











































Unicode solutions in Python 2 and 3

	00A	00B	00C	00D	00E	00F
000	À	Á	Â	Ã	Ä	Å
001	Æ	Ç	Ð	Ñ	Ò	Ó
002	Ô	Õ	Ö	×	Ø	Ù
003	Ú	Û	Ü	Ý	Þ	ß
004	à	á	â	ã	ä	å
005	æ	ç	ð	ñ	ò	ó
006	ô	õ	ö	÷	ø	ù
007	ú	û	ü	ý	þ	ÿ
008	À	Á	Â	Ã	Ä	Å
009	Æ	Ç	Ð	Ñ	Ò	Ó
00A	Ô	Õ	Ö	×	Ø	Ù
00B	Ú	Û	Ü	Ý	Þ	ß
00C	à	á	â	ã	ä	å
00D	æ	ç	ð	ñ	ò	ó
00E	ô	õ	ö	÷	ø	ù
00F	ú	û	ü	ý	þ	ÿ

1F61	1F62	1F63	1F64
			
1F610	1F620	1F630	1F640
			
1F611	1F621	1F631	1F641
			
1F612	1F622	1F632	1F642
			
1F613	1F623	1F633	
			
1F614	1F624	1F634	
			
1F615	1F625	1F635	1F645
			
1F616	1F626	1F636	1F646
			
1F617	1F627	1F637	1F647
			
1F618	1F628	1F638	1F648
			
1F619	1F629	1F639	1F649
			
1F61A	1F62A	1F63A	1F64A
			
1F61B	1F62B	1F63B	1F64B
			
1F61C	1F62C	1F63C	1F64C
			
1F61D	1F62D	1F63D	1F64D
			

053	054	055	056	057	058
	Ƶ 0540	ƶ 0550		Ʒ 0570	Ƹ 0580
Ƶ 0531	ƶ 0541	Ʒ 0551	Ƹ 0561	ƹ 0571	ƺ 0581
ƹ 0532	ƺ 0542	ƻ 0552	Ƽ 0562	ƽ 0572	ƾ 0582
ƽ 0533	ƿ 0543	ƺ 0553	ƻ 0563	Ƽ 0573	ƾ 0583
ƽ 0534	ƿ 0544	ƺ 0554	ƻ 0564	Ƽ 0574	ƾ 0584
ƽ 0535	ƿ 0545	ƺ 0555	ƻ 0565	Ƽ 0575	ƾ 0585
ƽ 0536	ƿ 0546	ƺ 0556	ƻ 0566	Ƽ 0576	ƾ 0586
ƽ 0537	ƿ 0547		ƻ 0567	Ƽ 0577	ƾ 0587
ƽ 0538	ƿ 0548		ƻ 0568	Ƽ 0578	
ƽ 0539	ƿ 0549	ƺ 0559	ƻ 0569	Ƽ 0579	ƾ 0589
ƽ 053A	ƿ 054A	ƺ 055A	ƻ 056A	Ƽ 057A	ƾ 058A
ƽ 053B	ƿ 054B	ƺ 055B	ƻ 056B	Ƽ 057B	
ƽ 053C	ƿ 054C	ƺ 055C	ƻ 056C	Ƽ 057C	
ƽ 053D	ƿ 054D	ƺ 055D	ƻ 056D	Ƽ 057D	
ƽ 053E	ƿ 054E	ƺ 055E	ƻ 056E	Ƽ 057E	

1314	1315	1316	1317	1318	1319	131A	131B

	0D1	0D2	0D3	0D4	0D5	0D6
	എ	റ	ര	ീ		ഋ
	0D19	0D20	0D30	0D40		0D60
		ഡ	റ	ു		ൠ
		0D21	0D31	0D41		0D61
	ഒ	ഘ	ല	ു		ു
	0D12	0D22	0D32	0D42		0D62
	ഓ	ണ	ള	ു		ു
	0D13	0D23	0D33	0D43		0D63
	ഔ	ത	ഴ	ു		
	0D14	0D24	0D34	0D44		
	ക	ഥ	വ			
	0D15	0D25	0D35			
	ഖ	ദ	ശ	െ		റ
	0D16	0D26	0D36	0D46		0D66
	ഗ	ധ	ഷ	േ	ൌ	൧
	0D17	0D27	0D37	0D47	0D57	0D67
	ഘ	ന	സ	ൈ		൨
	0D18	0D28	0D38	0D48		0D68
	ങ		ഹ			൩
	0D19	0D29	0D39			0D69
	ച	പ		ൊ		ർ
	0D1A	0D2A	0D3A	0D4A		0D6A
	ഛ	ഫ		ോ		ർ
	0D1B	0D2B		0D4B		0D6B
	ജ	ബ		ൗ		൩
	0D1C	0D2C		0D4C		0D6C
	ട	ഭ	്	്		൪
	0D1D	0D2D	0D3D	0D4D		0D6D
	ണ	മ	ാ			൮
	0D1E	0D2E	0D3E	0D4E		0D6E

