Python, Locales and Writing Systems

Rae Knowler

Swiss Python Summit

17th February 2017

About me

CKAN, Symfony, Django
@RaeKnowler
they/their/them

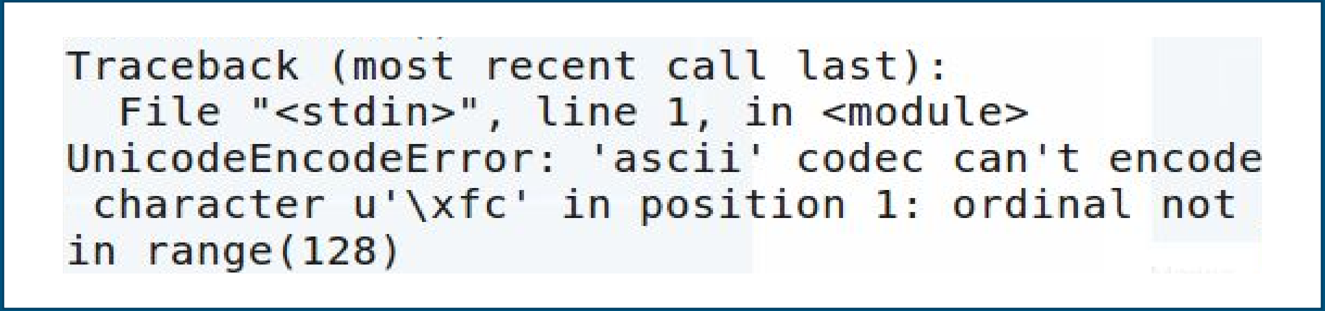
Python 3 is great

Unicode by default!

Source file encoding assumed to be UTF-8

No need to specify `u'foobar'` for non-ascii strings

Less of this:



```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode
 character u'\xfc' in position 1: ordinal not
in range(128)
```

Turkish i and ı

Ramazan Çalçoban

Emine Çalçoban

http://gizmodo.com/382026/a-cellphones-missing-dot-kills-two-people-puts-three-more-in-jail

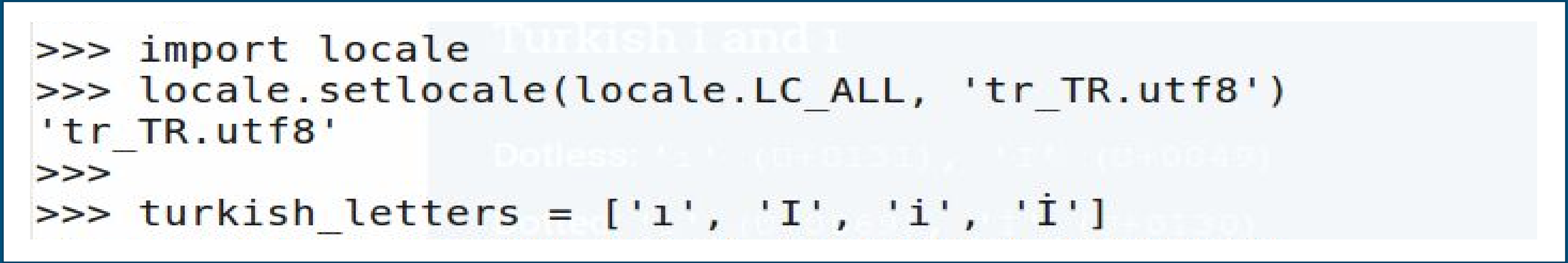
Turkish i and ı

Dotless: `'ı' (U+0131), 'I' (U+0049)`

Dotted: `'i' (U+0069), 'İ' (U+0130)`

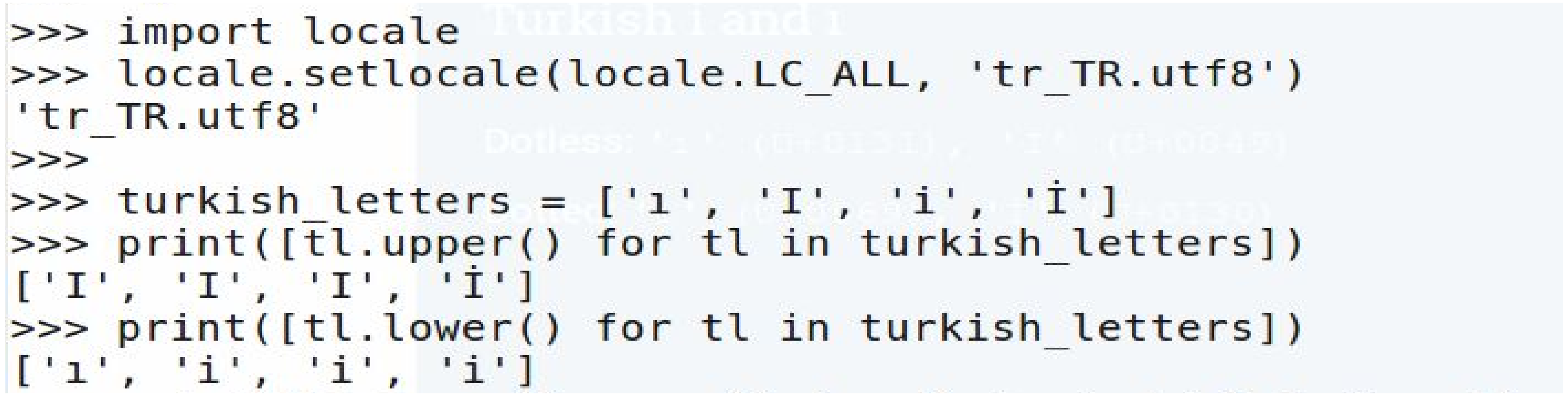
More details here:
http://www.i18nguy.com/unicode/turkish-i18n.html

