Python programming — encoding characters

Finn Årup Nielsen

DTU Compute
Technical University of Denmark

September 8, 2014

https://gist.github.com/3667575

Character encoding

Handling of characters encoding is a recurring annoying problem.

Python2 may uses “ordinary string” and Unicode formats and may read and write in ”other” formats, such as UTF-8.

Suggestions/recommendations:

- Write the script in UTF-8 (if non-ASCII is necessary).
- Never assume that a user-provided file is in the “correct” encoding.
- Write in UTF-8 unless the user wants it differently.

Encoding in Python 2

With a UTF-8 terminal:

```
$ python2.6
Python 2.6.6 (r266:84292, Sep 15 2010, 15:52:39)
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 'Finn Årup Nielsen'
'Finn \xc3\x85rup Nielsen'
>>> u'Finn Årup Nielsen'
u'Finn \xc5rup Nielsen'
>>> unicode('Finn Årup Nielsen', 'utf-8')
u'Finn \xc5rup Nielsen'
```

In the last 2 cases you have the correct unicode string in the variable. In the first case it is still a string with UTF-8 bytes (a bytestring).

So what is wrong with that?

```
>>> len(u'Finn Arup Nielsen')
17
>>> len('Finn Årup Nielsen')
18
>>> len(u'Finn Årup Nielsen')
17
```

'`len`' counts bytes in the "ordinary" string.

Another example of a problem: Finding words with regular expressions:

```
>>> re.findall('\w+', 'Finn Årup Nielsen')
['Finn', 'rup', 'Nielsen']
```

Agh, missed the "Å"! The fix is to convert to Unicode and use `re.UNICODE`

```
>>> re.findall('\w+', u'Finn Årup Nielsen', re.UNICODE)
[u'Finn', u'\xc5rup', u'Nielsen']
```

Surprises with encodings

```
>>> person = {'given_name': 'Finn', 'middle_name': u'Årup'}
>>> person
{'middle_name': u'\xc5rup', 'given_name': 'Finn'}
>>> "{middle_name}".format(**person)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode character u'\xc5'
in position 0: ordinal not in range(128)
```

What!?

Surprises with encodings

```
>>> person = {'given_name': 'Finn', 'middle_name': u'Årup'}
>>> person
{'middle_name': u'\xc5rup', 'given_name': 'Finn'}
>>> "{middle_name}".format(**person)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode character u'\xc5'
in position 0: ordinal not in range(128)
```

What!?

Make the first string a unicode string:

```
>>> u"{middle_name}".format(**person)
u'\xc5rup'
```

From the future: `unicode_literals`

```
from __future__ import unicode_literals
```

will make strings be interpreted as unicode

```
>>> person = {'given_name': 'Finn', 'middle_name': 'Årup'}
>>> person
{u'middle_name': u'\xc5rup', u'given_name': u'Finn'}
>>> "{middle_name}".format(**person)
u'\xc5rup'
```

Note that we do not have any 'u' prefix in front of strings to say it is a Unicode string.

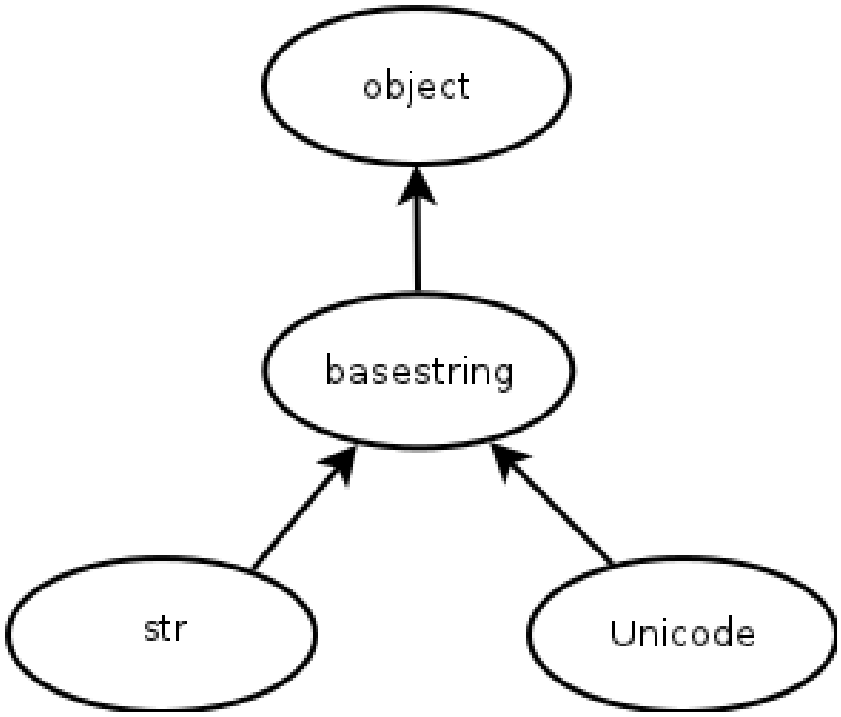
Python 2 recommendation

Convert strings from I/O as quickly as possible to Unicode.

Python 2 string hierarchy

```
>>> isinstance("I'm an ordinary string", basestring)
True
>>> isinstance(u"I'm a UNICODE", basestring)
True
>>> isinstance(u"I'm a UNICODE", str)
False
```

The basestring class is the parent of str (ordinary string) and unicode string in Python 2.



