Working with PDF files in Python

Difficulty Level : Hard ● Last Updated : 27 Nov, 2020

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

using simple python scripts!

Installation

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
- and more!

To install PyPDF2, run following command from 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

```
# importing required modules
import PyPDF2

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

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

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

# creating a page object
pageObj = pdfReader.getPage(0)

# extracting text from page
print(pageObj.extractText())

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

Output of 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.PdfFileReader(pdfFileObj)`

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

- `print(pdfReader.numPages)`

  numPages property gives the number of pages in the pdf file. For example, in our case, it is 20 (see first line of output).

- `pageObj = pdfReader.getPage(0)`

  Now, we create an object of **PageObject** class of PyPDF2 module. pdf reader object has function **getPage()** which takes page number (starting form index 0) as argument and returns the page object.

- `print(pageObj.extractText())`

Page object has function **extractText()** to extract text from the pdf page.

- `pdfFileObj.close()`

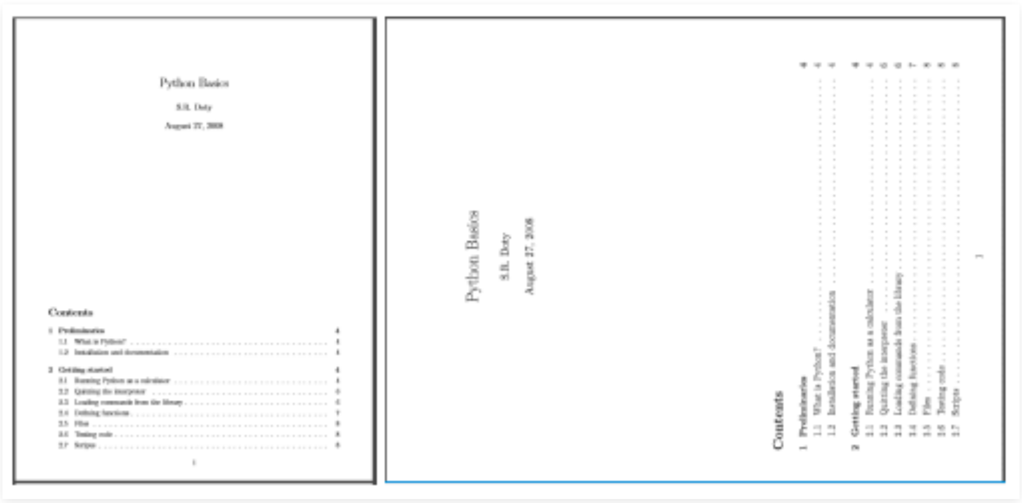
At last, we close the pdf file object.

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. There 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

```
# importing the required modules
import PyPDF2

def PDFrotate(origFileName, newFileName, rotation):

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

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

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

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

        # creating rotated page object
        pageObj = pdfReader.getPage(page)
        pageObj.rotateClockwise(rotation)

        # adding rotated page object to pdf writer
        pdfWriter.addPage(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:


Python Basics

S.R. Doty

August 27, 2008

Contents

1 Preliminaries 4
1.1 What is Python? 4
1.2 Installation and documentation 4

2 Getting started 4
2.1 Running Python as a calculator 4
2.2 Quitting the interpreter 6
2.3 Loading commands from the library 6
2.4 Defining functions 7
2.5 Files 8
2.6 Testing code 8
2.7 Scripts 8

1

Some important points related to above code:

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

-
  ```
  pdfWriter = PyPDF2.PdfFileWriter()
  ```

  Rotated pages will be written to a new pdf. For writing to pdfs, we use object of **PdfFileWriter** class of PyPDF2 module.

-
  ```
  for page in range(pdfReader.numPages):
          pageObj = pdfReader.getPage(page)
          pageObj.rotateClockwise(rotation)
          pdfWriter.addPage(pageObj)
  ```

  Now, we iterate each page of original pdf. We get page object by **getPage()** method of pdf reader class. Now, we rotate the page by **rotateClockwise()** method of page object class. Then, we add page to pdf writer object using **addPage()** 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

```
# importing required modules
import PyPDF2

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

    # appending pdfs one by one
    for pdf in pdfs:
        with open(pdf, 'rb') as f:
            pdfMerger.append(f)

    # 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'

    # calling pdf merge function
    PDFmerge(pdfs = pdfs, output = output)

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

Output of 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.PdfFileMerger()`

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

-
  ```
  for pdf in pdfs:
          with open(pdf, 'rb') as f:
              pdfMerger.append(f)
  ```

  Now, we append file object of each pdf to pdf merger object using **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

```
# importing the required modules
import PyPDF2

def PDFsplit(pdf, splits):
    # 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 = pdfReader.numPages

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

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

    # split page positions
    splits = [2,4]

    # calling PDFsplit function to split pdf
    PDFsplit(pdf, splits)

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)**.