Turkish i and ı



```
>>> import locale
>>> locale.setlocale(locale.LC_ALL, 'tr_TR.utf8')
'tr_TR.utf8'
>>>
>>> turkish_letters = ['ı', 'I', 'i', 'İ']
```

Turkish i and ı



```
>>> import locale
>>> locale.setlocale(locale.LC_ALL, 'tr_TR.utf8')
'tr_TR.utf8'
>>>
>>> turkish_letters = ['ı', 'I', 'i', 'İ']
>>> print([tl.upper() for tl in turkish_letters])
['I', 'I', 'I', 'İ']
>>> print([tl.lower() for tl in turkish_letters])
['ı', 'i', 'i', 'i']
```

#SPS17
@RaeKnowler

Turkish i and ı - Solutions

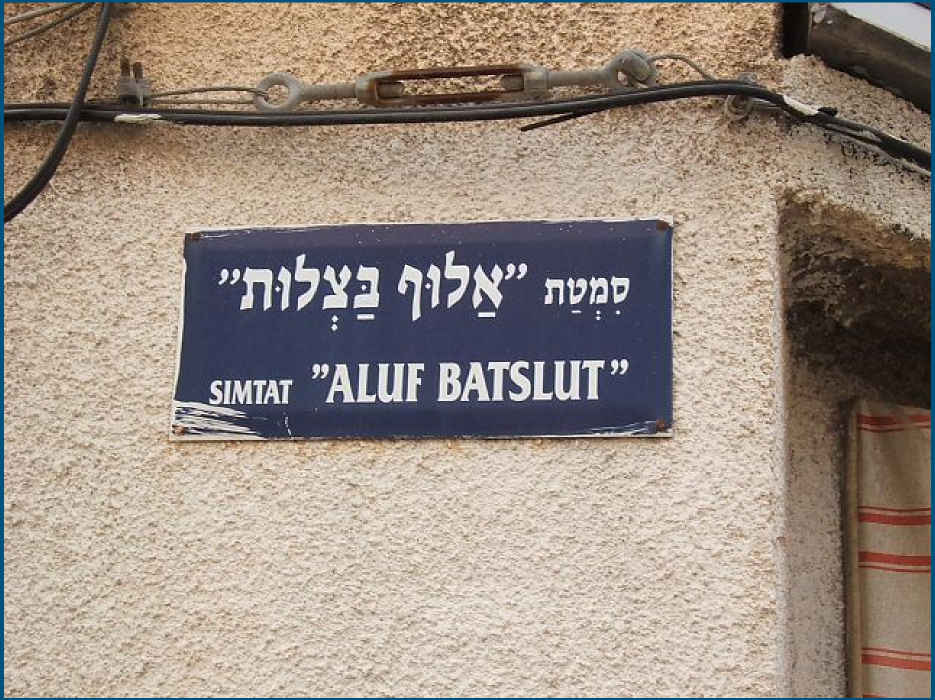
- PyICU: a Python extension wrapping IBM's International Components for Unicode C++ library (ICU).

  https://pypi.python.org/pypi/PyICU

- Or… make a translation table and use `str.translate()` to replace characters when changing the case

