pdfformfiller Documentation

Release 0.4

Daniel roesler

March 20, 2016

Contents

CHAPTER 1

Quickstart

Source Code: https://github.com/diafygi/pdfformfiller

Welcome to the documentation for PdfFormFiller! This is a library that lets you easy insert text into a pdf. It is super useful when you need to prefill an existing pdf template (for example, a grant application form) with your own data.

```
pip install pdfformfiller
```

Once installed, you can add text fields to any pdf. You specify the bounding box of the field, and the text will auto-resize to fit within that rectangle.

```
from pdfformfiller import PdfFormFiller
filler = PdfFormFiller("myform.pdf")
filler.add_text(text, pagenum, (x1, y1), (x2, y2))
filler.write(outfile)
```

In order to determine the correct (x1, y1), (x2, y2) coordinates for your test field bounding box, we recommend dumping your existing pdf template to images with 72 dpi and using an image editor (like GIMP) to find the pixel coordinates of the rectangle you want your bounding box to be.

```
pdftoppm -png -r 72 myform.pdf myform-pages
```

CHAPTER 2

Example use

2.1 Basic Example

Fill out a simple form by adding you name at the top of the first two pages.

```
>>> from pdfformfiller import PdfFormFiller
>>> filler = PdfFormFiller("myform.pdf")
>>> filler.add_text("Joe Smith", 0, (50, 50), (500, 100))
>>> filler.add_text("Joe Smith", 1, (50, 50), (500, 100))
>>> filler.write(outfile)
```

2.2 Default Styling

Set some default custom styles and padding for the text fields.

```
>>> from pdfformfiller import PdfFormFiller
>>> from reportlab.lib.styles import ParagraphStyle
>>> customstyle = ParagraphStyle("customstyle", backColor="#FF0000")
>>> custompadding = [6, 6, 6, 6]
>>> filler = PdfFormFiller("myform.pdf", style=customstyle, padding=custompadding)
>>> filler.add_text("Joe Smith", 0, (50, 50), (500, 100))
>>> filler.add_text("Joe Smith", 1, (50, 50), (500, 100))
>>> filler.write(outfile)
```

2.3 Field Styling

Set some default custom styles and padding for one text field.

```
>>> from pdfformfiller import PdfFormFiller
>>> from reportlab.lib.styles import ParagraphStyle
>>> customstyle = ParagraphStyle("customstyle", backColor="#FF0000")
>>> custompadding = [6, 6, 6, 6]
>>> filler = PdfFormFiller("myform.pdf")
>>> filler.add_text("Joe Smith", 0, (50, 50), (500, 100), style=customstyle, padding=custompaddi
>>> filler.add_text("Joe Smith", 1, (50, 50), (500, 100))
>>> filler.write(outfile)
```

2.4 Removing Fields

Remove the first text field on the 2nd page (will still write the text field on the 1st page).

```
>>> from pdfformfiller import PdfFormFiller
>>> filler = PdfFormFiller("myform.pdf", boxes=(0, 0, 255))
>>> filler.add_text("Joe Smith", 0, (50, 50), (500, 100))
>>> filler.add_text("Joe Smith", 1, (50, 50), (500, 100))
>>> del filler[1][0]
>>> filler.write(outfile)
```

2.5 Bounding Boxes

Show red bounding boxes on the page (useful for debugging).

```
>>> from pdfformfiller import PdfFormFiller
>>> filler = PdfFormFiller("myform.pdf", boxes=True)
>>> filler.add_text("Joe Smith", 0, (50, 50), (500, 100))
>>> filler.add_text("Joe Smith", 1, (50, 50), (500, 100))
>>> filler.write(outfile)
```

2.6 Custom Boxes

Show custom blue bounding boxes on the page.

```
>>> from pdfformfiller import PdfFormFiller
>>> filler = PdfFormFiller("myform.pdf", boxes=(0, 0, 255))
>>> filler.add_text("Joe Smith", 0, (50, 50), (500, 100))
>>> filler.add_text("Joe Smith", 1, (50, 50), (500, 100))
>>> filler.write(outfile)
```

CHAPTER 3

API

3.1 PdfFormFiller

class pdfformfiller.**PdfFormFiller**(*pdf*, *style=<ParagraphStyle 'Normal'>*, *padding=(0, 0, 0, 0)*, *boxes=False*)

Add text fields to a PDF. Useful for programmatically filling out forms.

