Python

Python PDF Packages Comparison, All You Need To Know (Updated 2024)

Posted: January 6, 2024 • 11 min read • pythonify

CONTENTS

Fun Fact About PDFs

Summary of Comparison

PyPDF

Pdfminer.six

PdfPlumber

PyMuPDF

OCRmyPDF

Conclusion

Portable Document Format (PDF) is a file format developed by Adobe in 1992 to present documents, including text formatting and images, in a manner independent of application software, hardware, and operating systems

To understand PDFs properly we need to know that there can be three types of PDF documents:

- **Digitally-born PDF files**: The file are created digitally on the computer. It can contain images, texts, links, outline items (a.k.a., bookmarks), JavaScript etc. If you Zoom in a lot, the text still looks sharp.
- **Scanned PDF files**: The pages in this pdf contains scanned image. The images are then stored in a PDF file. Hence the file is just a container for those images. You cannot copy the text, you don't have links, outline items, JavaScript.
- **OCRed PDF files**: The scanner ran OCR software and put the recognized text in the background of the image. Hence you can copy the text, but it still looks like a scan. If you zoom in enough, you can recognize pixels.

Fun Fact About PDFs

Most PDF files look like they contain well-structured text. But the reality is that a PDF file does not contain anything that resembles paragraphs, sentences, or even words. When it comes to text, a PDF file is only aware of the characters and their placement.

This makes extracting meaningful pieces of text from PDF files difficult. The characters that compose a paragraph are no different from those that compose the table, the page footer or the description of a figure. Unlike other document formats, like a .txt file or a word document, the PDF format does not contain a stream of text.

A PDF document consists of a collection of objects that together describe the appearance of one or more pages, possibly accompanied by additional interactive elements and higher-level application data. A PDF file contains the objects making up a PDF document along with associated structural information, all represented as a single self-contained sequence of bytes.

In this blog, we will discuss the following 5 libraries, the last one i.e. OCRmyPDF serves a different purpose but is a very useful tool for converting scanned pdf to searchable pdf.

1. PyPDF
2. PyMuPDF
3. PDFminer.six
4. PDFplumber
5. OCRmyPDF

Summary of Comparison

Features	PyPDF	PyMuPDF	PDFminer.six	PDFplumber
Main feature	PDF processing, Reading and writing PDFs, Applying transformations, Delete pages merging pages, etc	PDF manipulation, including advanced rendering and comprehensive data extraction	extracting text and other elements from PDFs with a focus on accurate layout analysis and conversion to other text formats	Uses PDFminer.six, to extract text, PDFPlumber excels at extracting and analyzing visual and tabular data. provides a more intuitive interface for exploring the layout and structure of a page

Features	PyPDF	PyMuPDF	PDFminer.six	PDFplumber
Text Extraction(Digital PDF)	✓	✓	✓	✓
Text Extraction(Scanned PDF)	Requires external OCR like Tesseract	Requires external OCR like Tesseract	Requires external OCR like Tesseract	Requires external OCR like Tesseract
Table Extraction	✗	✓	✗	✓
Merge PDF, Page Deletion, Addition, Reorder	✓	✓	✗	✗
Transformation like Rotation, Scaling	✓	✓	✗	✗
Complexity	Easiest and simple	Feature-rich but takes time to get things right	Moderate	Easier than pdfminer
Speed	Slowest	Fastest and performant	Moderate	Moderate

PyPDF

pypdf is a free and open-source pure-python PDF library capable of **splitting**, **merging**, **cropping**, and **transforming(rotation, scaling, translation)** the pages of PDF files. It can also add custom data, viewing options, and passwords to PDF files. pypdf can retrieve text and metadata from PDFs as well.

