"""Module containing wrapper class to write block format data"""

import ctypes as ct

from libtwml import CLIB

class BlockFormatWriter(object):

"""

Class to write block format file.

"""

def \_\_init\_\_(self, file\_name, records\_per\_block=100):

file\_name = file\_name

if not isinstance(file\_name, str):

raise ValueError("file\_name has to be of type str")

self.file\_name = ct.c\_char\_p(file\_name.encode())

self.records\_per\_block = ct.c\_int(int(records\_per\_block))

handle = ct.c\_void\_p(0)

err = CLIB.block\_format\_writer\_create(ct.pointer(handle),

self.file\_name,

self.records\_per\_block)

self.\_handle = None

# 1000 means TWML\_ERR\_NONE

if err != 1000:

raise RuntimeError("Error from libtwml")

self.\_handle = handle

@property

def handle(self):

"""

Return the handle

"""

return self.\_handle

def write(self, class\_name, record):

"""

Write a record.

Note: `record` needs to be in a format that can be converted to ctypes.c\_char\_p.

"""

if not isinstance(class\_name, str):

raise ValueError("class\_name has to be of type str")

record\_len = len(record)

class\_name = ct.c\_char\_p(class\_name.encode())

record = ct.c\_char\_p(record)

err = CLIB.block\_format\_write(self.\_handle, class\_name, record, record\_len)

if err != 1000:

raise RuntimeError("Error from libtwml")

def flush(self):

"""

Flush records in buffer to outputfile.

"""

err = CLIB.block\_format\_flush(self.\_handle)

if err != 1000:

raise RuntimeError("Error from libtwml")

def \_\_del\_\_(self):

"""

Delete the handle

"""

if self.\_handle:

CLIB.block\_format\_writer\_delete(self.\_handle)