PyPDF2 Library for Working with PDF Files in Python

BEGINNER PROJECT PYTHON

Introduction

PDF stands for Portable Document Format, is distinguished by its .pdf file extension. This format is predominantly utilized for document sharing due to its inherent property of preserving the original formatting, ensuring that documents appear consistent across various platforms, irrespective of the hardware, software, or operating system used. This consistency makes PDFs the format of choice for distributing, viewing, and ensuring the integrity of documents on a global scale.

Originally developed by Adobe, PDF has transcended its proprietary origins to become an open standard, governed by the International Organization for Standardization (ISO). This transition to an ISO standard has further cemented PDF's position as a cornerstone of digital document management, facilitating its adoption in a wide range of applications from academic publishing to business communications.

In this tutorial, we will learn how to work with PDF files in Python. The following topics will be covered:

- How to extract text from a PDF file.
- How to rotate pages of a PDF file.
- How to extract document information from a PDF file.
- How to split pages from a PDF file.
- How to encrypt PDF files.
- How to add a watermark to a PDF file.

This article was published as a part of the Data Science Blogathon.

Table of contents

- Frequently Asked Questions

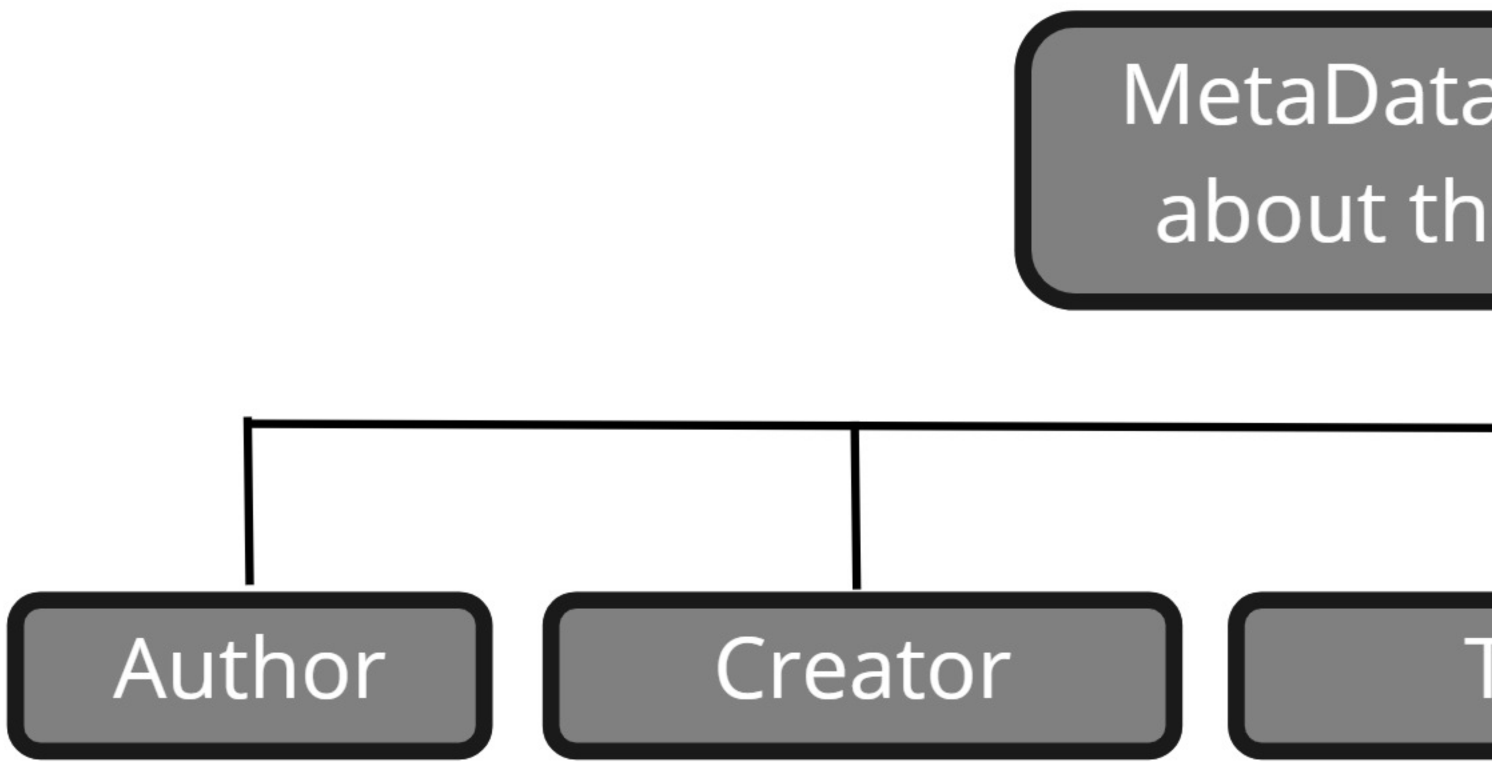
Some Common Libraries for PDFs in Python

