File Formats

INF 551

Wensheng Wu

File Formats

- Specify what information bits in file encode

- Example: text file
  - String of characters with particular encoding scheme, e.g., ASCII and Unicode
  - E.g., TXT, HTML, JSON, XML

- Others: xls, ppt, pdf, jpg, gif, mp3, png, etc.

Roadmap

- Character encoding
  - ASCII
  - Unicode
- JSON (done earlier)
- XML (will talk about it next)

Code space & points

- Code space
  - A range of numerical values available for encoding characters
  - E.g., 0 to 10FFFF for Unicode, 0 to 7F for ASCII

- Code point
  - A value for a character in a code space

- Unicode code point
  - U+ followed by its hexadecimal value, e.g., U+0058 for capital letter ‘X’)

Encoding (of code points)

- Code unit: the smallest unit (comprising a number of bits) used to construct an encoding for a code point
  - Code unit for UTF-8: 8-bit
  - UTF-16:16-bit
- UTF (Unicode Transformation Format) encoding
  - E.g., UTF-8 and UTF-16

Variable-length encoding

- Characters encoded using codes of different length

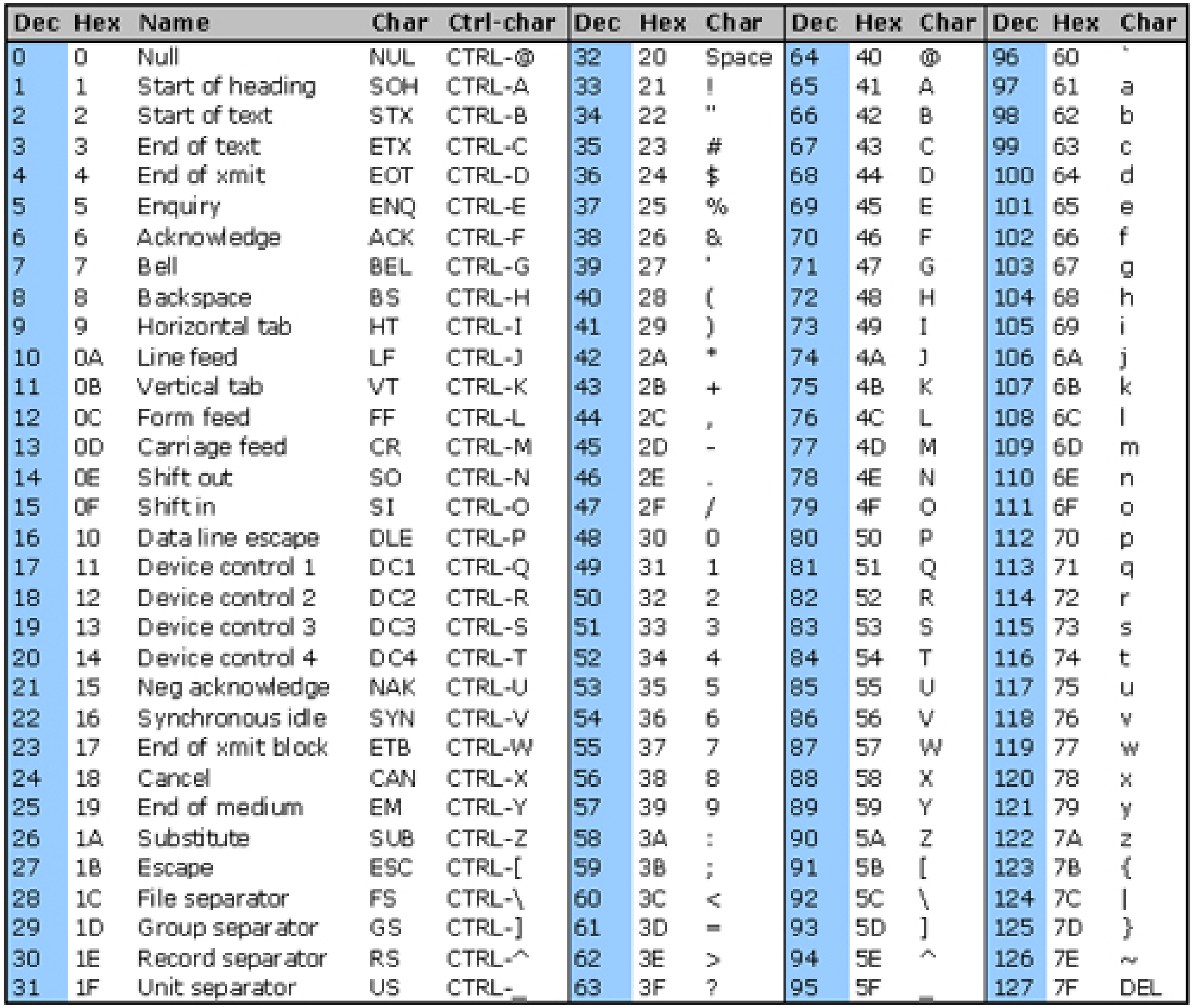
- In Unicode, a code point may be represented using multiple code units
  - E.g., 1-4 in UTF-8, 1-2 in UTF-16

