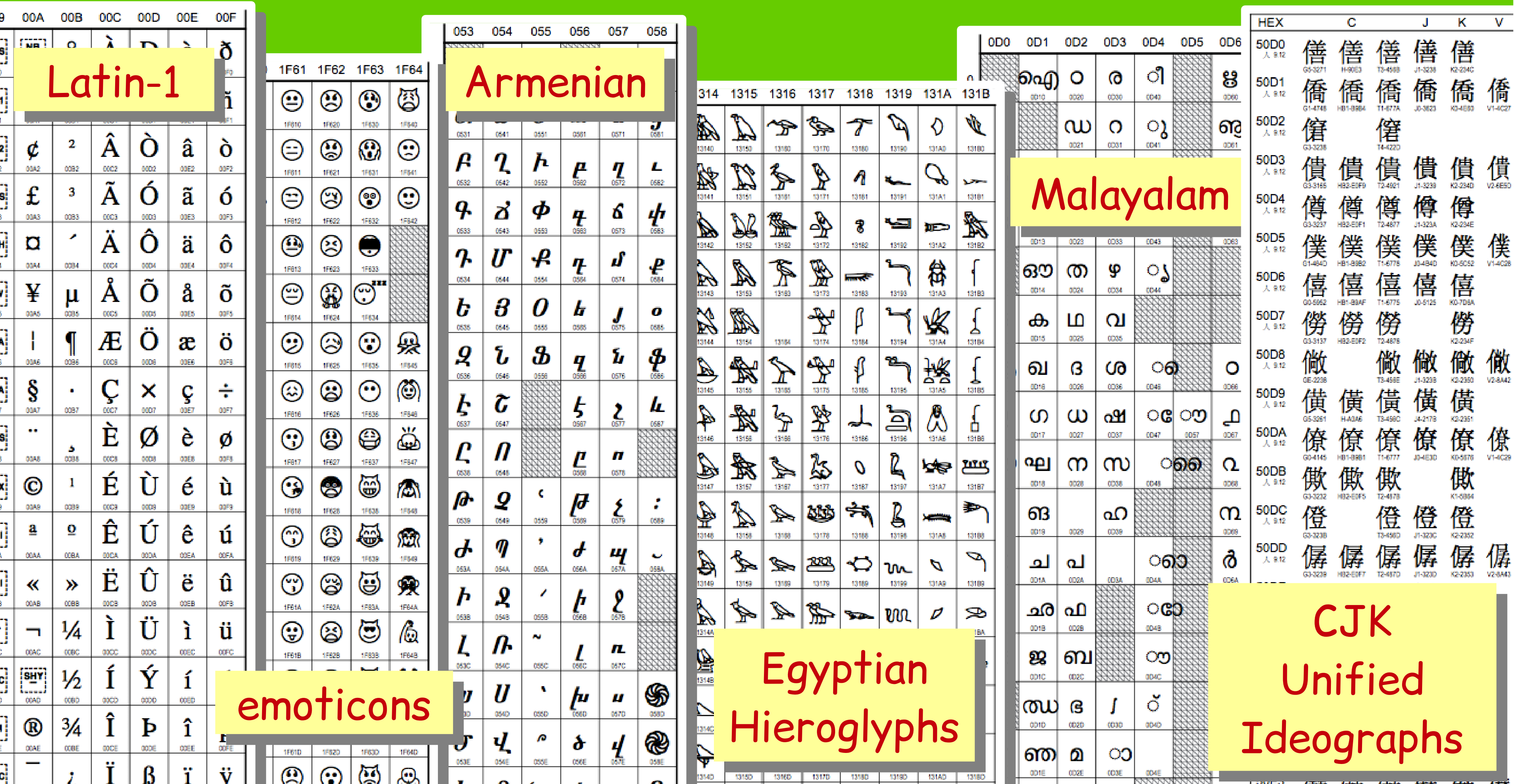
Unicode solutions in Python 2 and 3

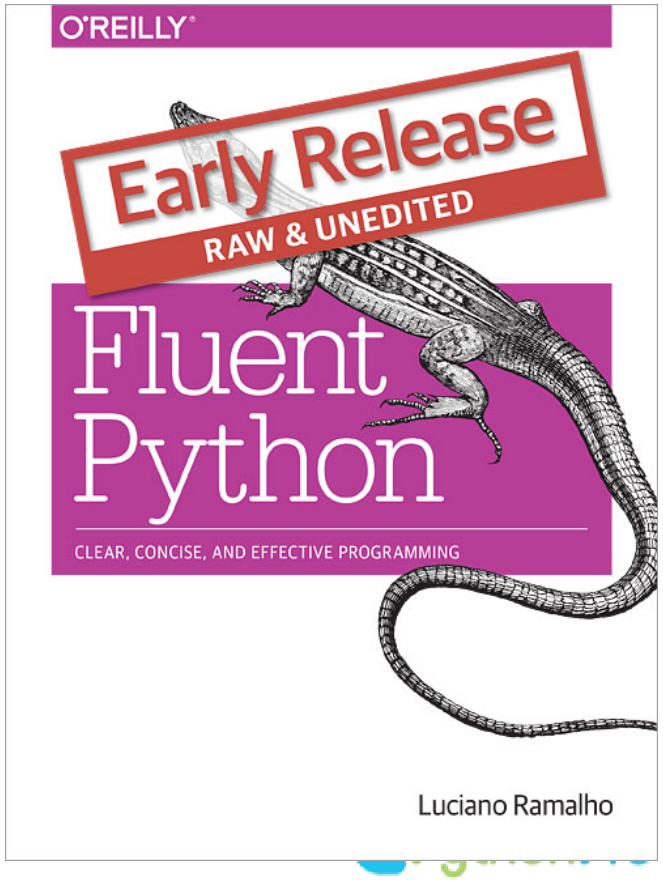
Unicode solutions in Python 2 and 3



About me: Luciano Ramalho

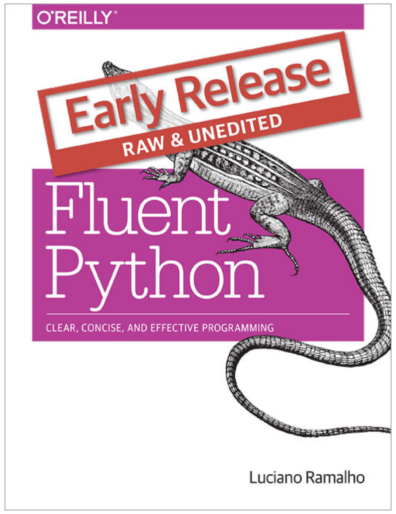
- Programming in Python since 1998
- Focus on content management (i.e. text wrangling)
- Teaching Python since 1999
- Speaker at PyCon US, OSCON, FISL, PythonBrasil, RuPy, QCon...
- Author of **Fluent Python**
- Twitter: @ramalhoorg
- Native language: Português
  - “ação”