HEX	C	J	K	V
50D0 人 9 12	僖 G5-3271 H4-9003	僖 T3-4588 J1-3238	僖 K2-234C J0-3623	僖 V1-4C27
50D1 人 9 12	僑 G1-4768 H81-8964	僑 T1-677A J0-3623	僑 K0-4089	僑
50D2 人 9 12	僭 G3-3238	僭 T4-422D		僭
50D3 人 9 12	僭 G3-3165 H82-EDF9	僭 T3-4021 J1-3239	僭 K2-234D	僭 V2-4E5D
50D4 人 9 12	僭 G3-3237 H82-EDF1	僭 T3-4877 J1-323A	僭 K2-234E	僭
50D5 人 9 12	僕 G1-484D H81-8982	僕 T4-4778 J0-494D	僕 K0-5C52	僕 V1-4C28
50D6 人 9 12	僖 G0-5562 H81-894F	僖 T4-4775 J0-6125	僖 K0-705A	僖
50D7 人 9 12	僭 G3-3137 H82-EDF2	僭 T3-4878	僭 K2-234F	僭
50D8 人 9 12	僭 GE-2238	僭 T3-458E J1-323B	僭 K2-235D	僭 V2-8A42
50D9 人 9 12	僭 G5-3261 H4-9046	僭 T3-459C J4-2175	僭 K2-2351	僭
50DA 人 9 12	僭 G0-4145 H81-8981	僭 T1-6777 J0-4E3D	僭 K0-5E7E	僭 V1-4C29
50DB 人 9 12	僭 G3-3232 H82-EDF 5	僭 T2-487D	僭 K1-688A	僭
50DC 人 9 12	僭 G3-3238	僭 T3-459D J1-323C	僭 K2-2352	僭
50DD 人 9 12	僭 G3-3209 H82-EDF7	僭 T2-487D J1-323D	僭 K2-2353	僭 V2-8A43
50DE 人 9 12	僭 G1-4E31 GE-2239	僭 T4-457E J0-5128	僭 K0-6A9A	僭
50DF 人 9 12	僭 G1-4E31 GE-2239	僭 T4-457E J1-323E	僭 K2-235A	僭
50E0 人 9 12	僭 GE-223A H82-EDFE	僭 T4-458D	僭 K2-2355	僭
50E1 人 9 12	僭 G5-3269 H-FC72	僭 T3-457D	僭 J4-217C	僭 K2-2356
50E2 人 9 12	僭 G5-326F	僭 T3-4575	僭 J1-323F	僭 K1-6F32
50E3 人 9 12	僭 G5-326F	僭 T3-4575	僭 J1-323F	僭 K1-6F32

Unicode solutions in Python 2 and 3

	00A	00B	00C	00D	00E	00F
	À	Á	Â	Ã	Ä	Å
	Æ	Ç	Ð	Ñ	Ò	Ó
	Ô	Õ	Ö	×	Ý	ÿ
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	Ä	Å	Æ	Ç	Ð	Ñ
	Ò	Ó	Ô	Õ	Ö	×
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	À	Á	Â	Ã	Ä	Å
	Æ	Ç	Ð	Ñ	Ò	Ó
	Ô	Õ	Ö	×	Ý	ÿ
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	Ä	Å	Æ	Ç	Ð	Ñ
	Ò	Ó	Ô	Õ	Ö	×
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	À	Á	Â	Ã	Ä	Å
	Æ	Ç	Ð	Ñ	Ò	Ó
	Ô	Õ	Ö	×	Ý	ÿ
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	Ä	Å	Æ	Ç	Ð	Ñ
	Ò	Ó	Ô	Õ	Ö	×
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	À	Á	Â	Ã	Ä	Å
	Æ	Ç	Ð	Ñ	Ò	Ó
	Ô	Õ	Ö	×	Ý	ÿ
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	Ä	Å	Æ	Ç	Ð	Ñ
	Ò	Ó	Ô	Õ	Ö	×
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	À	Á	Â	Ã	Ä	Å
	Æ	Ç	Ð	Ñ	Ò	Ó
	Ô	Õ	Ö	×	Ý	ÿ
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	Ä	Å	Æ	Ç	Ð	Ñ
	Ò	Ó	Ô	Õ	Ö	×
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	À	Á	Â	Ã	Ä	Å
	Æ	Ç	Ð	Ñ	Ò	Ó
	Ô	Õ	Ö	×	Ý	ÿ
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	Ä	Å	Æ	Ç	Ð	Ñ
	Ò	Ó	Ô	Õ	Ö	×
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	À	Á	Â	Ã	Ä	Å
	Æ	Ç	Ð	Ñ	Ò	Ó
	Ô	Õ	Ö	×	Ý	ÿ
	à	á	â	ã	ä	å
	æ	ç	ð	ñ	ò	ó
	ô	õ	ö	÷	ý	ÿ
	Ä	Å	Æ	Ç	Ð	Ñ
	Ò	Ó	Ô	Õ	Ö	×

