Internationalization and Localization in Python

... and pretty much elsewhere.

Pedro Ferreira, May 2016

I18n and L10n for short.

Localization refers to the **adaptation** of a product, application or document content to meet the language, cultural and other requirements of a specific target market (a *locale*).

Internationalization is the design and development of a product, application or document content that **enables** easy localization for target audiences that vary in culture, region, or language.

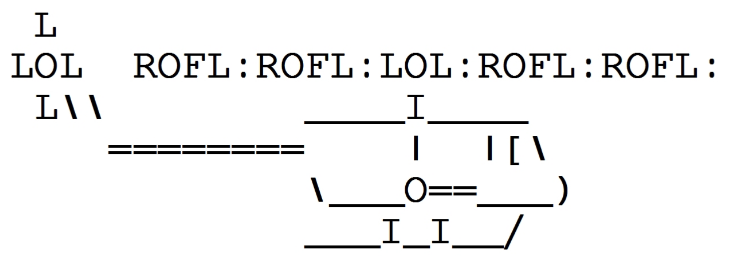
(W3C Internationalization Working Group)

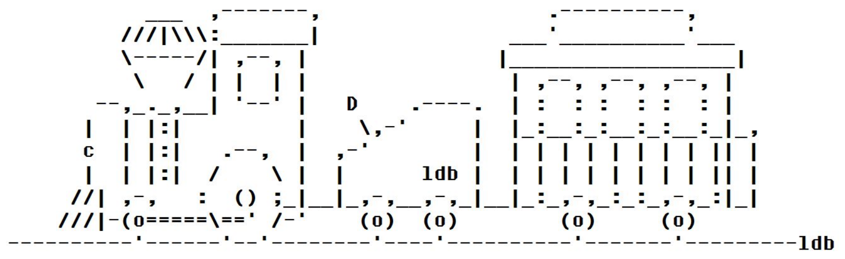
I18N + L10N

- Translation
- Numeric formats 1000 = 1,000 = 1.000 = 1’000
- Currency units € £ $ ¥
- Keyboards
- Icons, symbols 👆 👌 👊 ☠️ 🆒
- Collation and Sorting 漢 > A ?
- ...

Charsets in Python

What a fine m□lange.





ASCII

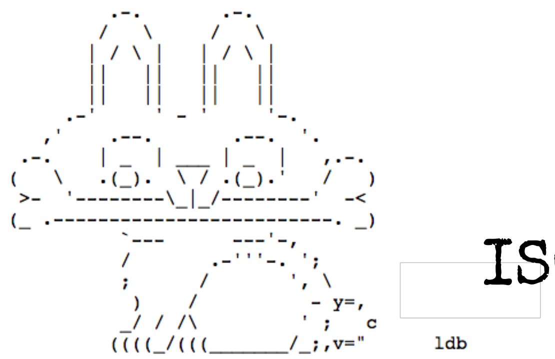
American Standard Code for Information Interchange

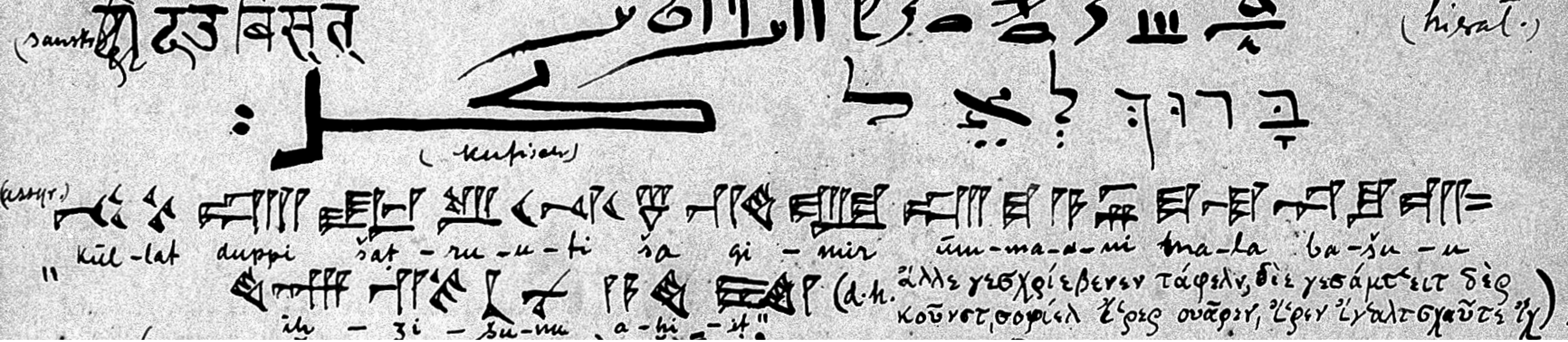
1963 ASA

7-bit = 128 symbols

(+ extensions = 8-bit)