ASCII

- American Standard Code for Information Interchange

- 128 characters: 7-bit code (code points: 0~7F)
  - Digits: 0-9 (0x30 – 0x39)
  - Uppercase letters: A-Z (0x41 – 0x5A)
  - Lowercase letters: a-z (0x61 – 0x7A)
  - White space (0x20)
  - Punctuation symbols
  - Control characters (e.g., Ctrl-C: 0x03)

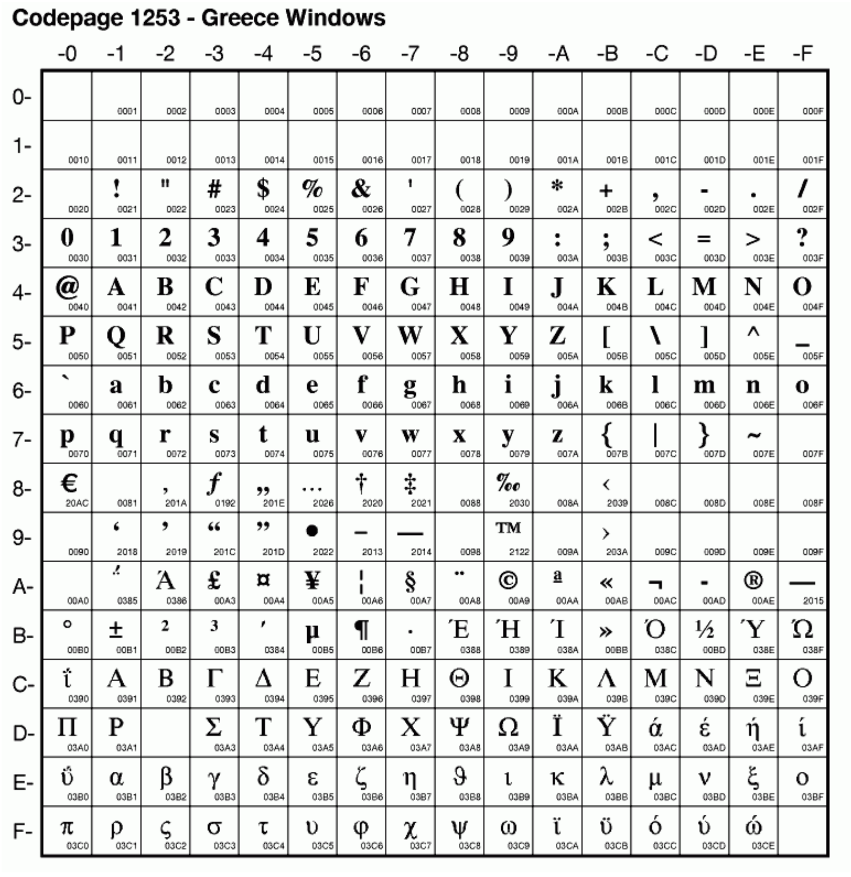
ASCII



Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Windows-1253

- Windows code page for Latin + Greek characters
- Use 8 bits
  - – 0x00 ~ 0xFF



Codepage 1253 - Greece Windows

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-		0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3-	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4-	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5-	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6-	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7-	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8-	€ 20AC	0081	‚ 201A	ƒ 0192	„ 201E	… 2026	† 2020	‡ 2021	0088	‰ 2030	008A	‹ 2039	008C	008D	008E	008F
9-	0090	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	0098	™ 2122	009A	› 203A	009C	009D	009E	009F
A-	00A0	΅ 0385	Ά 0386	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	― 2015
B-	° 00B0	± 00B1	² 00B2	³ 00B3	΄ 0384	µ 00B5	¶ 00B6	· 00B7	Έ 0388	Ή 0389	Ί 038A	» 00BB	Ό 038C	½ 00BD	Ύ 038E	Ώ 038F
C-	ΐ 0390	Α 0391	Β 0392	Γ 0393	Δ 0394	Ε 0395	Ζ 0396	Η 0397	Θ 0398	Ι 0399	Κ 039A	Λ 039B	Μ 039C	Ν 039D	Ξ 039E	Ο 039F
D-	Π 03A0	Ρ 03A1		Σ 03A3	Τ 03A4	Υ 03A5	Φ 03A6	Χ 03A7	Ψ 03A8	Ω 03A9	Ϊ 03AA	Ϋ 03AB	ά 03AC	έ 03AD	ή 03AE	ί 03AF
E-	ΰ 03B0	α 03B1	β 03B2	γ 03B3	δ 03B4	ε 03B5	ζ 03B6	η 03B7	ϑ 03B8	ι 03B9	κ 03BA	λ 03BB	μ 03BC	ν 03BD	ξ 03BE	ο 03BF
F-	π 03C0	ρ 03C1	ς 03C2	σ 03C3	τ 03C4	υ 03C5	φ 03C6	χ 03C7	ψ 03C8	ω 03C9	ϊ 03CA	ϋ 03CB	ό 03CC	ύ 03CD	ώ 03CE	

Unicode

- Unicode supports more characters than ASCII and various codepages

- Unicode separates code points from encoding
  - In contrast to ASCII, where code point = encoding

