text2pdf.py

```python
from __future__ import print_function
import fitz
import sys

assert len(sys.argv) == 2, "usage: python %s text.file" % (sys.argv[0],)
ifn = sys.argv[1]
ofn = ifn + ".pdf"  # name of PDF output
"""
------------------------------------------------------------------------------
A very basic text-to-PDF converter.
-------------------------------------
Any text file.xxx will be converted to file.xxx.pdf
Adjust preferred page format, fontsize, fontname, fontfile below.
Formula of lines per page (nlines) is also used by the 'insertPage' method.
------------------------------------------------------------------------------
"""

width, height = fitz.PaperSize("a4")  # choose paper format
fontsz = 10  # choose font size of text
lineheight = fontsz * 1.2  # line height is 20% larger
# the above gives the following lines per page:
nlines = int((height - 108.0) / lineheight)

# choose a nice mono-spaced font of the system, instead of 'Courier'.
# To use a standard PDF base14 font, e.g. set font='Courier' and ffile=None
ffile = "C:/windows/fonts/consola.ttf"  # font file
font = "F0"  # fontname

sourcefile = open(ifn)  # we are going to convert this file
line_ctr = 0  # page line counter
total_ctr = 0  # total line counter
out_ctr = 0  # count output lines
out_buf = ""  # text of one page

doc = fitz.open()  # new empty PDF


def page_out(b):  # only a shortcut
    return doc.insertPage(
        -1,
        fontsize=fontsz,
        text=b,
        fontname=font,
        fontfile=ffile,
        width=width,
        height=height,
    )


while 1:
    line = sourcefile.readline()  # read a text line
    if line == "":
        break  # eof encountered
    out_buf += line  # concat line to page buffer
    line_ctr += 1  # increase page ctr
    total_ctr += 1  # increase total ctr
    if line_ctr == nlines:  # page line limit reached
        out_ctr += page_out(out_buf)  # output page to PDF
        out_buf = ""  # clear page buffer
        line_ctr = 0  # clear page line ctr
```

```
if len(out_buf) > 0:  # output remaining stuff in buffer
    out_ctr += page_out(out_buf)

print("PDF conversion results for file '%s':" % (ifn,))
print(out_ctr, "lines read,", total_ctr, "lines written,", nlines, "lines per page.")
print(ofn, "contains", len(doc), "pages.")

# Now add some header and footer to each created page

hdr_fontsz = 16  # header fontsize
ftr_fontsz = 8  # footer fontsize
blue = (0, 0, 1)  # header / footer color
pspace = 500  # available line width

for page in doc:
    footer = "%i (%i)" % (page.number + 1, len(doc))  # footer text
    plen_ftr = fitz.getTextlength(footer, fontname="Helvetica", fontsize=ftr_fontsz)

    page.insertText(
        fitz.Point(50, 50),
        ifn,  # header = input filename
        color=blue,
        fontsize=hdr_fontsz,
    )

    page.drawLine(
        fitz.Point(50, 60),
        fitz.Point(50 + pspace, 60),  # line below hdr
        color=blue,
        width=0.5,
    )

    page.drawLine(
        fitz.Point(50, height - 33),  # line above footer
        fitz.Point(50 + pspace, height - 33),
        color=blue,
        width=0.5,
    )

    page.insertText(
        fitz.Point(
            50 + pspace - plen_ftr, height - 33 + ftr_fontsz * 1.2  # insert footer
        ),
        footer,
        fontsize=ftr_fontsz,
        color=blue,
    )

# finally provide some metadata
m = {
    "creationDate": fitz.getPDFnow(),  # current timestamp
    "modDate": fitz.getPDFnow(),  # current timestamp
    "creator": "text2pdf.py",
    "producer": "PyMuPDF %s" % fitz.VersionBind,
    "title": "Content of file " + ifn,
    "subject": "Demonstrate the use of methods insertPage, insertText and drawLine",
    "author": "Jorj McKie",
}

doc.setMetadata(m)
```

```
# and save the PDF
doc.save(ofn, garbage=4, deflate=True)
doc.close()
```