#SPS17
@RaeKnowler

Right-to-left writing systems



https://en.wikipedia.org/wiki/File:Simtat_Aluf_Batslut.JPG

Right-to-left writing systems

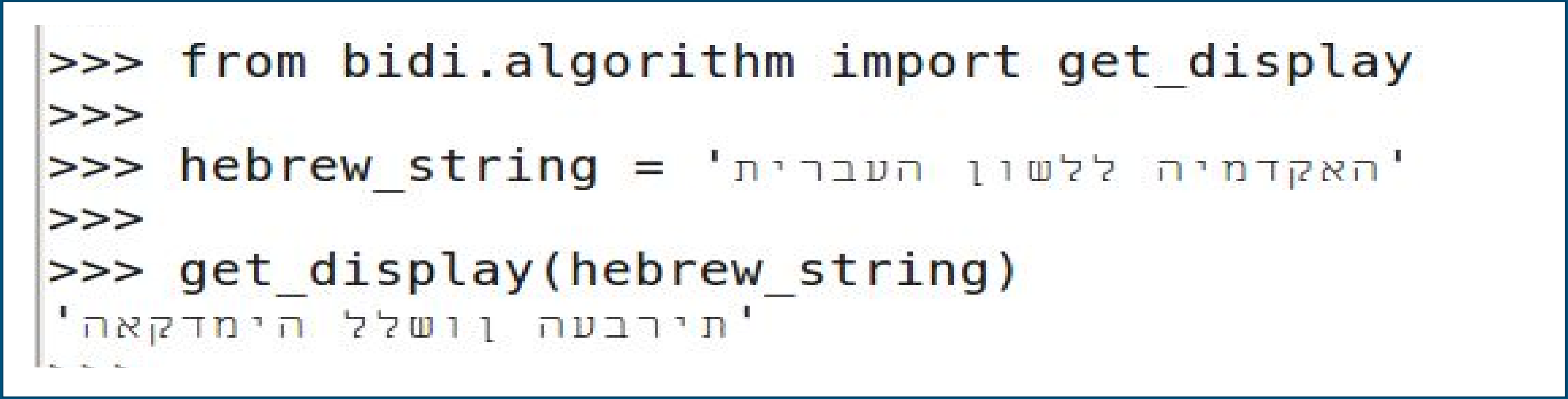
Unicode wants characters ordered **logically**, not **visually**

→ we need bidirectional (bidi) support

→ `pip install python-bidi`

#SPS17
@RaeKnowler

Right-to-left writing systems

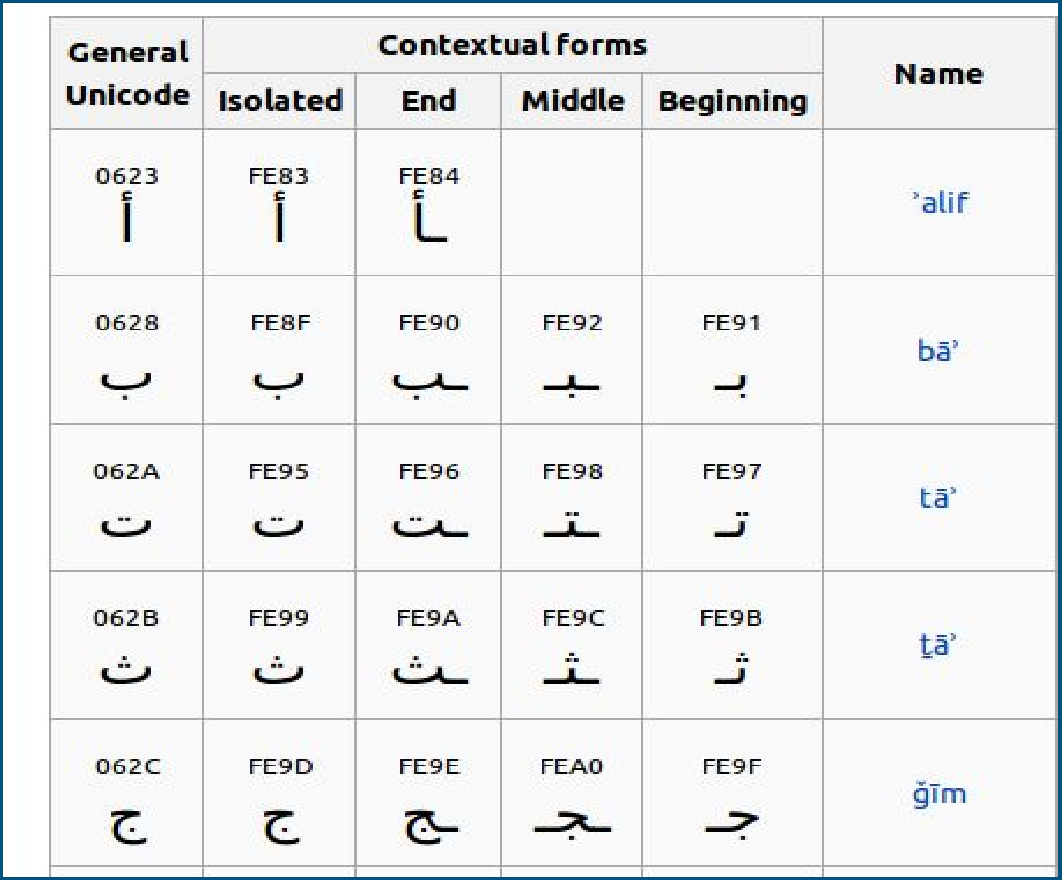


```
>>> from bidi.algorithm import get_display
>>>
>>> hebrew_string = 'האקדמיה ללשון העברית'
>>>
>>> get_display(hebrew_string)
'תירבעה ןושלל הימדקאה'
```

Right-to-left writing systems

Arabic letters have contextual forms:

Their placement in the text changes their shape.