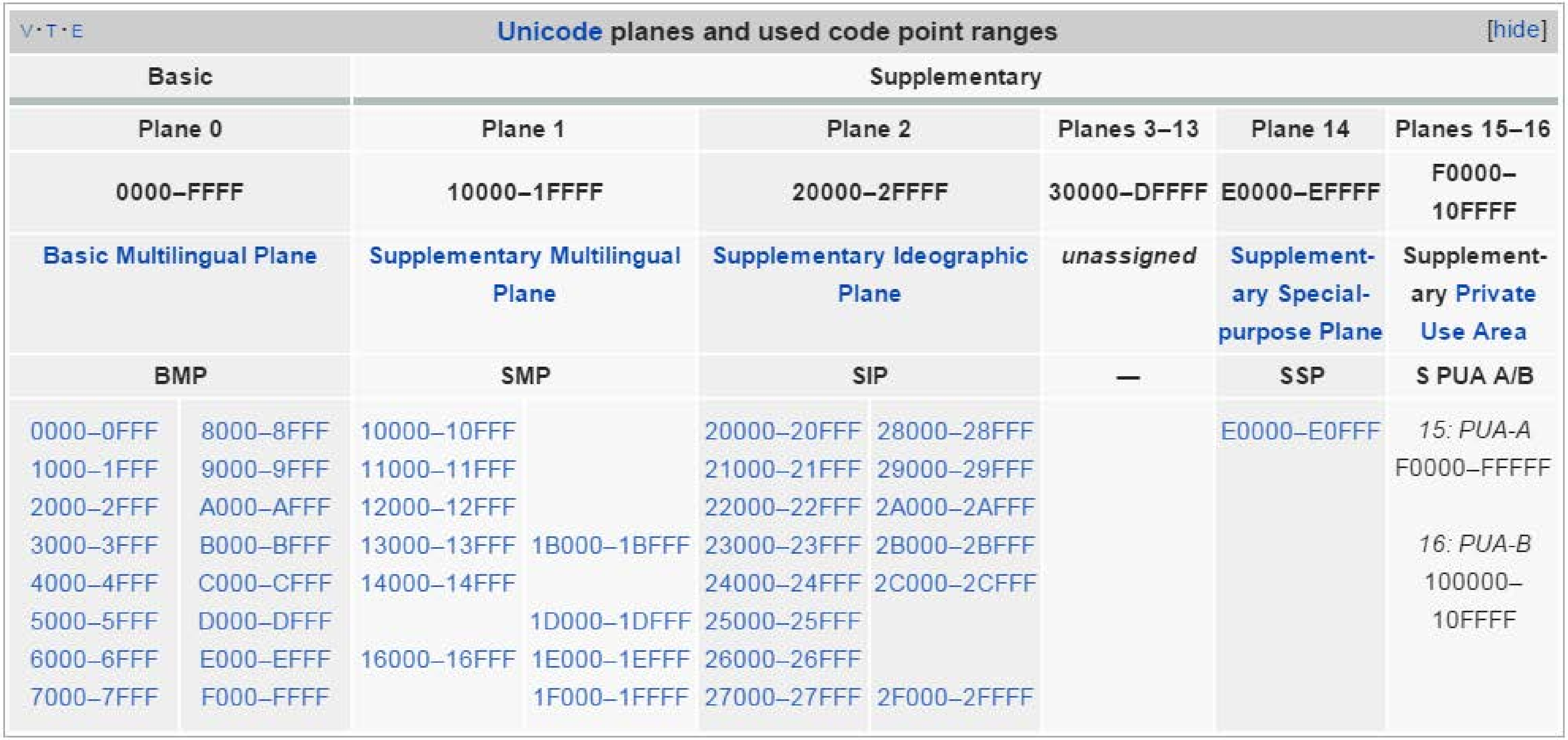
Unicode

- Code space is divided into 17 planes
- Each plane = contiguous 2^{16} code points
- Recall that code points range from 0 to 10FFFF