ISO-8859-1/2/5 - most European languages




(Sanskrit)
(hieroglyph.)
(Kufisch)
(assyr.)
kul-lat duppi šat-ru-u-ti ša gi-mir um-ma-a-ni ma-la ba-šu-u
ih-zi-šu-nu a-hi-iţ

Unicode

"International/multilingual text character encoding system"

1988 Xerox/Apple

> 120.000 symbols, 129 scripts

Common Locale Data Repository (CLDR)

Unicode ≠ UTF

UTF = Unicode Transformation Format

UTF-8 - var.-width (ASCII superset)

UTF-16 - var.-width

UTF-32 - fixed-width

Python 2

`'foo'` → encoded string (ASCII)

`u'Ça va'` → unicode object

`unicode('Ça va')` → `sys.getdefaultencoding()`

Python 2

```
u'Ça va'.encode('utf-8') → '\xc3\x87a va’
u'Ça va'.encode('ascii') → 💥
'Ça va'.decode('utf-8') → u'\xe7a va’
'Ça va'.decode('ascii') → 💥
u'Ça va'.encode('ascii', 'ignore') → 'a va'
```

Implicit Hell

```
'Hello ' + u'World' ✓
'Hello %s' % u'World' ✓
'What a nice ' + u'mélange' ✓
u'What a nice ' + 'mélange' 💥
'What a nice %s!' % u'mélange' ✓
u'What a nice %s!' % 'mélange' 💥
```

Python 3

`'foo'` → `str('foo')` → Unicode

`u'foo'` → still unicode (compatibility)

`b'foo'` → `bytes('foo')`

No implicit conversions!

Python 3

```
'I cannot ' + b'join unicode and bytes together' → 💥
'Frappé'.encode(‘utf-8’) → b'Frapp\xc3\xa9'
```

