

```
#!/usr/bin/env python

"""calie.py -- Converting between ASCII LaTeX input encodings.

Calie stands for '(C)onvert (A)SCII (L)aTeX (I)nput (E)ncodings'.
It is primarily supposed to convert LaTeX source files (ASCII
files with extension '.tex') from one encoding to another. All
encodings are treated that can be given as an optional parameter
to the LaTeX command '\usepackage{inputenc}'. Only characters
with an ASCII code higher than 127 are converted. As calie can
be used on non-LaTeX files as well, it is something like a poor
man's version of the GNU recode program on Unix. It does probably
do much less, but on more platforms. ;-)

Calie does its job by reading the '.def' files of your local LaTeX
installation (e.g. in C:\programs\texmf\tex\latex\base on a Win-
doze box running MiKTeX), defining the source as well as destina-
tion encoding. The conversion itself should be very fast as calie
builds an internal translation table that is used by the very fast
string.translate() function that is part of Python's standard
string library.

The need for this module came up after I started to write LaTeX
code on Windows and Macs on the same project and when I thought
that one could not use multiple encodings with LaTeX simultane-
ously (in different files). Of course, this is actually not quite
so, as \inputenc{<encoding>} can indeed be used to switch between
different encodings within LaTeX, but even then this script can
still be useful when the file you want to convert is not necess-
arily one to be processed with LaTeX or when you want to convert
many files without modifying them individually by changing a
LaTeX command. In this case keep in mind that only those charac-
ters are converted that LaTeX lists in its .def files. For all
these reasons, the module is still named calie and not some vari-
ant of pmrecode (for poor man's recode) such that you're reminded
of its LaTeX context.

Dinu Gherman
"""

__version__ = '0.5'
__author__ = 'Dinu C. Gherman'
__date__ = '2000-02-03'

import string, os, re

### Global variable (just one).

# Adapt to point to your 'LaTeX base' directory of your
# LaTeX installation!

# latexFolder = 'Macintosh HD:Programs:OzTeX 4.0:Tex:Inputs:Latex:base'
# latexFolder = 'C:/programme/texmf/tex/latex/base'
# latexFolder = '/mnt/windows/Programme/texmf/tex/latex/base'

### Convenience function.

def _invertDict(dict):
    "Convenience function to invert a dictionary."

    # All the necessary restrictions apply...
    inv = {}
    for k, v in dict.items():
        inv[v] = k
    return inv
```

```
### The real stuff (not to be called directly).

def _readLatexDefFile(folder, name):
    "Turn a LaTeX .def file into a mapping for input encodings."

    # Sample line of a .def file:
    # \DeclareInputText{159}{\u}

    # Mapping from ASCII codes to LaTeX strings
    # (<num> -> <sequence>).
    d = {}

    s = '\\DeclareInputText'
    pat = re.compile('\\" + s + '\\{(.*)\\}\{(.*)\\}')
    file = open(os.path.join(folder, name + '.def'))

    while 1:
        line = file.readline()
        if not line:
            break
        if len(line) > len(s) and line[:len(s)] == s:
            num, seq = pat.match(line[:-1]).groups()
            char = chr(string.atoi(num))
            d[char] = seq

    return d

def _makeTable(fromEnc, toEnc):
    "Return a translation table from one encoding to another."

    toEnc = _invertDict(toEnc)
    toTable = map(chr, range(256))
    for i in range(128, 256):
        c = toTable[i]
        if fromEnc.has_key(c):
            try:
                toTable[i] = toEnc[fromEnc[c]]
            except KeyError:
                pass

    fromTable = map(chr, range(256))
    fromTable = string.join(fromTable, '')
    toTable = string.join(toTable, '')

    return string.maketrans(fromTable, toTable)

def _convertStream(fromEnc, toEnc, stream):
    "Convert a stream from one encoding to another."

    len1 = len(stream)
    tab = _makeTable(fromEnc, toEnc)
    out = string.translate(stream, tab)
    len2 = len(out)
    if len1 != len2:
        msg = "Different lengths! Before: %d and after: %d" % (len1, len2)
        raise "ConversionError", msg
    return out

### New interface (inspired by Andreas Jung's PyRecode wrapper).

def string2string(src, dest, text):
    "Recode a string and return it as a new string."

    try:
        srcDict = _readLatexDefFile(latexFolder, src)
        destDict = _readLatexDefFile(latexFolder, dest)
    except NameError:
```

---

```
    msgP1 = "You must set calie's 'latexFolder' variable, "  
    msgP2 = "before you can use calie!"  
    raise "SetupError", msgP1 + msgP2  
  
    return _convertStream(srcDict, destDict, text)  
  
def string2file(src, dest, text, outFileName):  
    "Recode a string and save it into a file."  
  
    recText = string2string(src, dest, text)  
    file = open(outFileName, 'wb')  
    file.write(recText)  
    file.close()  
  
def file2string(src, dest, inFileName):  
    "Read a file, recode it and return its content as a new string."  
  
    file = open(inFileName, 'rb')  
    text = file.read()  
    file.close()  
    return string2string(src, dest, text)  
  
def file2file(src, dest, inFileName, outFileName):  
    "Read a file, recode it and save it as a new file."  
  
    recText = file2string(src, dest, inFileName)  
    file = open(outFileName, 'wb')  
    file.write(recText)  
    file.close()  
  
### Test functions.  
  
def _testStringConversion():  
    "Round-trip conversion test from ansinew to applemac to ansinew."  
  
    print "Round-trip conversion test from ansinew to applemac to ansinew."  
    print  
  
    textInAnsineW = 'äöüÄÖÛŞşääääèèêîîîóóóóúúúÁÀÂÊËËËÎÎÎÎÓÓÓÓÛÛÛ'  
    print "ansinew :", textInAnsineW  
    textInApplemac = string2string('ansinew', 'applemac', textInAnsineW)  
    print "applemac:", textInApplemac  
    textInAnsineWAgain = string2string('applemac', 'ansinew', textInApplemac)  
    print "ansinew :", textInAnsineWAgain  
  
    if textInAnsineW == textInAnsineWAgain:  
        print "ok!"  
    else:  
        print "failed!"  
  
    print  
  
def test():  
    "Do some simple tests."  
  
    _testStringConversion()  
  
### Main.  
  
if __name__ == '__main__':  
    test()
```