

ENCODING AND DECODING FOR NETWORK TEXT TRANSMISSION

Overview

- History of computer/network text encoding
 - ASCII
 - Limitations of the ASCII character set
- Unicode
 - Incorporation of all known human character sets
 - Code points
- Unicode Encoding and Decoding
 - UTF-8, UTF-16, UTF-32, etc.
- Unicode examples and Python 3 support

Early Computer Data Transmission

- Created for plain English letter/number focus
- Each letter, number, capital letters, etc., were encoded by assigned binary codes
- American Standard Code for Information Interchange (ASCII)
- Since early computer data transmission was often to peripherals, ASCII also included control character assignments

ASCII

- American Standards Association (now American National Standards Institute, (ANSI)) standard starting in 1960 and first published in 1963 (last update 1986).
- 7-bit codewords in 0-127 range, e.g., “T” : 84; “t” : 116; space: 32.
- 33 non-printing control characters. These affect how text and spaces are processed.
- Now properly referred to as US-ASCII.

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(88	58	1011000	130	X					
41	29	101001	51)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

What about the 8th Bit?

- Many computers were based on 8-bit data words (bytes)
- ASCII left half of the available codes undefined i.e., 128-255.
- Sometimes used as a parity bit
- The spare codewords started to be assigned in non-standard ways, e.g., some industry proprietary, some assigned in non-English speaking countries. This created an interoperability mess.

Text Encoding Example

- Extended Binary Coded Decimal Interchange Code (EBCDIC) devised by IBM (1963/1964) for peripheral communications. 8-bit code.
- EBCDIC defined different "code pages" to extend encoding to accommodate different languages.
- A code page is a table of values that describes the character set used for encoding
- Various "double byte" character sets were used to encode (mainly) Eastern languages.
- There are many other text encodings, e.g.,

ISO-8859-1 (Latin-1)

- Upward compatible ASCII extension that uses all 8 bits and adds 96 additional characters. Many Western European languages are completely covered:
- Afrikaans, Albanian, Basque, Breton, Corsican, Danish[a], English, Faroese, Galician, German, Icelandic, Irish, Indonesian, Italian, Kurdish, Leonese, Luxembourgish, Malay, Manx, Norwegian, Occitan, Portuguese, Rhaeto-Romanic, Scottish Gaelic, Spanish, Swahili, Swedish, Walloon
- About 7% of web sites used this encoding in 2016.
- ISO-8859-1 is the first 256 code points in Unicode

Other Language Encodings









































- Windows-1252 adds 27 more characters
- Latin-2, Latin-3, ..., Latin-9, (i.e., ISO-8859-2, ISO-8859-3,..., ISO-8859-9) target other (mainly Western European) languages.
- Many other languages are missing!
- How to accommodate all world-wide languages?

The Unicode Standard

Unicode

- Intended to be the final world-wide standard text representation mechanism, e.g., enables standard world-wide data communications.
- All the major living scripts are included.
- Defines "abstract characters" for all languages., i.e., basic units of textual information.
- Gives a name and a unique binary "code point" identifier to each abstract character.
- Code points are not intended for data communication.

Emojis have been recently added, e.g.,

<u>No</u>	<u>Code</u>	<u>Browser</u>	<u>Appl</u>	<u>Goog</u>	<u>FB</u>	<u>Wind</u>	<u>Twtr</u>	<u>Joy</u>	<u>Sams</u>
1	U+1F600								
2	U+1F603								
3	U+1F604								
4	U+1F601								
5	U+1F606								

For a complete list, go to:

<https://unicode.org/emoji/charts/full-emoji-list.html>

Unicode Example

- Character: M
- Name: LATIN CAPITAL LETTER M
- Categories: uppercase letter, Left-to-Right
- Codepoint: U+004D
- i.e., "M" (Latin Capital Letter M) is assigned unicode code point of 004D, written as U+004D, i.e., a 16 bit word.
- Legal Unicode code point values: U+0000 to U+10FFFF (over 1 million characters). Only about 10% have have been assigned.

