```
## {{{ http://code.activestate.com/recipes/532908/ (r3)
#! /usr/bin/env python
"""
 pyText2Pdf - Python script to convert plain text files into Adobe
 Acrobat PDF files with support for arbitrary page breaks etc.

 Version 2.0

 Author: Anand B Pillai <abpillai at gmail dot com>

"""

# Derived from http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/189858

import sys, os
import string
import time
import optparse
import re

LF_EXTRA=0
LINE_END='\015'
# form feed character (^L)
FF=chr(12)

ENCODING_STR = """\
/Encoding <<
/Differences [ 0 /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /space /exclam
/quotedbl /numbersign /dollar /percent /ampersand
/quoteright /parenleft /parenright /asterisk /plus /comma
/hyphen /period /slash /zero /one /two /three /four /five
/six /seven /eight /nine /colon /semicolon /less /equal
/greater /question /at /A /B /C /D /E /F /G /H /I /J /K /L
/M /N /O /P /Q /R /S /T /U /V /W /X /Y /Z /bracketleft
/backslash /bracketright /asciicircum /underscore
/quoteleft /a /b /c /d /e /f /g /h /i /j /k /l /m /n /o /p
/q /r /s /t /u /v /w /x /y /z /braceleft /bar /braceright
/asciitilde /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/.notdef /.notdef /.notdef /.notdef /.notdef /.notdef
/dotlessi /grave /acute /circumflex /tilde /macron /breve
```

```
/dotaccent /dieresis /.notdef /ring /cedilla /.notdef
/hungarumlaut /ogonek /caron /space /exclamdown /cent
/sterling /currency /yen /brokenbar /section /dieresis
/copyright /ordfeminine /guillemotleft /logicalnot /hyphen
/registered /macron /degree /plusminus /twosuperior
/threesuperior /acute /mu /paragraph /periodcentered
/cedilla /onesuperior /ordmasculine /guillemotright
/onequarter /onehalf /threequarters /questiondown /Agrave
/Aacute /Acircumflex /Atilde /Adieresis /Aring /AE
/Ccedilla /Egrave /Eacute /Ecircumflex /Edieresis /Igrave
/Iacute /Icircumflex /Idieresis /Eth /Ntilde /Ograve
/Oacute /Ocircumflex /Otilde /Odieresis /multiply /Oslash
/Ugrave /Uacute /Ucircumflex /Udieresis /Yacute /Thorn
/germandbls /agrave /aacute /acircumflex /atilde /adieresis
/aring /ae /ccedilla /egrave /eacute /ecircumflex
/edieresis /igrave /iacute /icircumflex /idieresis /eth
/ntilde /ograve /oacute /ocircumflex /otilde /odieresis
/divide /oslash /ugrave /uacute /ucircumflex /udieresis
/yacute /thorn /ydieresis ]
>>
"""

INTRO="""\
%prog [options] filename

PyText2Pdf  makes a 7-bit clean PDF file from any input file.

It reads from a named file, and writes the PDF file to a file specified by
the user, otherwise to a file with '.pdf' appended to the input file.

Author: Anand B Pillai."""


class PyText2Pdf(object):
    """ Text2pdf converter in pure Python """

    def __init__(self):
        # version number
        self._version="1.3"
        # iso encoding flag
        self._IsoEnc=False
        # formfeeds flag
        self._doFFs=False
        self._progname="PyText2Pdf"
        self._appname = " ".join((self._progname,str(self._version)))
        # default font
```

```
    self._font="/Courier"
    # default font size
    self._ptSize=10
    # default vert space
    self._vertSpace=12
    self._lines=0
    # number of characters in a row
    self._cols=80
    self._columns=1
    # page ht
    self._pageHt=792
    # page wd
    self._pageWd=612
    # input file
    self._ifile=""
    # output file
    self._ofile=""
    # default tab width
    self._tab=4
    # input file descriptor
    self._ifs=None
    # output file descriptor
    self._ofs=None
    # landscape flag
    self._landscape=False
    # Subject
    self._subject = ''
    # Author
    self._author = ''
    # Keywords
    self._keywords = []
    # Custom regexp  for page breaks
    self._pagebreakre = None

    # marker objects
    self._curobj = 5
    self._pageObs = [0]
    self._locations = [0,0,0,0,0,0]
    self._pageNo=0

    # file position marker
    self._fpos=0

def parse_args(self):

    """ Callback function called by argument parser.