There are many libraries available freely for working with PDFs:

- **PDFMiner**: It is an open-source tool for extracting text from PDF. It is used for performing analysis on the data. It can also be used as a PDF transformer or PDF parser.
- **PDFQuery**: It is a lightweight python wrapper around PDFMiner, lxml, and PyQuery. It is a fast, user-friendly PDF scraping library.
- **Tabula.py**: It is a python wrapper for tabula.java. It converts PDF files into Pandas' data frame and further all data manipulation operations can be performed on the data frame.
- **Xpdf**: It allows conversion of PDFs into text.
- **pdflib**: It is an extension of the poppler library with python bindings present in it.
- **Slate**: It is a Python package based on the PDFMiner and used for extraction of text from PDF.
- **PyPDF2**: It is a python library used for performing major tasks on PDF files such as extracting the document-specific information, merging the PDF files, splitting the pages of a PDF file, adding watermarks to a file, encrypting and decrypting the PDF files, etc. We will use the PyPDF2 library in this tutorial. It is a pure python library so it can run on any platform without any platform-related dependencies on any external libraries.

Getting Started with the PyPDF2 Library

PyPDF2 is a comprehensive Python library designed for the manipulation of PDF files. It enables users to create, modify, and extract content from PDF documents. Built entirely in Python, PyPDF2 does not rely on any external modules, making it an accessible tool for Python developers.



The library offers a dual API system to cater to different programming needs. The low-level API, inspired by Pygments, provides the capability to craft programs that can generate or manipulate documents with high efficiency. On the other hand, the high-level API, influenced by ReportLab, simplifies the creation of complex documents—ranging from forms to entire books or magazines—with minimal coding effort.

Key Features

- Transformation of PDFs into image formats like PNG or JPEG, as well as conversion into text files.
- Generation of new PDF documents from the ground up.
- Modification of existing PDFs through the addition, deletion, or alteration of pages.
- Advanced editing features such as page rotation, watermark addition, font adjustments, and more.
- The ability to secure documents with digital signatures, provided the necessary certificates are available.

Designed for efficiency, PyPDF2 leverages native C code for intensive operations like parsing, ensuring optimal performance without compromising the simplicity of its Pythonic interface. Additionally, the library is thread-safe, boasting a modest memory footprint approximately the size of Python's own (around 1MB), making it both powerful and lightweight for developers looking to manage PDF documents in their projects.

Use Cases of PyPDF2

PyPDF2's flexibility and command-line interface make it an ideal choice for integrating PDF processing into your workflow or Python projects. Below are some practical applications where PyPDF2 excels:

PDF Conversion to Word or Other Formats

Traditionally, converting PDFs into Word or other file formats requires specialized software for each conversion type, which can be inefficient, especially when handling multiple documents. PyPDF2 offers a streamlined alternative, enabling users to automate the conversion process within their Python scripts or via command-line instructions, significantly simplifying the task of converting PDF files into desired formats.

Merging Multiple PDF Documents

Whether you're compiling reports, combining chapters of a book, or consolidating financial statements, PyPDF2 simplifies the process of merging multiple PDF files into a single document. This capability is invaluable for creating cohesive documents from disparate sources, enhancing organization and accessibility.

Modifying PDF Document Contents

PyPDF2's functionality extends beyond basic file manipulation, allowing for detailed modifications within PDF documents. Users can add or remove pages, extract text for analysis, and even insert images or other objects into existing PDFs. This level of control makes PyPDF2 a versatile tool for tailoring documents to specific requirements.

Splitting PDFs into Smaller Segments

Large PDF documents can be unwieldy, making them difficult to share or process. PyPDF2 addresses this challenge by providing robust tools for splitting a single, large document into smaller, more manageable files. Whether you need to divide a document by page number, at regular intervals (every n pages), or according to document metadata such as author or title, PyPDF2 equips you with the necessary functionality.