4 non-ASCII characters here

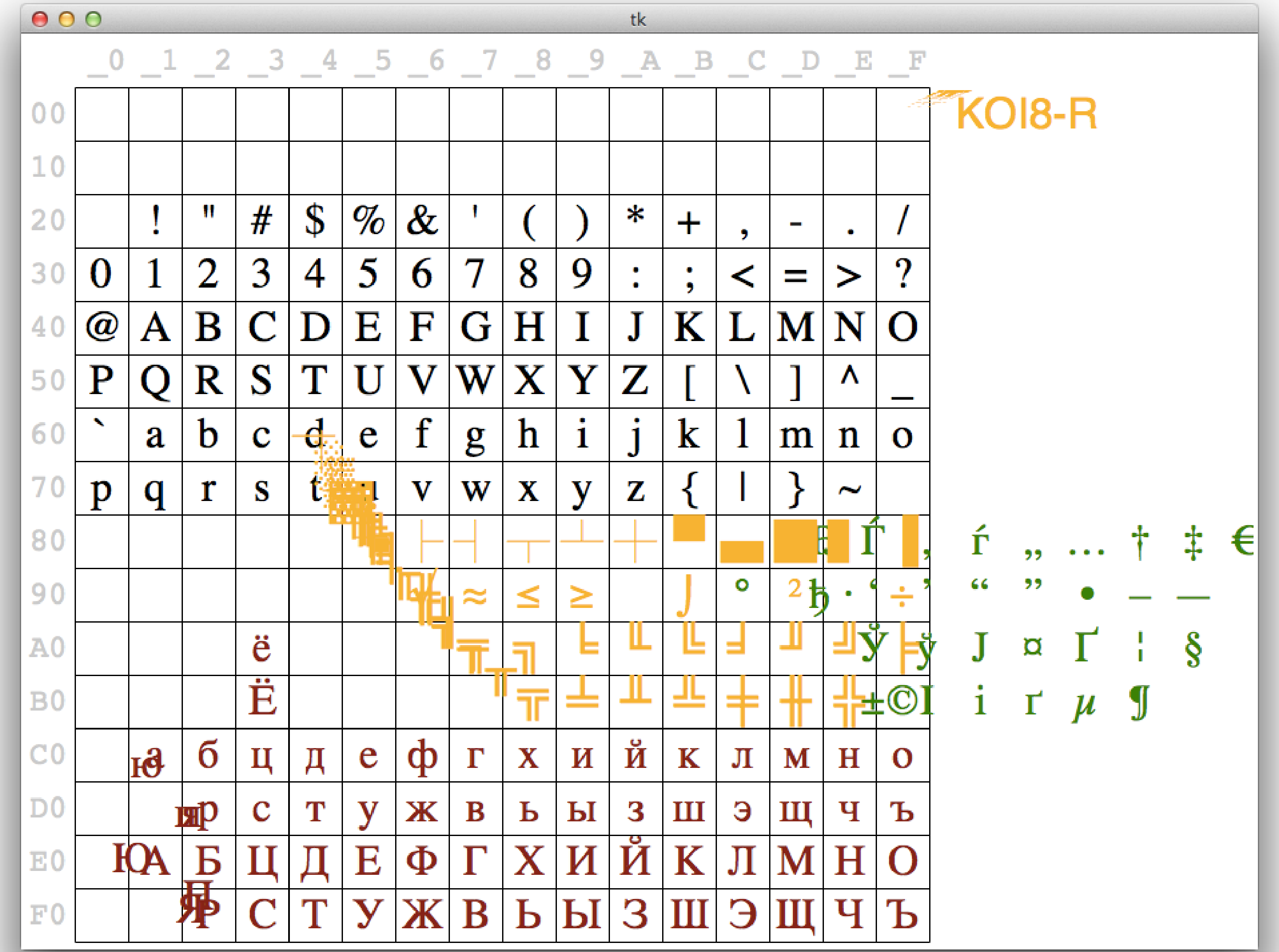


Resources

- 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							├	┤	┬	┴	┼	▀	▄	█	▌Ѓ	▐‚
90								≈	≤	≥		⌡	°	²ђ	· ‘	÷’
A0				ё				╥	╕	╘	╙	╚	╛	╜	╝Ў	╞ў
B0				Ё					╤	╧	╨	╩	╪	╫	╬±©І	
C0		юа	б	ц	д	е	ф	г	х	и	й	к	л	м	н	о
D0			яр	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ
E0	ЮА		Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О
F0			ЯП Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ

KOI8-R

ŕ „ … † ‡ €

“ ” • – —

Ј ¤ Ґ ¦ §

і ґ µ ¶

Video: `https://www.youtube.com/watch?v=J4qioAacrYo`
Source code: `http://bit.ly/1Oqt0MZ`

PythonPro

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**

	UTF-8	UTF-16LE
• `U+0041 →`	`0x41`	`0x41 0x00`
• `U+096C →`	`0xE0 0xA5 0xAC`	`0x6C 0x09`

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
ڿ	U+06BF	*	*	*	*	*	DA BF	BF 06
“	U+201C	*	*	93	*	A1 B0	E2 80 9C	1C 20
€	U+20AC	*	*	80	*	*	E2 82 AC	AC 20
┌	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