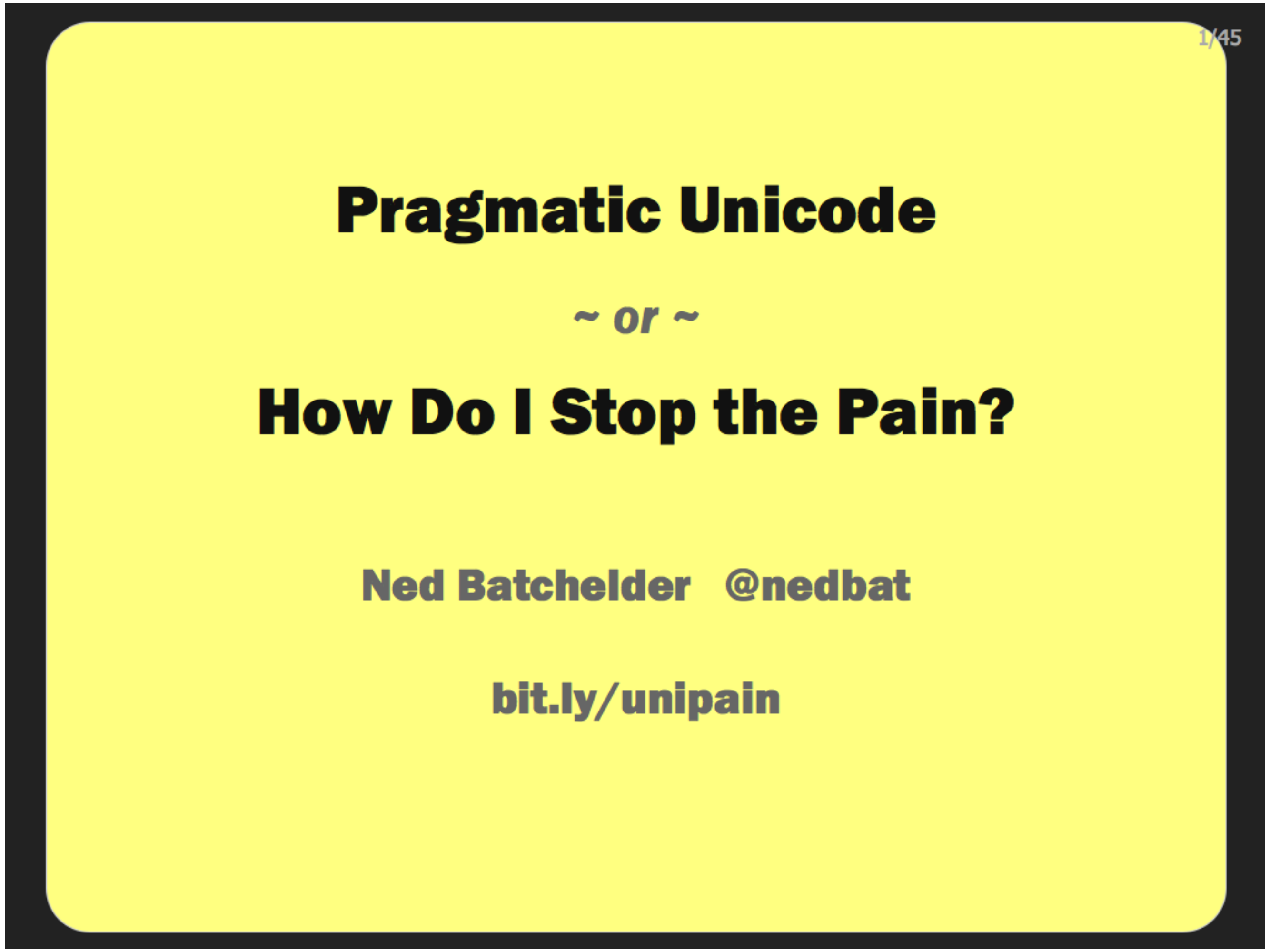
Tips

Always use Unicode

Except when serializing (e.g. I/O)

Python 3 > Python 2

Highly Recommended


1/45

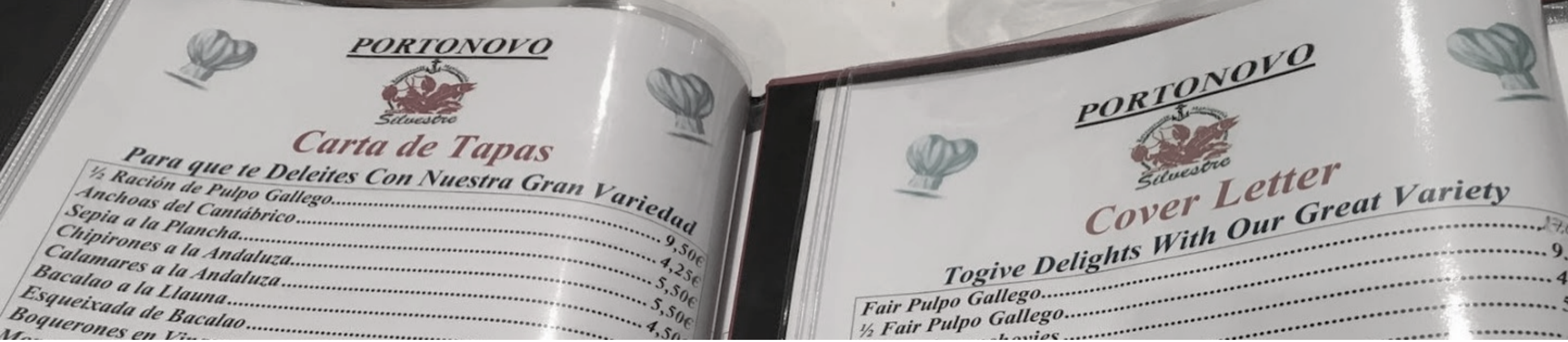
Pragmatic Unicode

~ or ~

How Do I Stop the Pain?