```

```
        Helps to remove duplicate code """

        if len(sys.argv)<2:
            sys.argv.append('-h')

        parser = optparse.OptionParser(usage=INTRO)
        parser.add_option('-o','--output',dest='outfile',help='Direct output to file
OUTFILE',metavar='OUTFILE')
        parser.add_option('-f','--font',dest='font',help='Use Postscript font FONT (must be in standard 14,
default: Courier)',
                          default='Courier')
        parser.add_option('-I','--isolatin',dest='isolatin',help='Use ISO latin-1
encoding',default=False,action='store_true')
        parser.add_option('-s','--size',dest='fontsize',help='Use font at PTSIZE points
(default=>10)',metavar='PTSIZE',default=10)
        parser.add_option('-v','--linespace',dest='linespace',help='Use line spacing LINESPACE (deault
12)',metavar='LINESPACE',default=12)
        parser.add_option('-l','--lines',dest='lines',help='Lines per page (default 60, determined
automatically if unspecified)',default=60, metavar=None)
        parser.add_option('-c','--chars',dest='chars',help='Maximum characters per line (default
80)',default=80,metavar=None)
        parser.add_option('-t','--tab',dest='tabspace',help='Spaces per tab character (default
4)',default=4,metavar=None)
        parser.add_option('-F','--ignoreff',dest='formfeed',help='Ignore formfeed character ^L (i.e, accept
formfeed characters as pagebreaks)',default=False,action='store_true')
        parser.add_option('-P','--papersize',dest='papersize',help='Set paper size (default is letter, accepted
values are "A4" or "A3")')
        parser.add_option('-W','--width',dest='width',help='Independent paper width in
points',metavar=None,default=612)
        parser.add_option('-H','--height',dest='height',help='Independent paper height in
points',metavar=None,default=792)
        parser.add_option('-2','--twocolumns',dest='twocolumns',help='Format as two
columns',metavar=None,default=False,action='store_true')
        parser.add_option('-L','--landscape',dest='landscape',help='Format in landscape
mode',metavar=None,default=False,action='store_true')
        parser.add_option('-R','--regexp',dest='pageregexp',help='Regular expression string to determine
page breaks (if supplied, this will be used to split text into pages, instead of using line
count)',metavar=None)
        parser.add_option('-S','--subject',dest='subject',help='Optional subject for the
document',metavar=None)
        parser.add_option('-A','--author',dest='author',help='Optional author for the
document',metavar=None)
        parser.add_option('-K','--keywords',dest='keywords',help='Optional list of keywords for the
document (separated by commas)',metavar=None)
```

```
optlist, args = parser.parse_args()
# print optlist.__dict__, args

if len(args)==0:
   sys.exit('Error: input file argument missing')
elif len(args)>1:
   sys.exit('Error: Too many arguments')

self._ifile = args[0]

d = optlist.__dict__
if d.get('isolatin'): self._IsoEnc=True
if d.get('formfeed'): self._doFFs = True
if d.get('twocolumns'): self._columns = 2
if d.get('landscape'): self._landscape = True

self._font = '/' + d.get('font')
psize = d.get('papersize')
if psize=='A4':
   self._pageWd=595
   self._pageHt=842
elif psize=='A3':
   self._pageWd=842
   self._pageHt=1190

fsize = int(d.get('fontsize'))
if fsize < 1: fsize = 1
self._ptSize = fsize

lspace = int(d.get('linespace'))
if lspace<1: lspace = 1
self._vertSpace = lspace

lines = int(d.get('lines'))
if lines<1: lines = 1
self._lines = int(lines)

chars = int(d.get('chars'))
if chars<4: chars = 4
self._cols = chars

tab = int(d.get('tabspace'))
if tab<1: tab = 1
self._tab = tab

w = int(d.get('width'))
```

```
    if w<72: w=72
    self._pageWd = w

    h = int(d.get('height'))
    if h<72: h=72
    self._pageHt = h

    # Very optional args
    author = d.get('author')
    if author: self._author = author

    subject = d.get('subject')
    if subject: self._subject = subject

    keywords = d.get('keywords')
    if keywords:
        self._keywords = keywords.split(',')

    pagebreak = d.get('pageregexp')
    if pagebreak:
        self._pagebreakre = re.compile(pagebreak, re.UNICODE|re.IGNORECASE)

    outfile = d.get('outfile')
    if outfile: self._ofile = outfile

    if self._landscape:
        print 'Landscape option on...'
    if self._columns==2:
        print 'Printing in two columns...'
    if self._doFFs:
        print 'Ignoring form feed character...'
    if self._IsoEnc:
        print 'Using ISO Latin Encoding...'

    print 'Using font',self._font[1:],'size =', self._ptSize

def writestr(self, str):
    """ Write string to output file descriptor.
    All output operations go through this function.