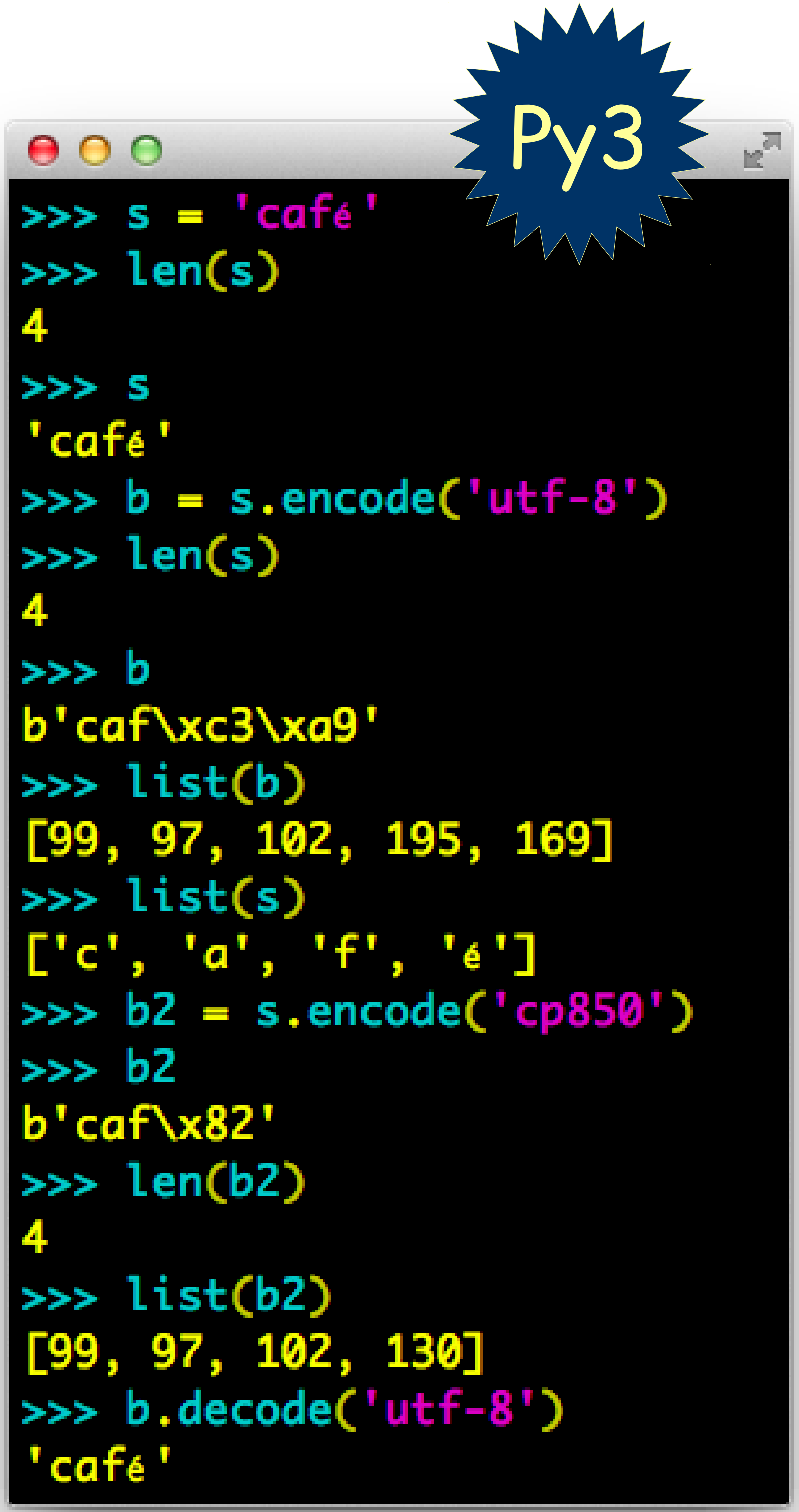
Figure 4-1 of Fluent Python

Data types for text or bytes

	Py2 **Python 2.7**	Py3 **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')

PythonPro

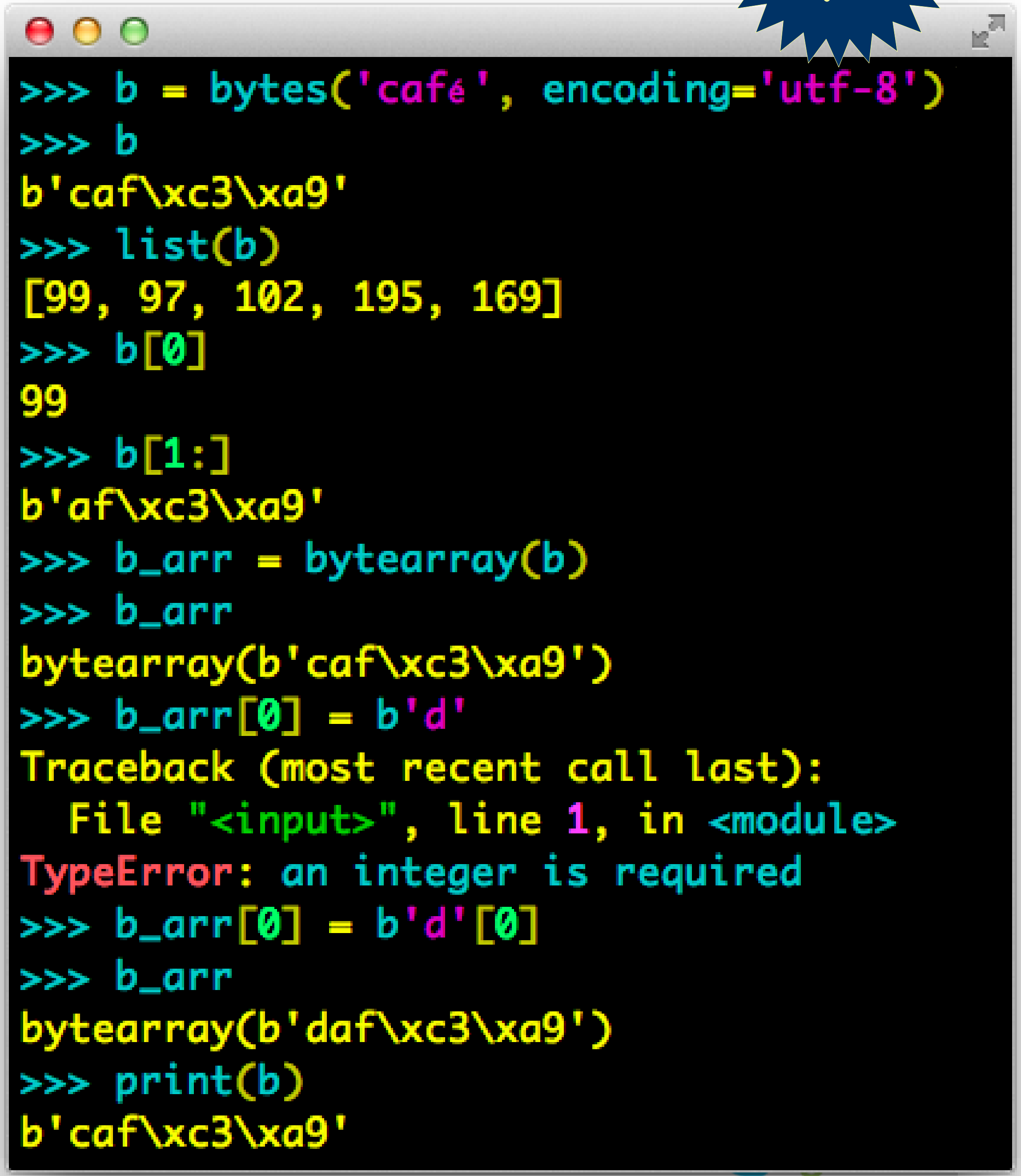
str v. **bytes** in Py3



```
>>> 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'
```

