

ENCODING TEXT

Periodic Table of Typefaces

Popular, Influential, & Notorious

Family and/or Class Rank

Symbol Typeface

Designers Year Designed

Ranking determined by voting and combining lists and opinions from the following sites:
The 100 Best Fonts Of All Time - <http://www.100bestfontsofallof.com/>
No include too few personal favorites from designers: dan.mahoney@strobil.com, Roger Black (robertblack.com),
Bernard Schmidt (bernard.schmidt@strobil.com), Stephen Cohen (stephen.cohen@strobil.com),
Paul Hoverson (paul.hoverson@strobil.com), and Claudia Quinonez (claudiaquinonez@strobil.com)
Paul Shaw's Top 100 Typefaces survey - <http://www.100typefaces.com/>
Top 100 Typefaces By Professional Designers - <http://www.typefaces.com/2006/10/05/21-most-used-fonts-by-professional-designers/>
Top 7 Fonts Used By Professionals in Graphic Design - <http://www.typefaces.com/2006/10/05/21-most-used-fonts-by-professional-designers/>
Top 7 Fonts Used By Professionals in Graphic Design - <http://www.typefaces.com/2006/10/05/21-most-used-fonts-by-professional-designers/>
30 Fonts That ALL Designers Must Know & Should Own - <http://www.typefaces.com/2006/10/05/21-most-used-fonts-by-professional-designers/>
Typefaces no one gets tired of using - <http://www.typefaces.com/2006/10/05/21-most-used-fonts-by-professional-designers/>
To include all serif and sans-serif options listed in the comments section

<http://www.designishistory.com/1450/type-classification/>

TEXT AND BINARY DATA

- Up till this point in the course, we've been moving up to bigger and bigger code structures
 - *From ints and floats, to lists and strings, to modules and classes*
- Today, we're heading in the other direction.
 - *We want to know how a computer can store all the types of data we use, when at some level it's all binary, just zeros and ones.*
- When we're just working in Python, we usually don't have to worry about this level of representation ... but ...

TEXT & BINARY DATA

- But as a data scientist, you'll often need to collect data from other sources
 - *Websites or servers with public apis*
 - *Files created by other people*
 - *Text that people type into your application*
- We can classify these sources as text data or binary data. Let's cover text data first.

TEXT

- How can we get a computer to process text characters?
- We know that at some level, text has to be binary, so need to map each character to a binary sequence
 - You can view every binary sequence as a number in base 2
 - 0000 is 0
 - 0001 is 1
 - 0010 is 2
 - And so on...
- The first major system for encoding text was called ASCII
 - *“American Standard Code for Information Interchange”*
 - *It was published in 1963*
- Every ASCII character takes 7 bits [tho there’s also 8-bit ASCII]
 - *This means that you can only get $2^7 = 128$ different characters*
 - *But there is also 8-bit ASCII [<https://www.sciencebuddies.org/science-fair-projects/references/table-of-8-bit-ascii-character-codes>]*

ASCII

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	
001	☺	SOH	033	!	065	A	097	a
002	☻	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	\$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	■	BS	040	(072	H	104	h
009	(tab)	HT	041)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	♪	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	▲	DLE	048	0	080	P	112	p
017	▼	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019		DC3	051	3	083	S	115	s
020	π	DC4	052	4	084	T	116	t
021	§	NAK	053	5	085	U	117	u
022	▬	SYN	054	6	086	V	118	v
023	↕	ETB	055	7	087	W	119	w
024	↕	CAN	056	8	088	X	120	x
025	↕	EM	057	9	089	Y	121	y
026	→	SUB	058	:	090	Z	122	z
027	←	ESC	059	;	091	[123	{
028	(cursor right)	FS	060	<	092	\	124	
029	(cursor left)	GS	061	=	093]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	☐

BEYOND ASCII

- Of course, ASCII is far too limiting.
 - *It doesn't even cover all the characters we use to write Latin-alphabet languages, such as English, let alone other major languages.*
- There have been a lot of attempts to create larger character encodings.
 - *Windows code, Latin-1, etc*
- These may be incompatible and cause errors if you don't know exactly how to process the bits you're getting.

MORE ON ENCODINGS...