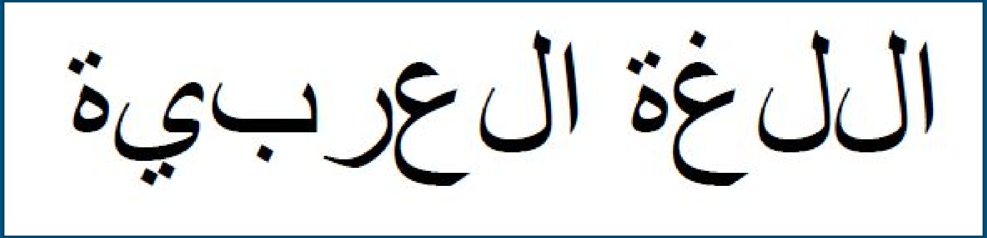


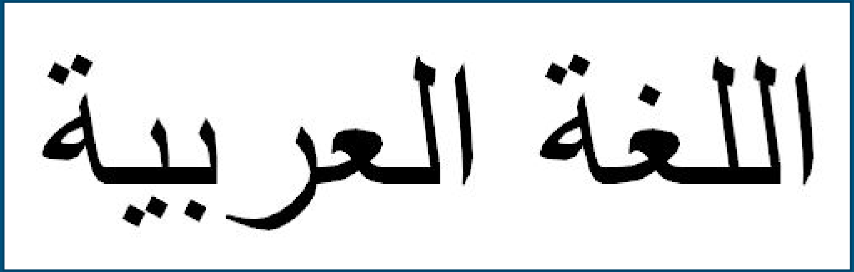
General Unicode	Contextual forms				Name
	Isolated	End	Middle	Beginning	
0623 أ	FE83 أ	FE84 ـأ			ʾalif
0628 ب	FE8F ب	FE90 ـب	FE92 ـبـ	FE91 بـ	bāʾ
062A ت	FE95 ت	FE96 ـت	FE98 ـتـ	FE97 تـ	tāʾ
062B ث	FE99 ث	FE9A ـث	FE9C ـثـ	FE9B ثـ	ṯāʾ
062C ج	FE9D ج	FE9E ـج	FEA0 ـجـ	FE9F جـ	ǧīm

https://en.wikipedia.org/wiki/Arabic_script_in_Unicode#Contextual_forms

Right-to-left writing systems

→ Python Arabic Reshaper to the rescue:
https://github.com/mpcabd/python-arabic-reshaper


ﺍﻝﻝﻍﺓ ﺍﻝﻉﺭﺏﻱﺓ


اللغة العربية

Fullwidth and halfwidth characters

Notice any difference?

Ｔｈｅ　ｑｕｉｃｋ　ｂｒｏｗｎ　ｆｏｘ　ｊｕｍｐｅｄ　ｏｖｅｒ　　ｔｈｅ　ｌａｚｙ　ｄｏｇ．

The quick brown fox jumped over the lazy dog.

Fullwidth and halfwidth characters

Courier New doesn’t even bother with the fullwidth characters.

Ｔｈｅ　ｑｕｉｃｋ　ｂｒｏｗｎ　ｆｏｘ　ｊｕｍｐｅｄ　ｏｖｅｒ　ｔｈｅ　ｌａｚｙ　ｄｏｇ．

The quick brown fox jumped over the lazy dog.

Fullwidth and halfwidth characters

假借字, 形声字

Han characters (in Chinese, Japanese, Korean) are fullwidth

Fullwidth and halfwidth characters

假借字, 形声字

ミムメモヤユヨラリルレロワン

ﾐﾑﾒﾓﾔﾕﾖﾗﾘﾙﾚﾛﾜﾝ

There are fullwidth and halfwidth kana (Japanese)

