# An Intro to pyfpdf – A Simple Python PDF Generation Library

[July 10, 2012](https://www.blog.pythonlibrary.org/2012/07/10/an-intro-to-pyfpdf-a-simple-python-pdf-generation-library/" \o "Permalink to An Intro to pyfpdf – A Simple Python PDF Generation Library)[Python](https://www.blog.pythonlibrary.org/category/python/)[Python PDF Series](https://www.blog.pythonlibrary.org/tag/python-pdf-series/)[Mike](https://www.blog.pythonlibrary.org/author/mld/" \o "View all posts by Mike)

Today we’ll be looking at a simple PDF generation library called [pyfpdf](https://github.com/reingart/pyfpdf" \t "https://www.blog.pythonlibrary.org/2012/07/10/an-intro-to-pyfpdf-a-simple-python-pdf-generation-library/_blank), a port of [FPDF](http://www.fpdf.org/" \t "https://www.blog.pythonlibrary.org/2012/07/10/an-intro-to-pyfpdf-a-simple-python-pdf-generation-library/_blank)which is a php library. This is not a replacement for Reportlab, but it does give you more than enough to create simple PDFs and may meet your needs. Let’s take a look and see what it can do!

### Installing pyfpdf

You can just use pip to install pyfpdf:

pip install pyfpdf

### Taking pyfpdf for a Test Drive

As with any new library, you need to actually write some code to see how it works. Here’s one of the simplest scripts you can make that will create a PDF:

**import** pyfpdf

pdf = pyfpdf.FPDF(format='letter')

pdf.add\_page()

pdf.set\_font("Arial", size=12)

pdf.cell(200, 10, txt="Welcome to Python!", align="C")

pdf.output("tutorial.pdf")

Note that when we initialize our FPDF object we need to tell it that we want the result to be “letter” size. It defaults to “A4”. Next we need to add a page, set the font and put some text in. The ****pdf.cell**** call was a little unintuitive. The first two arguments are width and height and kind of specify the location of the text you pass in. The ****align**** parameter only takes single letter abbreviations and you’ll have to look in the source to figure those out. In this case, we’re centering the text by passing it a “C”. The last line accepts two parameters: the pdf name and the destination. If there’s no destination present, then the PDF is output to the same directory that the script is run in.

What if you wanted to add another line? Well you can add more text to the end if you edit how big the cell is and just create another cell. If you need a line break, then you can change the code to the following:

**import** pyfpdf

pdf = pyfpdf.FPDF(format='letter')

pdf.add\_page()

pdf.set\_font("Arial", size=12)

pdf.cell(200, 10, txt="Welcome to Python!", ln=1, align="C")

pdf.cell(200,10,'Powered by FPDF',0,1,'C')

pdf.output("tutorial.pdf")

Which results in the following PDF: [tutorial.pdf](http://www.blog.pythonlibrary.org/wp-content/uploads/2012/07/tutorial.pdf)

### Adding Bling with Headers, Footers and PageBreaks

The tutorial that is included on pyfpdf’s Google Code site shows how to do headers, footers and page breaks. It doesn’t run due to a method name that must have changed and the code is written using “this” instead of “self”, so I rewrote it and cleaned it up a bit. Here’s the code:

**import** pyfpdf

*########################################################################***class** MyPDF(pyfpdf.FPDF):

""""""

*#----------------------------------------------------------------------*

**def** header(self):

"""

Header on each page

"""

*# insert my logo*

self.image("logo.png", x=10, y=8, w=23)

*# position logo on the right*

self.cell(w=80)

*# set the font for the header, B=Bold*

self.set\_font("Arial", style="B", size=15)

*# page title*

self.cell(40,10, "Python Rules!", border=1, ln=0, align="C")

*# insert a line break of 20 pixels*

self.ln(20)

*#----------------------------------------------------------------------*

**def** footer(self):

"""

Footer on each page

"""

*# position footer at 15mm from the bottom*

self.set\_y(-15)

*# set the font, I=italic*

self.set\_font("Arial", style="I", size=8)

