Python UnicodeEncodeError: 'ascii' codec can't encode character

Date: 2008 (/blog/2008/)-11-06 | Modified: 2012-06-19 | Tags: django (/blog/tag/django/), error (/blog/tag/error/), python (/blog/tag/python/), wide (/blog/tag/wide/) | 31 Legacy Comments (/blog/2008/11/python-unicodeencodeerror-ascii-codec-cant-encode-character/#comments) | (/blog/2008/11/python-unicodeencodeerror-ascii-codec-cant-encode-character/#disqus_thread)

```
UnicodeEncodeError: 'ascii' codec can't encode character u'\xa1'
in position 0: ordinal not in range(128)
```

If you've ever gotten this error, Django's `smart_str` function might be able to help. I found this from James Bennett's article, Unicode in the real world (http://www.b-list.org/weblog/2007/nov/10/unicode/). He provides a very good explanation of Python's Unicode and bytestrings, their use in Django, and using Django's Unicode utilities for working with non-Unicode-friendly Python libraries. Here are my notes from his article as it applies to the above error. Much of the wording is directly from James Bennett's article.

This error occurs when you pass a Unicode string containing non-English characters (Unicode characters beyond 128) to something that expects an ASCII bytestring. The default encoding for a Python bytestring is ASCII, "which handles exactly 128 (English) characters". This is why trying to convert Unicode characters beyond 128 produces the error.

The good news is that you can encode Python bytestrings in other encodings besides ASCII. Django's `smart_str` function in the `django.utils.encoding` module, converts a Unicode string to a bytestring using a default encoding of UTF-8.

Here is an example using the built-in function, `str`:

```
a = u'\xa1'
print str(a) # this throws an exception
```

Results:

```
Traceback (most recent call last):
  File "unicode_ex.py", line 3, in
    print str(a) # this throws an exception
UnicodeEncodeError: 'ascii' codec can't encode character u'\xa1' in position 0:
```

Here is an example using `smart_str`:

```
from django.utils.encoding import smart_str, smart_unicode

a = u'\xa1'
print smart_str(a)
```

Results:

```
¡
```

Definitions

- Unicode string: sequence of Unicode characters
- Python bytestring: a series of bytes which represent a sequence of characters. It's default encoding is ASCII. This is the "normal", non-Unicode string in Python <3.0.
- encoding: a code that pairs a sequence of characters with a series of bytes
- ASCII: an encoding which handles 128 English characters
- UTF-8: a popular encoding used for Unicode strings which is backwards compatible with ASCII for the first 128 characters. It uses one to four bytes for each character.

Operations related to str and unicode objects

- `unicode.encode()` (http://docs.python.org/library/stdtypes.html#str.encode) - converts to str
- `str.decode()` (http://docs.python.org/library/stdtypes.html#str.decode) - converts to unicode
- `unicode(str, encoding)` (http://docs.python.org/library/functions.html#unicode) - converts to unicode
- `ord(c)` (http://docs.python.org/library/functions.html#ord) - returns the Unicode code point of the character
- `chr(i)` (http://docs.python.org/library/functions.html#chr) - returns a str object for the given ASCII code (inverse of ord() for 8-bit strings)

- `unichr(i)` (http://docs.python.org/library/functions.html#unichr) - returns a unicode object for the given Unicode code (inverse of ord() for Unicode strings)

Table of operations on str types (Python 2.7)

```
| x ->                      | 'i'              | '¡'              | '\xa1'
|---------------------------+------------------+------------------+------------
| type(x)                   | <type 'str'>     | <type 'str'>     | <type 'str'>
| ord(x)                    | 105              | NA               | 161
| type(str(x))              | <type 'str'>     | <type 'str'>     | <type 'str'>
| type(unicode(x))          | <type 'unicode'> | DecodeError      | DecodeError
| type(unicode(x, 'utf-8')) | <type 'unicode'> | <type 'unicode'> | DecodeError
| type(unicode(x, 'ascii')) | <type 'unicode'> | DecodeError      | DecodeError
| type(x.decode('utf-8'))   | <type 'unicode'> | <type 'unicode'> | DecodeError
| type(x.encode('utf-8'))   | <type 'str'>     | DecodeError      | DecodeError
| type(x.decode('ascii'))   | <type 'unicode'> | DecodeError      | DecodeError
| type(x.encode('ascii'))   | <type 'str'>     | DecodeError      | DecodeError
```