Ned Batchelder @nedbat

bit.ly/unipain


PORTONOVO

Silvestre

Carta de Tapas

Para que te Deleites Con Nuestra Gran Variedad

½ Ración de Pulpo Gallego	9,50€
Anchoas del Cantábrico	4,25€
Sepia a la Plancha	5,50€
Chipirones a la Andaluza	5,50€
Calamares a la Andaluza	5,50€
Bacalao a la Llauna	4,50
Esqueixada de Bacalao	
Boquerones en Vi	

PORTONOVO

Silvestre

Cover Letter

Togive Delights With Our Great Variety

Fair Pulpo Gallego	9,
½ Fair Pulpo Gallego	4

Language

Machines are terrible at it

Impossible to do it 100% right

Not hard to do a reasonable job, though.

gettext

Present everywhere

Based on dictionaries

Compiled into binary

Word returned unchanged if not found

```
_(u'Hello')  →  u'Olá'
_(u'Salut')→  u'Salut'
```

gettext

```
msgid ""
msgstr ""
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=n != 1;\n"

msgid "Good morning!"
msgstr "Bom dia!"

msgid "My name is %s."
msgstr "Chamo-me %s."

msgid "I've found %d result"
msgid_plural "I've found %d results"
msgstr[0] "Encontrei %d resultado"
msgstr[1] "Encontrei %d resultados"
```

gettext

```
>>> import gettext
>>> t = gettext.translation('myapp', '/path/to/translations/dir')

>>> t.ugettext(u'My name is Pedro.')
u'My name is Pedro.'

>>> t.ugettext(u'My name is %s.') % u'Pedro'
u'Chamo-me Pedro.'

>>> t.ungettext(u"I've found %d result", u"I've found %d results", 1) % 1
u'Encontrei 1 resultado'

>>> t.ungettext(u"I've found %d result", u"I've found %d results", 2) % 2
u'Encontrei 2 resultados'
```

Jinja i18n

```
{% trans %}Good morning!{% endtrans %}
{% trans name=user.name %}
    My name is {{ name }}.
{% endtrans %}

{% trans count=list|length -%}
    I've found {{ count }} result
{%- pluralize -%}
    I've found {{ count }} results
{%- endtrans %}
```

Babel

Additional tools for i18n/l10n

Based on gettext

Date and time formatting too

Numbers and currencies, Units (CLDR)

Babel

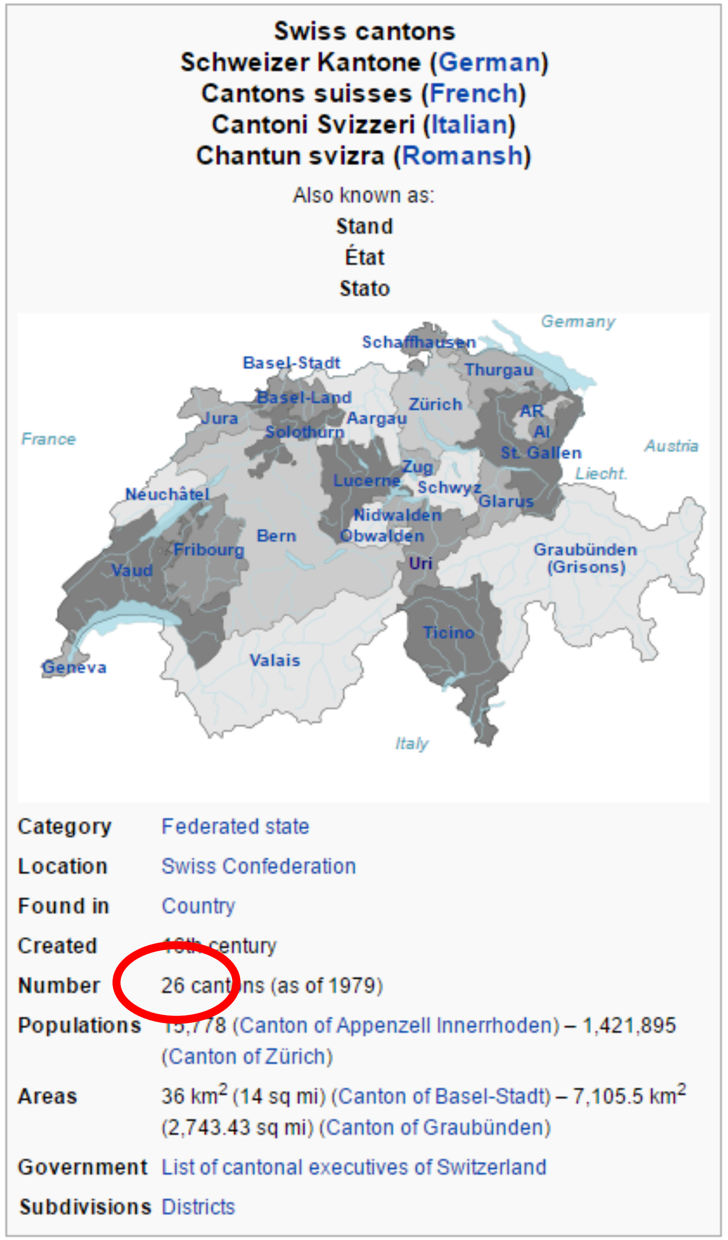
```
>>> from datetime import date
>>> from babel.dates import format_date
>>> d = date(2016, 5, 30)

>>> format_date(d, locale='pt_PT', format='full')
u'segunda-feira, 30 de maio de 2016’

>>> format_date(d, locale='en_US', format='short')
u'5/30/16'
```