Unicode

- Prior approaches involve assigning a given character to a unique binary word that is used for transmission, e.g., ASCII assigns "a" to 97, i.e., transmitted as 01100001.
- Unicode instead assigns each symbol in an alphabet to a unique binary "code point". Code points are not intended for communication.

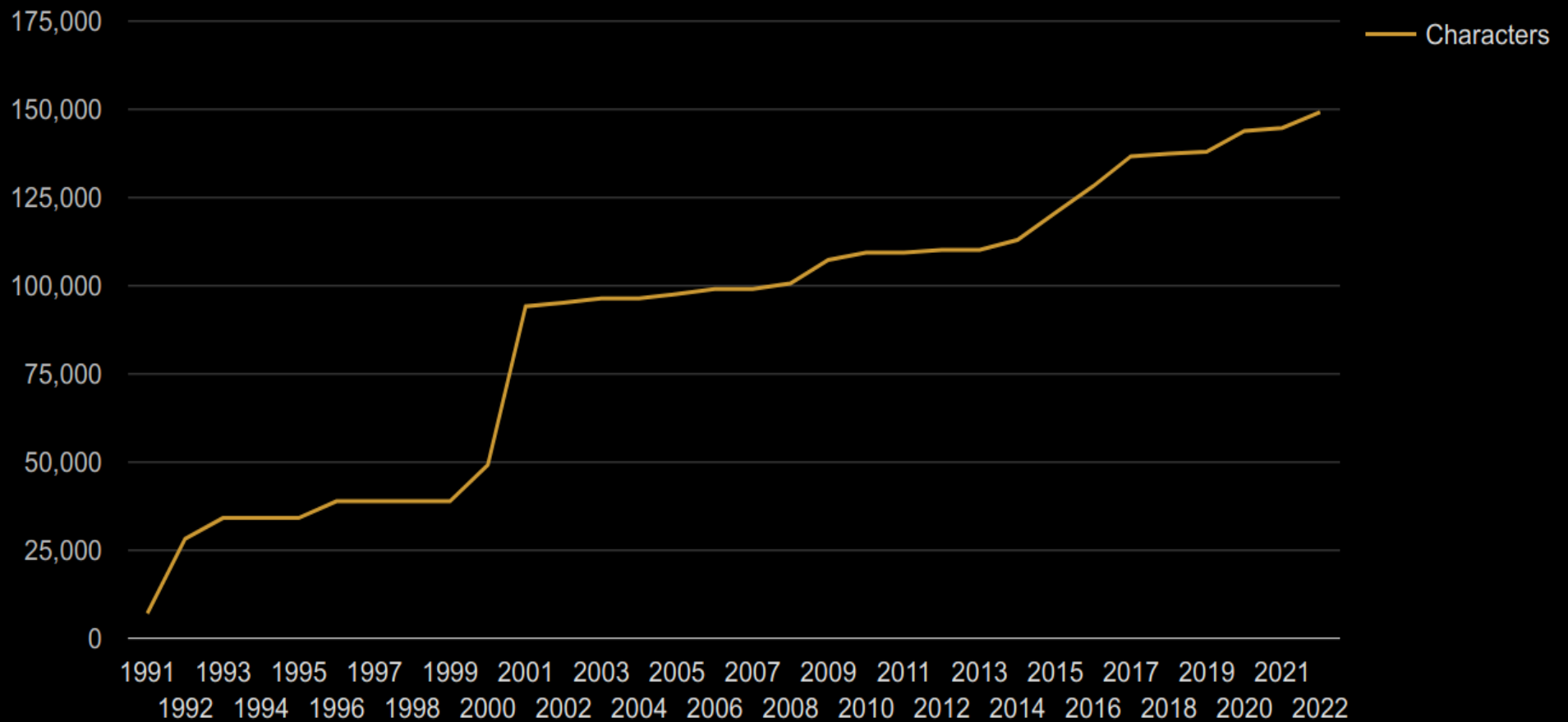
Unicode

- Example: "M" (Latin Capital Letter M) is assigned unicode code point of 004D, written as U+004D, i.e., a 16 bit word.
- Code points are multi-byte words. To transmit them, we would need to be careful about byte ordering, e.g., The word "cat" consists of the unicode sequence: U+0063 U+0061 U+0074
- Big-endian: 0063 0061 0074
- Little-endian: 0074 0061 0063
- Would probably need to include a byte order mark (BOM) in transmitted data.

