Working with PDF files in Python

Read Discuss Courses Practice

All of you must be familiar with what PDFs are. In fact, they are one of the most important and widely used digital media. PDF stands for **Portable Document Format**. It uses **.pdf** extension. It is used to present and exchange documents reliably, independent of software, hardware, or operating system. Invented by **Adobe**, PDF is now an open standard maintained by the International Organization for Standardization (ISO). PDFs can contain links and buttons, form fields, audio, video, and business logic.
In this article, we will learn, how we can do various operations like:

- Extracting text from PDF
- Rotating PDF pages
- Merging PDFs
- Splitting PDF
- Adding watermark to PDF pages

Installation: Using simple python scripts!
We will be using a third-party module, PyPDF2.
PyPDF2 is a python library built as a PDF toolkit. It is capable of:

- Extracting document information (title, author, ...)
- Splitting documents page by page
- Merging documents page by page
- Cropping pages
- Merging multiple pages into a single page
- Encrypting and decrypting PDF files

To install PyPDF2, run the following command from the command line:

```
pip install PyPDF2
```

This module name is case-sensitive, so make sure the **y** is lowercase and everything else is uppercase. All the code and PDF files used in this tutorial/article are available here.

1. Extracting text from PDF file

Python

```python
# importing required modules
import PyPDF2

# creating a pdf file object
pdfFileObj = open('example.pdf', 'rb')

# creating a pdf reader object
pdfReader = PyPDF2.PdfReader(pdfFileObj)

# printing number of pages in pdf file
print(len(pdfReader.pages))

# creating a page object
pageObj = pdfReader.pages[0]

# extracting text from page
print(pageObj.extract_text())
```

```
pdfFileObj.close()
```

The output of the above program looks like this:

```
20
PythonBasics
S.R.Doty
August27,2008
Contents

1Preliminaries
4
1.1WhatisPython?.................................
..4
1.2Installationanddocumentation...................
.........4 [and some more lines...]
```

Let us try to understand the above code in chunks:

```
pdfFileObj = open('example.pdf', 'rb')
```

- We opened the **example.pdf** in binary mode. And saved the file object as **pdfFileObj**.

```
pdfReader = PyPDF2.PdfReader(pdfFileObj)
```

- Here, we create an object of **PdfReader** class of PyPDF2 module and pass the PDF file object & get a PDF reader object.

```
print(len(pdfReader.pages))
```

- **pages** property gives the number of pages in the PDF file. For example, in

```
pageObj = pdfReader.pages[0]
```

- Now, we create an object of **PageObject** class of PyPDF2 module. PDF reader object has function **pages[]** which takes page number (starting from index 0) as argument and returns the page object.

```
print(pageObj.extract_text())
```

- Page object has function **extract_text()** to extract text from the PDF page.

```
pdfFileObj.close()
```

- At last, we close the PDF file object.

Note: While PDF files are great for laying out text in a way that's easy for people to print and read, they're not straightforward for software to parse into plaintext. As such, PyPDF2 might make mistakes when extracting text from a PDF and may even be unable to open some PDFs at all. It isn't much you can do about this, unfortunately. PyPDF2 may simply be unable to work with some of your particular PDF files.