Parameters **pdf** (*str or file*) – The pdf to which to add text fields. Can be a string path to a file, or a file-like object.

Keyword Arguments

- **style** (*ParagraphStyle*) – Custom style to apply to text fields. Default is black text, 20pt Times New Roman.
- **padding** (*tuple*) – Custom padding to apply to the text field. Tuple is a series of four floats or integers (leftPadding, bottomPadding, rightPadding, topPadding). Default is `[0, 0, 0, 0]`.
- **boxes** (*bool or tuple*) – Whether to show the text field bounding boxes in the final pdf. Can be `False` (default), `True` (default color red), or an `(r, g, b)` tuple for the bounding box color.

N

list[*TextField*]

List of the added text fields for page N. For example, to delete the 2nd text field from the 4th page, do `del filler[3][1]`. Note: This attribute is an integer, not "N".

Note: Coordinates use `points`, which represent 1/72 inch. The origin for `(0, 0)` is at the top left of the page. This is slightly different than normal pdf coordinates (which have the origin at the bottom left). We use a top left origin because it is easier to figure out bounding box coordinates using image editing software (which typically use a top left origin). You can quickly dump the pdf to a set of images at the correct dpi using `pdftoppm`:

```
pdftoppm -png -r 72 myform.pdf myform
```

---

add_text(*text*, *pagenum*, *upperLeft*, *lowerRight*, *style=None*, *padding=None*)

Add a text field with a bounding box.

Insert some text within a certain rectangle on a page. Text will be resized if needed to fit within the rectangle. NOTE: coordinates are measured from the top left of the page (i.e. top left is (0, 0)).

Parameters

- **text** (*str*) – Contents of the text field
- **pagenum** (*int*) – Page of the pdf to insert the field (0 is first page)
- **upperLeft** (*tuple*) – (x, y) coordinates for top left corner of bounding box rectangle
- **lowerRight** (*tuple*) – (x, y) coordinates for bottom right corner of bounding box rectangle

Keyword Arguments

- **style** (*ParagraphStyle*) – Custom style to apply to the text. Default is None (i.e. inherit style from class default).
- **padding** (*tuple*) – Custom padding to apply to the text field. Tuple is a series of four floats or integers (leftPadding, bottomPadding, rightPadding, topPadding). Default is None (i.e. inherit padding from class default).

Returns None

write(*outputFile*)

Writes the modified pdf to a file.

This method merges the original pdf with all of the added text fields together to create the final modified pdf. Text fields are written over top of the original pdf.

Parameters **outputFile** (*str or file*) – Output file to write to. Can be string path or any file-like object.

Returns None

3.2 TextField

class pdfformfiller.**TextField**(*text*, *x1*, *y1*, *width*, *height*, *style*, *padding*)

This is a namedtuple class used for storing text field parameters in a *PdfFormFiller* instance. Note: Coordinates in this class are based on the origin (0, 0) as the bottom left.

text

str

Contents of the text field

x1

int or float

X-coord for bottom left bounding box corner

y1

int or float

Y-coord for bottom left bounding box corner

width

int or float

Width of bounding box

height

int or float

Height of bounding box

style
ParagraphStyle or None

Optional custom style of text field

padding
tuple or None

Optional custom padding for text field

CHAPTER 4

Testing & Docs

To setup a development environment, you need to clone the repo and install dependencies.

```
git clone https://github.com/diafygi/pdfformfiller.git
cd pdfformfiller
virtualenv venv
source venv/bin/activate
pip install -r requirements.txt
```

You don't need anything extra to run the test suite.

```
$ python test.py
.........
----------------------------------------------------------------------
Ran 9 tests in 0.080s

OK
```

However, if you want to generate a coverage report or build the documentation, you will need to install the developer dependencies.

```
sudo apt-get install poppler-utils
pip install -r dev_requirements.txt

# to get code coverage
coverage run --omit="venv/*" test.py
coverage report

# to build the docs
cd docs
make html
```

A

H

N

P

S

T

W

X

Y