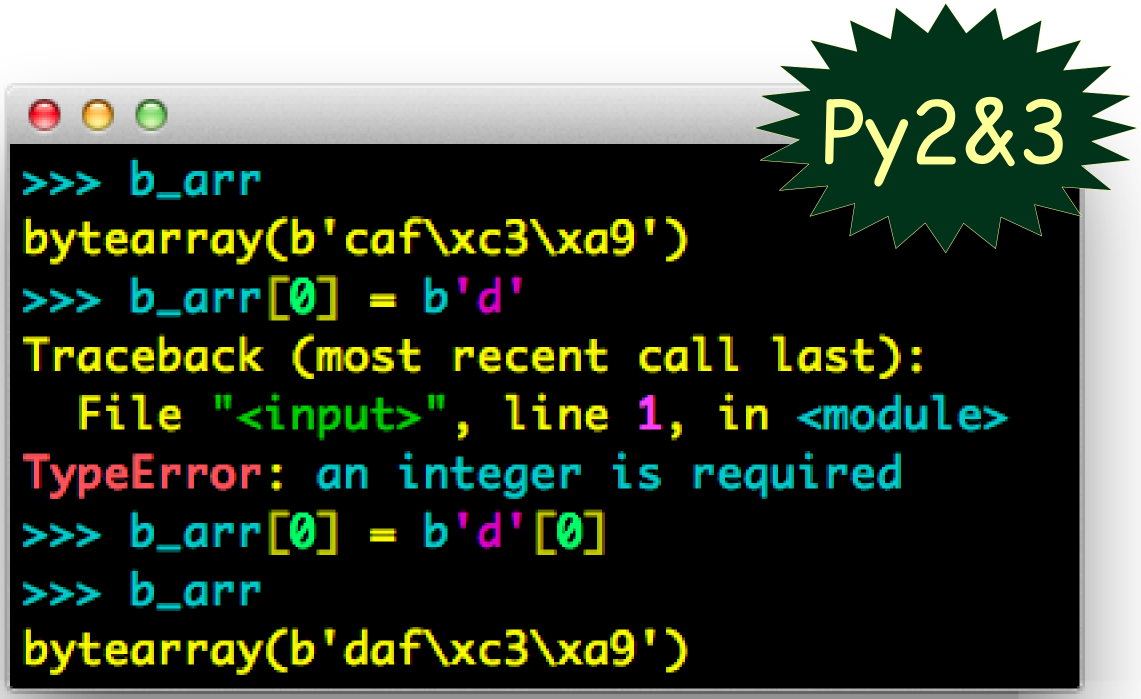
bytes in Py3

Py3



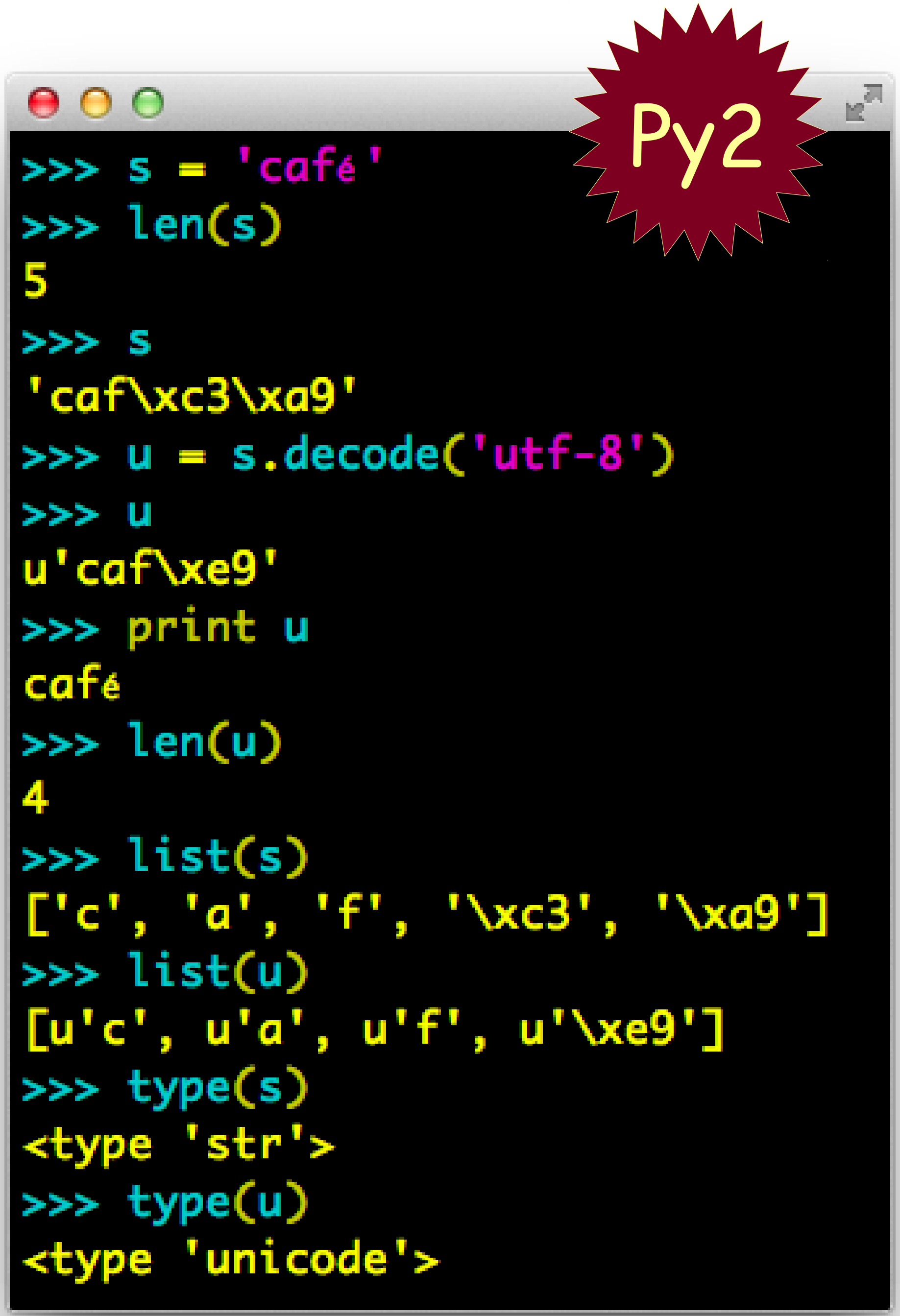
```
>>> b = bytes('café', encoding='utf-8')
>>> b
b'caf\xc3\xa9'
>>> list(b)
[99, 97, 102, 195, 169]
>>> b[0]
99
>>> b[1:]
b'af\xc3\xa9'
>>> b_arr = bytearray(b)
>>> b_arr
bytearray(b'caf\xc3\xa9')
>>> b_arr[0] = b'd'
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: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



```
>>> 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()

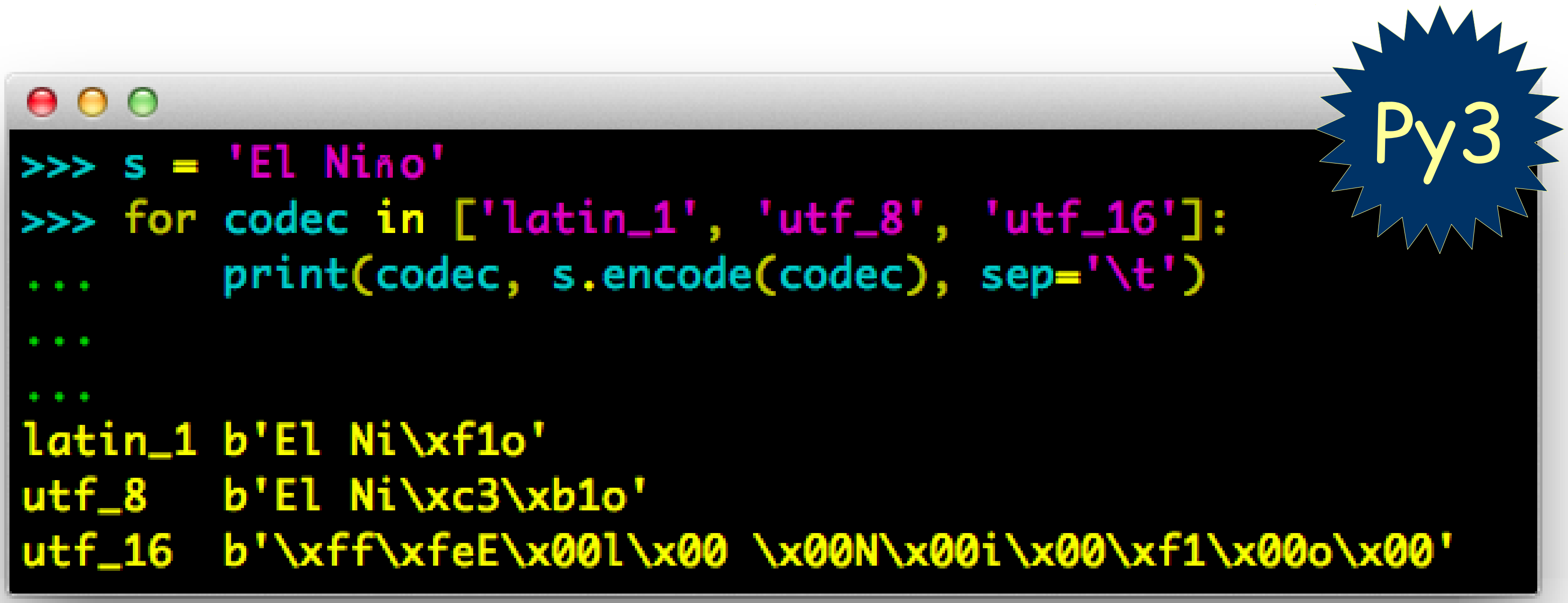
- “Humans use text. Computers speak bytes.”
  - Esther Nam and Travis Fischer in *Character encoding and Unicode in Python (Pycon US 2014)*

