












Unicode solutions in Python 2 and 3

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

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	13164	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

0D0	0D1	0D2	0D3	0D4	0D5	0D6
എ	ഐ	ഓ	ഐ	ഓ	ഐ	ഓ
0D10	0D20	0D30	0D40	0D50	0D60	0D70
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D21	0D31	0D41	0D51	0D61	0D71	0D81
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D12	0D22	0D32	0D42	0D52	0D62	0D72
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D13	0D23	0D33	0D43	0D53	0D63	0D73
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D14	0D24	0D34	0D44	0D54	0D64	0D74
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D15	0D25	0D35	0D45	0D55	0D65	0D75
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D16	0D26	0D36	0D46	0D56	0D66	0D76
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D17	0D27	0D37	0D47	0D57	0D67	0D77
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D18	0D28	0D38	0D48	0D58	0D68	0D78
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D19	0D29	0D39	0D49	0D59	0D69	0D79
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D1A	0D2A	0D3A	0D4A	0D5A	0D6A	0D7A
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D1B	0D2B	0D3B	0D4B	0D5B	0D6B	0D7B
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D1C	0D2C	0D3C	0D4C	0D5C	0D6C	0D7C
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D1D	0D2D	0D3D	0D4D	0D5D	0D6D	0D7D
ഐ	ഓ	ഐ	ഓ	ഐ	ഓ	ഐ
0D1E	0D2E	0D3E	0D4E	0D5E	0D6E	0D7E

HEX	C	J	K	V
50D0 人 9.12	僖 G5-5071	僖 H-9003	僖 J1-3038	僖 K2-204C
50D1 人 9.12	僑 G1-4748	僑 H81-0964	僑 J1-4774	僑 K2-4680
50D2 人 9.12	僭 G3-3238	僭 T4-4220		
50D3 人 9.12	債 G3-3165	債 H52-02F9	債 J1-3231	債 K2-204D
50D4 人 9.12	傳 G3-3237	傳 H83-02F1	傳 J1-323A	傳 K2-234E
50D5 人 9.12	僕 G1-4840	僕 H81-0982	僕 J4-0940	僕 K2-5C52
50D6 人 9.12	僖 G0-5952	僖 H81-094F	僖 J4-0125	僖 K2-708A
50D7 人 9.12	僂 G3-3137	僂 H82-02F2	僂 J2-4878	僂 K2-234F
50D8 人 9.12	倣 G5-2238	倣 T3-4685	倣 J1-3038	倣 K2-2350
50D9 人 9.12	僨 G5-3261	僨 H-A0A6	僨 J4-2178	僨 K2-2351
50DA 人 9.12	僚 G0-4145	僚 H81-0981	僚 J1-4777	僚 K2-5E78
50DB 人 9.12	倣 G3-3232	倣 H82-02F8	倣 J2-4879	倣 K1-0884
50DC 人 9.12	僦 G3-3238	僦 T3-4590	僦 J1-303C	僦 K2-2352
50DD 人 9.12	倣 G3-3239	倣 H82-02F7	倣 J1-303D	倣 K2-2353
50DE 人 9.12	僞 G1-4E31	僞 T3-4576	僞 J4-0128	僞 K2-6A6A
50DF 人 9.12	僣 G2-2239	僣 H-94A9	僣 J1-323E	僣 K2-235A
50E0 人 9.12	僖 G2-223A	僖 H82-02F6	僖 J2-4685	僖 K2-2355
50E1 人 9.12	僣 G5-3269	僣 H-PC72	僣 T3-4570	僣 J4-217C
50E2 人 9.12	僣 G2-326F	僣 T3-4575	僣 J1-323F	僣 K1-6F32
50E3 人 9.12	僣 G2-326F	僣 T3-4575	僣 J1-323F	僣 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	×	Ø	Ù
Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å
æ	ç	ð	ñ	ò	ó
ô	õ	ö	÷	ø	ù
ú	û	ü	ý	þ	ÿ