$\Rightarrow$Total code points = 17 * 2^{16} or 1,114,112
code points

Note 2^{16} = 65,536

Planes in Unicode

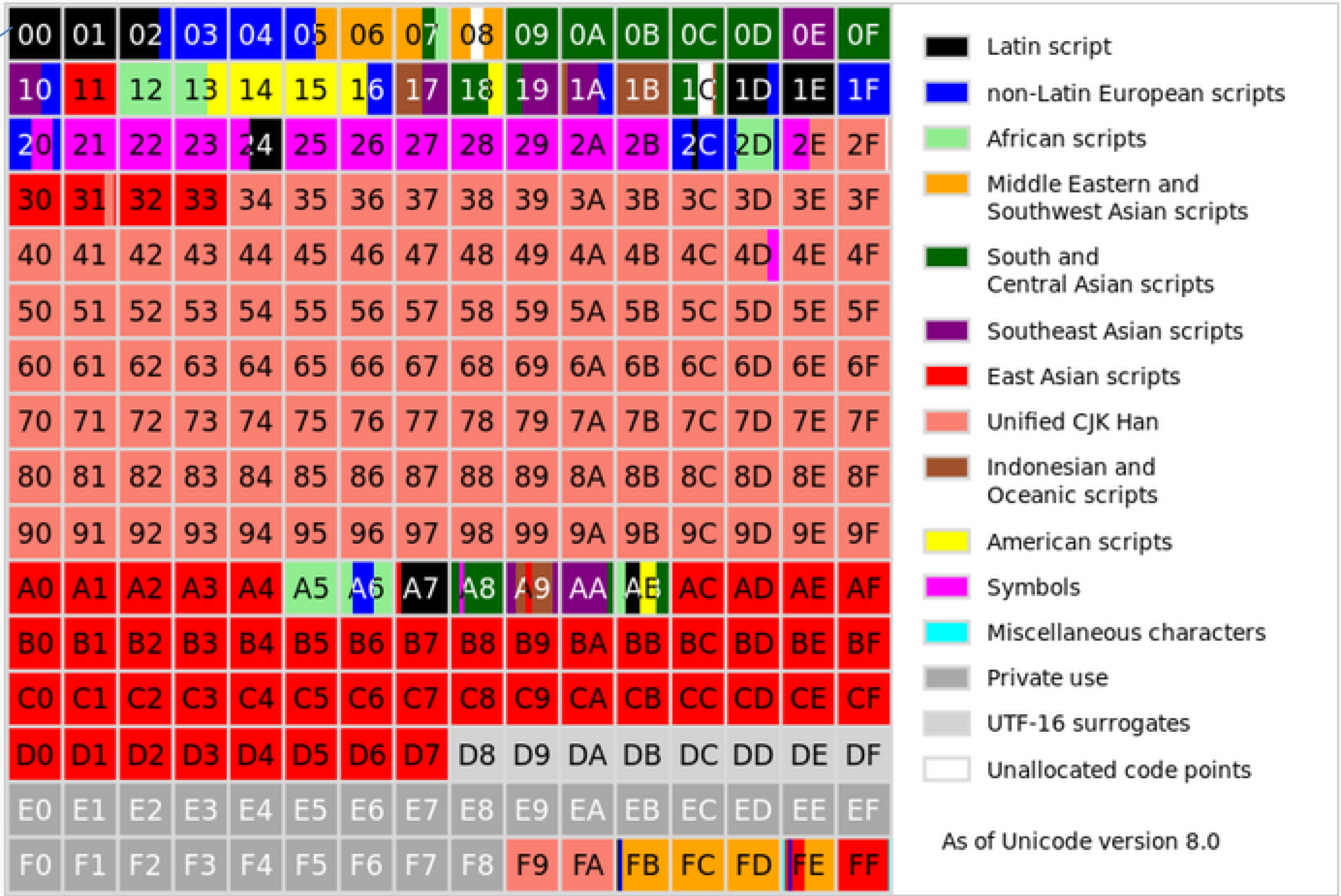


V·T·E Unicode planes and used code point ranges [hide]