- 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**

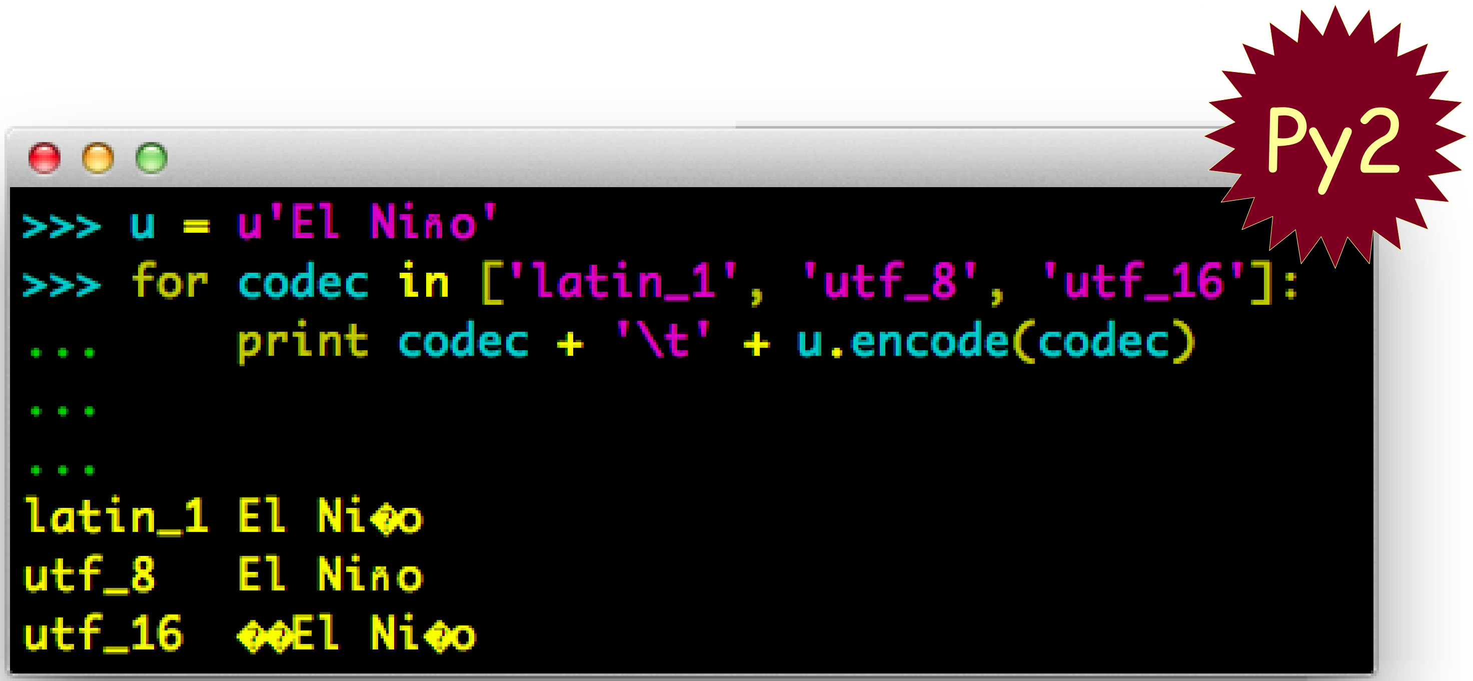
PythonPro

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\xf1o'
utf_8   b'El Ni\xc3\xb1o'
utf_16  b'\xff\xfeE\x00l\x00 \x00N\x00i\x00\xf1\x00o\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�o
utf_8   El Niño
utf_16  ��El Ni�o
```

Best practice

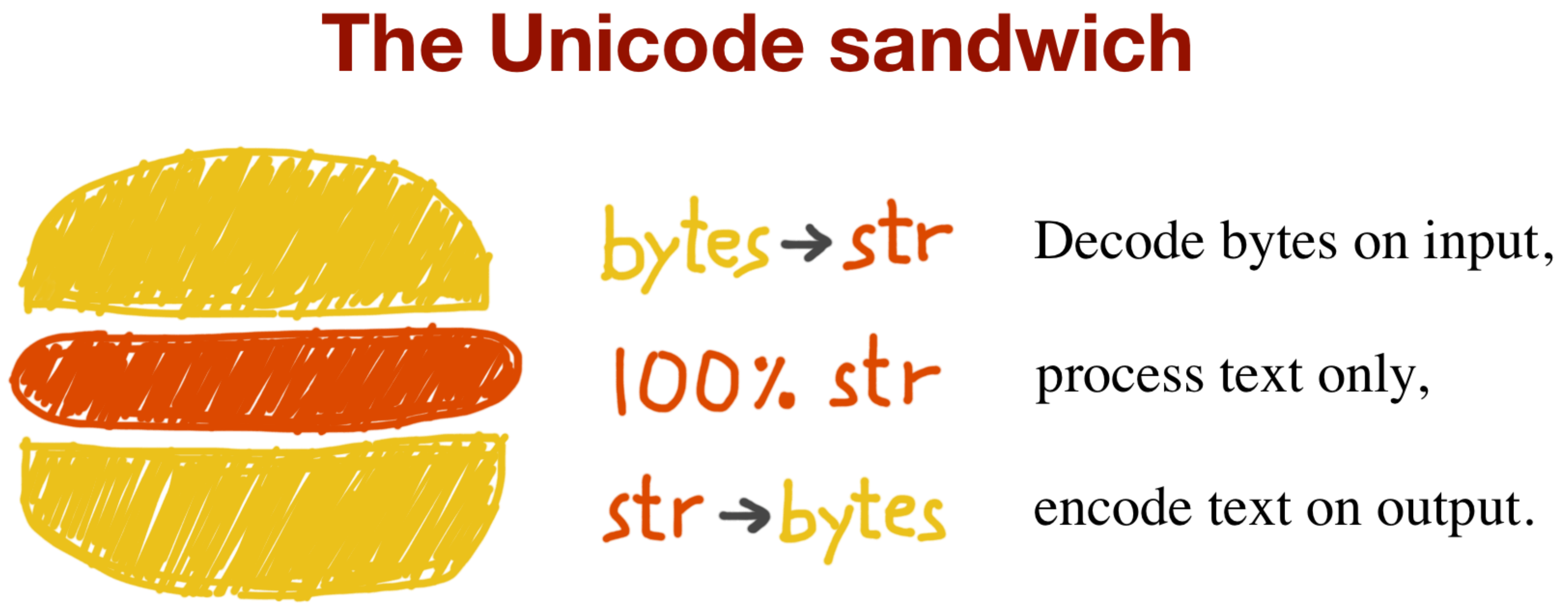


Figure 4-2 of Fluent Python