Enhancements and Additional Use Cases

- **Extracting and Analyzing PDF Content**: PyPDF2 can be employed for text mining and analysis, extracting text from PDFs for use in data analysis projects, natural language processing, or content aggregation.
- **Automating Report Generation**: Automate the creation of reports by compiling data and text into professionally formatted PDFs, complete with dynamically added graphs, tables, and text.
- **Securing PDFs**: Implement security measures by encrypting PDFs, setting permissions, or adding digital signatures to protect sensitive information.
- **Custom PDF Creation**: Generate customized PDFs from scratch, using PyPDF2 to programmatically create documents that meet precise layout and content specifications.In summary, PyPDF2 is not just a library for manipulating PDF files; it's a comprehensive toolset that caters to a wide array of PDF-related tasks, from document conversion and merging to complex modifications and automated report generation. Its capabilities make it an indispensable resource for professionals, developers, and hobbyists alike, seeking to streamline their document management processes.

Also Read: Transforming PDF Summary using Python

Installing the PyPDF2 Library

To install PyPDF2, copy the following commands in the command prompt and run:

```
pip install PyPDF2
```

Getting the Document Details

PyPDF2 provides metadata about the PDF document. This can be useful information about the PDF files. Information like the author of the document, title, producer, Subject, etc is available directly.

To extract the above information, run the following code:

```
from PyPDF2 import PdfFileReader pdf_path=r"C:UsersDellDesktopTesting Tesseractexample.pdf" with open(pdf_path, 'rb') as f: pdf = PdfFileReader(f) information = pdf.getDocumentInfo() number_of_pages = pdf.getNumPages() print(information)
```

The output of the above code is as follows:

Let us format the output:

```
print("Author" +': ' + information.author) print("Creator" +': ' + information.creator) print("Producer" +':
' + information.producer)
```

Extracting Text from PDF

Extracting text from PDFs with PyPDF2 can be challenging due to its restricted capabilities in text extraction. The output generated by the code might not be well-formatted, often resulting in an output cluttered with line break characters, a consequence of PyPDF2’s constrained text extraction support.

To extract text, we will read the file and create a PDF object of the file.

```
# creating a pdf file object pdfFileObject = open(pdf_path, 'rb')
```

Then we will create a PDFReader class object and pass PDF File Object to it.

# creating a pdf reader object

```
pdfReader = PyPDF2.PdfFileReader(pdfFileObject)
```

And Finally, we will extract each page and concatenate the text of each page.

```
text='' for i in range(0,pdfReader.numPages): # creating a page object pageObj = pdfReader.getPage(i) #
extracting text from page text=text+pageObj.extractText() print(text)
```

The output text is as follows:

Rotating the pages of a PDF

To rotate a page of a PDF file and save it another file, copy the following code and run it.

```
pdf_read = PdfFileReader(r"C:UsersDellDesktopstory.pdf") pdf_write = PdfFileWriter() # Rotate page 90 degrees
to the right page1 = pdf_read.getPage(0).rotateClockwise(90) pdf_write.addPage(page1) with
```

```
open(r'C:UsersDellDesktoprotate_pages.pdf', 'wb') as fh: pdf_write.write(fh)
```

Merging PDF files in Python

We can also merge two or more PDF files using the following commands:

```
pdf_read = PdfFileReader(r”C:UsersDellDesktopstory.pdf”) pdf_write = PdfFileWriter() # Rotate page 90 degrees
to the right page1 = pdf_read.getPage(0).rotateClockwise(90) pdf_write.addPage(page1) with
open(r'C:UsersDellDesktoprotate_pages.pdf', 'wb') as fh: pdf_write.write(fh)
```

The output PDF is shown below:

Splitting the Pages of PDF

We can split a PDF into separate pages and save them again as PDFs.

```
fname = os.path.splitext(os.path.basename(pdf_path))[0] for page in range(pdf.getNumPages()): pdfwrite = PdfFileWriter() pdfwrite.addPage(pdf.getPage(page)) outputfilename = '{}_page_{}.pdf'.format( fname, page+1) with open(outputfilename, 'wb') as out: pdfwrite.write(out) print('Created: {}'.format(outputfilename))
```

```
pdf = PdfFileReader(pdf_path)
```

Encrypting a PDF File

Encryption of a PDF file means adding a password to the file. Each time the file is opened, it prompts to give the password for the file. It allows the content to be password protected. The following popup comes up:

