Red Stork

Release 0.0.41

Jul 09, 2020

Manual:

Yet another PDF parser. This one is based on PDFium engine.

CHAPTER 1

Quick Start

Sample:

```
from redstork.document import Document

doc = Document('sample.pdf')
print('Number of pages:', len(doc))
```

1.1 Tutorial

The philosophy of **redstork** is to map API to standard and well undersood Python objects, like `list` and `dict`.

In this tutorial we will use the following `sample document`.

1.1.1 Version

There are two version values in *`redstork`* module: PDFium build version, and Python package version:

```
import redstork

redstork.__pdfium_version__
>> 'cromium/4097'

redstork.__version__
>> '0.0.1'
```

1.1.2 Document

`Document` is the top-level object, and the only object that can be instantiated directly:

```
from redstork import Document

doc = Document('sample.pdf')

len(doc)
>> 15
```

As you can see, *Document* resembles standard Python `list`, containing *Page* objects.

PDF file creators can attach arbitraty key-value strings to the document, that we call `meta` (official PDf specs call it `Document Information Dictionary`). Most commonly these values describe `Author`, `Title`, and the name of software that created this document. Lets see the meta in our sample:

```
doc.meta['Title']
>> 'Red Stork'
```

You can change `meta` content and save the updated document:

```
doc.meta['Title'] = 'Awesome PDF parsing library'
doc.save('awesome.pdf')
```

Document has a lazily populated collection of fonts. Initially this collection is empty. As pages are being accessed and parsed, this collection is being populated:

```
list(doc[0])  # read all objects from page 1
len(doc.fonts)
>> 2
```

1.1.3 Page

Page represents PDF page. Get page by indexing a *Document* object, just like a normal list:

```
page = doc[0]
page.crop_box
>> (0.0, 0.0, 612.0, 792.0)
```

Page has *Page.label*, representing the page label (like `xxi`, or `128`):

```
doc[2].label  # this is the label of the third page
>> 'i'
```

A page of PDF document is a list-like object, containing concrete instances of *PageObject*:

```
page = ...
len(page)  # how many objects on this page?
>> 17
```

1.1.4 PageObject

Abstract class *PageObject* describes an object on a PDF page. Concrete classes implementing *PageObject* are:

- *TextObject* - a string of characters
- *PathObject* - vector graphics
- *ImageObject* - a bitmap image

- *ShadingObject* - a shading object

Notable properties of all objects are:

- *PageObject.page()* - links back to the parent page
- *PageObject.matrix()* - transformation matrix of this object
- PageObject.rect() - rectangle of this object on the page

1.1.5 TextObject

Text object represents a string of characters. Each character is a three-tuple of (charcode, x, y), where charcode is a character code (this value is just an index in the font glyph table, not a text corresponding to this character!). x and y are placement coordinates of this character (in the coordinate system of this *TextObject* - first character typically has x,y == 0, 0.

Text object has font property. Here is how to use font to extract text of a *TextObject*:

```
def text_of(o):
    assert o.type == PageObject.OBJ_TYPE_TEXT, o
    text = []
    for c, x, y in o:
        text.append(o.font[c])
    return ''.join(text)

page = ...
for o in page:
    if o.type == PageObject.OBJ_TYPE_TEXT:
        text = text_of(o)
        print(text)
```

1.1.6 PathObject

Path object represents a set of vector drawing instructions.

1.1.7 ImageObject

Image object represents an embedded bitmap image. You can ge the pixel width and height of the image, using the properties *ImageObject.pixel_width()* and *ImageObject.pixel_height()*.

Example:

```
page = ...
for o in page:
    if o.type == PageObject.OBJ_TYPE_IMAGE:
        print(o.pixel_width, o.pixel_height)
```

1.1.8 Font

Font object is a look-up table for character text, and also holds character glyphs (shape).

Font names in PDF file have a special prefix. To get a human-friendly one use *Font.simple_name()*.

Document contains a lazy font collection *Document.fonts()*. It is lazy, because just after document is opened, it is empty. As pages are accessed and parsed, this collection is populated.

Here is how to get *Glyph* object:

```
page = ...
for o in page:
    if o.type == PageObject.OBJ_TYPE_TEXT:
        for code,_,_ in o:
            glyph = o.font.load_glyph(code)
            print('Character with code %d has %d glyph instructions', code,␣
↪len(glyph))
```

1.2 API Reference

1.2.1 Document

class redstork.**Document**(*file_name*, *password=None*)

PDF document.

A list-like container of pages. Sample use:

```
doc = Document('sample.pdf')
print("Number of pages:', len(doc))

for key, value in doc.meta.items():
    print('     ', key, ':', value)
```

__init__(*file_name*, *password=None*)

Create new PDF Document object, from a file.

Parameters

- **file_name** (*str*) – Name of PDF file
- **password** (*str*) – File password (optional)

numpages = None

int – total number of pages

meta = None

dict – document meta info (Author, Title, etc)

fonts = None

dict – font collection (populated lazily as pages are parsed)

__getitem__(*page_index*)

Returns *Page* at this index.

Example:

```
doc = ...

page = doc[0]  # first page
```

Parameters **page_index** (*int*) – zero-based page index

Returns *Page* object

__len__()
Returns number of pages in this document

__iter__()
Iterate over the pages of this document

get_all_pages_size()
get width and height of all pages, without loading each page

changed
True is PDF was changed since teh load (or last save)

save(*filename*)
Saves PDF file, resets *Document.changed()* to False

1.2.2 Page

class redstork.**Page**(*page*, *page_index*, *parent*)
Represents page of a PDF file.

crop_box
Page crop box.

media_box
Page media box.

rotation
Page rotation.