    We keep the current file position also here"""

    # update current file position
    self._fpos += len(str)
    for x in range(0, len(str)):
        if str[x] == '\n':
            self._fpos += LF_EXTRA
```

```
    try:
        self._ofs.write(str)
    except IOError, e:
        print e
        return -1

    return 0

def convert(self):
    """ Perform the actual conversion """

    if self._landscape:
        # swap page width & height
        tmp = self._pageHt
        self._pageHt = self._pageWd
        self._pageWd = tmp

    if self._lines==0:
        self._lines = (self._pageHt - 72)/self._vertSpace
    if self._lines < 1:
        self._lines=1

    try:
        self._ifs=open(self._ifile)
    except IOError, (strerror, errno):
        print 'Error: Could not open file to read --->', self._ifile
        sys.exit(3)

    if self._ofile=="":
        self._ofile = os.path.splitext(self._ifile)[0] + '.pdf'

    try:
        self._ofs = open(self._ofile, 'wb')
    except IOError, (strerror, errno):
        print 'Error: Could not open file to write --->', self._ofile
        sys.exit(3)

    print 'Input file=>',self._ifile
    print 'Writing pdf file',self._ofile, '...'
    self.writeheader()
    self.writepages()
    self.writerest()

    print 'Wrote file', self._ofile
    self._ifs.close()
    self._ofs.close()
```

```
    return 0

def writeheader(self):
    """Write the PDF header"""

    ws = self.writestr

    title = self._ifile

    t=time.localtime()
    timestr=str(time.strftime("D:%Y%m%d%H%M%S", t))
    ws("%PDF-1.4\n")
    self._locations[1] = self._fpos
    ws("1 0 obj\n")
    ws("<<\n")

    buf = "".join(("/Creator (", self._appname, " By Anand B Pillai )\n"))
    ws(buf)
    buf = "".join(("/CreationDate (", timestr, ")\n"))
    ws(buf)
    buf = "".join(("/Producer (", self._appname, "(\\251 Anand B Pillai))\n"))
    ws(buf)
    if self._subject:
        title = self._subject
        buf = "".join(("/Subject (",self._subject,")\n"))
        ws(buf)
    if self._author:
        buf = "".join(("/Author (",self._author,")\n"))
        ws(buf)
    if self._keywords:
        buf = "".join(("/Keywords (",' '.join(self._keywords),")\n"))
        ws(buf)

    if title:
        buf = "".join(("/Title (", title, ")\n"))
        ws(buf)

    ws(">>\n")
    ws("endobj\n")

    self._locations[2] = self._fpos

    ws("2 0 obj\n")
    ws("<<\n")
    ws("/Type /Catalog\n")
    ws("/Pages 3 0 R\n")
```

```
        ws(">>\n")
        ws("endobj\n")

        self._locations[4] = self._fpos
        ws("4 0 obj\n")
        ws("<<\n")
        buf = "".join(("/BaseFont ", str(self._font), " /Encoding /WinAnsiEncoding /Name /F1 /Subtype
/Type1 /Type /Font >>\n"))
        ws(buf)

        if self._IsoEnc:
            ws(ENCODING_STR)

        ws(">>\n")
        ws("endobj\n")

        self._locations[5] = self._fpos

        ws("5 0 obj\n")
        ws("<<\n")
        ws("  /Font << /F1 4 0 R >>\n")
        ws("  /ProcSet [ /PDF /Text ]\n")
        ws(">>\n")
        ws("endobj\n")

    def startpage(self):
        """ Start a page of data """

        ws = self.writestr

        self._pageNo += 1
        self._curobj += 1

        self._locations.append(self._fpos)
        self._locations[self._curobj]=self._fpos

        self._pageObs.append(self._curobj)
        self._pageObs[self._pageNo] = self._curobj

        buf = "".join((str(self._curobj), " 0 obj\n"))

        ws(buf)
        ws("<<\n")
        ws("/Type /Page\n")
        ws("/Parent 3 0 R\n")
        ws("/Resources 5 0 R\n")
```

```
    self._curobj += 1
    buf = "".join(("/Contents ", str(self._curobj), " 0 R\n"))
    ws(buf)
    ws(">>\n")
    ws("endobj\n")

    self._locations.append(self._fpos)
    self._locations[self._curobj] = self._fpos

    buf = "".join((str(self._curobj), " 0 obj\n"))
    ws(buf)
    ws("<<\n")

    buf = "".join(("/Length ", str(self._curobj + 1), " 0 R\n"))
    ws(buf)
    ws(">>\n")
    ws("stream\n")
    strmPos = self._fpos

    ws("BT\n");
    buf = "".join(("/F1 ", str(self._ptSize), " Tf\n"))
    ws(buf)
    buf = "".join(("1 0 0 1 50 ", str(self._pageHt - 40), " Tm\n"))
    ws(buf)
    buf = "".join((str(self._vertSpace), " TL\n"))
    ws(buf)

    return strmPos

def endpage(self, streamStart):
    """End a page of data """

    ws = self.writestr

    ws("ET\n")
    streamEnd = self._fpos
    ws("endstream\n")
    ws("endobj\n")

    self._curobj += 1