Here are some basic uses of this library

Merging and Transforming PDF files

```
python
```

Copy

```python
from pypdf import PdfWriter

merger = PdfWriter()

for pdf in ["file1.pdf", "file2.pdf", "file3.pdf"]:
    merger.append(pdf)

#transformation can be applied in page level
merger.pages[0].rotate(90)

#Transformation can also be applied using the add_transformation method
transformation = Transformation().rotate(45)
merger.pages[0].add_transformation(transformation)

merger.write("merged-pdf.pdf")
merger.close()
```

Text Extraction

We can extract text content from Digitally-born PDF files. Its limitation is that it does not work with scanned pdf.

python Copy

```python
from pypdf import PdfReader

reader = PdfReader("example.pdf")
```

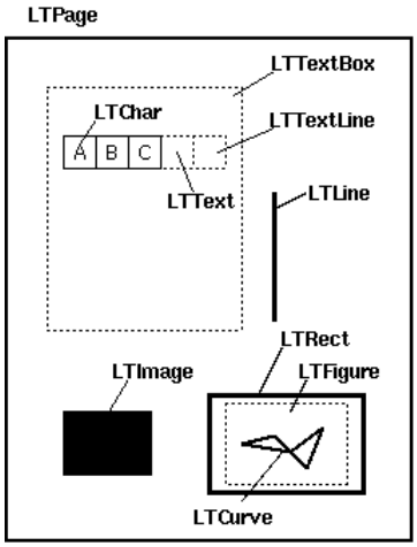
PYTHONIFY

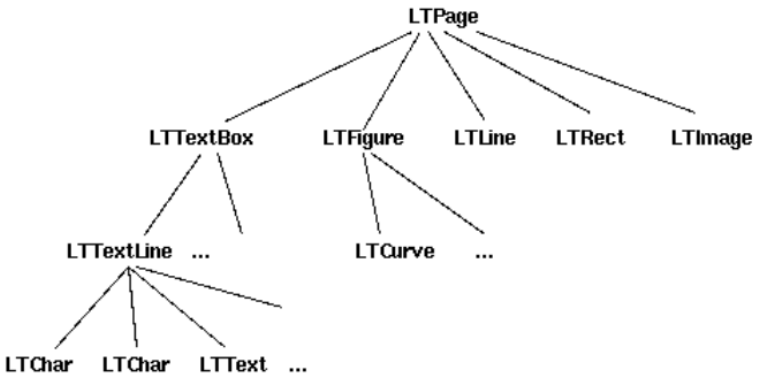
Pdfminer.six

Pdfminer.six is a Python package for extracting information from PDF documents. As we already know pdf is a binary format that consists of objects without proper structure unlike text files or other more familiar HTML files. pdfminer can help us extract different segments from pdf files like, text, figure, line, rect, and image.

```python
from pdfminer.high_level import extract_pages
for page_layout in extract_pages("test.pdf"):
    for element in page_layout:
        print(element)
```

Each `element` will be an `LTTextBox`, `LTFigure`, `LTLine`, `LTRect` or an `LTImage`. Some of these can be iterated further, for example iterating through an `LTTextBox` will give you an `LTTextLine`, and these, in turn, can be iterated through to get an `LTChar`. See the diagram here:





Let's say we want to extract all of the text. We could do:

```python
from pdfminer.high_level import extract_pages
from pdfminer.layout import LTTextContainer
for page_layout in extract_pages("test.pdf"):
    for element in page_layout:
        if isinstance(element, LTTextContainer):
            print(element.get_text())
```

Or, we could extract the fontname or size of each individual character:

```python
from pdfminer.high_level import extract_pages
from pdfminer.layout import LTTextContainer, LTChar
for page_layout in extract_pages("test.pdf"):
    for element in page_layout:
        if isinstance(element, LTTextContainer):
            for text_line in element:
                for character in text_line:
                    if isinstance(character, LTChar):
                        print(character.fontname)
                        print(character.size)
```

Extract all images from PDF into directory

This command extracts all the images from the PDF and saves them into the `my_images_dir` directory. If you have properly installed pdfminer.six, you should be able to run this command from the command line.

```shell
pdf2txt.py example.pdf --output-dir my_images_dir
```

PdfPlumber

PDFplumber Plumbs PDF for detailed information about each text character, rectangle, and line. Plus: Table extraction and visual debugging. it works best on machine-generated, rather than scanned, PDFs. It uses `pdfminer.six` under the hood. It adds **Table extraction** on top of `pdfminer.six`

Let's extract a table from this sample pdf file that looks like this:

Sample Table

This is a sample table. The pdfplumber detects the table in a pdf and extracts the table contents in a list. Then, we save the table data in pandas dataframe for further processing.