We can use the following code for the same:

```
for page in range(pdf.getNumPages()): pdfwrite.addPage(pdf.getPage(page)) pdfwrite.encrypt(user_pwd=password, owner_pwd=None, use_128bit=True) with open(outputpdf, 'wb') as fh: pdfwrite.write(fh)
```

Adding a Watermark to the PDF File

A watermark is an identifying image or pattern that appears on each page. It can be a company logo or any strong information to be reflected on each page. To add a watermark to each page of the PDF, copy the following code and run.

```
originalfile = r"C:UsersDellDesktopTesting Tesseractexample.pdf" watermark = r"C:UsersDellDesktopTesting
Tesseractwatermark.pdf" watermarkedfile = r"C:UsersDellDesktopTesting Tesseractwatermarkedfile.pdf" watermark
= PdfFileReader(watermark) watermarkpage = watermark.getPage(0) pdf = PdfFileReader(originalfile) pdfwrite =
PdfFileWriter() for page in range(pdf.getNumPages()): pdfpage = pdf.getPage(page)
pdfpage.mergePage(watermarkpage) pdfwrite.addPage(pdfpage) with open(watermarkedfile, 'wb') as fh:
pdfwrite.write(fh)
```

The above code reads two files- the input file and the watermark. Then after reading each page it attaches the watermark to each page and saves the new file in the same location.

Conclusion

PyPDF2 stands out as a highly accessible solution for PDF file conversion, celebrated for its open-source nature and integration capabilities. Its comprehensive online documentation, hosted on GitHub, ensures that even those pressed for time can quickly find their way through setup and execution, streamlining the learning curve with well-organized docs and examples. For those seeking further assistance or looking to contribute, the PyPDF2 community on GitHub welcomes inquiries and contributions, fostering an environment of support and continuous improvement.

This library is not only user-friendly but also designed with automation and integration in mind, making it a go-to choice for developers looking to incorporate PDF manipulation into their workflows or applications. Since PyPDF2 is available on PyPI, installing it is straightforward for any Python project, and its compatibility with HTML and other formats enhances its versatility in handling various document conversion tasks.

With no dependencies other than Python, PyPDF2 promises exceptional portability across different operating systems, ensuring developers can deploy it in diverse environments without compatibility issues. The BSD-style license under which PyPDF2 is released offers the flexibility to include it within commercial software packages without legal concerns.

In essence, PyPDF2 is an invaluable tool for Python developers interested in automating PDF manipulation, offering an optimal blend of ease of use, efficiency, and adaptability. Whether you're generating reports, converting documents, or integrating PDF functionalities into larger systems, PyPDF2's robust feature set and supportive community make it a highly recommended resource.

It never goes without saying:

```
Thanks for reading!
```

Frequently Asked Questions

Q1. Can I use Python 3 for PDF manipulation?

A. Yes, Python 3 supports various libraries for PDF manipulation, such as PyPDF2, PDFMiner, and pdflib. These libraries allow you to perform operations like extracting text, html, merging, splitting, and encrypting PDFs in a Python 3 environment.

Q2. What is PyPDF for Python?

A. PyPDF is a term often used to refer to libraries like PyPDF2 and PyPDF4, which are Python libraries for working with PDF files. They provide functionalities for extracting information, merging, splitting, encrypting, and decrypting PDF documents.

Q3. Can I append text to a PDF using Python?

A. Appending text directly to a PDF is complex due to the format's nature. Instead, you can use Python to add text as annotations or by creating a new PDF with the text and then merging it with the original PDF using PyPDF2.

Q4. Is it possible to decrypt PDF files with Python?

A. Yes, PyPDF2 allows you to decrypt PDF files, provided you have the necessary permissions and the password.

Q5. How do I manipulate Excel files in relation to PDFs in Python?

A. To work with Excel files and PDFs, you can use libraries like Pandas to manipulate Excel data and then use ReportLab or PyPDF2 to generate or manipulate PDFs based on that data.

Q6. What are some tips for working with PDFs on Linux using Python?

A. On Linux, ensure you have dependencies installed for libraries like PyPDF2 or PDFMiner. Use the Linux package manager to install any required system libraries for advanced operations like OCR.

The media shown in this article are not owned by Analytics Vidhya and are used at the Author's discretion.

Article Url - https://www.analyticsvidhya.com/blog/2021/09/pypdf2-library-for-working-with-pdf-files-in-python/

Siddharth Sachdeva

Computer science enthusiast