	1F61	1F62	1F63	1F64
1F610				
1F611				
1F612				
1F613				
1F614				
1F615				
1F616				
1F617				
1F618				
1F619				
1F61A				
1F61B				

053	054	055	056	057	058
Armenian					
0531	0541	0551	0561	0571	0581
Բ	Ղ	Ի	Բ	Ղ	Լ
0532	0542	0552	0562	0572	0582
Գ	Ճ	Փ	Դ	Ճ	Վ
0533	0543	0553	0563	0573	0583
Դ	Մ	Ք	Դ	Մ	Ք
0534	0544	0554	0564	0574	0584
Ե	Յ	Օ	Ե	Յ	Օ
0535	0545	0555	0565	0575	0585
Զ	Ն	Ծ	Զ	Ն	Փ
0536	0546	0556	0566	0576	0586
Է	Շ		Է	Շ	Լ
0537	0547		0567	0577	0587
Ը	Ո		Ը	Ո	
0538	0548		0568	0578	
Թ	Չ	՝	Թ	Չ	:
0539	0549	0559	0569	0579	0589
Ժ	Պ	՛	Ժ	Պ	՛
053A	054A	055A	056A	057A	058A
Ի	Ջ	՛	Ի	Ջ	
053B	054B	055B	056B	057B	
Լ	Ի	~	Լ	Ի	
053C	054C	055C	056C	057C	
Ս	Ս	՝	Խ	Ս	Ս
053D	054D	055D	056D	057D	058D
Տ	Վ	՝	Ծ	Վ	Ս
053E	054E	055E	056E	057E	058E

314	1315	1316	1317	1318	1319	131A	131B
13140	13150	13160	13170	13180	13190	131A0	131B0
13141	13151	13161	13171	13181	13191	131A1	131B1
13142	13152	13162	13172	13182	13192	131A2	131B2
13143	13153	13163	13173	13183	13193	131A3	131B3
13144	13154	13156	13174	13184	13194	131A4	131B4
13145	13155	13165	13175	13185	13195	131A5	131B5
13146	13156	13166	13176	13186	13196	131A6	131B6
13147	13157	13167	13177	13187	13197	131A7	131B7
13148	13158	13168	13178	13188	13198	131A8	131B8
13149	13159	13169	13179	13189	13199	131A9	131B9
1314A	1315A	1316A	1317A	1318A	1319A	131AA	131BA
1314B	1315B	1316B	1317B	1318B	1319B	131AB	131BB
1314C	1315C	1316C	1317C	1318C	1319C	131AC	131BC
1314D	1315D	1316D	1317D	1318D	1319D	131AD	131BD

Egyptian Hieroglyphs

	0D0	0D1	0D2	0D3	0D4	0D5	0D6
131B0	ഒ	ഐ	ഓ	ഔ	ഐ	ഐ	ഐ
131B1		ഡ	റ	ഠ	ഡ	ഢ	ണ
131B2	ഔ	ത	ഴ	ഠ	ഡ	ഢ	ണ
131B3	ക	ഥ	വ	ഠ	ഡ	ഢ	ണ
131B4	ഖ	ഭ	ശ	ഠ	ഡ	ഢ	ണ
131B5	ഗ	ധ	ഷ	ഠ	ഡ	ഢ	ണ
131B6	ഘ	ന	സ	ഠ	ഡ	ഢ	ണ
131B7	ങ		ഹ	ഠ	ഡ	ഢ	ണ
131B8	ച	പ		ഠ	ഡ	ഢ	ണ
131B9	ഛ	ഫ		ഠ	ഡ	ഢ	ണ
131BA	ജ	ബ		ഠ	ഡ	ഢ	ണ
131BB	ത	ഭ	ഠ	ഡ	ഢ	ണ	ണ
131BC	ന	മ	ഠ	ഡ	ഢ	ണ	ണ