Unicode

- Latest version of Unicode contains 149,813 characters that cover over 135 modern and historical scripts, as well as multiple symbol sets.
- Rules for things such as text normalization, composition, decomposition, collation, and bidirectional display order.
- Latest version is Unicode 15.1.0, maintained by the Unicode Consortium.

Unicode Codepoint Growth



Unicode Character Map

- Contains code planes of 64K (FFFF) code points each. Planes currently defined:
 - 0000 : Basic Multilingual Plane (BMP)
 - 0001 : Supplementary Multilingual Plane (SMP)
 - 0002 : Supplementary Ideographic Plane (SIP)
 - 0003 : Tertiary Ideographic Plane (TIP)
 - 000E: Supplementary Special-purpose Plane (SSP)
 - 000F: Private Use Plane (PUP)
 - 0010 : Private Use Plane (PUP)

Unicode Code Plane Geometry

- Each small block: $2^8 = 256$ code points
- Each row: 16 (2^4) small blocks = $2^4 \times 2^8 = 2^{12} = 4K$ (4096) code points
- Each plane: 16 (2^4) rows = $2^4 \times 2^{12} = 2^{16} = 64K$ (65536) code points

Plane 0: Basic Multilingual

(U+0000 to U+FFFF)

0000-00FF

0F00-0FFF

US-ASCII

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
- CJK characters
- Indonesian and Oceanic scripts
- American scripts
- Notational systems
- Symbols
- Private use
- UTF-16 surrogates
- No block

As of Unicode 15.1

F000-F0FF

FF00-FFFF

Plane 1: Supplementary Multilingual

(U+10000 to U+1FFFFF)

100	101	102	103	104	105	106	107	108	109	10A	10B	10C	10D	10E	10F
110	111	112	113	114	115	116	117	118	119	11A	11B	11C	11D	11E	11F
120	121	122	123	124	125	126	127	128	129	12A	12B	12C	12D	12E	12F
130	131	132	133	134	135	136	137	138	139	13A	13B	13C	13D	13E	13F
140	141	142	143	144	145	146	147	148	149	14A	14B	14C	14D	14E	14F
150	151	152	153	154	155	156	157	158	159	15A	15B	15C	15D	15E	15F
160	161	162	163	164	165	166	167	168	169	16A	16B	16C	16D	16E	16F
170	171	172	173	174	175	176	177	178	179	17A	17B	17C	17D	17E	17F
180	181	182	183	184	185	186	187	188	189	18A	18B	18C	18D	18E	18F
190	191	192	193	194	195	196	197	198	199	19A	19B	19C	19D	19E	19F
1A0	1A1	1A2	1A3	1A4	1A5	1A6	1A7	1A8	1A9	1AA	1AB	1AC	1AD	1AE	1AF
1B0	1B1	1B2	1B3	1B4	1B5	1B6	1B7	1B8	1B9	1BA	1BB	1BC	1BD	1BE	1BF
1C0	1C1	1C2	1C3	1C4	1C5	1C6	1C7	1C8	1C9	1CA	1CB	1CC	1CD	1CE	1CF
1D0	1D1	1D2	1D3	1D4	1D5	1D6	1D7	1D8	1D9	1DA	1DB	1DC	1DD	1DE	1DF
1E0	1E1	1E2	1E3	1E4	1E5	1E6	1E7	1E8	1E9	1EA	1EB	1EC	1ED	1EE	1EF
1F0	1F1	1F2	1F3	1F4	1F5	1F6	1F7	1F8	1F9	1FA	1FB	1FC	1FD	1FE	1FF



- 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
- Indonesian and Oceanic scripts
- American scripts
- Cuneiform
- Hieroglyphs
- Notational systems
- Symbols
- Unallocated code points

As of Unicode 15.1

Plane 2: Supplementary Ideographic

(U+20000 to U+2FFFF)