- 0 - no rotation
- 1 - rotated 90 degrees clock-wise
- 2 - rotated 180 degrees clock-wise
- 3 - rotated 270 degrees clock-wise

label
Page label.

__len__()
Number of objects on this page.

__getitem__(*index*)
Get object at this index.

__iter__()
Iterates over page objects.

flat_iter()
Iterates over all non-container objects (Text, Image, Path).

render_to_buffer(*scale=1.0*, *rect=None*)
Render page (or rectangle on the page) to memory (the pixel format is BGRx)

Parameters

- **scale** (*float*) – scale to use (default is 1.0, which will assume that 1pt takes 1px)
- **rect** (*tuple*) – optional rectangle to render. Value is a 4-tuple of (x0, y0, x1, y1) in PDF coordinates. if None, then page's *crop_box* will be used for rendering.

render(*file_name*, *scale=1.0*, *rect=None*)
Render page (or rectangle on the page) as PPM image file.

Parameters

- **file_name** (*str*) – name of the output file
- **scale** (*float*) – scale to use (default is 1.0, which will assume that 1pt takes 1px)
- **rect** (*tuple*) – optional rectangle to render. Value is a 4-tuple of (x0, y0, x1, y1) in PDF coordinates. if None, then page's *crop_box* will be used for rendering.

PageObject

class redstork.**PageObject**(*obj*, *index*, *typ*, *parent*)

OBJ_TYPE_TEXT = 1
see *TextObject*

OBJ_TYPE_PATH = 2
see *PathObject*

OBJ_TYPE_IMAGE = 3
see *ImageObject*

OBJ_TYPE_SHADING = 4
see *ShadingObject*

OBJ_TYPE_FORM = 5
Common superclass of all page objects

type = None
type of this object

matrix = None
transformation matrix of this object

page
Links back to the parent page

TextObject

class redstork.**TextObject**(*obj*, *index*, *typ*, *parent*)
Represents a string of text on a page

font = None
Font for this text object

font_size = None
font size of this text object

matrix = None
matrix for this page object

__len__()
Number of items in this string

__getitem__(*index*)
Returns item at this index.

Each item is a 3-tuple: (charcode, x, y).

__iter__()
Iterates over items.

char_iter()
Iterates over characters (skips kerns)

text_geometry_iter()
Iterates over characters and returns character text and bounds

effective_font_size
Returns effective (user-visible) font size

scale_y
Returns Y-scale of text matrix transformation

scale_x
Returns X-scale of text matrix transformation

skew
Returns skew value of text matrix.

box(*x0*, *y0*, *x1*, *y1*)
Computes bounding box after transformation with text matrix

Font

class redstork.**Font**(*font*, *parent*)
Represents font used in a PDF file.

FLAGS_NORMAL = 0
Normal font

FLAGS_FIXED_PITCH = 1
Fixed pitch font

FLAGS_SERIF = 2
Serif font

FLAGS_SYMBOLIC = 4
Symbolic font

FLAGS_SCRIPT = 8
Script font

FLAGS_NONSYMBOLIC = 32
Non-symbolic font

FLAGS_ITALIC = 64
Italic font

FLAGS_ALLCAP = 65536
All-cap font

FLAGS_SMALLCAP = 131072
Small-cap font

FLAGS_FORCE_BOLD = 262144
Force-bold font

name
Font name in the PDF document.

simple_name
Font name without PDF-specific prefix.

flags
Font flags.

weight
Font weight.

is_vertical
True for vertical writing systems (CJK)

id
Tuple of (Object_id, Generation_id), identifying underlaying stream in PDF file

load_glyph(*charcode*)
Load glyph, see *Glyph*

Parameters **charcode** (*int*) – the character code (see *TextObject*)

__getitem__(*charcode*)
Returns Unicode text of this character.

Parameters **charcode** (int) - the character code (see *TextObject*) –

is_editable
True if font encoding can be changed

__setitem__(*charcode*, *text*)
Updates font encoding.

Parameters

- **charcode** (*int*) – character code
- **text** (*str*) – new text for this character code

Raises `ReadOnlyEncodingError` – if encoding is read-one (no "ToUnicode" map in the font dictionary)

Glyph

class redstork.**Glyph**(*glyph*, *parent*)
Represents Glyph drawing instructions

LINETO = 0
LineTo instruction

CURVETO = 1
CurveTo instruction

MOVETO = 2
MoveTo instruction

__getitem__(*i*)
Returns a 4-tuple representing this drawing instruction: (x, y, type, close).

Parameters **i** (*int*) – index of the instruction

ImageObject

class redstork.**ImageObject**(*obj*, *index*, *typ*, *parent*)
Represents image on a page.

matrix = None
matrix for this page object

pixel_width
width of the bitmap, in pixels

pixel_height
height of the bitmap, in pixels

PathObject

class redstork.**PathObject**(*obj*, *index*, *typ*, *parent*)
Represents vector graphics on a aage.

matrix = None
matrix for this page object

ShadingObject

class redstork.**ShadingObject**(*obj*, *index*, *typ*, *parent*)
Represents a shading object on a page.

FormObject

class redstork.**FormObject**(*obj*, *index*, *typ*, *parent*)
Represents a form (XObject) on a page - a container of other page objects (used internally).

matrix = None
matrix for this page object

form_matrix = None
transformation matrix for contained objects

flat_iter()
Iterates over all non-container objects in this form.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search
- Glossary

Python Module Index

r

Index

Symbols

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

W