HEX		C	J	K	V		
50D0 人 9 12	僖	僖 G5-3271	僖 H-90C3	僖 T3-458B	僖 J1-323B	僖 K2-234C	
50D1 人 9 12	僑	僑 G1-474B	僑 H81-938A	僑 T1-677A	僑 J6-3623	僑 K2-463B	僑 V1-4C27
50D2 人 9 12	僭	僭 G3-323B		僭 T4-422D			
50D3 人 9 12	僨	僨 G3-3165	僨 H82-63F9	僨 T2-4821	僨 J1-323B	僨 K2-234D	僨 V2-6E5D
50D4 人 9 12	僭	僭 G3-3237	僭 H82-63F1	僭 T2-4877	僭 J1-323A	僭 K2-234E	
50D5 人 9 12	僨	僨 G14-840	僨 H81-9382	僨 T1-677B	僨 J0-4B4D	僨 K0-5C52	僨 V1-4C28
50D6 人 9 12	僖	僖 G0-5952	僖 H81-93AF	僖 T1-6775	僖 J0-5125	僖 K0-7D8A	
50D7 人 9 12	僖	僖 G3-3157	僖 H82-63F2	僖 T2-487B		僖 K2-234F	
50D8 人 9 12	僨	僨 G8-323B		僨 T3-458E	僨 J1-323B	僨 K2-235D	僨 V2-8A42
50D9 人 9 12	僨	僨 G5-3261	僨 H403A6	僨 T3-458C	僨 J4-217B	僨 K2-2351	
50DA 人 9 12	僨	僨 G0-414D	僨 H81-9381	僨 T1-6777	僨 J0-4E3D	僨 K0-567B	僨 V1-4C29
50DB 人 9 12	僨	僨 G3-3232	僨 H82-63F5	僨 T2-487B		僨 K1-689A	
50DC 人 9 12	僨	僨 G0-323B		僨 T3-458D	僨 J1-323C	僨 K2-2352	
50DD 人 9 12	僨	僨 G3-323B	僨 H82-63F7	僨 T2-487D	僨 J1-323D	僨 K2-2353	僨 V2-8A43

CJK

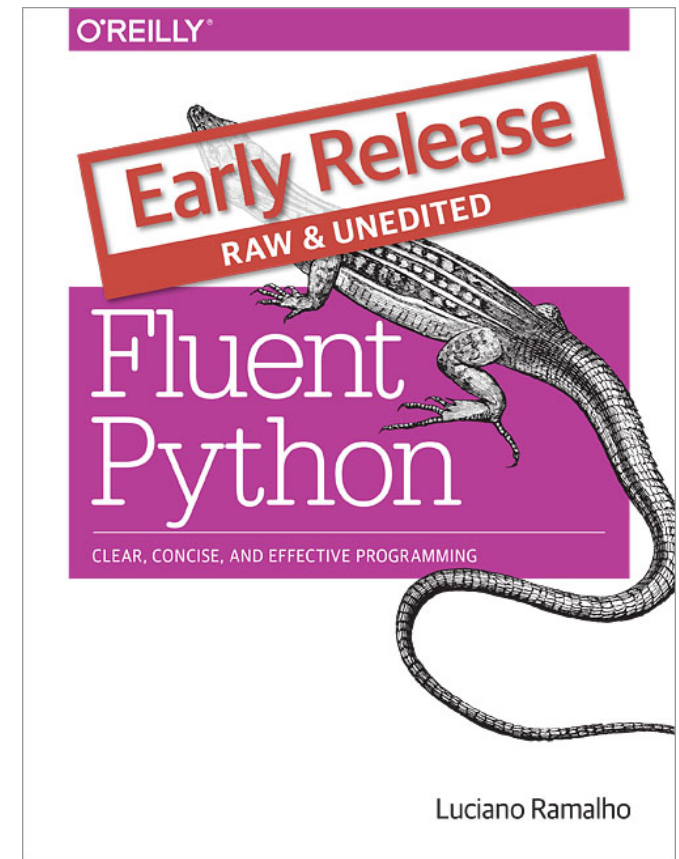
Unified

Ideographs

About me: Luciano Ramalho

- Programming in Python since 1998
- Focus on content management (i.e. text wrangling)
- Teaching Python since 1999
- Speaker at PyCon US, OSCON, FISL, PythonBrasil, RuPy, QCon...
- Author of **Fluent Python**
- Twitter: @ramalhoorg
- Native language: Português
 - “ação”