Latin-1

1F61	1F62	1F63	1F64
😡	😢	😣	😤
😥	😦	😧	😨
😩	😪	😫	😬
😭	😮	😯	😰
😱	😲	😳	😴
😵	😶	😷	😸
😹	😺	😻	😼
😽	😾	😿	🐼

emoticons

053	054	055	056	057	058
Ա	Բ	Գ	Դ	Ե	Զ
Է	Ը	Թ	Ձ	Ղ	Ճ
Մ	Ն	Շ	Ո	Պ	Պ
Ղ	Ճ	Մ	Ն	Շ	Ո
Պ	Պ	Ղ	Ճ	Մ	Ն
Շ	Ո	Պ	Պ	Ղ	Ճ
Ո	Պ	Պ	Ղ	Ճ	Մ
Պ	Պ	Ղ	Ճ	Մ	Ն

Armenian

1314	1315	1316	1317	1318	1319	131A	131B
𐤀	𐤁	𐤂	𐤃	𐤄	𐤅	𐤆	𐤇
𐤈	𐤉	𐤊	𐤋	𐤌	𐤍	𐤎	𐤏
𐤐	𐤑	𐤒	𐤓	𐤔	𐤕	𐤖	𐤗
𐤘	𐤙	𐤚	𐤛	𐤜	𐤝	𐤞	𐤟
𐤠	𐤡	𐤢	𐤣	𐤤	𐤥	𐤦	𐤧
𐤨	𐤩	𐤪	𐤫	𐤬	𐤭	𐤮	𐤯
𐤰	𐤱	𐤲	𐤳	𐤴	𐤵	𐤶	𐤷

Egyptian
Hieroglyphs

0D0	0D1	0D2	0D3	0D4	0D5	0D6
ഐ	ഓ	ഔ	ഓ	ഔ	ഓ	ഔ
ഐ	ഓ	ഔ	ഓ	ഔ	ഓ	ഔ
ഐ	ഓ	ഔ	ഓ	ഔ	ഓ	ഔ
ഐ	ഓ	ഔ	ഓ	ഔ	ഓ	ഔ
ഐ	ഓ	ഔ	ഓ	ഔ	ഓ	ഔ
ഐ	ഓ	ഔ	ഓ	ഔ	ഓ	ഔ
ഐ	ഓ	ഔ	ഓ	ഔ	ഓ	ഔ

Malayalam

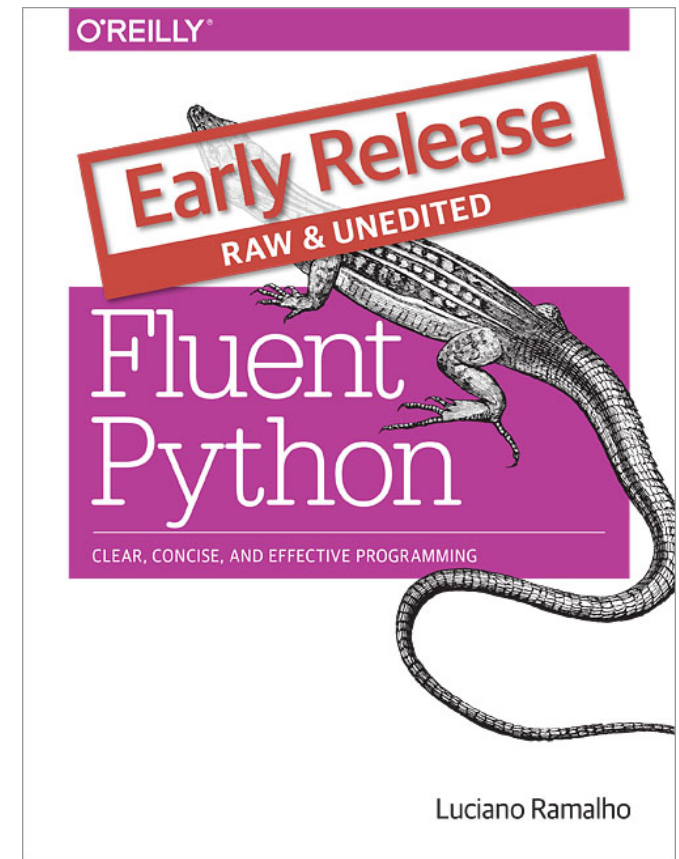
HEX	C	J	K	V
50D0	僣	僣	僣	僣
50D1	僣	僣	僣	僣
50D2	僣	僣	僣	僣
50D3	僣	僣	僣	僣
50D4	僣	僣	僣	僣
50D5	僣	僣	僣	僣
50D6	僣	僣	僣	僣
50D7	僣	僣	僣	僣
50D8	僣	僣	僣	僣
50D9	僣	僣	僣	僣
50DA	僣	僣	僣	僣
50DB	僣	僣	僣	僣
50DC	僣	僣	僣	僣
50DD	僣	僣	僣	僣