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()`

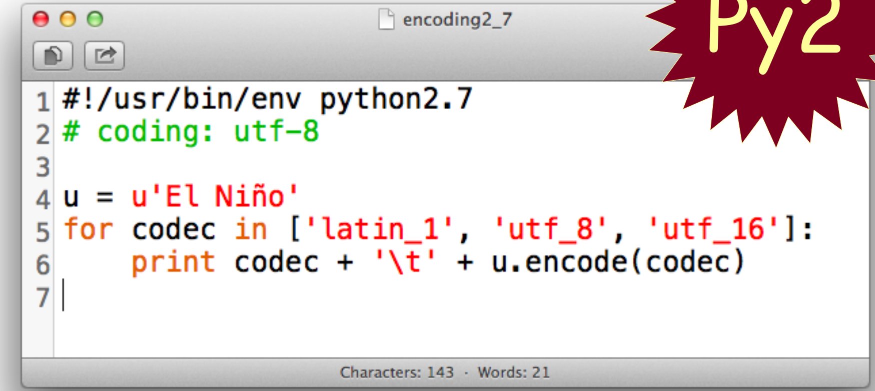
2.7 gotcha:
no way to specify
encoding in built-in open(...).
Must use io.open(...).

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

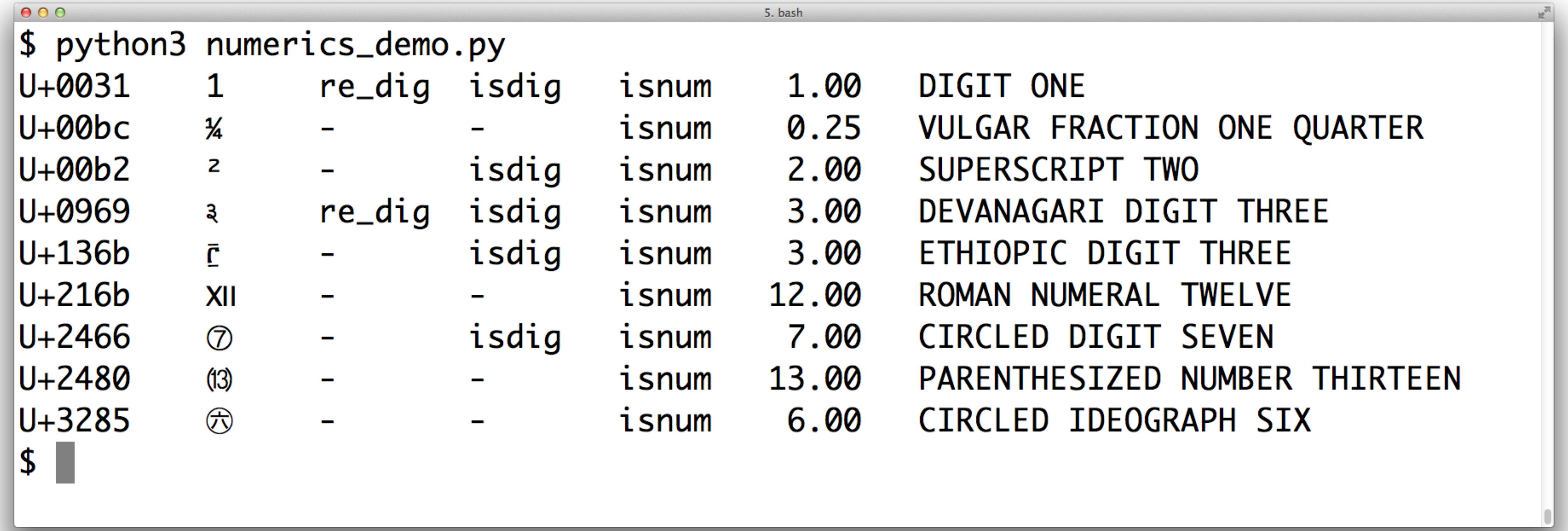


```
#!/usr/bin/env python2.7
# coding: utf-8

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

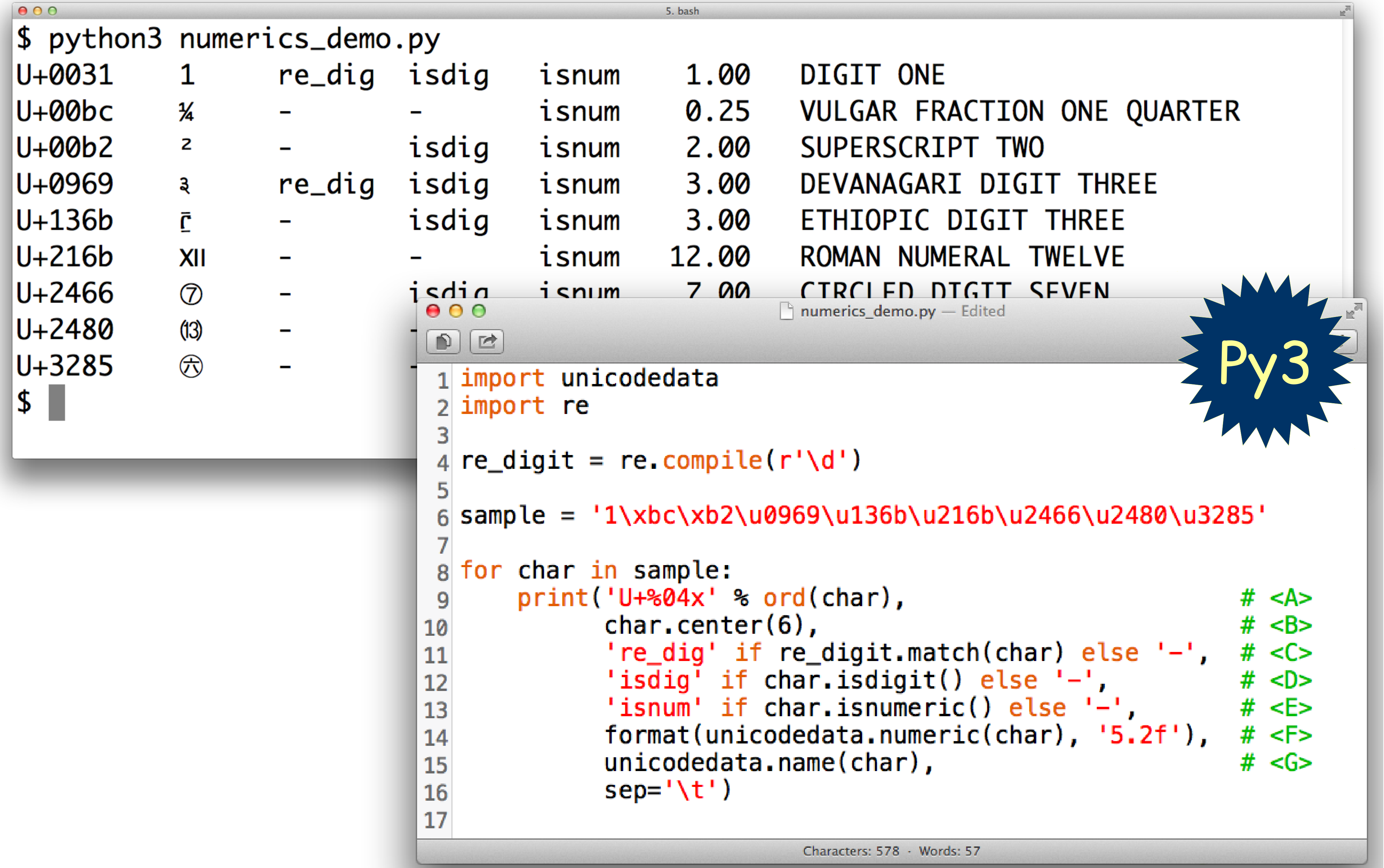
2.7 gotcha: default source encoding is ASCII