4 non-ASCII
characters here



Material de apoio

- Repositório de exemplos:
 - https://github.com/pythonprobr/unicode_pybr
- **Python Fluente**
 - <http://www.novatec.com.br/livros/pythonfluente/>
 - Conteúdo relevante e exemplos:
 - Capítulo 4: *Texto versus Bytes*
 - <http://bit.ly/pyflu04>
 - Capítulo 18: *Concorrência com asyncio*
 - os exemplos *charfinder*



The single-byte codepage ballet

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

KOI8-R

í „ ... † ‡ €

“ ” • — —

J α Γ ∫ §

i γ μ ¶

Video: <https://www.youtube.com/watch?v=J4qioAacrYo>

Source code: <http://bit.ly/10qt0MZ>

Why Unicode

- Too many incompatible byte encodings
- Separate concepts:
 - character identity: one **code point** for each abstract character
 - U+0041 → LATIN CAPITAL LETTER A
 - U+096C → DEVANAGARI DIGIT SIX
 - binary representation: multiple **encodings**

• U+0041 → 0x41

• U+096C → 0xE0 0xA5 0xAC

0x41 0x00

0x6C 0x09

UTF-8

UTF-16LE

A sample of encodings

char.	code point	ascii	latin1	cp1252	cp437	gb2312	utf-8	utf-16le
A	U+0041	41	41	41	41	41	41	41 00
¿	U+00BF	*	BF	BF	A8	*	C2 BF	BF 00
Ã	U+00C3	*	C3	C3	*	*	C3 83	C3 00
á	U+00E1	*	E1	E1	A0	A8 A2	C3 A1	E1 00
Ω	U+03A9	*	*	*	EA	A6 B8	CE A9	A9 03
ꣳ	U+06BF	*	*	*	*	*	DA BF	BF 06
“	U+201C	*	*	93	*	A1 B0	E2 80 9C	1C 20
€	U+20AC	*	*	80	*	*	E2 82 AC	AC 20
Г	U+250C	*	*	*	DA	A9 B0	E2 94 8C	0C 25
气	U+6C14	*	*	*	*	C6 F8	E6 B0 94	14 6C
氣	U+6C23	*	*	*	*	*	E6 B0 A3	23 6C
♫	U+1D11E	*	*	*	*	*	F0 9D 84 9E	34 D8 1E DD

Figure 4-1 of Fluent Python

Data types for text or bytes



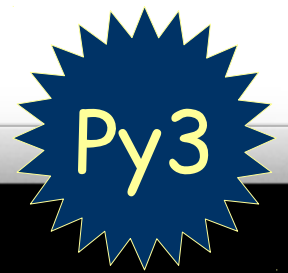
Python 2.7



Python 3.4

	Python 2.7	Python 3.4
Human text	unicode u'café', u'caf\xe9'	str 'café', u'café'
Bytes (immutable)	str 'café', 'caf\xe9', b'café'...	bytes b'caf\xc3\xa9'
Bytes (mutable)	bytearray bytearray(b'caf\xc3\xa9')	bytearray bytearray(b'caf\xc3\xa9')

str v. bytes in Py3



```
>>> s = 'café'
>>> len(s)
4
>>> s
'café'
>>> b = s.encode('utf-8')
>>> len(s)
4
>>> b
b'caf\xc3\xa9'
>>> list(b)
[99, 97, 102, 195, 169]
>>> list(s)
['c', 'a', 'f', 'é']
>>> b2 = s.encode('cp850')
>>> b2
b'caf\x82'
>>> len(b2)
4
>>> list(b2)
[99, 97, 102, 130]
>>> b.decode('utf-8')
'café'
```

bytes in Py3



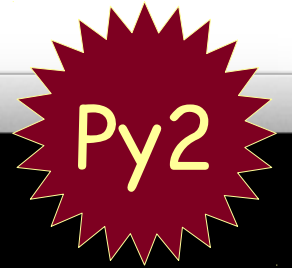
```
>>> b = bytes('café', encoding='utf-8')
>>> b
b'caf\xc3\xa9'
>>> list(b)
[99, 97, 102, 195, 169]
>>> b[0]
99
>>> b[1:]
b'af\xc3\xa9'
>>> b_arr = bytearray(b)
>>> b_arr
bytearray(b'caf\xc3\xa9')
>>> b_arr[0] = b'd'
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: an integer is required
>>> b_arr[0] = b'd'[0]
>>> b_arr
bytearray(b'daf\xc3\xa9')
>>> print(b)
b'caf\xc3\xa9'
```