Encoding in Python 3

Python version 3 “fixes” the problem with default representation in Unicode

```
$ python3
Python 3.1.2 (release31-maint, Sep 17 2010, 20:34:23)
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 'Finn Årup Nielsen'
'Finn Årup Nielsen'
>>> u'Finn Årup Nielsen'
  File "<stdin>", line 1
    u'Finn Årup Nielsen'
                       ^
SyntaxError: invalid syntax
```

Encoding from Unicode

Encode a Unicode string to, e.g., “ascii”, “latin2”, “iso-8859-1”, . . .

```
>>> u'Rådvad Æblerød'.encode('utf-8')
'R\xc3\xa5dvad \xc3\x86bler\xc3\xb8d'

>>> u'Rådvad Æblerød'.encode('ascii')
UnicodeEncodeError: 'ascii' codec can't encode character u'\xe5' ...

>>> u'Rådvad Æblerød'.encode('ascii', 'ignore')
'Rdvad blerd'

>>> u'Rådvad Æblerød'.encode('ascii', 'replace')
'R?dvad ?bler?d'

>>> u'Rådvad Æblerød'.encode('ascii', 'xmlcharrefreplace')
'R&#229;dvad &#198;bler&#248;d'
```

Construction of a UTF-8 test file

Write 14-characters “Rådvad Æbelrød” file:

```
f = open('text-with-utf-8.txt', 'wb')
f.write('R\xc3\xa5dvad \xc3\x86belr\xc3\xb8d')
f.close()
```

View the UTF-8 file that contains 17 bytes:

```
$ hexdump -C text-with-utf-8.txt
00000000  52 c3 a5 64 76 61 64 20  c3 86 62 65 6c 72 c3 b8  |R..dvad ..belr..|
00000010  64                                                |d|
```

File I/O with Python 2

Naïve Python 2 reading results in too many characters:

```
$ python2.6
>>> print(len(open('text-with-utf-8.txt').read()))
17
```

Call the `unicode` function to turn the UTF-8 string into Unicode:

```
$ python2.6
>>> print(len(unicode(open('text-with-utf-8.txt').read(), 'utf-8')))
14
```

```
>>> import codecs
>>> print(len(codecs.open('text-with-utf-8.txt', encoding='utf-8').read()))
14
```

File I/O with Python 2 default encoding

```
$ python2.6
>>> print(len(open('text-with-utf-8.txt').read()))
17
>>> print(len(unicode(open('text-with-utf-8.txt').read())))
UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 in ...
```

Use "sys reload"-trick:

```
$ python2.6
>>> import sys
>>> reload(sys)
>>> sys.setdefaultencoding('utf-8')
>>> print(len(open('text-with-utf-8.txt').read()))
17
>>> print(len(unicode(open('text-with-utf-8.txt').read())))
14
```

No explicit mentioning of encoding

File I/O with Python 3

Python 3 reading with UTF-8 environment

```
$ LANG=en_US.utf8 ; python3
>>> print(len(open('text-with-utf-8.txt').read()))
14
>>> print(len(open('text-with-utf-8.txt', encoding='utf-8').read()))
14
```

Python 3 reading with non-UTF-8 environment

```
$ LANG=C; python3
>>> print(len(open('text-with-utf-8.txt').read()))
UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 ...
>>> print(len(open('text-with-utf-8.txt', encoding='utf-8').read()))
14
```

More Python 3 issues

With non-UTF-8 environment: "`$ LANG=C; python3`"

```
for enc in ['ascii', 'ISO8859-1', 'latin1', 'utf-8']:
  try:
    s = open('text-with-utf-8.txt', encoding=enc).read()
  except UnicodeDecodeError:
    continue
  print("Read with encoding =", enc)
  break
```

Erroneous success reporting "`Read with encoding = ISO8859-1`"

```
>>> print(s)
```

gives you a `UnicodeEncodeError` exception: Call `encode` with replace:

```
>>> print(s.encode('ascii', 'replace'))
b'R??dvad ??belr??d'
```

Encoding in source code

Python 2 script with UTF-8 encoding

```
#!/usr/bin/python2.6
# -*- coding: utf-8 -*-
print("Rådvad Knivfabrik")
```

Without the "`# -*- coding: utf-8 -*-`" line you will get an error: "SyntaxError: Non-ASCII character"

In Python 3 script you don't need the "coding" line as the script is assumed to be in UTF-8:

```
#!/usr/bin/python3
print("Rådvad Knivfabrik")
```

Python 3 variables

You can have variables with non-ASCII characters:

```
>>> Æ = 3
>>> A = 1
>>> A + Æ
4
```

But this is not necessarily a good idea (i.e., it is a bad idea!).

Character encoding on the Web

Issues that are not specific to Python: In the header set:

```
Content-type: text/plain; charset=utf-8
```

Or/also set the meta-field in the `<head>`

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

In the Python web script you can then write out in UTF-8:

```
print(u'Rådvad Æblerød'.encode('utf-8'))
```

A bit more information

Unicode HOWTO in Python documentation

http://diveintopython.org/xml_processing/unicode.html

(Pilgrim, 2004, pages 125+)

http://nltk.googlecode.com/svn/trunk/doc/book/ch03.html

(Bird et al., 2009, pages 93+, section 3.3)

Kumar McMillan's talk Unicode In Python, Completely Demystified from PyCon 2008.

Conclusion

Expect headache

Conclusion

But

Conclusion

Do not ignore encoding issues, e.g., `s.decode('ascii', errors='ignore')` as people have died for not handling a difference such as between

"Zaten sen sıkışınca konuyu değiştiriyorsun."

and

"Zaten sen sikişınce konuyu değiştiriyorsun."

See Two Dots Too Many and Küçücük bir nokta tam 5 kişiyi yaktı. (Thanks to Vladimir Keleshev for the link)

References

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly, Sebastopol, California. ISBN 9780596516499.

Pilgrim, M. (2004). *Dive into Python*.