Babel

```
>>> from babel import Locale
>>> fr_ch = Locale.parse('fr_CH')
>>> print fr_ch.get_display_name('en')
French (Switzerland)
>>> print fr_ch.get_display_name('jpn')
フランス語 (スイス)
>>> print fr_ch.get_territory_name('de')
Schweiz
>>> print fr.territories['BR']
Brésil
>>> print fr.currencies['USD']
dollar des États-Unis
```

pycountry

```
>>> import pycountry
>>> list(pycountry.countries)[0].name
'Afghanistan'
>>> list(pycountry.countries)[0].official_name
'Islamic Republic of Afghanistan'
>>> len(pycountry.subdivisions.get(country_code='US'))
57
>>> len(pycountry.subdivisions.get(country_code='CH'))
24
>>> pycountry.subdivisions.get(code='CH-GE').name
'Genève'
```



Swiss cantons
Schweizer Kantone (German)
Cantons suisses (French)
Cantoni Svizzeri (Italian)
Chantun svizra (Romansh)

Also known as:
Stand
État
Stato

Germany
Schaffhausen
Basel-Stadt
Thurgau
Basel-Land
Zürich
Jura
Aargau
AR
Solothurn
AI
France
St. Gallen
Austria
Zug
Liecht.
Lucerne
Schwyz
Neuchâtel
Glarus
Nidwalden
Bern
Obwalden
Fribourg
Graubünden (Grisons)
Vaud
Uri
Ticino
Geneva
Valais
Italy

Category	Federated state
Location	Swiss Confederation
Found in	Country
Created	16th century
Number	26 cantons (as of 1979)
Populations	15,778 (Canton of Appenzell Innerrhoden) – 1,421,895 (Canton of Zürich)
Areas	36 km^2 (14 sq mi) (Canton of Basel-Stadt) – 7,105.5 km^2 (2,743.43 sq mi) (Canton of Graubünden)
Government	List of cantonal executives of Switzerland
Subdivisions	Districts

Time Zones

UTC vs. Local Time

Daylight Saving Time (DST)

Time zones keep changing (Egypt, Venezuela 2016)

stdlib + pytz

```
>>> from datetime import datetime
>>> from pytz import timezone
>>> dt = datetime(2016, 5, 30, 10, 0, 0)
>>> zurich_time = timezone('Europe/Zurich')
>>> shanghai_time = timezone('Asia/Shanghai')
>>> print(zurich_time.localize(dt).astimezone(shanghai_time))
2016-05-30 16:00:00+08:00
>>> pytz.OLSON_VERSION
'2016d'
```

Summarizing...

Unicode and UTF-8 → stdlib (Python 3 = better!)

Translation → gettext + Babel (+ templating support!)

Locale-specific data → Babel (CLDR)

Lists of countries and regions → PyCountry (ISO)

Time zones → stdlib + pytz

Conclusions

Even acceptable I18N is hard to achieve

But the Python community provides great tools.

Think of i18n from the beginning,

Make as few assumptions as possible.

Use locale definitions when possible,

Use 3rd party standard-based libs...

But leave some room for exceptions!