bytearray in Py2 & 3

Py2&3

```
>>> b_arr
bytearray(b'caf\xc3\xa9')
>>> b_arr[0] = b'd'
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: an integer is required
>>> b_arr[0] = b'd'[0]
>>> b_arr
bytearray(b'daf\xc3\xa9')
```

unicode v. str in Py2



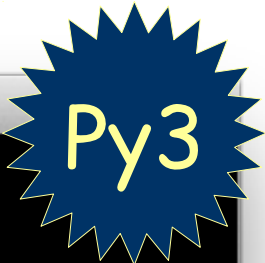
```
>>> s = 'café'
>>> len(s)
5
>>> s
'caf\xc3\xa9'
>>> u = s.decode('utf-8')
>>> u
u'caf\xe9'
>>> print u
café
>>> len(u)
4
>>> list(s)
['c', 'a', 'f', '\xc3', '\xa9']
>>> list(u)
[u'c', u'a', u'f', u'\xe9']
>>> type(s)
<type 'str'>
>>> type(u)
<type 'unicode'>
```

.encode() vs .decode()

- “Humans use text. Computers speak bytes.”
 - Esther Nam and Travis Fischer in *Character encoding and Unicode in Python (Pycon US 2014)*
- Use .encode() to convert **human** text to **bytes**
- Use .decode() to convert **bytes** to **human** text


2.7 gotcha:
the methods
.encode() and .decode()
exist in **str** and **unicode**

str.encode(...) in Py3



```
>>> s = 'El Niño'
>>> for codec in ['latin_1', 'utf_8', 'utf_16']:
...     print(codec, s.encode(codec), sep='\t')
...
...
latin_1 b'El Ni\xff1o'
utf_8   b'El Ni\xc3\xb1o'
utf_16  b'\xff\xfeE\x00l\x00 \x00N\x00i\x00\x00\x00\x00\x00\x00\x00\x00'
```

unicode.encode(...) in Py2

A red starburst badge with the text "Py2" in yellow.

```
>>> u = u'El Niño'
>>> for codec in ['latin_1', 'utf_8', 'utf_16']:
...     print codec + '\t' + u.encode(codec)
...
...
latin_1 El Niño
utf_8   El Niño
utf_16  \x00\x00El Niño
```

Best practice

The Unicode sandwich



bytes → str

100% str

str → bytes

Decode bytes on input,

process text only,

encode text on output.

How to implement the sandwich (1)

- Always specify encoding when reading/writing text files
 - that way you get text, and not bytes
 - in Python 2.7, use `io.open()`

2.7 gotcha:
no way to specify
encoding in built-in `open(...)`.
Must use `io.open(...)`.

Coping with Unicode Errors

- **SyntaxError**

- A .py file is loaded with contents in an unexpected encoding

- **UnicodeDecodeError**

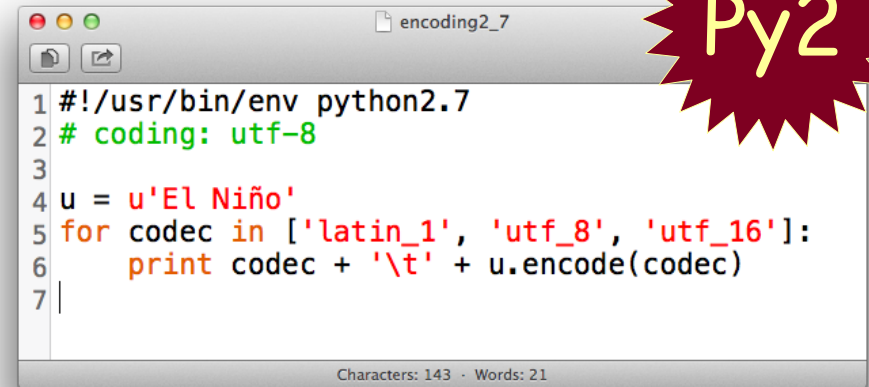
- A binary sequence is contains bytes that are not valid in the expected encoding

- **UnicodeEncodeError**

- A Unicode string contains codepoints that have no representation in the desired encoding