    self._locations.append(self._fpos)
    self._locations[self._curobj] = self._fpos

    buf = "".join((str(self._curobj), " 0 obj\n"))
    ws(buf)
```

```
    buf = "".join((str(streamEnd - streamStart), '\n'))
    ws(buf)
    ws('endobj\n')

def writepages(self):
    """Write pages as PDF"""

    ws = self.writestr

    beginstream=0
    lineNo, charNo=0,0
    ch, column=0,0
    padding,i=0,0
    atEOF=0
    linebuf = ''

    while not atEOF:
        beginstream = self.startpage()
        column=1

        while column <= self._columns:
            column += 1
            atFF=0
            atBOP=0
            lineNo=0
            # Special flag for regexp page break
            pagebreak = False

            while lineNo < self._lines and not atFF and not atEOF and not pagebreak:
                linebuf = ''
                lineNo += 1
                ws("(")
                charNo=0

                while charNo < self._cols:
                    charNo += 1
                    ch = self._ifs.read(1)
                    cond = ((ch != '\n') and not(ch==FF and self._doFFs) and (ch != ''))
                    if not cond:
                        # See if this dude matches the pagebreak regexp
                        if self._pagebreakre and self._pagebreakre.search(linebuf.strip()):
                            pagebreak = True

                        linebuf = ''
                        break
                    else:
```

```
            linebuf = linebuf + ch

        if ord(ch) >= 32 and ord(ch) <= 127:
            if ch == '(' or ch == ')' or ch == '\\':
                ws("\\")
            ws(ch)
        else:
            if ord(ch) == 9:
                padding =self._tab - ((charNo - 1) % self._tab)
                for i in range(padding):
                    ws(" ")
                charNo += (padding -1)
            else:
                if ch != FF:
                    # write \xxx form for dodgy character
                    buf = "".join(('\\', ch))
                    ws(buf)
                else:
                    # dont print anything for a FF
                    charNo -= 1

    ws(")'\n")
    if ch == FF:
        atFF=1
    if lineNo == self._lines:
        atBOP=1

    if atBOP:
        pos=0
        ch = self._ifs.read(1)
        pos= self._ifs.tell()
        if ch == FF:
            ch = self._ifs.read(1)
            pos=self._ifs.tell()
        # python's EOF signature
        if ch == '':
            atEOF=1
        else:
            # push position back by one char
            self._ifs.seek(pos-1)

    elif atFF:
        ch = self._ifs.read(1)
        pos=self._ifs.tell()
        if ch == '':
            atEOF=1
```

```
                else:
                    self._ifs.seek(pos-1)

            if column < self._columns:
                buf = "".join(("1 0 0 1 ",
                               str((self._pageWd/2 + 25)),
                               " ",
                               str(self._pageHt - 40),
                               " Tm\n"))
                ws(buf)

        self.endpage(beginstream)

def writerest(self):
    """Finish the file"""

    ws = self.writestr
    self._locations[3] = self._fpos

    ws("3 0 obj\n")
    ws("<<\n")
    ws("/Type /Pages\n")
    buf = "".join(("/Count ", str(self._pageNo), "\n"))
    ws(buf)
    buf = "".join(("/MediaBox [ 0 0 ", str(self._pageWd), " ", str(self._pageHt), " ]\n"))
    ws(buf)
    ws("/Kids [ ")

    for i in range(1, self._pageNo+1):
        buf = "".join((str(self._pageObs[i]), " 0 R "))
        ws(buf)

    ws("]\n")
    ws(">>\n")
    ws("endobj\n")

    xref = self._fpos
    ws("xref\n")
    buf = "".join(("0 ", str((self._curobj) + 1), "\n"))
    ws(buf)
    buf = "".join(("0000000000 65535 f ", str(LINE_END)))
    ws(buf)

    for i in range(1, self._curobj + 1):
        val = self._locations[i]
        buf = "".join((string.zfill(str(val), 10), " 00000 n ", str(LINE_END)))
```

```
            ws(buf)

        ws("trailer\n")
        ws("<<\n")
        buf = "".join(("/Size ", str(self._curobj + 1), "\n"))
        ws(buf)
        ws("/Root 2 0 R\n")
        ws("/Info 1 0 R\n")
        ws(">>\n")

        ws("startxref\n")
        buf = "".join((str(xref), "\n"))
        ws(buf)
        ws("%%EOF\n")


def main():

    pdfclass=PyText2Pdf()
    pdfclass.parse_args()
    pdfclass.convert()

if __name__ == "__main__":
    main()
## end of http://code.activestate.com/recipes/532908/ }}}
```