200	201	202	203	204	205	206	207	208	209	20A	20B	20C	20D	20E	20F
210	211	212	213	214	215	216	217	218	219	21A	21B	21C	21D	21E	21F
220	221	222	223	224	225	226	227	228	229	22A	22B	22C	22D	22E	22F
230	231	232	233	234	235	236	237	238	239	23A	23B	23C	23D	23E	23F
240	241	242	243	244	245	246	247	248	249	24A	24B	24C	24D	24E	24F
250	251	252	253	254	255	256	257	258	259	25A	25B	25C	25D	25E	25F
260	261	262	263	264	265	266	267	268	269	26A	26B	26C	26D	26E	26F
270	271	272	273	274	275	276	277	278	279	27A	27B	27C	27D	27E	27F
280	281	282	283	284	285	286	287	288	289	28A	28B	28C	28D	28E	28F
290	291	292	293	294	295	296	297	298	299	29A	29B	29C	29D	29E	29F
2A0	2A1	2A2	2A3	2A4	2A5	2A6	2A7	2A8	2A9	2AA	2AB	2AC	2AD	2AE	2AF
2B0	2B1	2B2	2B3	2B4	2B5	2B6	2B7	2B8	2B9	2BA	2BB	2BC	2BD	2BE	2BF
2C0	2C1	2C2	2C3	2C4	2C5	2C6	2C7	2C8	2C9	2CA	2CB	2CC	2CD	2CE	2CF
2D0	2D1	2D2	2D3	2D4	2D5	2D6	2D7	2D8	2D9	2DA	2DB	2DC	2DD	2DE	2DF
2E0	2E1	2E2	2E3	2E4	2E5	2E6	2E7	2E8	2E9	2EA	2EB	2EC	2ED	2EE	2EF
2F0	2F1	2F2	2F3	2F4	2F5	2F6	2F7	2F8	2F9	2FA	2FB	2FC	2FD	2FE	2FF


 CJK characters
 Unallocated code points


As of Unicode 15.1

Plane 3: Tertiary Ideographic

(U+30000 to U+3FFFF)

300	301	302	303	304	305	306	307	308	309	30A	30B	30C	30D	30E	30F
310	311	312	313	314	315	316	317	318	319	31A	31B	31C	31D	31E	31F
320	321	322	323	324	325	326	327	328	329	32A	32B	32C	32D	32E	32F
330	331	332	333	334	335	336	337	338	339	33A	33B	33C	33D	33E	33F
340	341	342	343	344	345	346	347	348	349	34A	34B	34C	34D	34E	34F
350	351	352	353	354	355	356	357	358	359	35A	35B	35C	35D	35E	35F
360	361	362	363	364	365	366	367	368	369	36A	36B	36C	36D	36E	36F
370	371	372	373	374	375	376	377	378	379	37A	37B	37C	37D	37E	37F
380	381	382	383	384	385	386	387	388	389	38A	38B	38C	38D	38E	38F
390	391	392	393	394	395	396	397	398	399	39A	39B	39C	39D	39E	39F
3A0	3A1	3A2	3A3	3A4	3A5	3A6	3A7	3A8	3A9	3AA	3AB	3AC	3AD	3AE	3AF
3B0	3B1	3B2	3B3	3B4	3B5	3B6	3B7	3B8	3B9	3BA	3BB	3BC	3BD	3BE	3BF
3C0	3C1	3C2	3C3	3C4	3C5	3C6	3C7	3C8	3C9	3CA	3CB	3CC	3CD	3CE	3CF
3D0	3D1	3D2	3D3	3D4	3D5	3D6	3D7	3D8	3D9	3DA	3DB	3DC	3DD	3DE	3DF
3E0	3E1	3E2	3E3	3E4	3E5	3E6	3E7	3E8	3E9	3EA	3EB	3EC	3ED	3EE	3EF
3F0	3F1	3F2	3F3	3F4	3F5	3F6	3F7	3F8	3F9	3FA	3FB	3FC	3FD	3FE	3FF

 CJK characters

 Unallocated code points

As of Unicode 15.1

Planes 4 to 13: Unused
(U+40000 to U+DFFFF)

Plane 14: Supplementary Special-purpose

(U+E0000 to U+EFFFFF)