Coping with SyntaxError

- A .py file is loaded with contents in an unexpected source code encoding
 - The source file encoding is not the default, and no `# coding` comment was found.
 - The source file encoding is not the one declared in the `# coding` comment
- Default source encoding:
 - Python 2.7 == ASCII
 - Python 3.x == UTF-8



```
1#!/usr/bin/env python2.7
2# coding: utf-8
3
4u = u'El Niño'
5for codec in ['latin_1', 'utf_8', 'utf_16']:
6    print codec + '\t' + u.encode(codec)
7|
```

Characters: 143 · Words: 21

2.7 gotcha:
default source
encoding is ASCII

Unicode database

```
5. bash
$ python3 numerics_demo.py
U+0031    1      re_dig isdig  isnum  1.00  DIGIT ONE
U+00bc    ¼      -      -      isnum  0.25  VULGAR FRACTION ONE QUARTER
U+00b2    ²      -      isdig  isnum  2.00  SUPERSCRIPT TWO
U+0969    ३      re_dig isdig  isnum  3.00  DEVANAGARI DIGIT THREE
U+136b    ፫      -      isdig  isnum  3.00  ETHIOPIC DIGIT THREE
U+216b    XII     -      -      isnum 12.00  ROMAN NUMERAL TWELVE
U+2466    ⑦      -      isdig  isnum  7.00  CIRCLED DIGIT SEVEN
U+2480    (13)    -      -      isnum 13.00  PARENTHESESIZED NUMBER THIRTEEN
U+3285    ⑥      -      -      isnum  6.00  CIRCLED IDEOGRAPH SIX
$ █
```

Unicode database

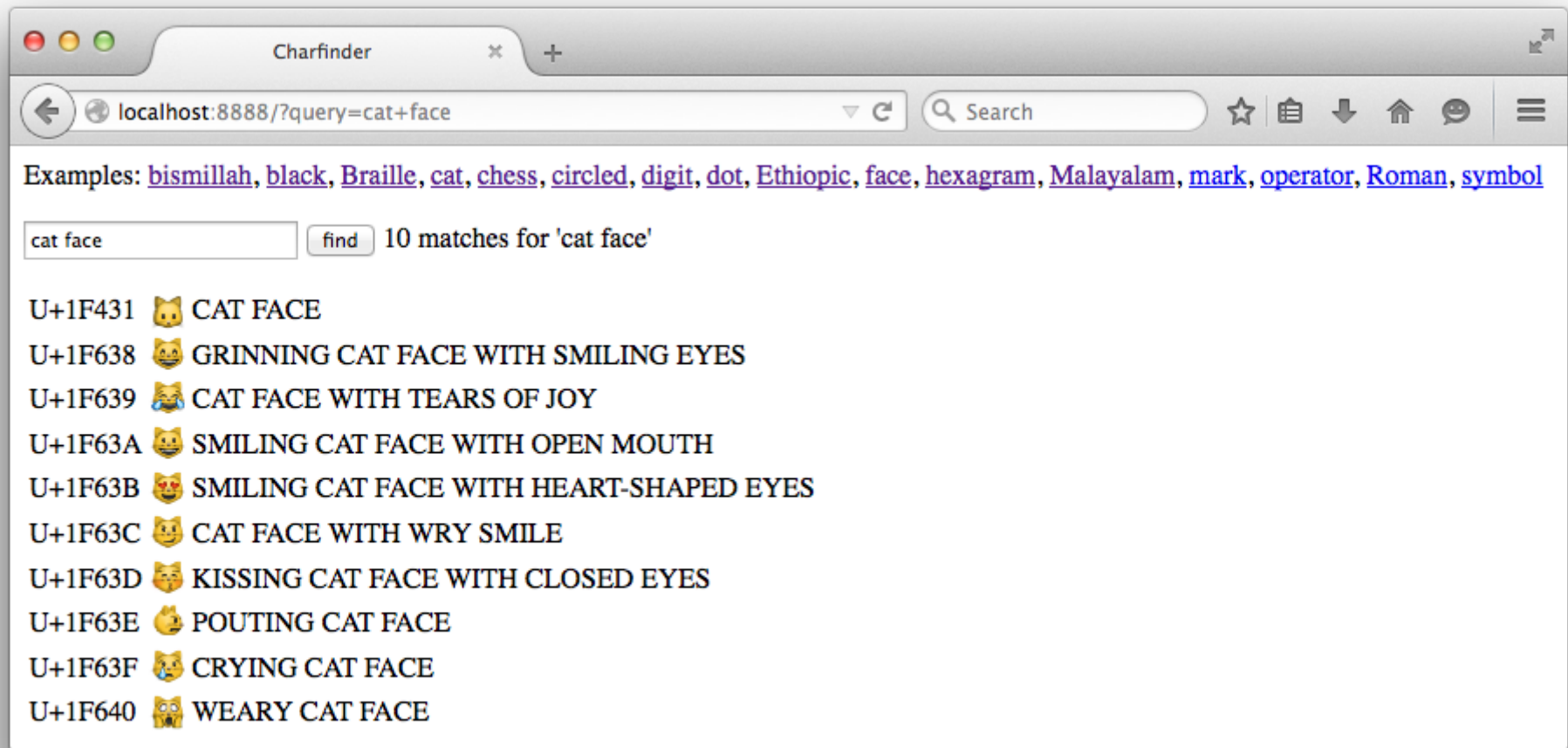
```
$ python3 numerics_demo.py
U+0031      1      re_dig isdig  isnum  1.00  DIGIT ONE
U+00bc      ¼      -      -      isnum  0.25  VULGAR FRACTION ONE QUARTER
U+00b2      ²      -      isdig  isnum  2.00  SUPERSCRIPT TWO
U+0969      ३      re_dig isdig  isnum  3.00  DEVANAGARI DIGIT THREE
U+136b      ፫      -      isdig  isnum  3.00  ETHIOPIC DIGIT THREE
U+216b      XII    -      -      isnum  12.00 ROMAN NUMERAL TWELVE
U+2466      ⑦      -      isdig  isnum  7.00  CIRCLED DIGIT SEVEN
U+2480      (13)   -      -      -
U+3285      ⑆      -      -      -
$
```

```
1 import unicodedata
2 import re
3
4 re_digit = re.compile(r'\d')
5
6 sample = '1\xbc\xb2\u0969\u136b\u216b\u2466\u2480\u3285'
7
8 for char in sample:
9     print('U+%04x' % ord(char),          # <A>
10          char.center(6),                 # <B>
11          're_digit' if re_digit.match(char) else '-', # <C>
12          'isdig' if char.isdigit() else '-',         # <D>
13          'isnum' if char.isnumeric() else '-',       # <E>
14          format(unicodedata.numeric(char), '5.2f'),  # <F>
15          unicodedata.name(char),                    # <G>
16          sep='\t')
17
```