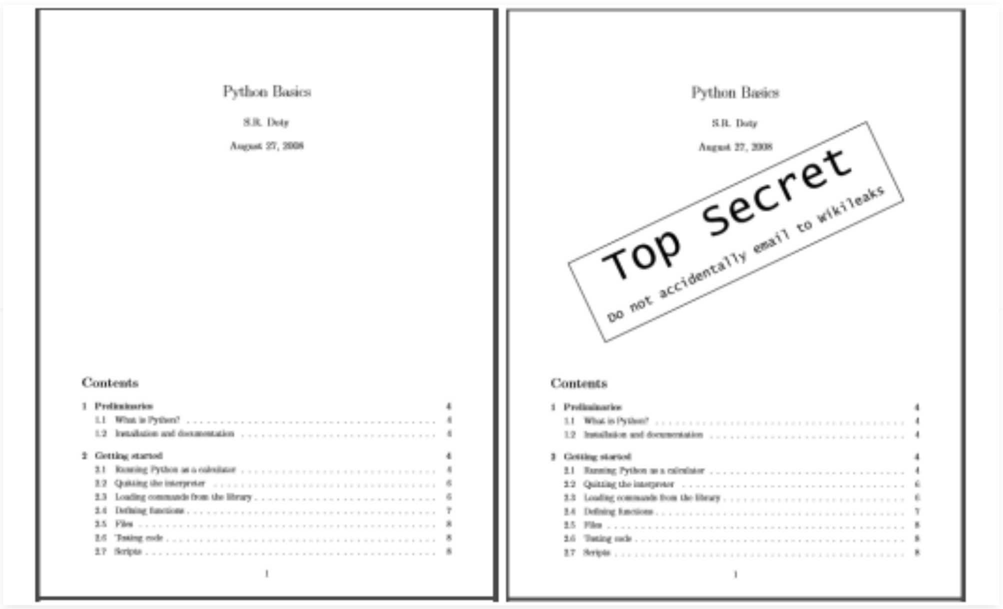
No new function or class has been used in above python program. Using simple logic and iterations, we created the splits of passed **pdf** according to the passed list **splits**.

5. Adding watermark to PDF pages

```
# 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.PdfFileReader(wmFileObj)

    # merging watermark pdf's first page with passed page object
    pageObj.mergePage(pdfReader.getPage(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'

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

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

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

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

    # adding watermark to each page
    for page in range(pdfReader.numPages):
        # creating watermarked page object
        wmpageObj = add_watermark(mywatermark, pdfReader.getPage

        # adding watermarked page object to pdf writer
        pdfWriter.addPage(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 first page of original (left) and watermarked (right) pdf file looks like:



Python Basics

S.R. Doty

August 27, 2008

Top Secret

Do not accidentally email to wikileaks

Contents

1

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

  ```
  wmpageObj = add_watermark(mywatermark, pdfReader.getPage
  ```

  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.PdfFileReader(wmFileObj)
  pageObj.mergePage(pdfReader.getPage(0))
  wmFileObj.close()
  return pageObj
  ```

  First of all, we create a pdf reader object of **watermark.pdf**. To the passed page object, we use **mergePage()** function and pass the page object of first page of 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.
Now, you can easily create your own PDF manager!

References:

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

This article is contributed by **Nikhil Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

|< Previous

Python | Reading contents of PDF using OCR (Optical Character Recognition)

Next >|

Extract text from PDF File using Python

RECOMMENDED ARTICLES

Page : 1 2 3

01 Send PDF File through Email using pdf-mail module
01, May 20

02 How to Crack PDF Files in Python?
25, Dec 20

03 Working with csv files in Python
09, Dec 16

04 Working with wav files in Python using Pydub
15, Aug 20

05 Working with zip files in Python
14, Dec 16

06 Working with Excel files in Python using Xlwings
03, Jan 21

07 Working with excel files using Pandas
13, Feb 20

08 Why are there two folders - Program Files and Program Files (x86) in 64-bit Windows OS?
17, Jul 19

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Hard

Easy Normal Medium Hard Expert

Improved By : Vijay Sirra

Article Tags : GBlog , Python

Improve Article Report Issue

C

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org