Table of operations on unicode types (Python 2.7)

```
| x ->                      | u'i'             | u'¡'             | u'\xa1'
|---------------------------+------------------+------------------+------------
| type(x)                   | <type 'unicode'> | <type 'unicode'> | <type 'unic
| ord(x)                    | 105              | 161              | 161
| type(str(x))              | <type 'str'>     | EncodeError      | EncodeError
| type(unicode(x))          | <type 'unicode'> | <type 'unicode'> | <type 'unic
| type(unicode(x, 'utf-8')) | <not supported>  | <not supported>  | <not suppor
| type(unicode(x, 'ascii')) | <not supported>  | <not supported>  | <not suppor
| type(x.decode('utf-8'))   | <type 'unicode'> | EncodeError      | EncodeError
| type(x.encode('utf-8'))   | <type 'str'>     | <type 'str'>     | <type 'str'>
| type(x.decode('ascii'))   | <type 'unicode'> | EncodeError      | EncodeError
| type(x.encode('ascii'))   | <type 'str'>     | EncodeError      | EncodeError
```

Unicode unit tests (Python 2.7)

```
import io
import os.path
import shutil
import tempfile
import unittest


class UnicodeTestCase(unittest.TestCase):
    codepoint105_as_unicode = unichr(105)
    codepoint105_as_bytestring = 'i'
    codepoint105_as_bytestring_ascii = unichr(105).encode('ascii')
    codepoint105_as_bytestring_utf8 = unichr(105).encode('utf-8')
    codepoint105_as_bytestring_latin1 = unichr(105).encode('latin-1')
    codepoint105_as_bytestring_cp950 = unichr(105).encode('cp950')

    codepoint161_as_unicode = unichr(161)
    codepoint161_as_bytestring_utf8 = unichr(161).encode('utf-8')
    codepoint161_as_bytestring_latin1 = unichr(161).encode('latin-1')

    def setUp(self):
        self.tempdir = tempfile.mkdtemp(prefix='tmp-ditest-')
        self.codepoint105_ascii_filepath = os.path.join(self.tempdir, 'codepoint
        self.codepoint105_utf8_filepath = os.path.join(self.tempdir, 'codepoint1
        self.codepoint105_latin1_filepath = os.path.join(self.tempdir, 'codepoi
        self.codepoint161_ascii_filepath = 'codepoint 161 cannot be encoded usi
        self.codepoint161_utf8_filepath = os.path.join(self.tempdir, 'codepoint1
        self.codepoint161_latin1_filepath = os.path.join(self.tempdir, 'codepoi

        with io.open(self.codepoint105_ascii_filepath, 'w', encoding='ascii') as
            f.write(self.codepoint105_as_unicode)
        with io.open(self.codepoint105_utf8_filepath, 'w', encoding='utf8') as f
            f.write(self.codepoint105_as_unicode)
        with io.open(self.codepoint105_latin1_filepath, 'w', encoding='latin1')
            f.write(self.codepoint105_as_unicode)

        with io.open(self.codepoint161_utf8_filepath, 'w', encoding='utf8') as f
            f.write(self.codepoint161_as_unicode)
        with io.open(self.codepoint161_latin1_filepath, 'w', encoding='latin1')
            f.write(self.codepoint161_as_unicode)
```

```
    def tearDown(self):
        shutil.rmtree(self.tempdir)

    def test_encoding_decoding_latin1_utf8(self):
        self.assertEqual(u'\xa1', unichr(161))
        self.assertEqual(
            unichr(161).encode('utf-8').decode('utf-8'),
            unichr(161))
        self.assertEqual(
            unichr(161).encode('latin-1').decode('latin-1'),
            unichr(161))
        self.assertNotEqual(
            unichr(161).encode('utf-8').decode('latin-1'),
            unichr(161))
        with self.assertRaises(UnicodeDecodeError):