Why care about this?

- *Internationalization* (“i18”) - mapping between data streams & localization
- *Legacy Systems* - converting data from older systems
- *Standards* - ANSI and ISO and others
- *Typography* - OpenFont, TrueType, others, support “GID” (graphic ID that are not part of the ASCII code page planes) for historical/graphic graphemes

SOME SITES TO VISIT...

Character encodings for beginners

<https://www.w3.org/International/questions/qa-what-is-encoding>

What Every Programmer Absolutely, Positively Needs To Know About Encodings And Character Sets To Work With Text

<http://kunststube.net/encoding/>

Unicode Basics

http://ergoemacs.org/emacs/unicode_basics.html

WIN-1252

https://www.w3schools.com/charsets/ref_html_ansi.asp

ISO-8859-x

<https://www.i18nqa.com/debug/table-iso8859-1-vs-windows-1252.html>

UNICODE

• <http://unicode.org>

		→ 2160 I roman numeral one
004A	J	LATIN CAPITAL LETTER J
004B	K	LATIN CAPITAL LETTER K
		→ 212A K kelvin sign
004C	L	LATIN CAPITAL LETTER L
		→ 2112 L script capital l
004D	M	LATIN CAPITAL LETTER M
		→ 2133 M script capital m
004E	N	LATIN CAPITAL LETTER N
		→ 2115 N double-struck capital n
004F	O	LATIN CAPITAL LETTER O
0050	P	LATIN CAPITAL LETTER P
		→ 2119 P double-struck capital p
0051	Q	LATIN CAPITAL LETTER Q
		→ 211A Q double-struck capital q
0052	R	LATIN CAPITAL LETTER R
		→ 211B R script capital r
		→ 211C R black-letter capital r
		→ 211D R double-struck capital r
0053	S	LATIN CAPITAL LETTER S
0054	T	LATIN CAPITAL LETTER T
0055	U	LATIN CAPITAL LETTER U
0056	V	LATIN CAPITAL LETTER V
		→ 2164 V roman numeral five
0057	W	LATIN CAPITAL LETTER W
0058	X	LATIN CAPITAL LETTER X
0059	Y	LATIN CAPITAL LETTER Y
005A	Z	LATIN CAPITAL LETTER Z
		→ 2124 Z double-struck capital z
		→ 2128 Z black-letter capital z

ASCII punctuation and symbols

005B	[LEFT SQUARE BRACKET
		= opening square bracket (1.0)
		• other bracket characters: 27E6 [−27EB] ,
		2983 [−2998] , 3008 [−301B]

Lowercase Latin alphabet








































0061	a	LATIN SMALL LETTER A
0062	b	LATIN SMALL LETTER B
0063	c	LATIN SMALL LETTER C
0064	d	LATIN SMALL LETTER D
0065	e	LATIN SMALL LETTER E
		→ 212E E estimated symbol
		→ 212F e script small e
0066	f	LATIN SMALL LETTER F
0067	g	LATIN SMALL LETTER G
		→ 0261 g latin small letter script g
		→ 210A g script small g
0068	h	LATIN SMALL LETTER H
		→ 04BB h cyrillic small letter shha
		→ 210E h planck constant
0069	i	LATIN SMALL LETTER I
		• Turkish and Azerbaijani use 0130 İ for uppercase
		→ 0131 i latin small letter dotless i
		→ 1D6A4 i mathematical italic small dotless i
006A	j	LATIN SMALL LETTER J
		→ 0237 j latin small letter dotless j
		→ 1D6A5 j mathematical italic small dotless j
006B	k	LATIN SMALL LETTER K
006C	l	LATIN SMALL LETTER L
		→ 2113 l script small l
		→ 1D4C1 l mathematical script small l
006D	m	LATIN SMALL LETTER M
006E	n	LATIN SMALL LETTER N
		→ 207F n superscript latin small letter n
006F	o	LATIN SMALL LETTER O
		→ 2134 o script small o
0070	p	LATIN SMALL LETTER P
0071	q	LATIN SMALL LETTER Q
0072	r	LATIN SMALL LETTER R
0073	s	LATIN SMALL LETTER S
0074	t	LATIN SMALL LETTER T