Fullwidth and halfwidth characters

假借字, 形声字

ミムメモヤユヨラリルレロワン

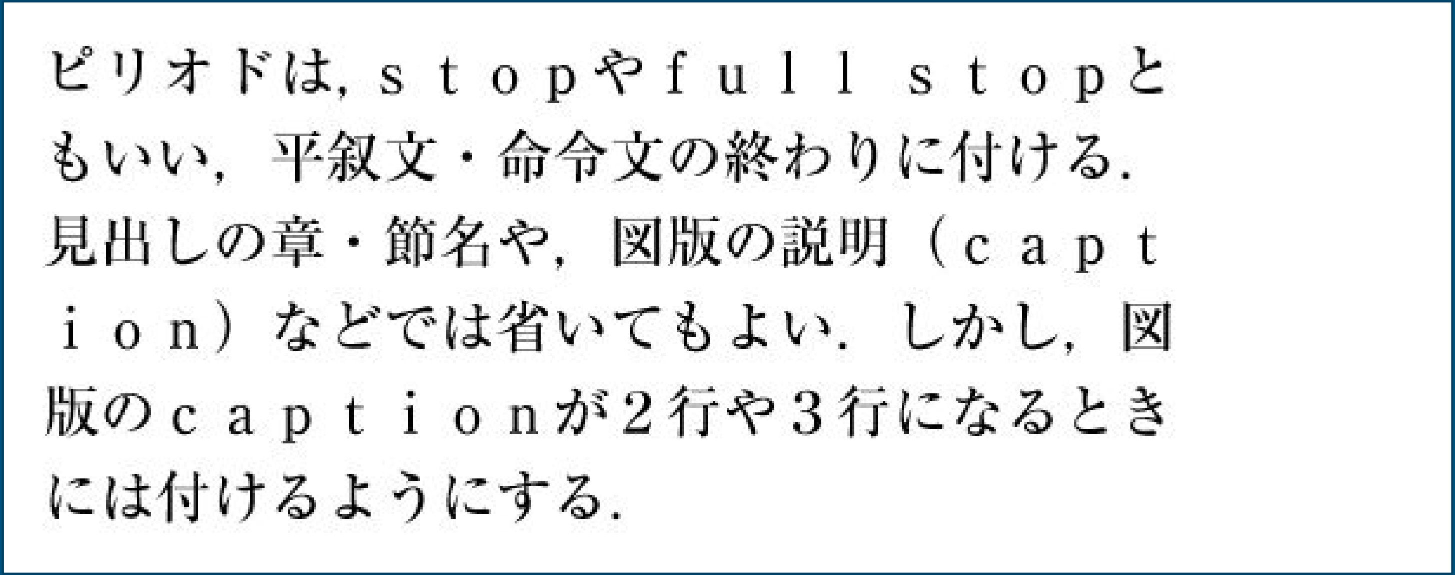
ﾐﾑﾒﾓﾔﾕﾖﾗﾘﾙﾚﾛﾜﾝ

なにぬねのは

Hiragana (Japanese) are always fullwidth

#SPS17
@RaeKnowler

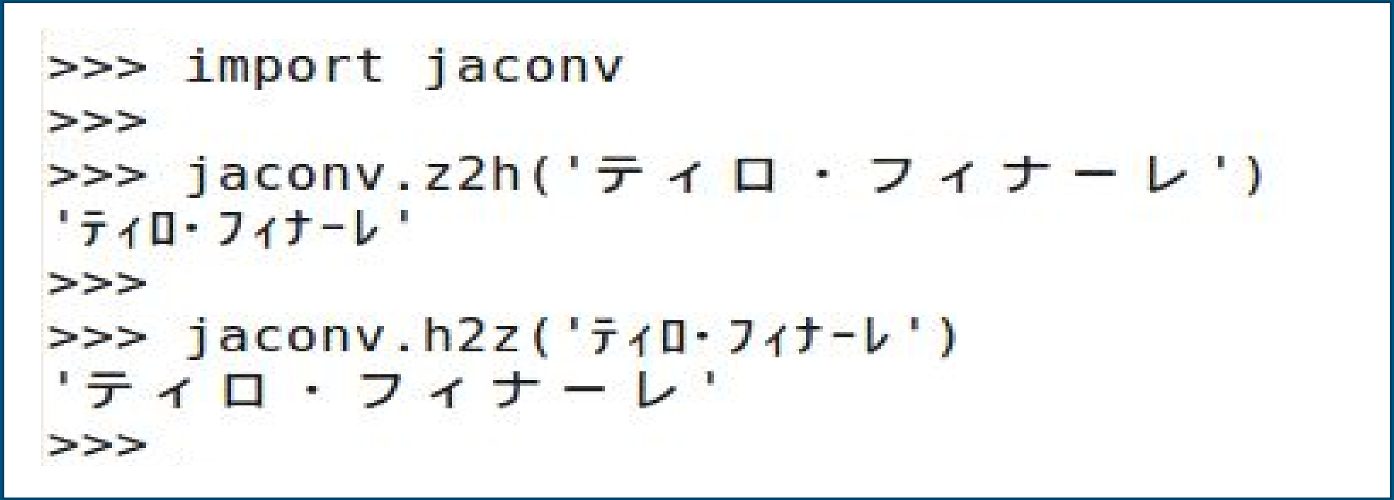
Fullwidth and halfwidth characters



ピリオドは，ｓｔｏｐやｆｕｌｌ ｓｔｏｐと
もいい，平叙文・命令文の終わりに付ける．
見出しの章・節名や，図版の説明（ｃａｐｔ
ｉｏｎ）などでは省いてもよい．しかし，図
版のｃａｐｔｉｏｎが２行や３行になるとき
には付けるようにする．

