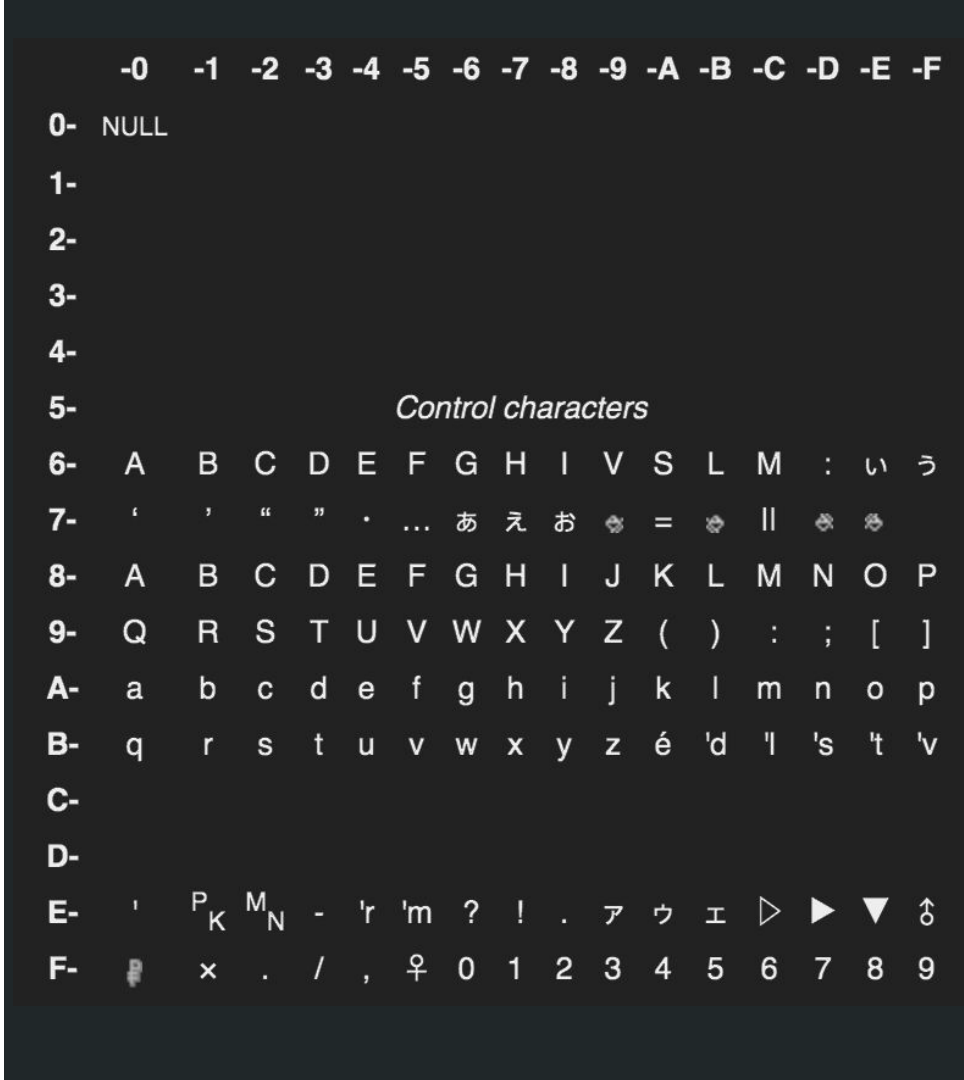
Greek


869 MS-DOS Greek 2
20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0
0 1 2 3 4 5 6 7 8 9 A B C D E F
NOT USED
SHY
NBSP

Hebrew


862 MS-DOS Hebrew
20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0
0 1 2 3 4 5 6 7 8 9 A B C D E F
NBSP

?


