This folder is a subpackage of ttLib. Each module here is a

specialized TT/OT table converter: they can convert raw data

to Python objects and vice versa. Usually you don't need to

use the modules directly: they are imported and used

automatically when needed by ttLib.

If you are writing you own table converter the following is

important.

The modules here have pretty strange names: this is due to the

fact that we need to map TT table tags (which are case sensitive)

to filenames (which on Mac and Win aren't case sensitive) as well

as to Python identifiers. The latter means it can only contain

[A-Za-z0-9\_] and cannot start with a number.

ttLib provides functions to expand a tag into the format used here:

>>> from fontTools import ttLib

>>> ttLib.tagToIdentifier("FOO ")

'F\_O\_O\_'

>>> ttLib.tagToIdentifier("cvt ")

'\_c\_v\_t'

>>> ttLib.tagToIdentifier("OS/2")

'O\_S\_2f\_2'

>>> ttLib.tagToIdentifier("glyf")

'\_g\_l\_y\_f'

>>>

And vice versa:

>>> ttLib.identifierToTag("F\_O\_O\_")

'FOO '

>>> ttLib.identifierToTag("\_c\_v\_t")

'cvt '

>>> ttLib.identifierToTag("O\_S\_2f\_2")

'OS/2'

>>> ttLib.identifierToTag("\_g\_l\_y\_f")

'glyf'

>>>

Eg. the 'glyf' table converter lives in a Python file called:

\_g\_l\_y\_f.py

The converter itself is a class, named "table\_" + expandedtag. Eg:

class table\_\_g\_l\_y\_f:

etc.

Note that if you \_do\_ need to use such modules or classes manually,

there are two convenient API functions that let you find them by tag:

>>> ttLib.getTableModule('glyf')

<module 'ttLib.tables.\_g\_l\_y\_f'>

>>> ttLib.getTableClass('glyf')

<class ttLib.tables.\_g\_l\_y\_f.table\_\_g\_l\_y\_f at 645f400>

>>>

You must subclass from DefaultTable.DefaultTable. It provides some default

behavior, as well as a constructor method (\_\_init\_\_) that you don't need to

override.

Your converter should minimally provide two methods:

class table\_F\_O\_O\_(DefaultTable.DefaultTable): # converter for table 'FOO '

def decompile(self, data, ttFont):

# 'data' is the raw table data. Unpack it into a

# Python data structure.

# 'ttFont' is a ttLib.TTfile instance, enabling you to

# refer to other tables. Do \*\*\*not\*\*\* keep a reference to

# it: it will cause a circular reference (ttFont saves

# a reference to us), and that means we'll be leaking

# memory. If you need to use it in other methods, just

# pass it around as a method argument.

def compile(self, ttFont):

# Return the raw data, as converted from the Python

# data structure.

# Again, 'ttFont' is there so you can access other tables.

# Same warning applies.

If you want to support TTX import/export as well, you need to provide two

additional methods:

def toXML(self, writer, ttFont):

# XXX

def fromXML(self, (name, attrs, content), ttFont):

# XXX