            unichr(161).encode('latin-1').decode('utf-8'),

    def test_bif_open_read(self):
        with open(self.codepoint161_utf8_filepath) as f:
            text = f.read()
            self.assertEqual(text, self.codepoint161_as_bytestring_utf8)
            self.assertEqual(type(text), type(self.codepoint161_as_bytestring_ut
        with open(self.codepoint161_latin1_filepath) as f:
            text = f.read()
            self.assertEqual(text, self.codepoint161_as_bytestring_latin1)
            self.assertEqual(type(text), type(self.codepoint161_as_bytestring_la

    def test_io_open_utf8_read(self):
        with io.open(self.codepoint161_utf8_filepath, encoding='utf-8') as f:
            text = f.read()
            self.assertEqual(text, self.codepoint161_as_unicode)
            self.assertEqual(type(text), type(self.codepoint161_as_unicode))
        with io.open(self.codepoint161_latin1_filepath, encoding='utf-8') as f:
            with self.assertRaises(UnicodeDecodeError):
                f.read()

    def test_io_open_latin1_read(self):
        with io.open(self.codepoint161_utf8_filepath, encoding='latin-1') as f:
            text = f.read()
            self.assertNotEqual(text, self.codepoint161_as_unicode)
        with io.open(self.codepoint161_latin1_filepath, encoding='latin-1') as
```

```
            text = f.read()
            self.assertEqual(text, self.codepoint161_as_unicode)
            self.assertEqual(type(text), type(self.codepoint161_as_unicode))

    def test_bif_open_write(self):
        with open('test.txt', 'w') as f:
            f.write(self.codepoint105_as_bytestring)
            f.write(self.codepoint105_as_unicode)
            f.write(self.codepoint161_as_bytestring_utf8)
            f.write(self.codepoint161_as_bytestring_latin1)
            with self.assertRaises(UnicodeEncodeError):
                f.write(self.codepoint161_as_unicode)

    def test_io_open_write(self):
        with io.open('test.txt', 'w') as f:
            f.write(self.codepoint105_as_unicode)
            f.write(self.codepoint161_as_unicode)

            with self.assertRaises(TypeError):
                f.write(self.codepoint105_as_bytestring)
            with self.assertRaises(TypeError):
                f.write(self.codepoint161_as_bytestring_utf8)
            with self.assertRaises(TypeError):
                f.write(self.codepoint161_as_bytestring_latin1)

    def test_io_open_utf8_write(self):
        with io.open('test.txt', 'w', encoding='utf-8') as f:
            f.write(self.codepoint105_as_unicode)
            f.write(self.codepoint161_as_unicode)

            with self.assertRaises(TypeError):
                f.write(self.codepoint105_as_bytestring)
            with self.assertRaises(TypeError):
                f.write(self.codepoint161_as_bytestring_utf8)
            with self.assertRaises(TypeError):
                f.write(self.codepoint161_as_bytestring_latin1)

    def test_io_open_latin1_write(self):
        with io.open('test.txt', 'w', encoding='latin-1') as f:
            f.write(self.codepoint105_as_unicode)
            f.write(self.codepoint161_as_unicode)
```

```
        with self.assertRaises(TypeError):
            f.write(self.codepoint105_as_bytestring)
        with self.assertRaises(TypeError):
            f.write(self.codepoint161_as_bytestring_utf8)
        with self.assertRaises(TypeError):
            f.write(self.codepoint161_as_bytestring_latin1)
```