Name	Age	Occupation	City
John	28	Engineer	New York
Emily	22	Teacher	Los Angeles
Mike	35	Doctor	Chicago
Sarah	30	Designer	San Francisco
Alex	25	Developer	Boston

Here. is the example code to extract the Table from the above pdf using `PdfPlumber`:

python Copy

```python
import pdfplumber
import pandas as pd

pdf = pdfplumber.open("./sample_pdf_with_table.pdf")
page = pdf.pages[0]

table = page.extract_table()

# Create a dataframe from the table data
# table[1:]-> Skip the first row of the table
# table[0]-> use the first row as the column names
df = pd.DataFrame(table[1:], columns=table[0])

# Remove spaces from the column
for column in ["Name", "Age", "Occupation", "City"]:
    df[column] = df[column].str.replace(" ", "")

# Save to CSV
df.to_csv("sample_pdf_with_table.csv", index=False)
```

We get the `sample_pdf_with_table.csv` file where we extract the table data from the pdf as follows:

```shell
Name,Age,Occupation,City
John,28,Engineer,NewYork
Emily,22,Teacher,LosAngeles
Mike,35,Doctor,Chicago
Sarah,30,Designer,SanFrancisco
Alex,25,Developer,Boston
```

Copy

PyMuPDF

PyMuPDF is a high-performance Python library for data extraction, analysis, conversion & manipulation of PDF **(and other)** documents. It is based on MuPDF which is a `C` library for pdf data extraction.

Some of the features it supports are Text extraction, Merging PDFs, Transforming PDFs, Inserting pages, Deleting pages, rearranging pages, etc.PyMuPDF does all this way faster than other libraries. Below I have shown a comparison of the most important features listed on the pymupdf's official website, to see the full list visit this link.

Feature	PyMuPDF	PyPDF2	pdfplumber / pdfminer
Supports Multiple Document Formats	PDF XPS EPUB MOBI FB2 CBZ SVG TXT Image	PDF	PDF
Implementation	C and Python	Python	Python
Render Document Pages	All document types	No rendering	No rendering
Extract Text	All document types	PDF only	PDF only
Extract Tables	✓	✗	PDF plumber can
Extract Vector Graphics	All document types	✗	Limited
Encrypted PDFs	✓	Limited	Limited

Feature	PyMuPDF	PyPDF2	pdfplumber / pdfminer
Integrates with Jupyter and IPython Notebooks	✓	✗	✓
Joining / Merging PDF with other Document Types	All document types	PDF only	PDF only
OCR API for Seamless Integration with Tesseract	All document types	✗	✗
PDF Form Fields	Create, read, update	Limited, no creation	✗

Below I have provided snippets for basic features supported by PyMuPDF. To use PyMuPDF, you need to install it using pip. All the features are accessed using the `fitz` module.

Merging PDF files

To merge PDF files, do the following:

```python
import fitz

doc_a = fitz.open("a.pdf") # open the 1st document
doc_b = fitz.open("b.pdf") # open the 2nd document

doc_a.insert_pdf(doc_b) # merge the docs
doc_a.save("a+b.pdf") # save the merged document with a new filename
```

Rotating a PDF

To add a rotation to a page, do the following:

```python
```

```python
import fitz

doc = fitz.open("test.pdf") # open document
page = doc[0] # get the 1st page of the document
page.set_rotation(90) # rotate the page
doc.save("rotated-page-1.pdf")
```

Deleting Pages

To delete a page from a document, do the following:

python Copy

```python
import fitz

doc = fitz.open("test.pdf") # open a document
doc.delete_page(0) # delete the 1st page of the document
doc.save("test-deleted-page-one.pdf") # save the document
```