UNICODE

/uxxxx

/Uxxxx

/N}

	1F60	1F61	1F62	1F63	1F64
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					

	059	05A	05B	05C	05D	05E	05F
0							
1							
2							
3							
4							
5							
6							
7							

3400 1.4 GKX-0018.01	北	北	北	3414 5.9 K3-212F	𠂔	3428 6.7 G5-3044	𠂔	3429 7.6 GKZ-10021.01	𠂔
3401 1.5 G5-3024	𠂔	𠂔	𠂔	3415 5.5 K3-2130	𠂔	342A 8.4 JA-2128	𠂔	342B 8.4 GKZ-10080.02	𠂔
3402 1.5 JA-2123	𠂔	𠂔	𠂔	3416 5.8 G3-3032	𠂔	342C 8.5 G5-3040	𠂔	342D 8.6 GKX-0080.01	𠂔
3403 2.2 K3-2122	𠂔	𠂔	𠂔	3417 5.8 K3-2131	𠂔	342E 8.11 GK-2-10191.08	𠂔	342F 8.15 GK-2-10191.08	𠂔
3404 2.2 GKX-0079.02	𠂔	𠂔	𠂔	3418 5.8 K3-2132	𠂔				
3405 4.1 GKX-0081.18	𠂔	𠂔	𠂔	3419 5.7 K3-2133	𠂔				
3406 4.5 G5-3076	𠂔	𠂔	𠂔	341A 5.7 K3-2134	𠂔				
3407 5.2 K3-2123	𠂔	𠂔	𠂔	341B 5.7 K3-2136	𠂔				
3408 5.2 K3-2134	𠂔	𠂔	𠂔	341C 5.8 G3-3024	𠂔				
3409 5.2 K3-2135	𠂔	𠂔	𠂔	341D 5.8 K3-2137	𠂔				
340A 5.3 K3-2136	𠂔	𠂔	𠂔	341E 5.8 K3-2138	𠂔				
340B 5.3 K3-2138	𠂔	𠂔	𠂔	341F 5.8 K3-2139	𠂔				

	090	091	092	093	094	095	096	097
0	ॐ	ऐ	ठ	र	ी	ॐ	ॐ	ॐ
1	ॐ	ऑ	ड	र	ु	ॐ	ॐ	ॐ
2	ॐ	ओ	ढ	ल	ॐ	ॐ	ॐ	ॐ
3	ॐ	ओ	ण	ळ	ॐ	ॐ	ॐ	ॐ
4	ॐ	औ	त	ळ	ॐ	ॐ	ॐ	ॐ
5	ॐ	अ	क	थ	व	ॐ	ॐ	ॐ
6	ॐ	आ	ख	द	श	ॐ	ॐ	ॐ
7	ॐ	इ	ग	ध	ष	ॐ	ॐ	ॐ
8	ॐ	ई	घ	न	स	ॐ	ॐ	ॐ

	000	001	002	003	004	005	006	007
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7				7	G	W	g	w
8				8	H	X	h	x
9				9	I	Y	i	y
10				:	J	Z	j	z
11				;	K	[k	{
12				<	L	\	l	
13				=	M]	m	}
14				>	N	^	n	~
15				?	O	_	o	DEL

UNICODE

- The closest thing we have to a standard is a system called Unicode.
 - *This defines a set of more than 120,000 different characters.*
 - *Modern languages, ancient languages, mathematical symbols.*
- Unicode is actually an abstraction, essentially just assigning an order to characters.
- There are then character encodings that take each unicode character and map it to a sequence of bytes.
 - The most popular one is UTF-8 [though CJK others in UTF-16]
 - The 8 stands for 8 bits, or one byte. Every character in UTF-8 gets 1 to 4 bytes.
 - The original ASCII characters are encoded with 1 byte using their original ASCII encoding. So UTF-8 is very compact for these characters.

UNICODE

- UTF-8 is becoming a popular standard.
 - 85.1% of all Web pages in September 2015
 - It is the required encoding for many file types (.json, .xml, etc.)
- This is what you want to use if you have a choice.
- But there are times when you have to interact with other encodings in Python.
- We'll explore how to encode and decode text in the next segments.