

PDF e Word em Python

Roger Tiago

Abril 2016

Sumário

0.1	Introdução	2
0.2	Documentos	2
1	PDF	3
1.1	Documentos em PDF	3
1.1.1	Extraindo textos de PDF's	3
1.1.2	Descriptografando PDF's	4
1.1.3	Criando PDF's	4
1.1.4	Copiando páginas	4
1.1.5	Rotacionando páginas	5
1.1.6	Sobrepondo páginas	5
1.1.7	Criptografando	6
2	Word	7
2.1	Documentos em .docx	7

0.1 Introdução

Este documento visa auxiliar o trabalho com arquivos PDF e .docx utilizando Python.

0.2 Documentos

Documentos em PDF e .docx são arquivos binários, oque dificulta o trabalho com esses arquivos, porem o python conta com módulos para facilitar estes processos.

Capítulo 1

PDF

1.1 Documentos em PDF

Documentos em PDF são arquivos .pdf, PDF's podem conter textos, imagens e vídeos, aqui será focado em editar e ler textos de arquivos PDF, para isso usaremos o módulo PyPDF2.

1.1.1 Extrair textos de PDF's

O módulo PyPDF2 não consegue extrair imagens, gráficos e outras mídias dos arquivos, porém pode extrair textos e retornar como uma string.

Para extrair um texto de um PDF é necessário a importação do módulo PyPDF2, cria-se um objeto para ler o pdf em modo binário e armazenar, é possível obter o número de páginas do documento que começam a ser contadas a partir do 0, para extrair o texto propriamente dito usamos a função `.getPage(n)` no objeto criado anteriormente onde `n` representa a página que desejamos o texto (contada a partir do 0), após isso usa-se a função `.extractText()` no objeto que contém o pdf já convertido em binário e o índice da página, será mostrada uma string contendo o texto.

Código exemplo

```
>>> import PyPDF2
>>> pdfFileObj = open('meetingminutes.pdf', 'rb')
>>> pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
>>> pdfReader.numPages
>>> pageObj = pdfReader.getPage(0)
>>> pageObj.extractText()
```

1.1.2 Descriptografando PDF's

Alguns PDF's contam com um sistema de criptografia, e não será possível ler o documento até que seja descriptografado. Todo objeto PdfFileReader conta com um atributo isEncrypted que se tiver o valor True não será possível ler o PDF. Neste caso para ler um PDF criptografado basta usar a função .decrypt() no objeto com a senha do documento como argumento em string, se a senha estiver correta será possível acessar o documento, caso contrário ainda haverá erros. A função .decrypt() funciona apenas para o objeto e não para o arquivo em si.

Código exemplo

```
>>> import PyPDF2
>>> pdfReader = PyPDF2.PdfFileReader(open('encrypted.pdf', 'rb'))
>>> pdfReader.isEncrypted
>>> pdfReader.getPage(0)
```

1.1.3 Criando PDF's

PyPDF2 pode criar novos PDF's, no entanto não é possível escrever arbitrariamente, apenas copiar páginas de outros PDF's, rotacionar, sobrepor e criptografar. PyPDF2 não pode editar diretamente um PDF, é necessário criar um novo PDF e importar textos já existentes. Para fazer isso é necessário abrir um ou mais PDF's fontes como objetos com PdfFileReader, criar um novo objeto PdfFileWriter, copiar as páginas do objeto PdfFileReader para o objeto PdfFileWriter, finalmente usamos o objeto PdfFileWriter para escrever o PDF final. O objeto PdfFileWriter não cria o arquivo em si, para isto é necessário usar o método write().

1.1.4 Copiando páginas

É possível utilizar o PyPDF2 para copiar páginas de um documento PDF para outro, isto permite combinar múltiplos arquivos PDF's, cortar páginas indesejadas e reordenar as páginas.

