Internationalization

The locale module

This module provides an interface to C's localization functions. It also provides functions to convert between numbers and strings based on the current locale (functions like **int** and **float**, as well as the numeric conversion functions in **string**, are not affected by the current locale).

Example: Using the locale module for data formatting

```
# File:locale-example-1.py

import locale

print "locale", "=>", locale.setlocale(locale.LC_ALL, "")

# integer formatting
value = 4711
print locale.format("%d", value, 1), "==",
print locale.atoi(locale.format("%d", value, 1))

# floating point
value = 47.11
print locale.format("%f", value, 1), "==",
print locale.atof(locale.format("%f", value, 1))

info = locale.localeconv()
print info["int_curr_symbol"]
```

```
locale => Swedish_Sweden.1252
4 711 == 4711
47,110000 == 47.11
SEK
```

Example: Using the locale module to get the platform locale

```
# File:locale-example-2.py

import locale

language, encoding = locale.getdefaultlocale()

print "language", language
print "encoding", encoding
```

```
language sv_SE
encoding cp1252
```

The unicodedata module

(New in 2.0) This module contains Unicode character properties, such as character categories, decomposition data, and numerical values.

Example: Using the unicodedata module

```
# File:unicodedata-example-1.py

import unicodedata

for char in [u"A", u"-", u"1", u"\N{LATIN CAPITAL LETTER O WITH DIAERESIS}"]:
    print repr(char),
    print unicodedata.category(char),
    print repr(unicodedata.decomposition(char)),
    print unicodedata.decimal(char, None),
    print unicodedata.numeric(char, None)
```

```
u'A' Lu '' None None
u'-' Pd '' None None
u'1' Nd '' 1 1.0
u'Ö' Lu '004F 0308' None None
```

Note that in Python 2.0, properties for CJK ideographs and Hangul syllables are missing. This affects characters in the range 0x3400-0x4DB5, 0x4E00-0x9FA5, and 0xAC00-D7A3. The first character in each range has correct properties, so you can work around this problem by simply mapping each character to the beginning:

```
def remap(char):
    # fix for broken unicode property database in Python 2.0
    c = ord(char)
    if 0x3400 <= c <= 0x4DB5:
        return unichr(0x3400)
    if 0x4E00 <= c <= 0x9FA5:
        return unichr(0x4E00)
    if 0xAC00 <= c <= 0xD7A3:
        return unichr(0xAC00)
    return char
```

This bug has been fixed in Python 2.1.

The ucnhash module

(Implementation, 2.0 only) This module is an implementation module, which provides a name to character code mapping for Unicode string literals. If this module is present, you can use \N{} escapes to map Unicode character names to codes.

In Python 2.1 and later, the services provided by this module has been moved to the **unicodedata** module.

Example: Using the ucnhash module

```
# File:ucnhash-example-1.py

# Python imports this module automatically, when it sees
# the first \N{} escape
# import ucnhash

print repr(u"\N{FROWN}")
print repr(u"\N{SMILE}")
print repr(u"\N{SKULL AND CROSSBONES}")
```

```
u'\u2322'
u'\u2323'
u'\u2620'
```