Going International

Apostolis Bessas
mpessas@transifex.com
@mpessas

July 3, 2012
Transifex
Unicode
100100011001011101100110110011011110101100010000010101011110
1111111001011011001100100001000010001010
Encodings
A character encoding system consists of a code that pairs each character from a given repertoire with something else.

Wikipedia
ASCII

- 7-bit code
- Characters for the English alphabet.
ASCII

- 7-bit code
- Characters for the English alphabet.
Unicode

Assign every possible character a unique code point.
Assign every possible character a unique *code point*.

- $A \rightarrow \text{U+0041}$
- $a \rightarrow \text{U+0061}$
UTF-8

Just another character encoding for Unicode.
Python 2.x
s = 'A string'

- Encoded strings.
- ASCII by default.
unicode

# -*- coding: utf-8 -*-
u = u'A string'

→ Strings stored in the internal representation.
→ Unicode literals
Conversion

```python
u.encode('UTF-8').decode('UTF-8')
```
Best practices

- Always use unicode strings.
- Decode in input and encode in output.
- Test against unicode strings.
Best practices

- Always use unicode strings.
- Decode in input and encode in output.
- Test against unicode strings.

```python
import codecs
codecs.open(filename, encoding=encoding)
```
Python 3

- Strings and bytes
Strings and bytes (Unicode literals are back in 3.3)
- Strings and bytes (*Unicode literals are back in 3.3*)
- No need to use the *codecs* module any more.
i18n & l10n
Formats

- Gettext (PO files)
- TS files (Qt)
- YAML
Choice?

Use a real format:

➔ Plurals support
➔ Context
➔ Comments
➔ Suggestions
Mark translation strings.
Extract them (PO files).
Translate them.
Compile them (MO files).
Load in the application.
Source code

https://github.com/mpessasgoing_international
import gettext

# Set up message catalog access
_t = gettext.translation('myapplication', 'locale', fallback=True)
_ = _t.ugettext
Usage

def greet_user(user):
    print _(u'Hello, %s.') % user
Plurals

children = {'John': 1, 'Mary': 3}

def report_children(user):
    print t.ungettext(
        u'You have %s child',
        u'You have %s children',
        children[user]
    ) % children[user]
xgettext -d myapplication -o app.pot l10n.py
vim app.pot
POT file headers

#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: 0.1\n"
"Report-Msgid-Bugs-To: http://github.com/mpessas/
going_international/issues\n"
"POT-Creation-Date: 2012-06-30 09:45+0300\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=INTEGER; plural=EXPRESSION;\n"
#: l10n.py:10
#, python-format
msgid "Hello, %s."
msgstr ""

#: l10n.py:17
#, python-format
msgid "You have %s child"
msgid_plural "You have %s children"
msgstr[0] ""
msgstr[1] ""
PO files

```bash
mkdir -p locale/en/LC_MESSAGES/

mkdir -p locale/el/LC_MESSAGES/
msginit -i app.pot -o locale/el/LC_MESSAGES/el.po -l el
vim locale/el/LC_MESSAGES/el.po
msgfmt locale/el/LC_MESSAGES/el.po -o \\locale/el/LC_MESSAGES/myapplication.mo

mkdir -p locale/it/LC_MESSAGES/
msginit -i app.pot -o locale/it/LC_MESSAGES/it.po -l it
vim locale/it/LC_MESSAGES/it.po
msgfmt locale/el/LC_MESSAGES/el.po -o \\locale/el/LC_MESSAGES/myapplication.mo
```
bash> **LANG=it** python2 l10n.py
Ciao, John.
You have 1 child
Ciao, Mary.
You have 3 children
Plural equation for arabic

\[
\begin{align*}
    n &== 0 \quad ? \quad 0 : \\
    n &== 1 \quad ? \quad 1 : \\
    n &== 2 \quad ? \quad 2 : \\
    n \mod 100 &\geq 3 & n \mod 100 &\leq 10 \quad ? \quad 3 : \\
    n \mod 100 &\geq 11 & n \mod 100 &\leq 99 \quad ? \quad 4 : \\
    5
\end{align*}
\]
Timezone handling
The mess with timezones

→ Daylight Saving Time (DST)
→ Past changes
UTC

- Coordinated Universal Time
- All timezones are based on that.
UTC

- Coordinated Universal Time
- All timezones are based on that.

Internally, only use times based on UTC. Convert them to localtime on output.
- Naive (does not have timezone information attached)
- Aware (has timezone information attached)
Naive (does not have timezone information attached)
Aware (has timezone information attached)

The two do **not** work together.
pytz

- Timezone database
- Saner conversions
import pytz
from datetime import datetime
u = datetime.utcnow().replace(tzinfo=pytz.utc)
r = u.astimezone(pytz.timezone('Europe/Rome'))
Questions?