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    ⒀     -       -       isnum   13.00   PARENTHESIZED NUMBER THIRTEEN
U+3285    ㊅     -       -       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    ⒀     -       -
U+3285    ㊅     -       -
$ 
```

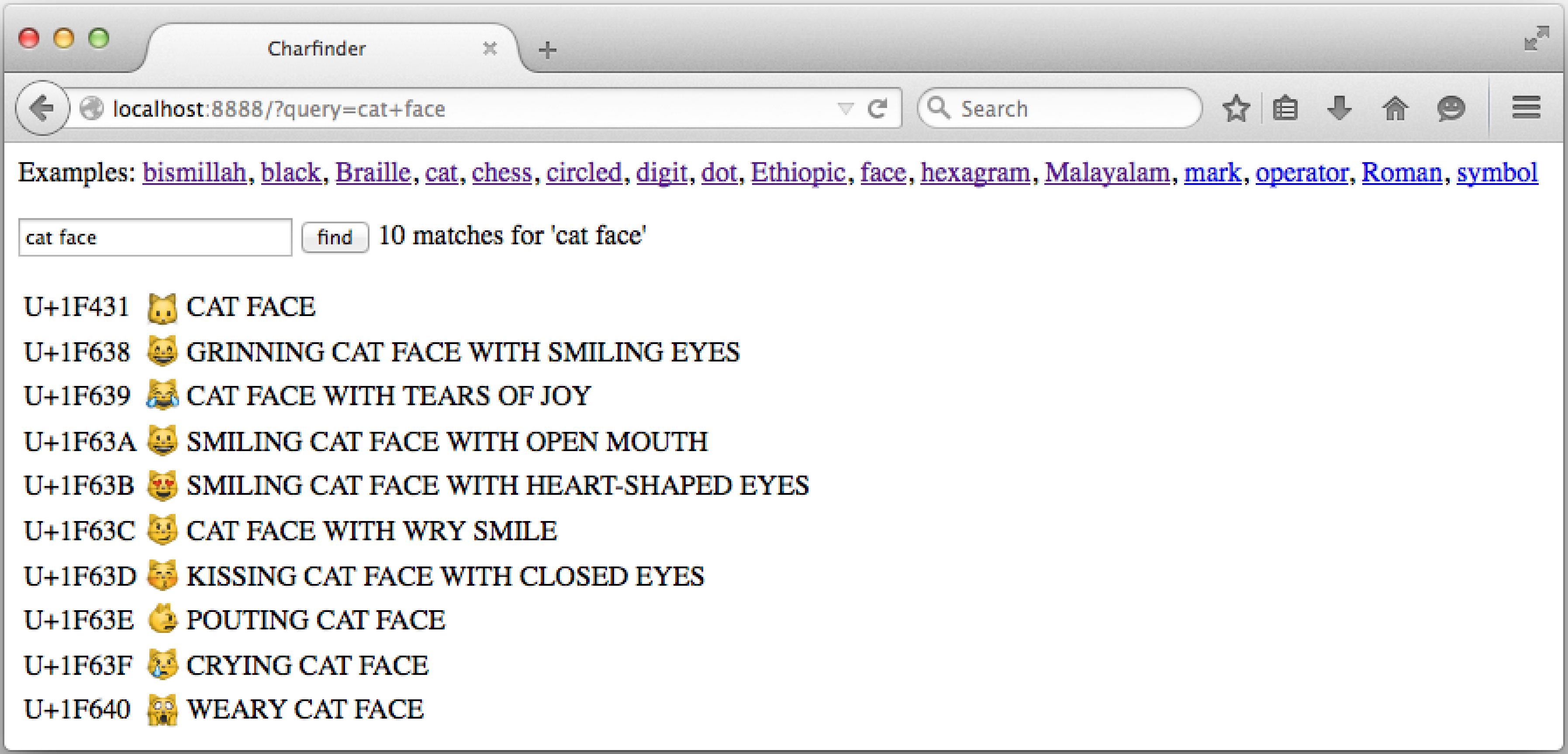
```python
import unicodedata
import re

re_digit = re.compile(r'\d')

sample = '1\xbc\xb2\u0969\u136b\u216b\u2466\u2480\u3285'