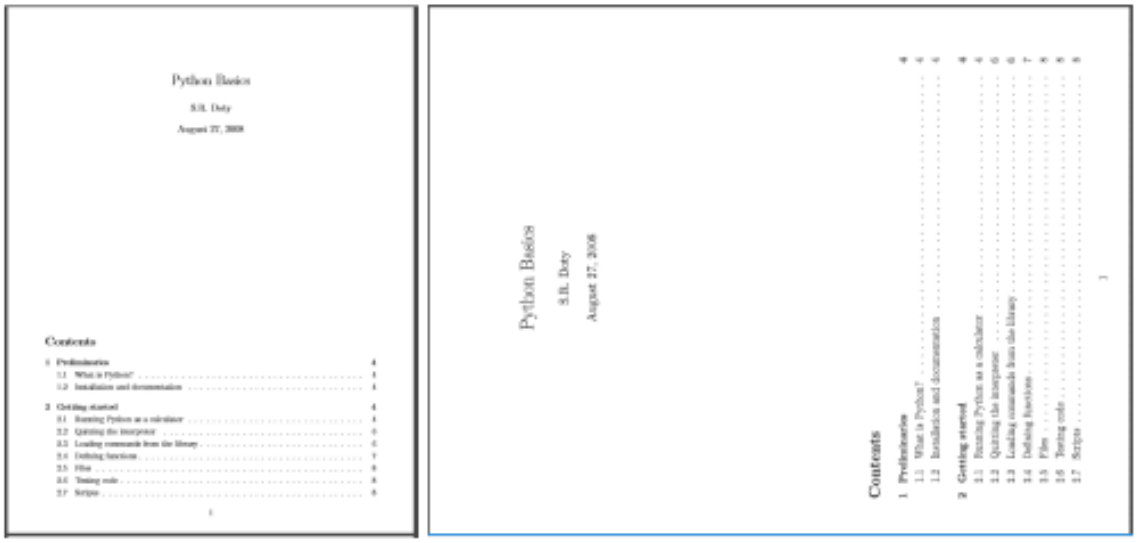
2. Rotating PDF pages

Python

```python
# importing the required modules
import PyPDF2

def PDFrotate(origFileName, newFileName, rotation):

    # creating a pdf File object of original pdf
```

```
    pdfReader = PyPDF2.PdfReader(pdfFileObj)

    # creating a pdf writer object for new pdf
    pdfWriter = PyPDF2.PdfWriter()

    # rotating each page
    for page in range(len(pdfReader.pages)):

        # creating rotated page object
        pageObj = pdfReader.pages[page]
        pageObj.rotate(rotation)

        # adding rotated page object to pdf writer
        pdfWriter.add_page(pageObj)

        # new pdf file object
        newFile = open(newFileName, 'wb')

        # writing rotated pages to new file
        pdfWriter.write(newFile)

    # closing the original pdf file object
    pdfFileObj.close()

    # closing the new pdf file object
    newFile.close()


def main():

    # original pdf file name
    origFileName = 'example.pdf'

    # new pdf file name
    newFileName = 'rotated_example.pdf'

    # rotation angle
    rotation = 270

    # calling the PDFrotate function
    PDFrotate(origFileName, newFileName, rotation)

if __name__ == "__main__":
    # calling the main function
    main()
```

Here, you can see how the first page of **rotated_example.pdf** looks like (right image) after rotation:



There is a tool called UPDF that can be used to [Rotate a PDF](#).

Some important points related to the above code:

- For rotation, we first create a PDF reader object of the original PDF.

```
pdfWriter = PyPDF2.PdfWriter()
```

- Rotated pages will be written to a new PDF. For writing to PDFs, we use the object of **PdfWriter** class of PyPDF2 module.

```
for page in range(len(pdfReader.pages)):
        pageObj = pdfReader.pages[page]
        pageObj.rotate(rotation)
        pdfWriter.add_page(pageObj)
```

- Now, we iterate each page of the original PDF. We get page object by **.pages[]** method of PDF reader class. Now, we rotate the page by **rotate()**

using **addage()** method of PDF writer class by passing the rotated page object.

```
newFile = open(newFileName, 'wb')
pdfWriter.write(newFile)
pdfFileObj.close()
newFile.close()
```

- Now, we have to write the PDF pages to a new PDF file. Firstly, we open the new file object and write PDF pages to it using **write()** method of PDF writer object. Finally, we close the original PDF file object and the new file object.

3. Merging PDF files

Python

```python
# importing required modules
import PyPDF2


def PDFmerge(pdfs, output):
    # creating pdf file merger object
    pdfMerger = PyPDF2.PdfMerger()

    # appending pdfs one by one
    for pdf in pdfs:
        pdfMerger.append(pdf)

        # writing combined pdf to output pdf file
        with open(output, 'wb') as f:
            pdfMerger.write(f)


def main():
    # pdf files to merge
    pdfs = ['example.pdf', 'rotated_example.pdf']

    # output pdf file name
    output = 'combined_example.pdf'
```

```
if __name__ == "__main__":
    # calling the main function
    main()
```

The output of the above program is a combined PDF, **combined_example.pdf**, obtained by merging **example.pdf** and **rotated_example.pdf**.