Fullwidth and halfwidth characters

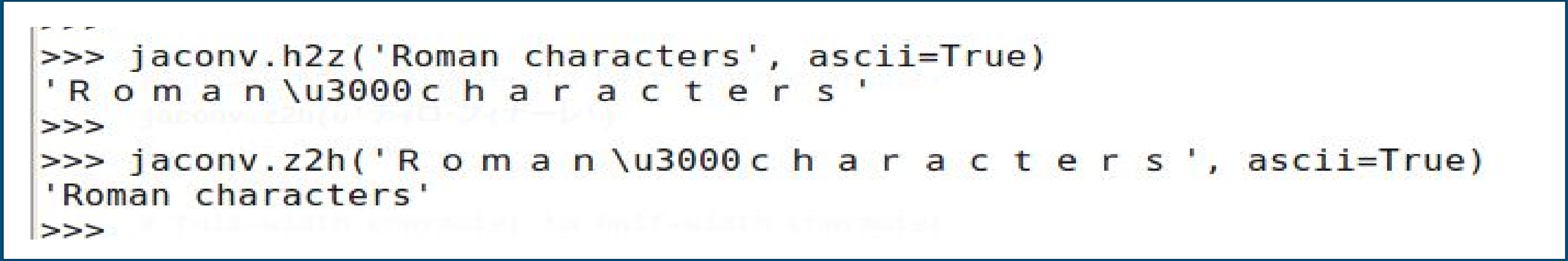
```
pip install jaconv
```



```
>>> import jaconv
>>>
>>> jaconv.z2h('ティロ・フィナーレ')
'ﾃｨﾛ･ﾌｨﾅｰﾚ'
>>>
>>> jaconv.h2z('ﾃｨﾛ･ﾌｨﾅｰﾚ')
'ティロ・フィナーレ'
>>>
```

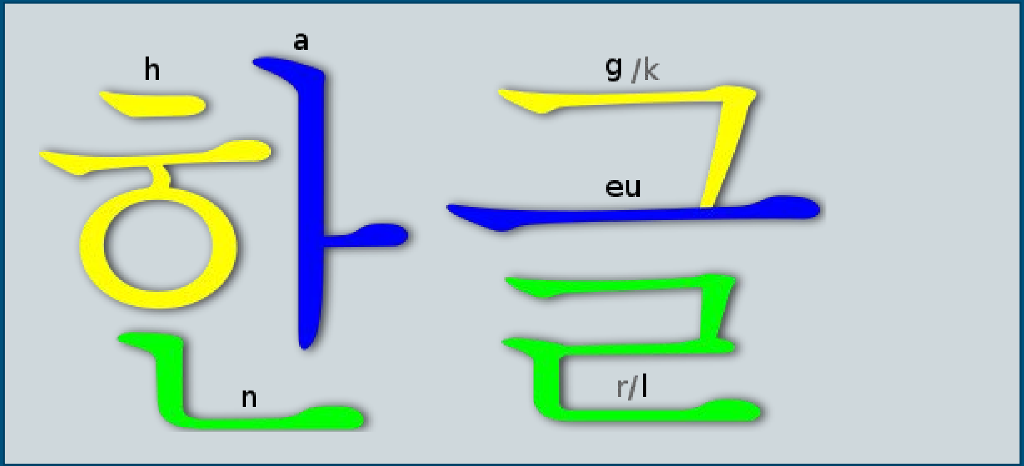
Fullwidth and halfwidth characters

```
pip install jaconv
```



```
>>> jaconv.h2z('Roman characters', ascii=True)
'Ｒｏｍａｎ\u3000ｃｈａｒａｃｔｅｒｓ'
>>>
>>> jaconv.z2h('Ｒｏｍａｎ\u3000ｃｈａｒａｃｔｅｒｓ', ascii=True)
'Roman characters'
>>>
```

Korean text



Lots more detail here:
http://www.gernot-katzers-spice-pages.com/var/korean_hangul_unicode.html

Korean text

Unicode canonical equivalence:

You can build the same character in several different ways, and they mean the same thing.

한 means the same as ㅎ ㅏ ㄴ

Korean text

Unicode canonical equivalence:

You can build the same character in several different ways, and they mean the same thing.