Código exemplo

```
>>> import PyPDF2
>>> pdf1File = open('meetingminutes.pdf', 'rb')
>>> pdf2File = open('meetingminutes2.pdf', 'rb')
```

```

>>> pdf1Reader = PyPDF2.PdfFileReader(pdf1File)
>>> pdf2Reader = PyPDF2.PdfFileReader(pdf2File)
>>> pdfWriter = PyPDF2.PdfFileWriter()
>>> for pageNum in range(pdf1Reader.numPages):
>>>     pageObj = pdf1Reader.getPage(pageNum)
>>>     pdfWriter.addPage(pageObj)
>>> for pageNum in range(pdf2Reader.numPages):
>>>     pageObj = pdf2Reader.getPage(pageNum)
>>>     pdfWriter.addPage(pageObj)
>>> pdfOutputFile = open('combinedminutes.pdf', 'wb')
>>> pdfWriter.write(pdfOutputFile)
>>> pdfOutputFile.close()
>>> pdf1File.close()
>>> pdf2File.close()

```

1.1.5 Rotacionando páginas

É possível rotacionar páginas de um PDF em incrementos de 90° com os métodos `rotateClockwise()` e `rotateCounterClockwise()` passando o argumento 90, 180 ou 270.

Código exemplo

```

>>> import PyPDF2
>>> minutesFile = open('meetingminutes.pdf', 'rb')
>>> pdfReader = PyPDF2.PdfFileReader(minutesFile)
>>> page = pdfReader.getPage(0)
>>> page.rotateClockwise(90)
>>> pdfWriter = PyPDF2.PdfFileWriter()
>>> pdfWriter.addPage(page)
>>> resultPdfFile = open('rotatedPage.pdf', 'wb')
>>> pdfWriter.write(resultPdfFile)
>>> resultPdfFile.close()
>>> minutesFile.close()

```

1.1.6 Sobrepondo páginas

Utilizando PyPDF2 é possível sobrepor páginas, facilmente adicionando logos ou marcas d'água a múltiplas páginas.

Código exemplo

```
>>> import PyPDF2
>>> minutesFile = open('meetingminutes.pdf', 'rb')
>>> pdfReader = PyPDF2.PdfFileReader(minutesFile)
>>> minutesFirstPage = pdfReader.getPage(0)
>>> pdfWatermarkReaderPyPDF2.PdfFileReader(open('watermark.pdf', 'rb'))
>>> minutesFirstPage.mergePage(pdfWatermarkReader.getPage(0))
>>> pdfWriter = PyPDF2.PdfFileWriter()
>>> pdfWriter.addPage(minutesFirstPage)

>>> for pageNum in range(1, pdfReader.numPages):
>>>     pageObj = pdfReader.getPage(pageNum)
>>>     pdfWriter.addPage(pageObj)
>>> resultPdfFile = open('watermarkedCover.pdf', 'wb')
>>> pdfWriter.write(resultPdfFile)
>>> minutesFile.close()
>>> resultPdfFile.close()
```

1.1.7 Criptografando

O objeto PdfFileWriter pode adicionar uma senha a um arquivo PDF com o método `.encrypt()` passando como argumento a senha desejada.

Código exemplo

```
>>> import PyPDF2
>>> pdfFile = open('meetingminutes.pdf', 'rb')
>>> pdfReader = PyPDF2.PdfFileReader(pdfFile)
>>> pdfWriter = PyPDF2.PdfFileWriter()
>>> for pageNum in range(pdfReader.numPages):
>>>     pdfWriter.addPage(pdfReader.getPage(pageNum))

>>> pdfWriter.encrypt('swordfish')
>>> resultPdf = open('encryptedminutes.pdf', 'wb')
>>> pdfWriter.write(resultPdf)
>>> resultPdf.close()
```

Capítulo 2

Word

2.1 Documentos em .docx

Python pode criar e modificar documentos .docx, para isso é usado o módulo python-docx. Documentos .docx possuem muita estrutura, as quais são representadas por 3 diferentes tipos de data no módulo Python-Docx. No nível mais alto o objeto Document contém o documento inteiro e uma lista do objeto Paragraph para os parágrafos no documento(um paragrafo novo é criado a cada 'enter' teclado), cada um desses objetos Paragraph contém um ou mais objetos Run. Um texto em .docx é mais que uma string, ele contém fonte, cor e tamanho, um objeto Run é necessário sempre que o texto muda de estilo;