-0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F
0- NULL
1-
2-
3-
4-
5-
Control characters
6- A B C D E F G H I V S L M : い う
7- ‘ ’ “ ” ・ … あ え お = ‖
8- A B C D E F G H I J K L M N O P
9- Q R S T U V W X Y Z ( ) : ; [ ]
A- a b c d e f g h i j k l m n o p
B- q r s t u v w x y z é 'd 'l 's 't 'v
C-
D-
E- ' PK MN - 'r 'm ? ! . ア ウ エ ▷ ▶ ▼
F- × . / , ♀ 0 1 2 3 4 5 6 7 8 9

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	NULL															
1-																
2-																
3-																
4-																
5-	*Control characters*															
6-	A	B	C	D	E	F	G	H	I	V	S	L	M	:	ぃ	ぅ
7-	‘	’	“	”	・	…	あ	え	お	▯	=	▯	‖	▯	▯	
8-	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
9-	Q	R	S	T	U	V	W	X	Y	Z	(	)	:	;	[	]
A-	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
B-	q	r	s	t	u	v	w	x	y	z	é	'd	'l	's	't	'v
C-																
D-																
E-	'	P_K	M_N	-	'r	'm	?	!	.	ァ	ゥ	ェ	▷	▶	▼	♂
F-	▯	×	.	/	,	♀	0	1	2	3	4	5	6	7	8	9

Meanwhile, in Japan...