한 means the same as ㅎㅏㄴ

Normal Form D (NFD): ㅎㅏㄴ

Normal Form C (NFC): 한

Korean text

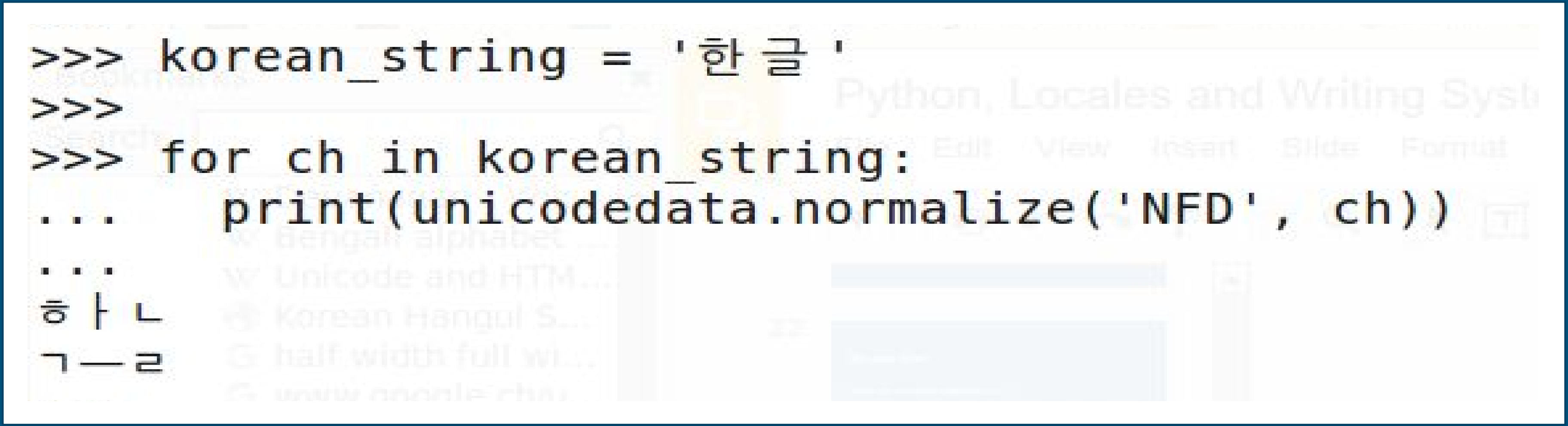
Unicode compatibility equivalence:

There are multiple code points for identical characters, for backwards compatibility reasons

U+2160 (ROMAN NUMERAL ONE) is really the same thing as U+0049 (LATIN CAPITAL LETTER I)

(https://docs.python.org/2/library/unicodedata.html)

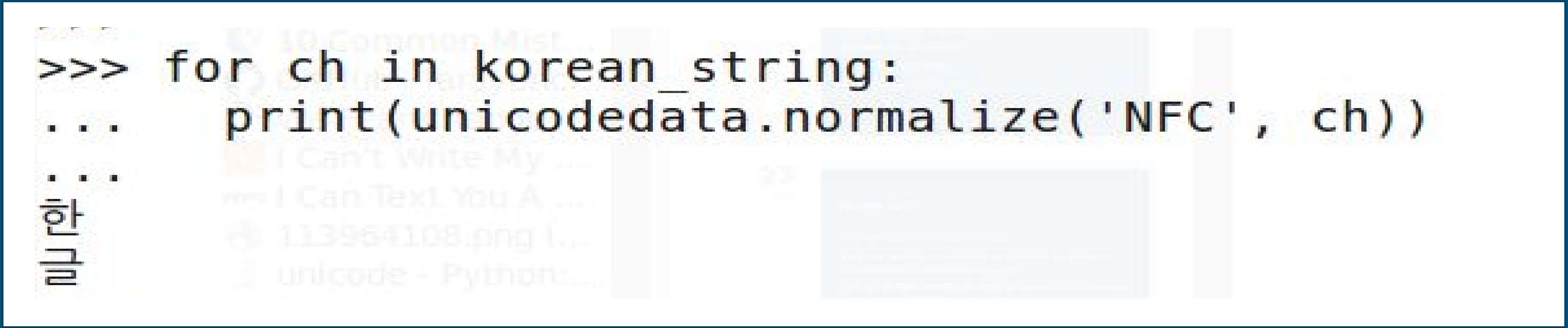
Korean text



```
>>> korean_string = '한글'
>>>
>>> for ch in korean_string:
...     print(unicodedata.normalize('NFD', ch))
...
ㅎㅏㄴ
ㄱㅡㄹ
```