CJK
Unified
Ideographs

About me: Luciano Ramalho

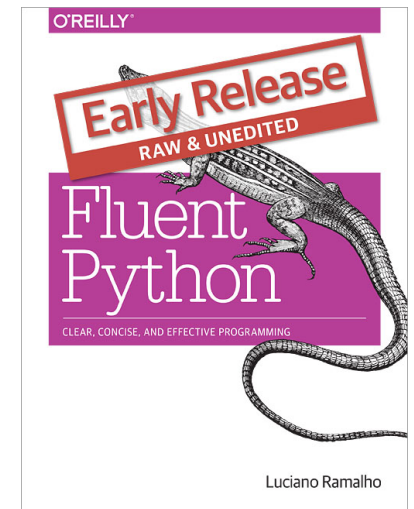
- Programming in Python since 1998
- Focus on content management (i.e. text wrangling)
- Teaching Python since 1999
- Speaker at PyCon US, OSCON, FISL, PythonBrasil, RuPy, QCon...
- Author of **Fluent Python**
- Twitter: @ramalhoorg
- Native language: Português
 - “ação”

4 non-ASCII
characters here



Resources

- All code, slides and images used in this talk:
 - <https://github.com/fluentpython/unicode-solutions>
- **Fluent Python**
 - <http://shop.oreilly.com/product/0636920032519.do>
 - Relevant content and examples:
 - Chapter 4: *Text versus Bytes*
 - all 39 pages
 - Chapter 18: *Concurrency with asyncio*
 - the *charfinder* examples



The single-byte codepage ballet

tk

0 1 2 3 4 5 6 7 8 9 A B C D E F

00

10

20 ! " # \$ % & ' () * + , - . /

30 0 1 2 3 4 5 6 7 8 9 : ; < = > ?