Kanji

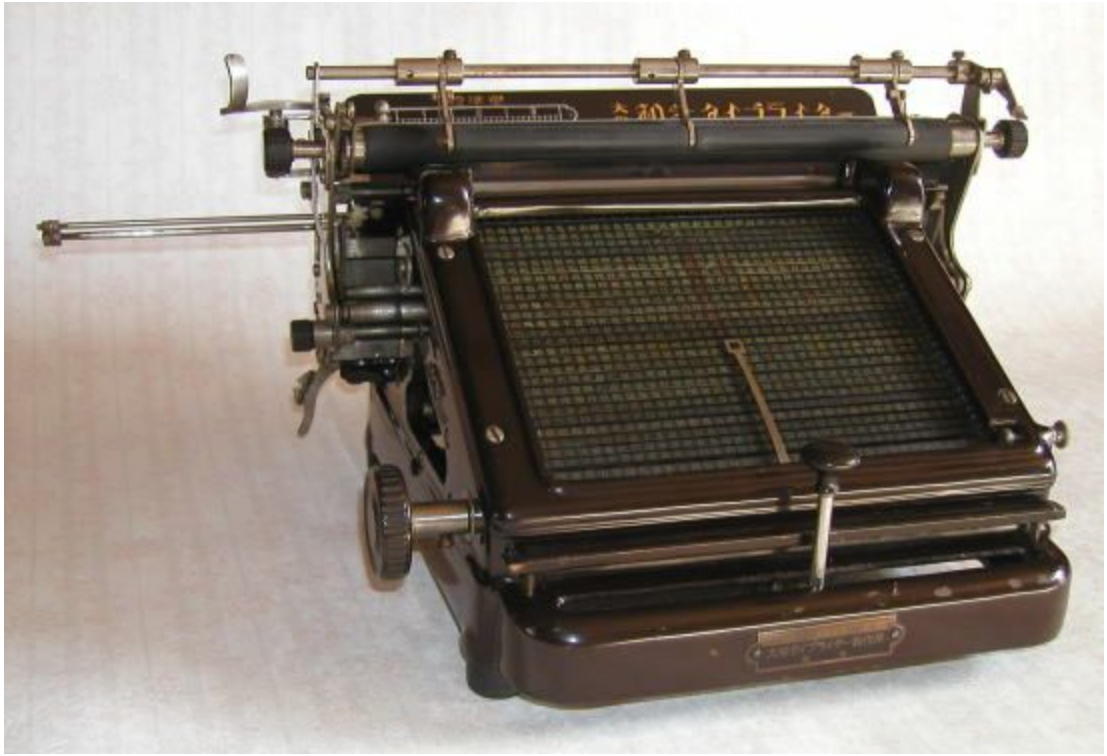
- Nouns
- Verbs
- Adverbs

Hiragana

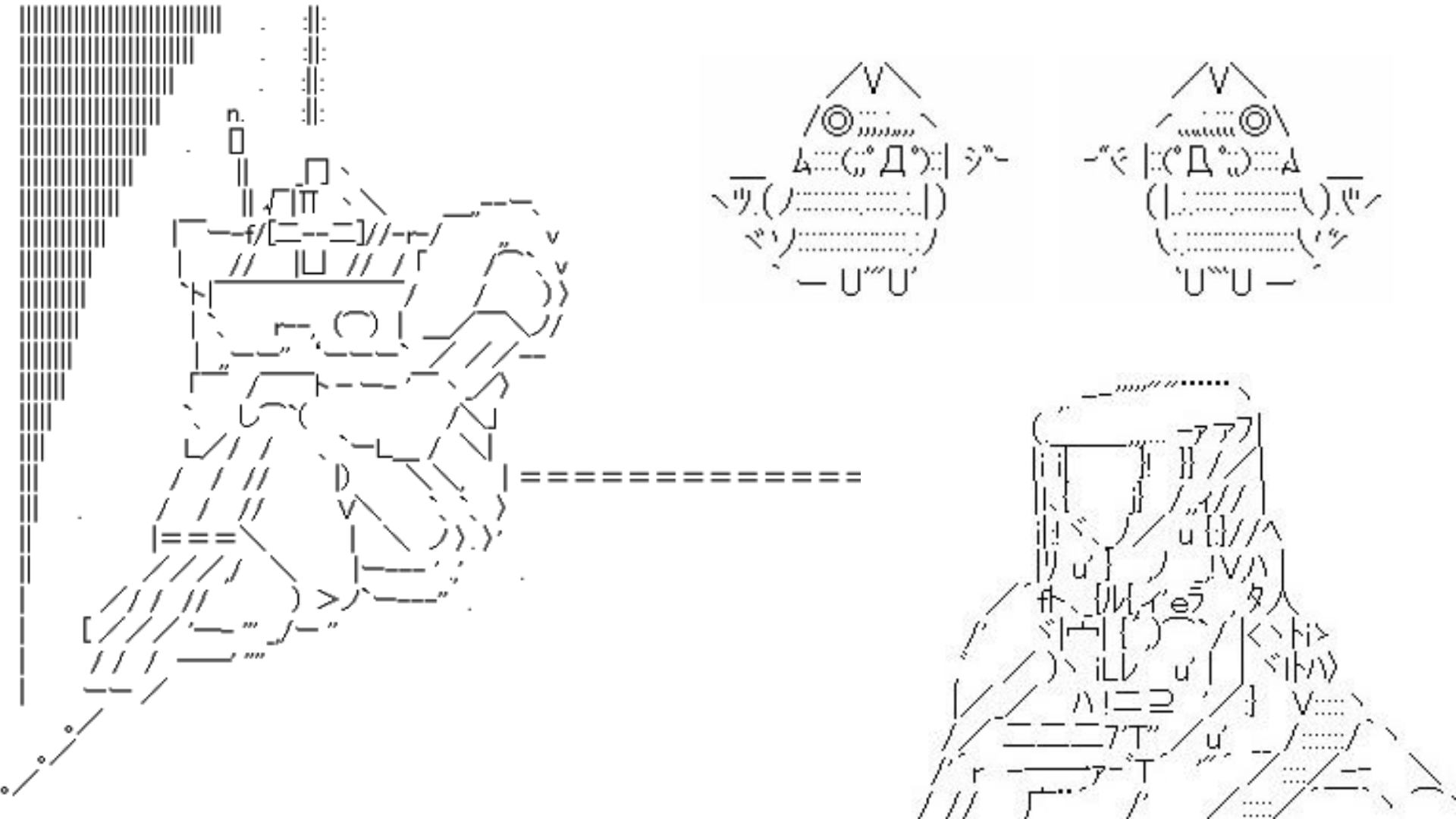
- inflectional endings
- pronunciations
- phonetic renderings of kanji