Extracting Tables from a Page

Tables can be found and extracted from any document Page

python Copy

```python
import fitz
from pprint import pprint

doc = fitz.open("test.pdf") # open document
page = doc[0] # get the 1st page of the document
tabs = page.find_tables() # locate and extract any tables on page
print(f"{len(tabs.tables)} found on {page}") # display number of found tables
if tabs.tables:  # at least one table found?
   pprint(tabs[0].extract())  # print content of first table/
```

OCRmyPDF

OCRmyPDF is a Python package and command line application that adds text “layers” to images in PDFs, making scanned image PDFs searchable and copyable/pasteable. It uses OCR to guess the text contained in images. OCRmyPDF also supports plugins that enable customization of its processing steps, and it is highly tolerant of PDFs containing scanned images and “born digital” content that doesn’t require text recognition.

```shell
ocrmypdf                    # it's a scriptable command line program
   -l eng+fra               # it supports multiple languages
   --rotate-pages           # it can fix pages that are misrotated
   --deskew                 # it can deskew crooked PDFs!
   --title "My PDF"         # it can change output metadata
   --jobs 4                 # it uses multiple cores by default
   --output-type pdfa       # it produces PDF/A by default
   input_scanned.pdf        # takes PDF input (or images)
   output_searchable.pdf    # produces validated PDF output
```

Copy

Features

- Converts regular PDFs into searchable PDF/A format.
- Accurately overlays OCR text on images to facilitate copying and pasting often without altering other content.
- Maintains original image resolutions.
- Reduces file sizes through image optimization.
- Offers image deskewing and cleaning before OCR, if needed.
- Utilizes all CPU cores for efficient processing.
- Employs Tesseract OCR to recognize over 100 languages.

Limitations

OCRmyPDF is subject to limitations imposed by the Tesseract OCR engine. These limitations are inherent to any software relying on Tesseract:

- OCR accuracy may not be perfect and could be inferior to commercial OCR tools.

- Handwriting recognition is not supported.
- Accuracy decreases for languages not included in the `-l LANG` setting.
- Tesseract may misinterpret the layout, like failing to recognize multiple columns.
- Tesseract doesn't identify font types or structure text into paragraphs or headings, only providing text and its location.

Conclusion

In conclusion, this comprehensive blog has navigated through the intricate world of PDF processing in Python, showcasing the strengths and particularities of five key libraries: PyPDF, PyMuPDF, PDFminer.six, PDFplumber, and OCRmyPDF. While PyPDF and PyMuPDF excel in general PDF manipulation and rendering, PDFminer.six and PDFplumber are more adept at extracting text and other elements like lines,rects and figures, with PDFplumber offering an intuitive interface for layout exploration on top of pdfminer. OCRmyPDF, on the other hand, stands out for its ability to add searchable text layers to scanned PDFs, enhancing their accessibility and functionality. Each of these tools, with its unique features and capabilities, caters to various needs within the PDF processing spectrum, making them invaluable assets for developers and data analysts dealing with diverse PDF-related challenges. This guide not only illuminates the capabilities of each library but also serves as a navigational beacon for selecting the right tool for specific PDF handling tasks, ensuring efficient and effective outcomes in the realm of digital document processing.