40 @ A B C D E F G H I J K L M N O

50 P Q R S T U V W X Y Z [\] ^ _

60 ` a b c d e f g h i j k l m n o

70 p q r s t u v w x y z { | } ~

80

90

A0

B0

C0

D0

E0

F0

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



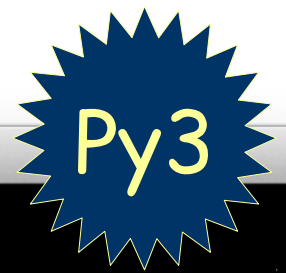
Python 2.7



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



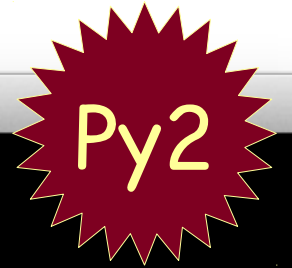
```
>>> b = bytes('café', encoding='utf-8')
>>> b
b'caf\xc3\xa9'
>>> list(b)
[99, 97, 102, 195, 169]
>>> b[0]
99
>>> b[1:]
b'af\xc3\xa9'
>>> b_arr = bytearray(b)
>>> b_arr
bytearray(b'caf\xc3\xa9')
>>> b_arr[0] = b'd'
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: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



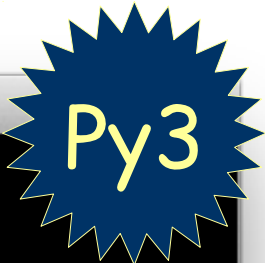
```
>>> s = 'café'
>>> len(s)
5
>>> s
'caf\xc3\xa9'
>>> u = s.decode('utf-8')
>>> u
u'caf\xe9'
>>> print u
café
>>> len(u)
4
>>> list(s)
['c', 'a', 'f', '\xc3', '\xa9']
>>> list(u)
[u'c', u'a', u'f', u'\xe9']
>>> type(s)
<type 'str'>
>>> type(u)
<type 'unicode'>
```

.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

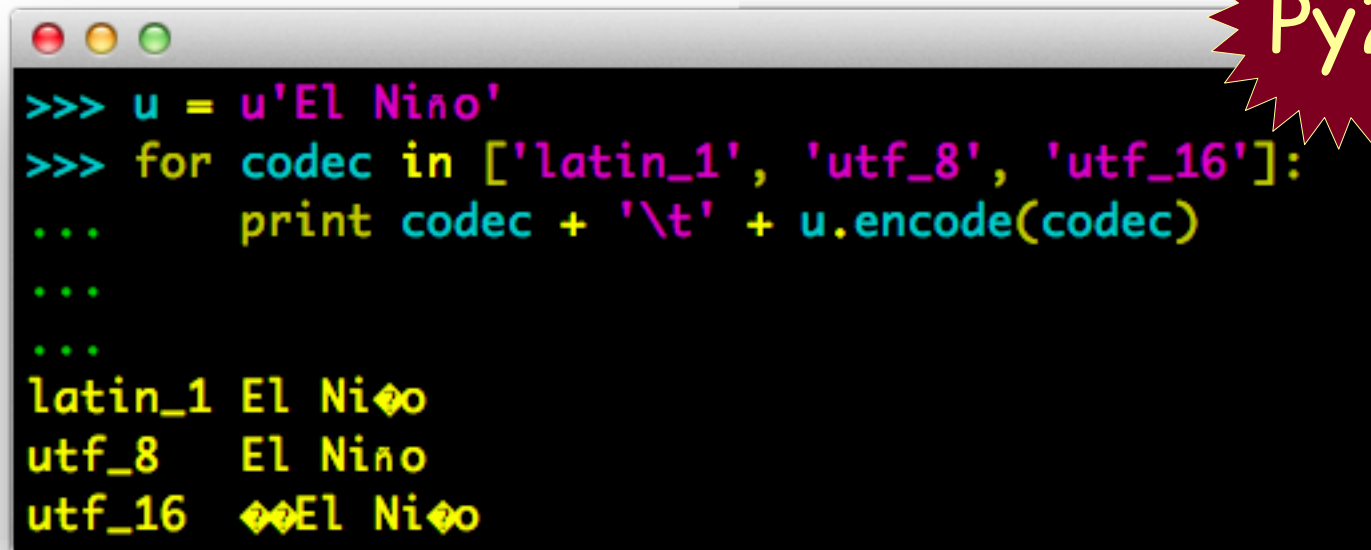
2.7 gotcha:
the methods
.encode() and .decode()
exist in **str** and **unicode**

str.encode(...) in Py3

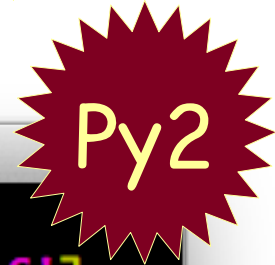