Katakana

- foreign words
- technical terms



Row	Column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Bit Pat. →	00				01				10				11			
	↓	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0	0000			SP	0	@	P	`	p			KS	-	タ	ミ		
1	0001			!	1	A	Q	a	q			。	ア	チ	ム		
2	0010			"	2	B	R	b	r			「	イ	ツ	メ		
3	0011			#	3	C	S	c	s			」	ウ	テ	モ		
4	0100			$	4	D	T	d	t			、	エ	ト	ヤ		
5	0101			%	5	E	U	e	u			・	オ	ナ	ユ		
6	0110			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7	0111			'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8	1000			(	8	H	X	h	x			ィ	ク	ネ	リ		
9	1001			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
10	1010			*	:	J	Z	j	z			ェ	コ	ハ	レ		
11	1011			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
12	1100			,	<	L	¥	l	¦			ャ	シ	フ	ワ		
13	1101			-	=	M	]	m	}			ュ	ス	ヘ	ン		
14	1110			.	>	N	^	n	~			ョ	セ	ホ	゛		
15	1111			/	?	O	_	o	DEL			ッ	ソ	マ	°		

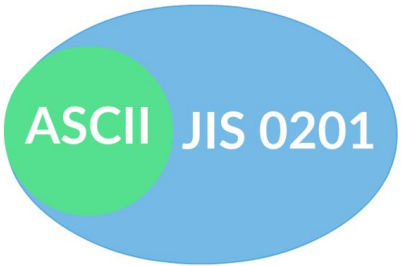


Japanese Character Sets

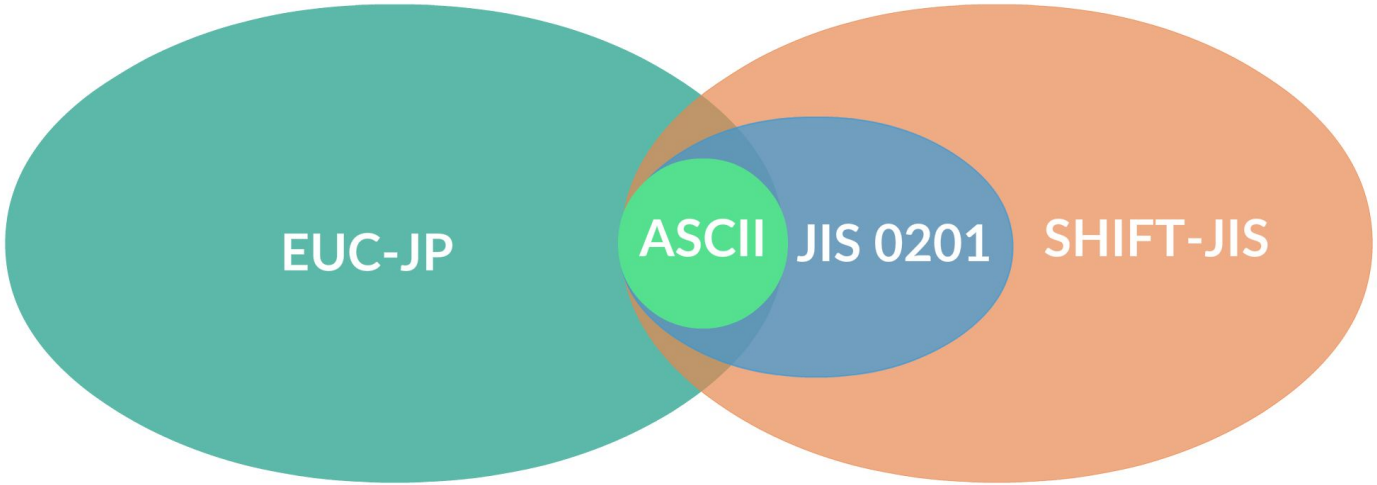
Name	Year	Supports			
		Latin	Katakana	Hiragana	Kanji
ASCII	1963	✔			
JIS X 0201	1969	✔	✔		
JIS X 0208	1978	✔	✔	✔	✔
JIS X 0212	1990	✔	✔	✔	✔✔
JIS X 0213	2000	✔	✔	✔	✔✔✔