References / See Also

- James Bennett's Unicode in the real world (2007) (http://www.b-list.org/weblog/2007/nov/10/unicode/)
- Django Unicode documentation (http://docs.djangoproject.com/en/dev/ref/unicode/)
- Joel Spolsky's “The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)” (2003) (http://www.joelonsoftware.com/articles/Unicode.html)
- Wikipedia on Unicode (http://en.wikipedia.org/wiki/Unicode), UTF-8 (http://en.wikipedia.org/wiki/UTF-8), ASCII (http://en.wikipedia.org/wiki/Ascii), character encoding (http://en.wikipedia.org/wiki/Character_encoding).
- Python Unicode HOWTO (http://docs.python.org/howto/unicode.html)
- Making Sense of Python Unicode (2009) (http://lobstertech.com/python_unicode.html)
- Unicode for dummies — Encoding (2012) (http://pythonconquerstheuniverse.wordpress.com/2012/02/01/unicode-for-dummies-encoding/)
- Overcoming frustration: Correctly using unicode in python2 (http://packages.python.org/kitchen/unicode-frustrations.html)
- Ned Batchelder: Pragmatic Unicode (2012) (http://nedbatchelder.com/text/unipain.html)

Comments

#1 Arthur Buliva commented on 2008-11-20:

A simpler way to do this is:

print unicode(u'\xa1').encode("utf-8")

#2 Eliot (http://www.saltycrane.com/blog/) commented on 2008-11-21:

Arthur, thanks for the tip. I'm not sure what differences the Django utility functions have. I will have to look into this further. For other readers, here is the documentation for `encode`:
http://docs.python.org/library/stdtypes.html#str.encode
(http://docs.python.org/library/stdtypes.html#str.encode)

#3 enoola commented on 2009-02-07:

Hi mates, I wanted to print a string with chinese I found that simple and usefull : cf :
http://members.shaw.ca/akochoi-old/blog/2005/10-02/index.html

```
# Simple unicode string
y = unicode(' 麻  婆 豆 腐', 'utf-8')


# Problem with this simple call..
# print y
# UnicodeEncodeError: 'ascii' codec can't encode character u'\u9ebb' in positi
on 1: ordinal not in range(128)

# Solution
print y.encode('utf8')
```

#4 Lukas Monk commented on 2009-02-27:

I use this in my main unit :

```
reload(sys)
sys.setdefaultencoding( "latin-1" )

a = u'\xa1'
print str(a) # no exception
```

#5 Low Kian Seong (http://lowkster.blogspot.com/) commented on 2009-03-19:

Thanks for this man. Now my edit page does not bomb out anymore.

#6 Barbara (http://www.djangrrl.com/) commented on 2009-05-27:

Thanks so much for this - it's exactly what I needed, at exactly the right moment. :)

#7 Gregory Saxton (http://newmediaandcapitalmarkets.org/) commented on 2009-09-06:

Thanks--exactly what I needed.

#8 Steve commented on 2009-11-02:

Cheers - really handy!

#9 William commented on 2009-11-10:

Python for windows do not have the attribute setdefaultencoding: What can I do to display utf-8 characters in text mode?

#10 Eliot (http://www.saltycrane.com/blog/) commented on 2009-11-10:

William,
It seems like `sys.setdefaultencoding` is not designed for us to use. From the sys module documentaton (http://docs.python.org/library/sys.html#sys.setdefaultencoding):

> *This function is only intended to be used by the site module implementation and, where needed, by sitecustomize. Once used by the site module, it is removed from the sys module's namespace.*

If you're not using Django, using the encode (http://docs.python.org/library/stdtypes.html#str.encode) string method (described by Arthur and enoola above) seems good. From the Django source code for smart_str (http://code.djangoproject.com/browser/django/tags/releases/1.1.1/django/utils/encoding.py#L95), it looks like `smart_str` uses `encode` with some other logic that you may or may not need.

#11 Klaus commented on 2009-11-17:

the tip of Lukas Monk works perfect. As well on MS Windows.

#12 Nikolai (http://www.schneevonmorgen.com/) commented on 2009-11-27:

saved my day! thx!

#13 Tacyt (http://www.mytwstats.com/) commented on 2009-12-17:

Thank you for this advice!

#14 chyro commented on 2010-02-01:

Same here, I found Lukas Monk's tip most useful as I don't want to use the "encode" function on every single string. I'm very puzzled about the "reload(sys)" part though. Why is the "setdefaultencoding" function not present until the module is reloaded? How come it makes any difference?