E00	E01	E02	E03	E04	E05	E06	E07	E08	E09	E0A	E0B	E0C	E0D	E0E	E0F
E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E1A	E1B	E1C	E1D	E1E	E1F
E20	E21	E22	E23	E24	E25	E26	E27	E28	E29	E2A	E2B	E2C	E2D	E2E	E2F
E30	E31	E32	E33	E34	E35	E36	E37	E38	E39	E3A	E3B	E3C	E3D	E3E	E3F
E40	E41	E42	E43	E44	E45	E46	E47	E48	E49	E4A	E4B	E4C	E4D	E4E	E4F
E50	E51	E52	E53	E54	E55	E56	E57	E58	E59	E5A	E5B	E5C	E5D	E5E	E5F
E60	E61	E62	E63	E64	E65	E66	E67	E68	E69	E6A	E6B	E6C	E6D	E6E	E6F
E70	E71	E72	E73	E74	E75	E76	E77	E78	E79	E7A	E7B	E7C	E7D	E7E	E7F
E80	E81	E82	E83	E84	E85	E86	E87	E88	E89	E8A	E8B	E8C	E8D	E8E	E8F
E90	E91	E92	E93	E94	E95	E96	E97	E98	E99	E9A	E9B	E9C	E9D	E9E	E9F
EA0	EA1	EA2	EA3	EA4	EA5	EA6	EA7	EA8	EA9	EAA	EAB	EAC	EAD	EAE	EAH
EB0	EB1	EB2	EB3	EB4	EB5	EB6	EB7	EB8	EB9	EBA	EBB	EBC	EBD	EBE	EBF
EC0	EC1	EC2	EC3	EC4	EC5	EC6	EC7	EC8	EC9	ECA	ECB	ECC	ECD	ECE	ECF
ED0	ED1	ED2	ED3	ED4	ED5	ED6	ED7	ED8	ED9	EDA	EDB	EDC	EDD	EDE	EDF
EE0	EE1	EE2	EE3	EE4	EE5	EE6	EE7	EE8	EE9	EEA	EEB	EEC	EED	EEE	EEF
EF0	EF1	EF2	EF3	EF4	EF5	EF6	EF7	EF8	EF9	EFA	EFB	EFC	EFD	EFE	EFF

Tags

Variation Selectors

Unallocated code points

As of Unicode 15.1

Planes 15/16: Supplementary Private Use

(U+F0000 to U+10FFFF)

To see the Unicode character
definitions, go to:

<http://unicode.org/charts/>

Unicode

Unicode Support

Привет мир

Γειά σου Κόσμε

你好，世界

Bonjour le monde

Hallo Welt

Helló Világ

שלום עולם

مرحبا بالعالم

こんにちは世界

Ahoj světe

Selam Dünya

Unicode Encodings

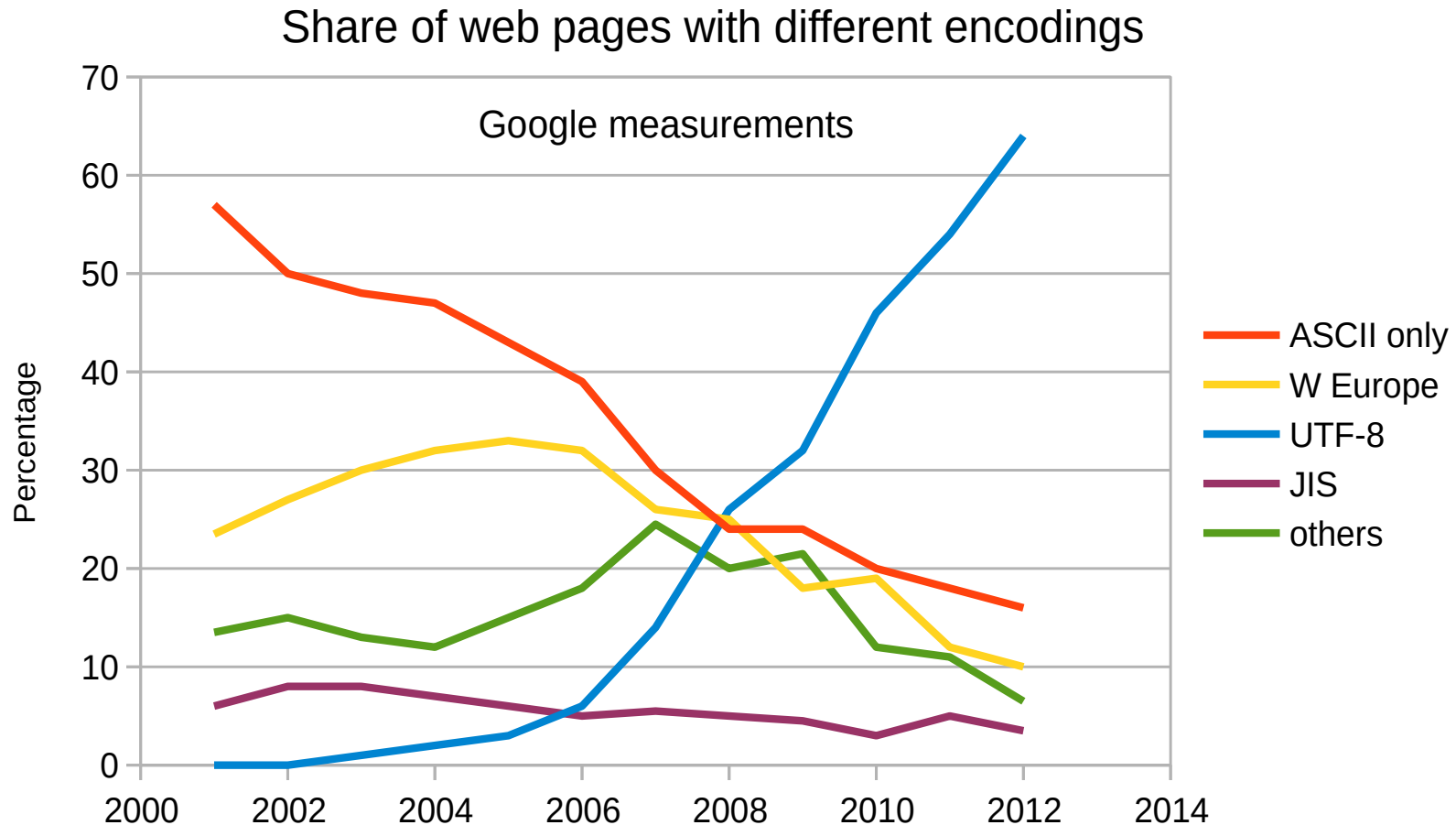
Unicode Encodings

- Instead of transmitting unicode code points, unicode defines different "encodings" that can be applied prior to transmission.
- Unicode Transformation Format (UTF)
 - UTF defines mappings of Unicode code points to a unique byte sequence.
 - The UTF mappings are reversible in that a given character is uniquely defined by both its unicode code point and its UTF encodings.
 - Most text in documents and webpages is encoded using some of the various UTF encodings

Unicode Encodings

- Unicode Transformation Format (UTF)
 - The conversions between all UTF encodings are algorithmically based, fast and lossless
 - Makes it easy to support data input or output in multiple formats, while using a particular UTF for internal storage or processing
 - Common encodings:
 - UTF-8
 - UTF-16
 - UTF-32.

Web Page Encoding Usage



(By Chris55 (Own work) [CC BY-SA 4.0 (<http://creativecommons.org/licenses/by-sa/4.0>)], via Wikimedia Commons)

As of 2024, over 98% of web pages use UTF-8 encoding. The second is ISO-8859-1 with about 1.3%.

UTF-8 Encoding

	Code Point	Byte 1	Byte 2	Byte 3	Byte 4
7 bits	0000-007F	0xxxxxxx			
11 bits	0080-07FF	110xxxxx	10xxxxxx		
16 bits	0800-D7FF	1110xxxx			
16 bits	E000-FFFF			10xxxxxx	
21 bits	10000-10FFFF	11110xxx			10xxxxxx

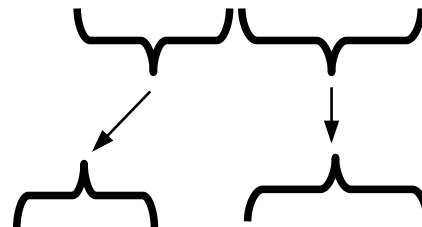
- Variable length (1 to 4 bytes). Sent as a sequence of bytes, i.e., no byte order issues.
- Note that the number of leading ones tells the number of bytes in the UTF-8 encoding (except for the 1-byte case).
- "continuation bytes" all start with 10.