Japanese Encodings

Japanese Encodings

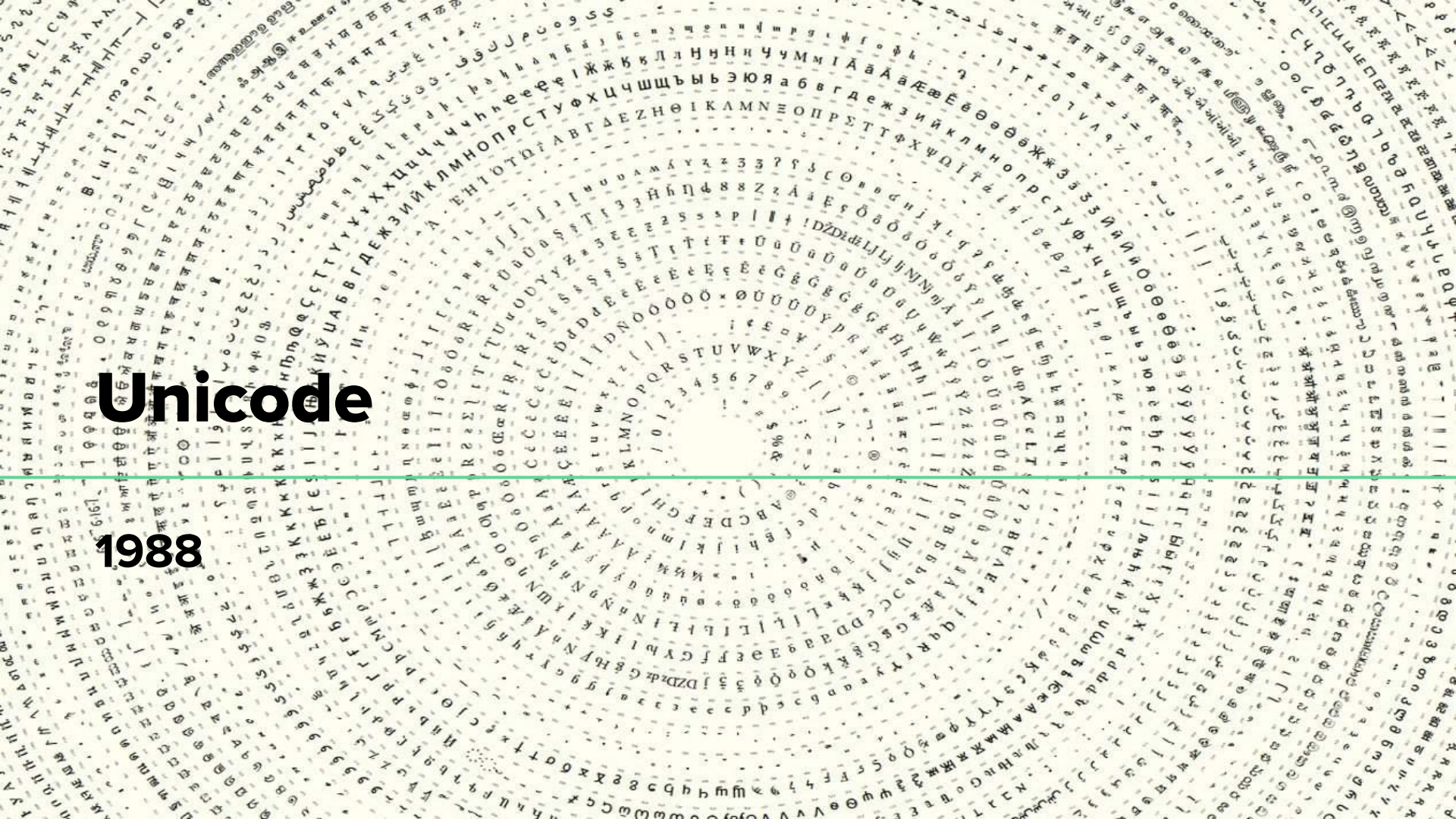


Japanese Encodings



Japanese Encodings

Name	Bytes	Supports			
		ASCII	JIS X 0201	JIS X 0208	JIS X 0213
ASCII	<1	✔			
JIS X 0201	1	✔	✔		
SHIFT-JIS	2	✔	✔	✔	
EUC-JP	2-3	✔	✔	✔	✔


Unicode
1988


今晩は

“enabling people around the world to use computers in any language”

Character Set

A list of characters recognized by hardware.

Unicode is a character set.