Basic		Supplementary						
Plane 0		Plane 1		Plane 2		Planes 3–13	Plane 14	Planes 15–16
0000–FFFF		10000–1FFFF		20000–2FFFF		30000–DFFFF	E0000–EFFFF	F0000–10FFFF
Basic Multilingual Plane		Supplementary Multilingual Plane		Supplementary Ideographic Plane		*unassigned*	Supplementary Special-purpose Plane	Supplementary Private Use Area
BMP		SMP		SIP		—	SSP	S PUA A/B
0000–0FFF	8000–8FFF	10000–10FFF		20000–20FFF	28000–28FFF		E0000–E0FFF	*15: PUA-A* F0000–FFFFF *16: PUA-B* 100000–10FFFF
1000–1FFF	9000–9FFF	11000–11FFF		21000–21FFF	29000–29FFF			
2000–2FFF	A000–AFFF	12000–12FFF		22000–22FFF	2A000–2AFFF			
3000–3FFF	B000–BFFF	13000–13FFF	1B000–1BFFF	23000–23FFF	2B000–2BFFF			
4000–4FFF	C000–CFFF	14000–14FFF		24000–24FFF	2C000–2CFFF			
5000–5FFF	D000–DFFF		1D000–1DFFF	25000–25FFF				
6000–6FFF	E000–EFFF	16000–16FFF	1E000–1EFFF	26000–26FFF				
7000–7FFF	F000–FFFF		1F000–1FFFF	27000–27FFF	2F000–2FFFF			

Plane 0: BMP (Basic Multilingual Plane)

Block 00
Represents
0000~00FF



00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

- ⬛ Latin script
- 🟦 non-Latin European scripts
- 🟩 African scripts
- 🟧 Middle Eastern and Southwest Asian scripts
- 🟩 South and Central Asian scripts
- 🟪 Southeast Asian scripts
- 🟥 East Asian scripts
- 🟥 Unified CJK Han
- 🟫 Indonesian and Oceanic scripts
- 🟨 American scripts
- 🟪 Symbols
- 🟦 Miscellaneous characters
- ⬜ Private use
- ⬜ UTF-16 surrogates
- ⬜ Unallocated code points

As of Unicode version 8.0

A map of the Basic Multilingual Plane. Each numbered box represents 256 code points.

Each block represents 256 code points

UTF-8

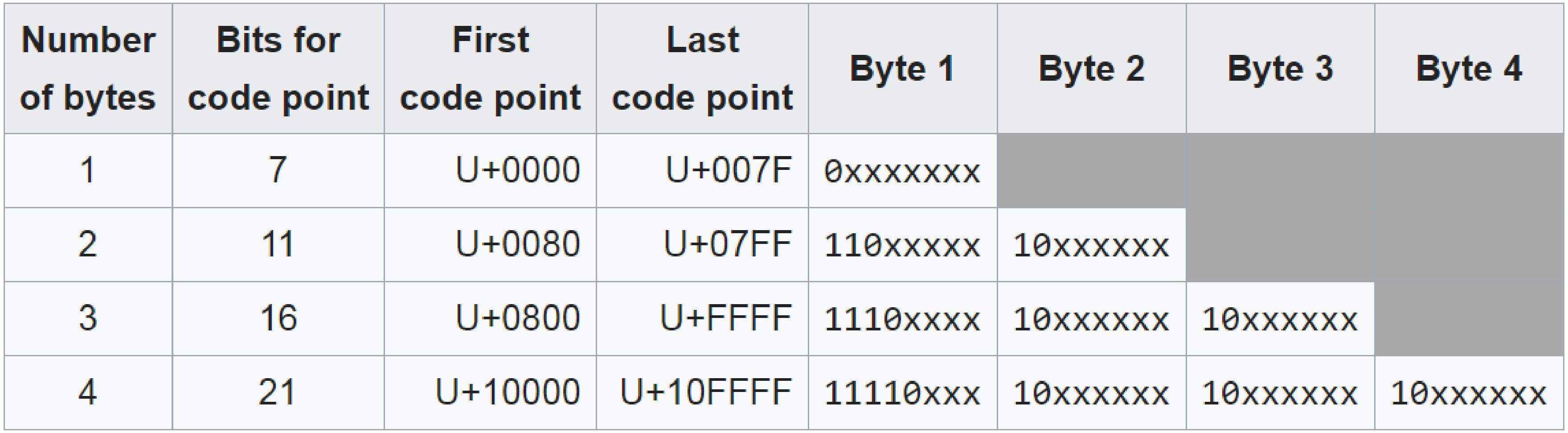
- Encoding scheme for Unicode code space