UTF-8 Encoding

	Code Point	Byte 1	Byte 2	Byte 3	Byte 4
7 bits	0000-007F	0xxxxxxx			
11 bits	0080-07FF	110xxxxx	10xxxxxx		
16 bits	0800-D7FF	1110xxxx		10xxxxxx	
16 bits	E000-FFFF				
21 bits	10000-10FFFF	11110xxx			10xxxxxx

- Example: Ω : Greek Capital Letter Omega

Unicode: U+03A9 (0000 0011 1010 1001)



UTF-8: 110 01110 10 101001 (CEA9 hex)

- Note that Unicode code points for US-ASCII characters is their US-ASCII encoding and also their UTF-8 encoding.

UTF-16 Encoding

	Code Point	16 bit word 1	16 bit word 2
16 bits	0000 - D7FF	xxxxxxxxxxxxxxxxxxxx	
16 bits	E000 - FFFF		
20 bits	10000 - 10FFFF	110110xxxxxxxxxxxx	110111xxxxxxxxxxxx

- Variable length (1 or 2 16-bit words). Sent as a sequence of sixteen bit words, i.e., byte order issues, i.e., UTF-16BE and UTF-16LE.
- Encoding uses gaps in code point assignment:

110110xxxxxxxxxx = D800 - DBFF

110111xxxxxxxxxx = DC00 – DFFF

i.e., code points in the above ranges would start with 110110 or 110111 which interfere with the 6-bit UTF-16 preambles so those code points have been reserved, never to be assigned a Unicode character.

Plane 0: Basic Multilingual

(U+0000 to U+FFFF)

0000-00FF

0F00-0FFF

US-ASCII

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
- CJK characters
- Indonesian and Oceanic scripts
- American scripts
- Notational systems
- Symbols
- Private use
- UTF-16 surrogates
- No block

As of Unicode 15.1

F000-F0FF

FF00-FFFF

UTF-32 Encoding

	Code Point	32 bit word
21 bits	0000 - D7FF	0000000000000000xxxxxxxxxxxxxxxxxxxxxxxxxxxx
21 bits	E000 - 10FFFF	

- Fixed length (one 32 bit word). Need to deal with byte order, i.e., UTF-32BE and UTF-32LE.
- Very simple but lots of overhead.
- Its main use is in internal APIs where the data is single code points or glyphs, rather than strings of characters.

Python 3 Unicode Support

- Python 3 has full support for Unicode.
- `str` type is stored as Unicode characters. Unicode chars can be entered as literals using name or code point, e.g.,

```
d = "\N{GREEK CAPITAL LETTER OMEGA}"
```

```
e = "\u0394"
```

- `chr(i)` returns the string representing a character whose Unicode code point is the integer `i`. Valid range for the argument is from 0 through 1,114,111 (0x10FFFF hex).

See `chr_examples.py`.

- `ord(c)`: Given a string representing one Unicode character, return an integer representing the Unicode code point of that character.

`ord(c)` and `chr(i)` are inverses.

See `ord_examples.py`

Python3 Unicode Support

- Encode/decode functions convert between unicode and unicode encodings, e.g.,
- `str.encode()` : returns a bytes object representation of the Unicode string, encoded in the requested encoding.
- `bytes.decode()` : returns a Unicode string, decoded using the requested decoding.
- In our network transmissions of text we encode prior to transmission and decode after reception.

Python Unicode Support

- Python3 encode/decode functions convert between unicode and unicode encodings, e.g.,

```
str_1 = "hello"
```

```
str_1_utf8 = str_1.encode('utf-8')
```

- This creates a "bytes" object with UTF-8 encoding. To decode we can

```
str_1_decode = str_1.decode('utf-8')
```

- See `encodeddecode_examples.py`.
- See `print_encodings.py`