Encoding

Is a system of rules that converts a character set to and from binary

UTF-8, UTF-16, and UTF-32 are encodings

Name	SNOWMAN
Category	Symbol, Other
Code Point	U+2603

UTF-8	`11100010`	`10011000`	`10000011`	
UTF-16	`00100110`	`00000011`		
UTF-32	`00000000`	`00000000`	`00000011`	`00000011`

UTF-16

(originally USC-2)

- U+0000 to U+FFFF are encoded as a single 2-byte character
- U+10000 to U+10FFFF are encoded as two 2-byte characters called “surrogate pairs”

UTF-32

(originally USC-4)

- U+0000 to U+10FFFF are encoded as a single 4-byte character

UTF-8

- U+0000 to U+007F are encoded as a 1 byte character
- U+0080 to U+07FF are encoded as a 2 byte character
- U+0800 to U+FFFF are encoded as a 3 byte character
- U+10000 to U+10FFFF are encoded as a 4 byte character

UTF-8 vs UTF-16 vs UTF-32

Glyph	Name	Code Point	UTF-8 (hex)	UTF-16 (hex)	UTF-32 (hex)
a	LATIN SMALL LETTER A	U+0061	61	00 61	00 00 00 61
â	LATIN SMALL LETTER A WITH CIRCUMFLEX	U+00E2	C3 A2	00 E2	00 00 00 E2
本	---	U+672C	E6 9C AC	67 2C	00 00 67 2C
	PILE OF POO	U+1F4A9	F0 9F 92 A9	D8 3D DC A9	00 01 F4 A9

UTF-8

Code Points	Byte 1	Byte 2	Byte 3	Byte 4
U+0000 – U+007F	0xxxxxxx			
U+0080 – U+0800	110xxxxx	10xxxxxx		
U+0800 – U+10000	1110xxxx	10xxxxxx	10xxxxxx	
U+10000 – U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

ASCII is a subset of UTF-8

Code Points	Byte 1	Byte 2	Byte 3	Byte 4
U+0000 – U+007F	0xxxxxxx			
U+0080 – U+0800	110xxxxx	10xxxxxx		
U+0800 – U+10000	1110xxxx	10xxxxxx	10xxxxxx	
U+10000 – U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

```
>>> u_snowman = u'☃'
>>> u_snowman
U'\u2603'

>>> b_snowman = u_snowman.encode('utf-8')
>>> b_snowman
'\xe2\x98\x83'

>>> b_snowman.decode('utf-8')
U'\u2603'
```

```
>>> u_snowman = '☃'
>>> u_snowman
'☃'

>>> b_snowman = u_snowman.encode('utf-8')
>>> b_snowman
B'\xe2\x98\x83'

>>> b_snowman.decode('utf-8')
'☃'
```

```
>>> u_snowman = u'☃'
>>> u_snowman
U'\u2603'
```

```
>>> b_snowman = u_snowman.encode('utf-8')
>>> b_snowman
'\xe2\x98\x83'
```

```
>>> b_snowman.decode('utf-8')
U'\u2603'
```

```
>>> u_snowman = '☃'
>>> u_snowman
'☃'
```

```
>>> b_snowman = u_snowman.encode('utf-8')
>>> b_snowman
B'\xe2\x98\x83'
```

```
>>> b_snowman.decode('utf-8')
'☃'
```

	Bytes	Unicode Code Points
Python 2.7	<type “str”>	<type “unicode”>
Python 3+	<class “bytes”>	<class “str”>

```
UnicodeDecodeError: 'ascii' codec can't
decode byte 0xc3 in position 6: ordinal not
in range(128)
```


pied piper


Jay-Z

In Python 2.7

```
>>> new_user = 'Jay-Z'
```

In Python 2.7

```
>>> new_user = 'Jay-Z'
>>> welcome_message = u'Welcome to my ☃ startup' + new_user
```

In Python 2.7

```
>>> new_user = 'Jay-Z'
>>> welcome_message = u'Welcome to my ☃ startup' + new_user
>>> print welcome_message.encode('utf-8')
Welcome to my ☃ startup Jay-Z
```


pied piper


Beyoncé


Beyonc\xc3\xa9

In Python 2.7

```
>>> new_user = 'Beyonc\xc3\xa9'
```

In Python 2.7

```
>>> new_user = 'Beyonc\xc3\xa9'
>>> welcome_message = u'Welcome to my ☃ startup' + new_user
```