- Code unit = 8 bits

- Variable length
  - Code point may be represented using 1-4 code units

UTF-8 Design

- ASCII characters use one code unit
  - First bit is zero
- Other Unicode characters use up to 4 units



Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

UTF-8 Features

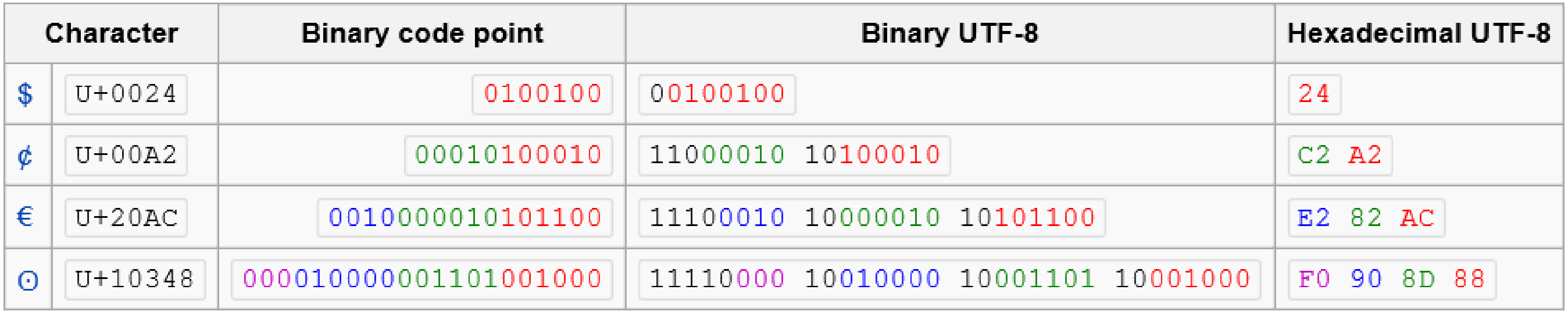
- Backward compatibility
  - One byte for ASCII, leading bit of byte is zero
- Clear distinction btw single- vs. multi-byte characters
  - Single-byte/multi-byte: start with 0/1 respectively
- Multiple length
  - a leading byte starts with 2 or more 1’s, followed by a 0, e.g., ‘110’, ‘1110’, etc.
  - One or more continuation bytes all start with ‘10’

UTF-8 Features

- Clear indication of code sequence length
  - By # of 1’s in leading byte (for multi-byte)
- Self-synchronization
  - Can find start of characters by backing up at most 3 bytes

Example

- Encode ‘€’ using UTF-8
- Code point = U+20AC
- Need 3 bytes in UTF-8



Character		Binary code point	Binary UTF-8	Hexadecimal UTF-8
$	U+0024	0100100	00100100	24
¢	U+00A2	00010100010	11000010 10100010	C2 A2
€	U+20AC	0010000010101100	11100010 10000010 10101100	E2 82 AC
𐍈	U+10348	000010000001101001000	11110000 10010000 10001101 10001000	F0 90 8D 88

Unicode in Python

- >>> a = u'\u20AC' # note need u before '
- >>> print a
- €

u indicates it is a Unicode string

- >>> e = u'€'
- >>> e
- u'\u20ac'

Unicode in Python

- >>> b = '€'
- >>> b
- '\xe2\x82\xac'
  - UTF-8 encoding of €

- >>> u'€'.encode('utf-8')
- '\xe2\x82\xac'

Resources

- UTF-8
  - https://en.wikipedia.org/wiki/UTF-8
- UTF-16
  - https://en.wikipedia.org/wiki/UTF-16