*# display the page number and center it*

pageNum = "Page %s/{nb}" % self.page\_no()

self.cell(0, 10, pageNum, align="C")

*#----------------------------------------------------------------------***if** \_\_name\_\_ == "\_\_main\_\_":

pdf = MyPDF()

pdf.alias\_nb\_pages()

pdf.add\_page()

pdf.set\_font("Times", size=12)

*# put some lines on the page*

**for** i **in** range(1, 50):

pdf.cell(0, 10, "Line number %s" % i, border=0, ln=1)

pdf.output("tutorial2.pdf")

Which creates the following PDF: [tutorial2.pdf](http://www.blog.pythonlibrary.org/wp-content/uploads/2012/07/tutorial2.pdf)

This time around, we create a subclass of FPDF and override its ****header**** and ****footer****methods as they don’t actually do anything anyway (i.e. they’re stubs). In our header, we create an image object and pass in a logo file as well as the x/y location and the width (23) of the logo. You can also pass a height too if you care about keeping the aspect ratio sane. Then we position it and insert a string of text for the title. Finally, we put in a line break.

The footer is set up to be 15 mm from the bottom. It’s also in 8pt Arial Italic. The error in the official tutorial is that it calls self.PageNo(), but that doesn’t exist. However there is a ****page\_no**** method that seems to replace it, so I used that. Then at the bottom of the script, we actually create our pdf object and tell it to write a bunch of lines. If you run this script, you should get a 3-page document.

### Using pyfpdf to Generate a PDF from HTML

One of my [readers](http://www.reddit.com/r/Python/comments/wcsl7/an_intro_to_pyfpdf_a_simple_python_pdf_generation/" \t "https://www.blog.pythonlibrary.org/2012/07/10/an-intro-to-pyfpdf-a-simple-python-pdf-generation-library/_blank) pointed out that pyfpdf can also generate a PDF from basic HTML. I don’t know how I missed that, but it’s true! You can! You can read about it on the project’s [wiki](https://code.google.com/p/pyfpdf/wiki/WriteHTML" \t "https://www.blog.pythonlibrary.org/2012/07/10/an-intro-to-pyfpdf-a-simple-python-pdf-generation-library/_blank). I am reproducing a slightly modified example below:

html = """

<H1 align="center">html2fpdf</H1>

<h2>Basic usage</h2>

<p>You can now easily print text while mixing different

styles : <B>bold</B>, <I>italic</I>, <U>underlined</U>, or

<B><I><U>all at once</U></I></B>!

<BR>You can also insert hyperlinks

like this <A HREF="http://www.mousevspython.com">www.mousevspython.comg</A>,

or include a hyperlink in an image. Just click on the one below.<br>

<center>

<A HREF="http://www.mousevspython.com"><img src="tutorial/logo.png" width="150" height="150"></A>

</center>

<h3>Sample List</h3>

<ul><li>option 1</li>

<ol><li>option 2</li></ol>

<li>option 3</li></ul>

<table border="0" align="center" width="50%">

<thead><tr><th width="30%">Header 1</th><th width="70%">header 2</th></tr></thead>

<tbody>

<tr><td>cell 1</td><td>cell 2</td></tr>

<tr><td>cell 2</td><td>cell 3</td></tr>

</tbody>

</table>

"""

**from** pyfpdf **import** FPDF, HTMLMixin

**class** MyFPDF(FPDF, HTMLMixin):

**pass**

pdf=MyFPDF()*#First page*

pdf.add\_page()

pdf.write\_html(html)

pdf.output('html.pdf','F')

Which will create the following PDF: [html.pdf](http://www.blog.pythonlibrary.org/wp-content/uploads/2012/07/html.pdf)

### Wrapping Up

There’s one more tutorial on their website that talks about colors and line breaks, but I’ll leave that for you to do as an exercise. I didn’t see anything in here about drawing, inserting tables or graphs, custom font embedding or the many other features that are available in Reportlab, but then again this is supposed to be a simple PDF generation library. If you need more advanced features, then you should certainly look at Reportlab or even some of the offshoot projects that are based on it (rst2pdf or xhtml2pdf come to mind).