#15 chyro commented on 2010-02-01:

Sorry about the double post, it seems I'll answer my own question. I found more information on that function here: http://blog.ianbicking.org/illusive-setdefaultencoding.html (http://blog.ianbicking.org/illusive-setdefaultencoding.html) It mostly raises the same question I did (in more details obviously). The real answers come in the second comment: http://blog.ianbicking.org/illusive-setdefaultencoding-comment-2.html (http://blog.ianbicking.org/illusive-setdefaultencoding-comment-2.html) That would kind of explain it. I'd still prefer everything being UTF-8.

#16 Eliot (http://www.saltycrane.com/blog/) commented on 2010-02-02:

chyro, Thanks for adding this information. Also, I changed your plain-text URLs into clickable links.

#17 Wayle Chen commented on 2010-03-17:

@Lukas: Thanks, your tips works for me.

#18 PhilGo20 (http://philgo20.com/) commented on 2010-04-02:

You saved me some time ..again. Thanks

I would add that if one is using DOM to output to file (dom.toxml() or dom.toprettyxml()), make sure to add "encoding="utf-8"" parameters or you will also generate the same type of errors.

```
UnicodeEncodeError: 'ascii' codec can't encode character
```

#19 Loe Spee (http://twitter.com/lgespee) commented on 2010-05-05:

If you get this error when serializing data to JSON, it might be caused by the "ensure_ascii=False" option. Leaving this option out prevents the error form happening.

This will cause the error:

```
serializers.serialize('json', [data], ensure_ascii=False)
```

This will prevent the error:

```
serializers.serialize('json', [data])
```

More info at:

http://groups.google.com/group/django-users/browse_thread/thread/4f5f99b730ee0aae/ (http://groups.google.com/group/django-users/browse_thread/thread/4f5f99b730ee0aae/)

http://groups.google.com/group/django-users/browse_thread/thread/87b1478c02d743e0/ (http://groups.google.com/group/django-users/browse_thread/thread/87b1478c02d743e0/)

#20 Carlo Pires commented on 2010-05-31:

With python2.6 you can do:

```
a = u'\xa1'
print format(a)
```

#21 Eliot (http://www.saltycrane.com/blog/) commented on 2010-06-01:

Carlo: Very nice. I see that format() (http://docs.python.org/library/stdtypes.html#str.format) is the preferred method for formatting strings going forward in 3.0. Thanks for the tip!

#22 Mark R (http://60bits.net/) commented on 2010-07-20:

Lucas's suggestion:

```
reload(sys) sys.setdefaultencoding( "latin-1" )
```

worked great for me on ActiveState Python 2.6.5 on my Windows box.

#23 Federico Capoano (http://nemesisdesign.net/) commented on 2010-10-13:

Thanks for this info, I was just looking for this!

#24 Rippa commented on 2011-02-03:

Adding the following works for me.

reload(sys)

sys.setdefaultencoding("latin-1")

Thanks! :)

#25 czemiello commented on 2011-02-18:

You saved my life !

#26 Chris (http://www.chrisspen.com/) commented on 2011-08-20:

Thanks, this fixed the error I was getting when I attempted to print a QuerySet. Apparently, my model's repr() was returning Unicode, but Django's QuerySet repr() expects Ascii, causing this very confusing error. Wrapping my models repr() output with smart_str() fixed the problem.

#27 Joe commented on 2011-10-16:

Adding the following works for me.

reload(sys)

sys.setdefaultencoding("latin-1")

Thanks! :)

#28 EngineeringDuniya (http://www.engineeringduniya.com/) commented on 2012-06-27:

Thanks. Wonderful !

#29 Jonatas CD commented on 2012-07-27:

You just seved me!

thanks.

I've also shared on DjangoBrasil google-group.

#30 Vitor Mendes commented on 2012-10-24:

i know i dont need to comment but this helped me so much i needed to thank someone somehow so thank you guys you rock!

#31 Andrew P commented on 2014-06-04:

Exactly what I was looking for, thanks for the quality article!