- Let us have a look at important aspects of this program:

```
pdfMerger = PyPDF2.PdfMerger()
```

- For merging, we use a pre-built class, **PdfMerger** of PyPDF2 module. Here, we create an object **pdfMerger** of PDF merger class

```
for pdf in pdfs:
    pdfmerger.append(open(focus, "rb"))
```

- Now, we append file object of each PDF to PDF merger object using the **append()** method.

```
with open(output, 'wb') as f:
    pdfMerger.write(f)
```

- Finally, we write the PDF pages to the output PDF file using **write** method of PDF merger object.

4. Splitting PDF file

Python

```
# importing the required modules
```

```
    # creating input pdf file object
    pdfFileObj = open(pdf, 'rb')

    # creating pdf reader object
    pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

    # starting index of first slice
    start = 0

    # starting index of last slice
    end = splits[0]


    for i in range(len(splits)+1):
        # creating pdf writer object for (i+1)th split
        pdfWriter = PyPDF2.PdfFileWriter()

        # output pdf file name
        outputpdf = pdf.split('.pdf')[0] + str(i) + '.pdf'

        # adding pages to pdf writer object
        for page in range(start,end):
            pdfWriter.addPage(pdfReader.getPage(page))

            # writing split pdf pages to pdf file
            with open(outputpdf, "wb") as f:
                pdfWriter.write(f)

            # interchanging page split start position for next split
            start = end
            try:
                # setting split end position for next split
                end = splits[i+1]
            except IndexError:
                # setting split end position for last split
                end = len(pdfReader.pages)

    # closing the input pdf file object
    pdfFileObj.close()

def main():
    # pdf file to split
    pdf = 'example.pdf'

    # split page positions
    splits = [2,4]
```

```
if __name__ == "__main__":
    # calling the main function
    main()
```

Output will be three new PDF files with **split 1 (page 0,1), split 2(page 2,3), split 3(page 4-end)**.
No new function or class has been used in the above python program. Using simple logic and iterations, we created the splits of passed PDF according to the passed list **splits**.