for char in sample:
    print('U+%04x' % ord(char),                       # <A>
          char.center(6),                              # <B>
          're_dig' if re_digit.match(char) else '-',   # <C>
          'isdig' if char.isdigit() else '-',          # <D>
          'isnum' if char.isnumeric() else '-',        # <E>
          format(unicodedata.numeric(char), '5.2f'),   # <F>
          unicodedata.name(char),                      # <G>
          sep='\t')
```

flupy-ch18/http_charfinder.py

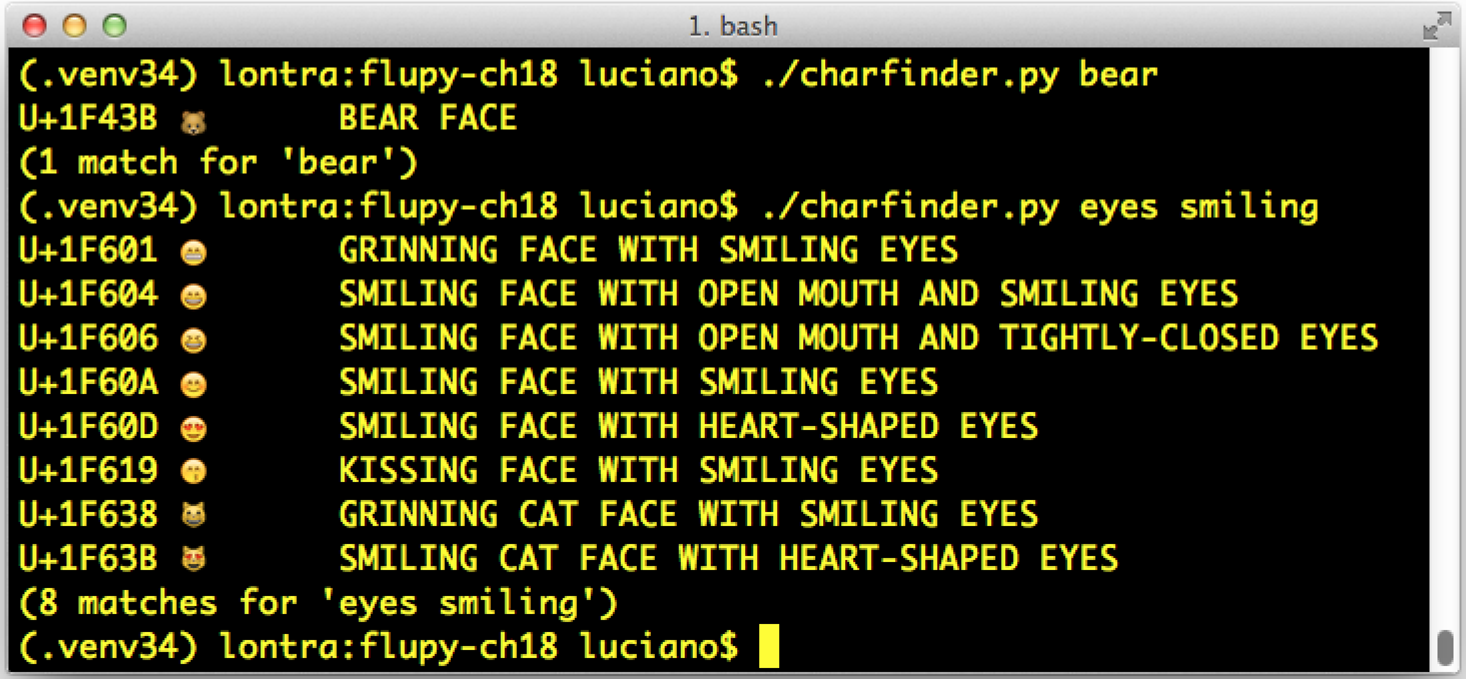


Examples: bismillah, black, Braille, cat, chess, circled, digit, dot, Ethiopic, face, hexagram, Malayalam, mark, operator, Roman, symbol

cat face find 10 matches for 'cat face'

```
U+1F431 🐱 CAT FACE
U+1F638 😸 GRINNING CAT FACE WITH SMILING EYES
U+1F639 😹 CAT FACE WITH TEARS OF JOY
U+1F63A 😺 SMILING CAT FACE WITH OPEN MOUTH
U+1F63B 😻 SMILING CAT FACE WITH HEART-SHAPED EYES
U+1F63C 😼 CAT FACE WITH WRY SMILE
U+1F63D 😽 KISSING CAT FACE WITH CLOSED EYES
U+1F63E 😾 POUTING CAT FACE
U+1F63F 😿 CRYING CAT FACE
U+1F640 🙀 WEARY CAT FACE
```

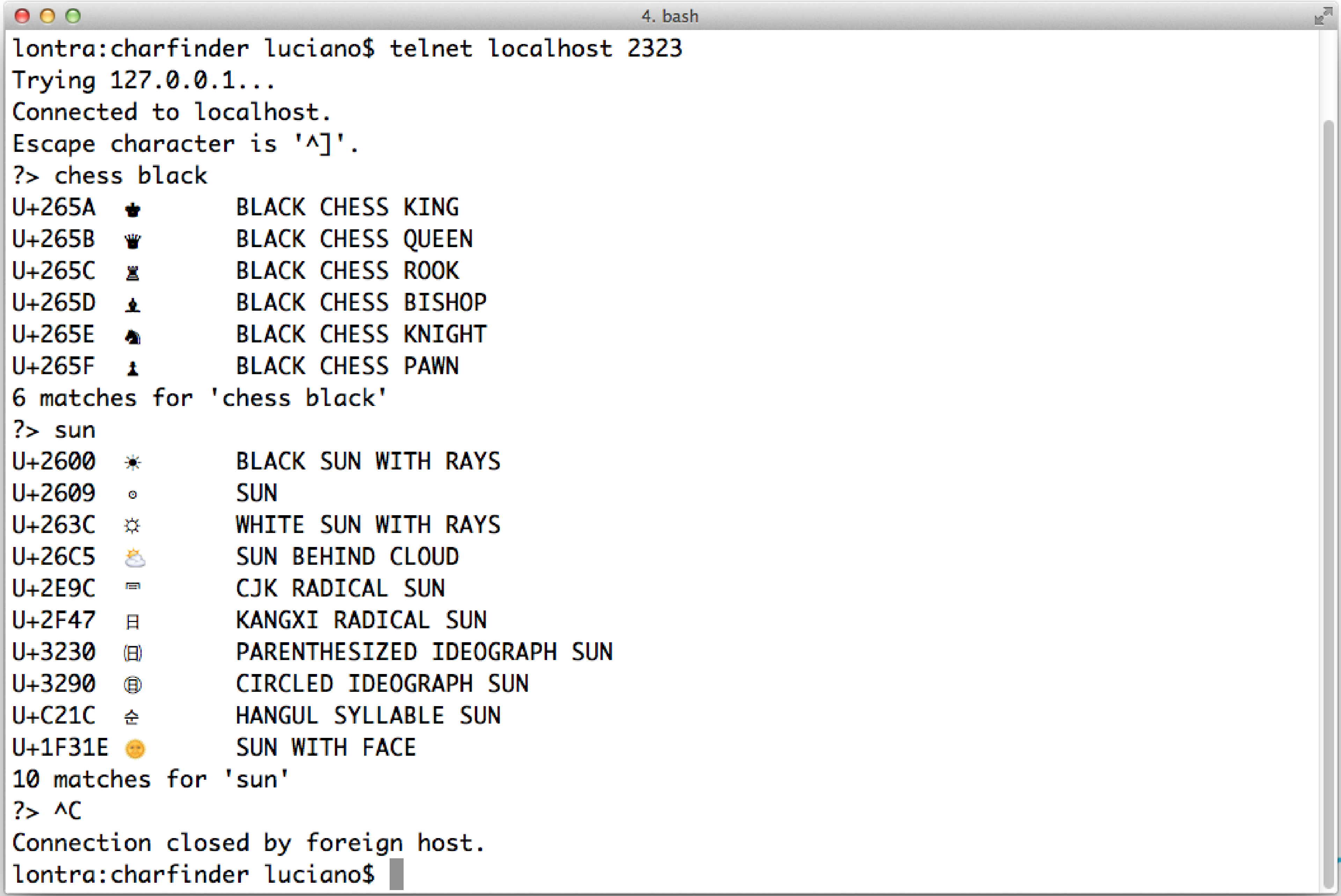
flupy-ch18/charfinder.py



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  ⺜       CJK RADICAL SUN
U+2F47  ⽇       KANGXI RADICAL SUN
U+3230  ㈰       PARENTHESIZED IDEOGRAPH SUN
U+3290  ㊐       CIRCLED IDEOGRAPH SUN
U+C21C  순       HANGUL SYLLABLE SUN
U+1F31E 🌞      SUN WITH FACE
10 matches for 'sun'
?> ^C
Connection closed by foreign host.
lontra:charfinder luciano$
```