In Python 2.7

```
>>> new_user = 'Beyonc\xc3\xa9'
>>> welcome_message = u'Welcome to my ☃ startup' + new_user
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 in position 6:
ordinal not in range(128)
```


pied piper

In Python 2.7

```
>>> new_user = 'Beyonc\xc3\xa9'
>>> welcome_message = u'Welcome to my ☃ startup' + new_user
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 in position 6:
ordinal not in range(128)
```

In Python 2.7

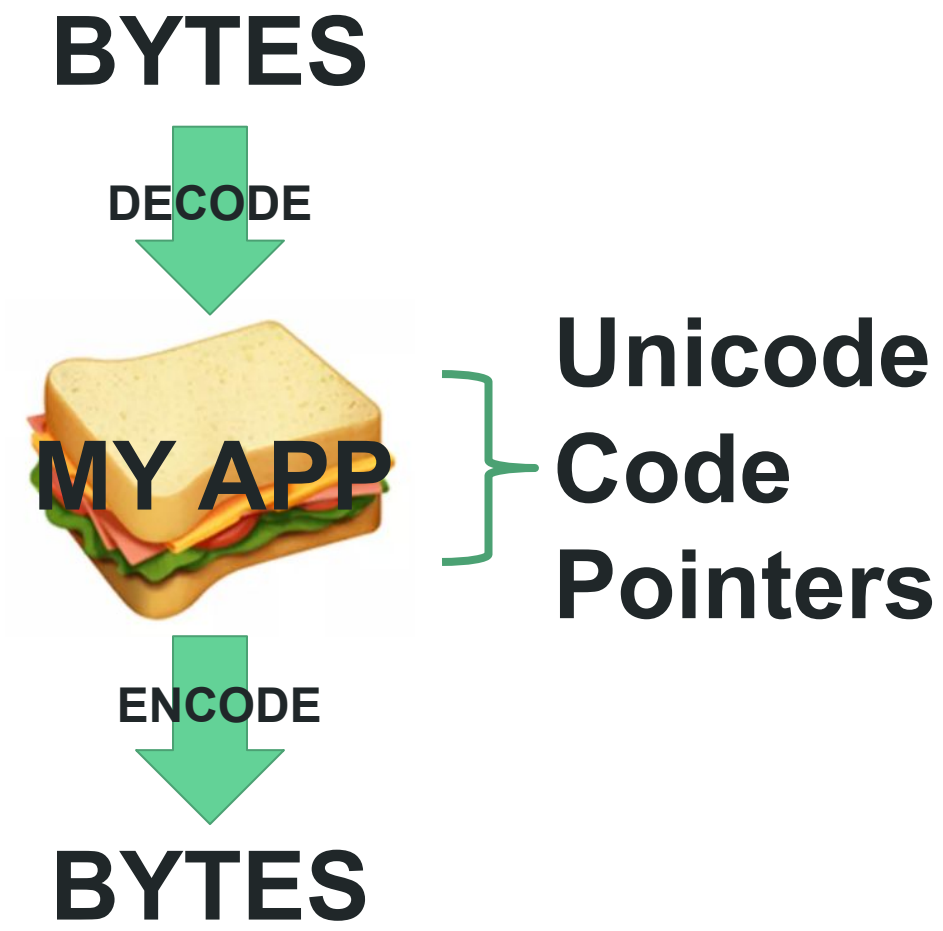
```
>>> new_user = 'Beyonc\xc3\xa9'
>>> welcome_message = u'Welcome to my ⛄ startup' + new_user
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 in position 6:
ordinal not in range(128)
```

In Python 3

```
>>> new_user = b'Jay-Z'
>>> welcome_message = 'Welcome to my ☃ startup' + new_user
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'bytes' object to str implicitly
```

Python 3 implicitly converts nothing

Best Practices


BYTES
DECODE
MY APP
Unicode
Code
Pointers
ENCODE
BYTES


Beyoncé

ありがとうございます

Thank You