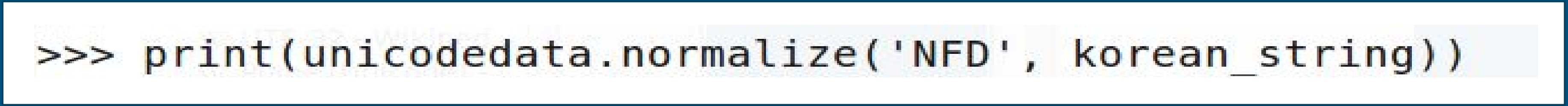
#SPS17
@RaeKnowler

Korean text



```
>>> for ch in korean_string:
...     print(unicodedata.normalize('NFC', ch))
...
한
글
```

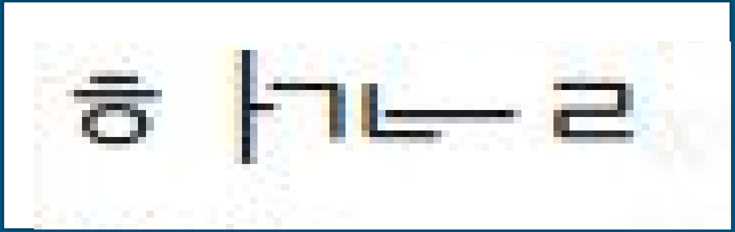
Korean text



```
>>> print(unicodedata.normalize('NFD', korean_string))
```

Korean text

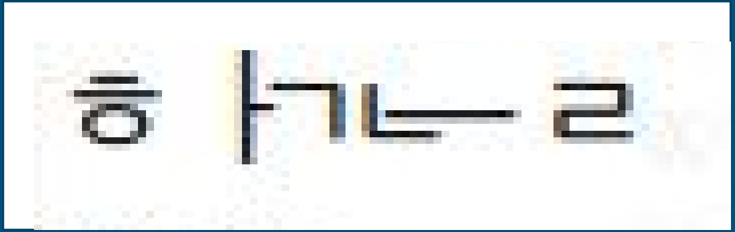
```
>>> print(unicodedata.normalize('NFD', korean_string))
```



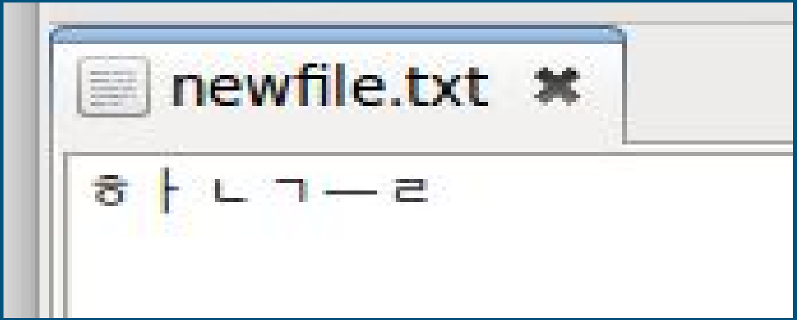
```
ㅎ ㅏㄱㄴㅡ ㄹ
```

Korean text

```
>>> print(unicodedata.normalize('NFD', korean_string))
```



ㅎ ㅏㄱㄴㅡ ㄹ



newfile.txt

ㅎ ㅏ ㄴ ㄱ ㅡ ㄹ

Security

This is a huge topic!

A couple of quick examples...

Security - SQL Injection

User input:

I don't like raisins

Sanitised user input:

```
'I don\'t like raisins'
```

Hex encoding of \ is `0x5C`

Security - SQL Injection

Hex encoding for 稞: `0xb8 0x5c`

User input:

0xb8' OR 1=1

Sanitised user input:

```
'稞 OR 1=1'
```

Security - SQL Injection

More details here:

http://howto.hackallthethings.com/2016/06/using-multi-byte-characters-to-nullify.html

Security - Address Bar Spoofing

A nice google.com link:

http://google.com/test/test/test/عربي.امارات

This actually led to:

http://عربي.امارات/google.com/test/test/test

Security - Address Bar Spoofing

More details here:
http://www.rafayhackingarticles.net/2016/08/google-chrome-firefox-address-bar.html

Conclusions

This stuff isn't easy … but it *is* interesting!

There are a lot of useful libraries out there. You won't be the first person to have your particular problem.

Python 3 makes dealing with Unicode a lot easier.

Further links

- The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!): http://www.joelonsoftware.com/articles/Unicode.html
- Dark corners of Unicode: https://eev.ee/blog/2015/09/12/dark-corners-of-unicode
- I Can Text You A Pile of Poo, But I Can't Write My Name: https://modelviewculture.com/pieces/i-can-text-you-a-pile-of-poo-but-i-cant-write-my-name
- Nope, Not Arabic: http://nopenotarabic.tumblr.com/