```
>>> s = 'El Niño'
>>> for codec in ['latin_1', 'utf_8', 'utf_16']:
...     print(codec, s.encode(codec), sep='\t')
...
...
latin_1 b'El Ni\xff1o'
utf_8   b'El Ni\xc3\xb1o'
utf_16  b'\xff\xfeE\x00l\x00 \x00N\x00i\x00\x00\x00\x00\x00\x00\x00\x00'
```

unicode.encode(...) in Py2



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



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**

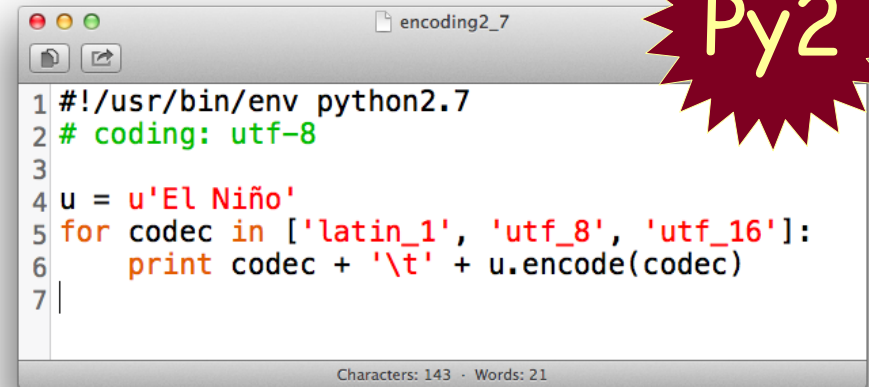
- A binary sequence contains bytes that are not valid in the expected encoding

- **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

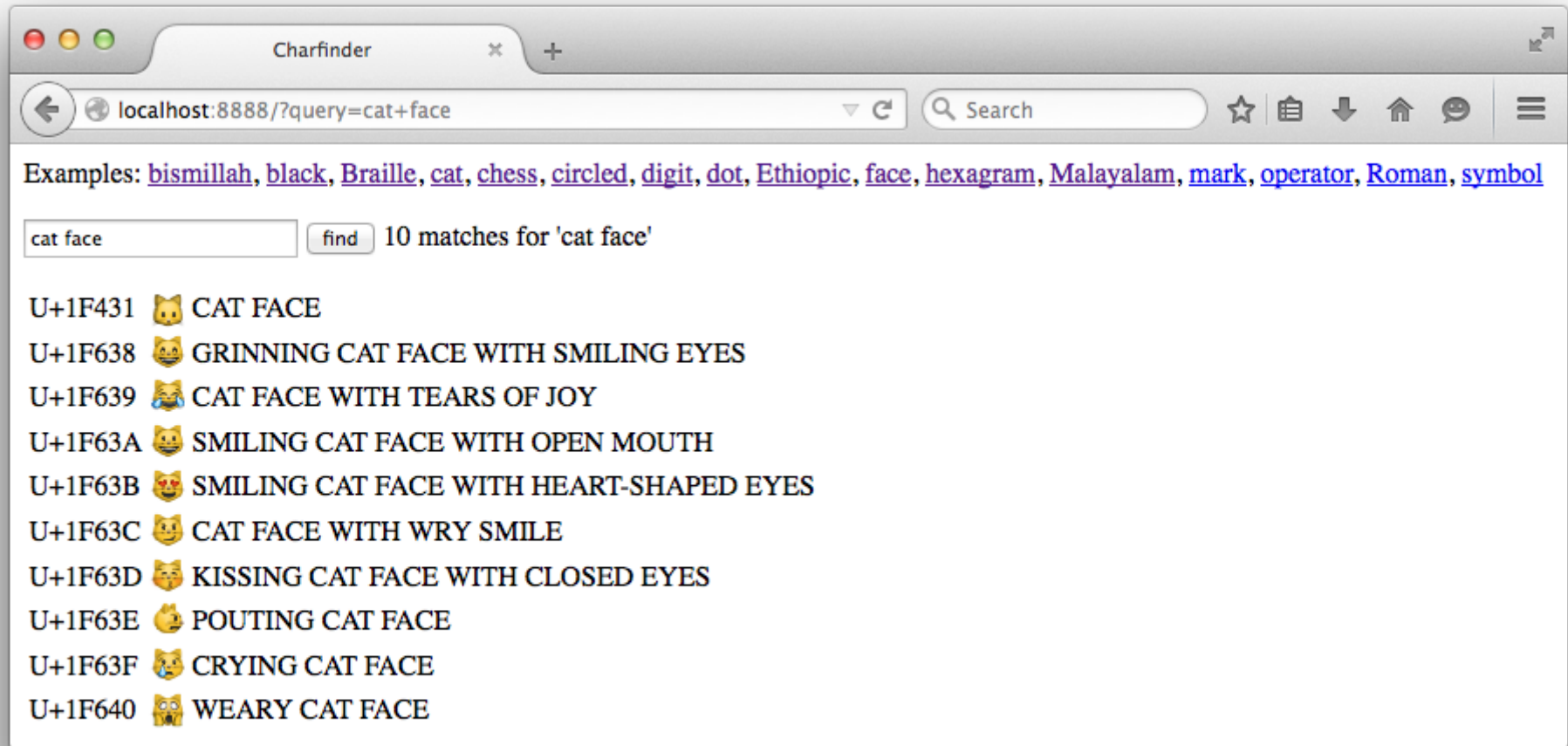
```
$ python3 numerics_demo.py
U+0031      1      re_dig isdig  isnum  1.00  DIGIT ONE
U+00bc      ¼      -      -      isnum  0.25  VULGAR FRACTION ONE QUARTER
U+00b2      ²      -      isdig  isnum  2.00  SUPERSCRIPT TWO
U+0969      ३      re_dig isdig  isnum  3.00  DEVANAGARI DIGIT THREE
U+136b      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      ⑥      -      -      -
$
```

```
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) PARENTHESESIZED 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$
```