Py3

Characters: 578 - Words: 57

flupy-ch18/http_charfinder.py



flupy-ch18/charfinder.py

```
1. bash
(.venv34) lontra:flupy-ch18 luciano$ ./charfinder.py bear
U+1F43B 🐻 BEAR FACE
(1 match for 'bear')
(.venv34) lontra:flupy-ch18 luciano$ ./charfinder.py eyes smiling
U+1F601 😄 GRINNING FACE WITH SMILING EYES
U+1F604 😁 SMILING FACE WITH OPEN MOUTH AND SMILING EYES
U+1F606 😊 SMILING FACE WITH OPEN MOUTH AND TIGHTLY-CLOSED EYES
U+1F60A 😇 SMILING FACE WITH SMILING EYES
U+1F60D 😍 SMILING FACE WITH HEART-SHAPED EYES
U+1F619 😗 KISSING FACE WITH SMILING EYES
U+1F638 😺 GRINNING CAT FACE WITH SMILING EYES
U+1F63B 😻 SMILING CAT FACE WITH HEART-SHAPED EYES
(8 matches for 'eyes smiling')
(.venv34) lontra:flupy-ch18 luciano$
```


flupy-ch18/tcp_charfinder.py

```
4. bash
lontra:charfinder luciano$ telnet localhost 2323
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
?> chess black
U+265A ♜ BLACK CHESS KING
U+265B ♝ BLACK CHESS QUEEN
U+265C ♞ BLACK CHESS ROOK
U+265D ♟ BLACK CHESS BISHOP
U+265E ♠ BLACK CHESS KNIGHT
U+265F ♟ BLACK CHESS PAWN
6 matches for 'chess black'
?> sun
U+2600 ☀ BLACK SUN WITH RAYS
U+2609 ☉ SUN
U+263C ☼ WHITE SUN WITH RAYS
U+26C5 ☁ SUN BEHIND CLOUD
U+2E9C ☞ CJK RADICAL SUN
U+2F47 日 KANGXI RADICAL SUN
U+3230 (日) PARENTHESES IDEOGRAPH SUN
U+3290 (☉) CIRCLED IDEOGRAPH SUN
U+C21C ☼ HANGUL SYLLABLE SUN
U+1F31E 🌞 SUN WITH FACE
10 matches for 'sun'
?> ^C
Connection closed by foreign host.
lontra:charfinder luciano$
```