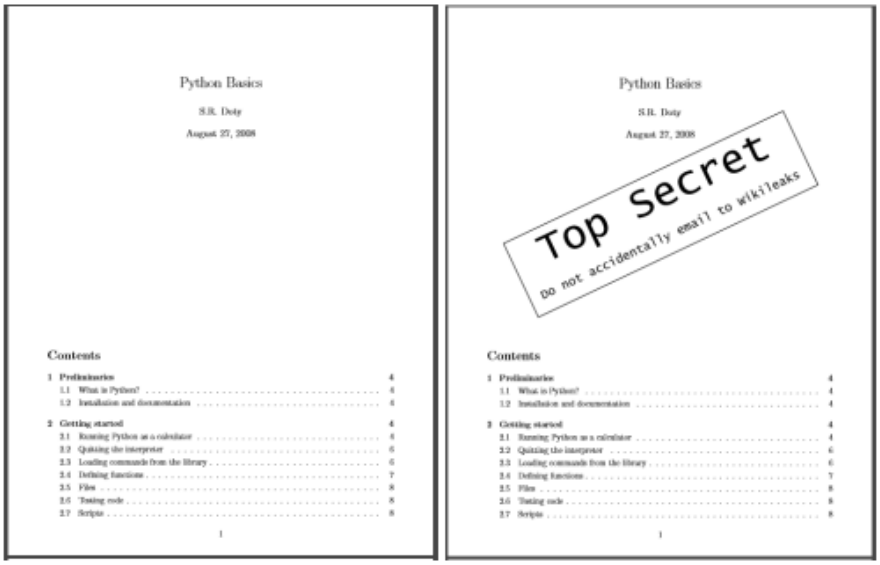
5. Adding watermark to PDF pages

Python

```
# importing the required modules
import PyPDF2

def add_watermark(wmFile, pageObj):
    # opening watermark pdf file
    wmFileObj = open(wmFile, 'rb')

    # creating pdf reader object of watermark pdf file
    pdfReader = PyPDF2.PdfReader(wmFileObj)

    # merging watermark pdf's first page with passed page object.
    pageObj.merge_page(pdfReader.pages[0])

    # closing the watermark pdf file object
    wmFileObj.close()

    # returning watermarked page object
    return pageObj

def main():
    # watermark pdf file name
    mywatermark = 'watermark.pdf'

    # original pdf file name
    origFileName = 'example.pdf'
```

```
    # creating pdf File object of original pdf
    pdfFileObj = open(origFileName, 'rb')

    # creating a pdf Reader object
    pdfReader = PyPDF2.PdfReader(pdfFileObj)

    # creating a pdf writer object for new pdf
    pdfWriter = PyPDF2.PdfWriter()

    # adding watermark to each page
    for page in range(len(pdfReader.pages)):
        # creating watermarked page object
        wmpageObj = add_watermark(mywatermark, pdfReader.pages[page])

        # adding watermarked page object to pdf writer
        pdfWriter.add_page(wmpageObj)

        # new pdf file object
        newFile = open(newFileName, 'wb')

        # writing watermarked pages to new file
        pdfWriter.write(newFile)

    # closing the original pdf file object
    pdfFileObj.close()
    # closing the new pdf file object
    newFile.close()

if __name__ == "__main__":
    # calling the main function
    main()
```

Here is how the first page of original (left) and watermarked (right) PDF file looks like:



- All the process is same as the page rotation example. Only difference is:

```
wmpageObj = add_watermark(mywatermark, pdfReader.pages[page])
```

- Page object is converted to watermarked page object using **add_watermark()** function.
- Let us try to understand **add_watermark()** function:

```
wmFileObj = open(wmFile, 'rb')
pdfReader = PyPDF2.PdfReader(wmFileObj)
pageObj.merge_page(pdfReader.pages[0])
wmFileObj.close()
return pageObj
```

- Foremost, we create a PDF reader object of **watermark.pdf**. To the passed page object, we use **merge_page()** function and pass the page object of the first page of the watermark PDF reader object. This will overlay the watermark over the passed page object.

And here we reach the end of this long tutorial on working with PDF files in python.

References:

- https://automatetheboringstuff.com/chapter13/
- https://pythonhosted.org/PyPDF2/

If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.
Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

If you have better suggestions about the products/services/tools/brands listed above or feel like something missing, please Contact Us and share your suggestions.

Last Updated : 05 Sep, 2023

26

Similar Reads

Send PDF File through Email using pdf-mail module

Interact with PDF with PDF ChatBot

How to Crack PDF Files in Python?

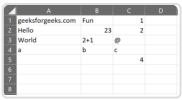
How to Scrape all PDF files in a Website?

How to create PDF files in Flask

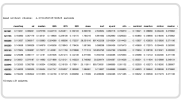
Working with csv files in Python

Working with zip files in Python

Working with wav files in Python using Pydub



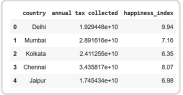
Working with Excel files in Python using Xlwings



Working with large CSV files in Python

Related Tutorials

OpenAI Python API - Complete Guide



Pandas AI: The Generative AI Python Library

Python for Kids - Fun Tutorial to Learn Python Programming

Data Analysis Tutorial

Flask Tutorial

Previous

ZeroDivisionError: float division by zero in Python

Next

Inplace vs Standard Operators in Python

Article Contributed By :

N Nikhil Kumar

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : Vijay Sirra, himanshutiwarihouse, ericcheng26, richasalan57, chprudhvi1003

Article Tags : Listicles , python , Python

Improve Article Report Issue

GeeksforGeeks
Sanchhaya Education Private Limited
A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org