Popular Tags

#all

#python

#programming

#document-processing

#asyncio

#library

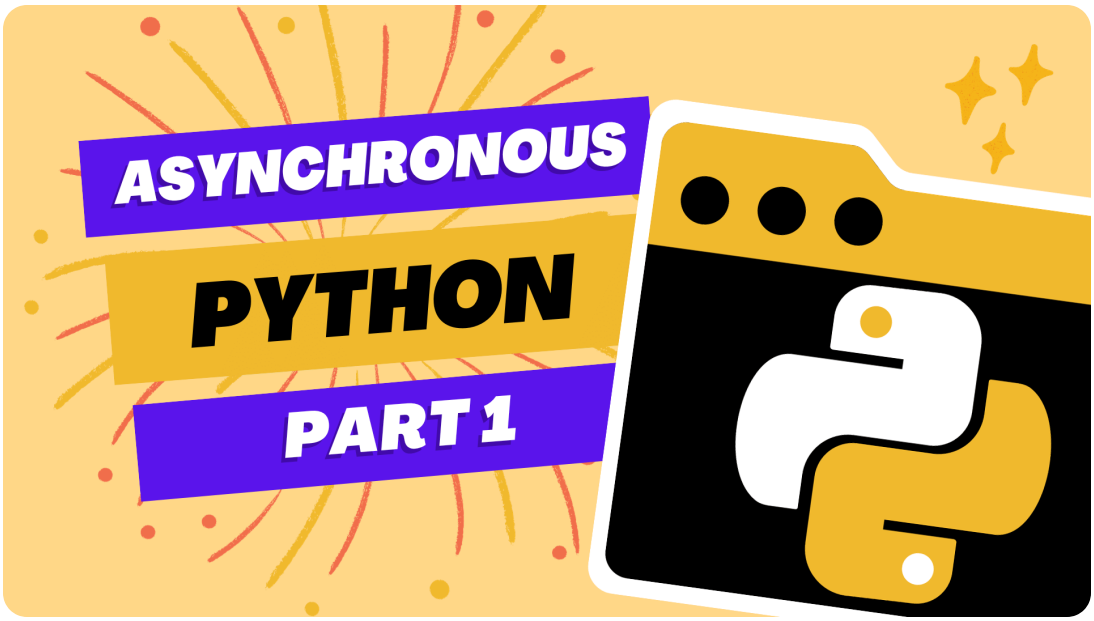
#java

#automation

#tricks

#tips

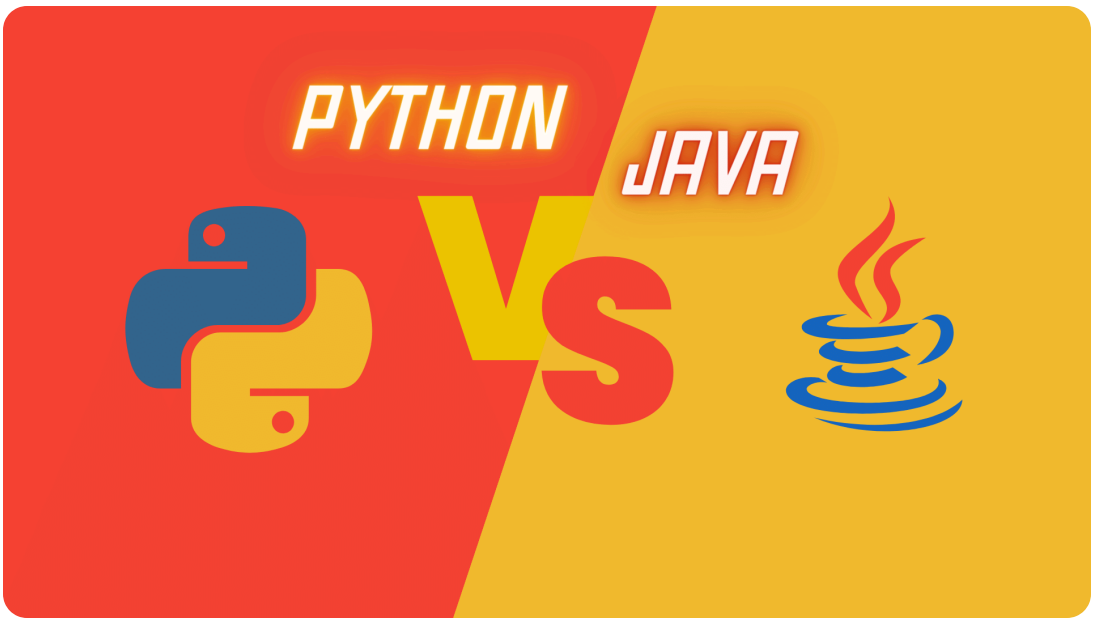
More Reads



ASYNCIO

Asynchronous Python [Part 1]: Mastering GIL, Multithreading, And Multiprocessing

January 06, 2024



PYTHON

Python For Java Programmers

December 30, 2023

PYTHONIFY

Get In Touch

contact@pythonify.com