Capítulo 13 - Trabalhando com documentos em PDF's e Word

Laura Ribeiro

Maio de 2017

Segundo Sprint - Engenharia de Software/2017-1

Professor: Marcelo Rodrigues De Sousa

1 Introdução

PDF e Word são documentos de arquivos/tipo binários, que podem ser muito mais complexo do que aparentam. Em adição em um texto, há uma gigantesca abrangência de core, fontes, tipo, informações, linguagens, símbolos, etc.

Se você quiser que seus programas leiam ou escrevam documentos em PDFs ou Word, você precisará de uma simples ferramenta, a função open().

Felizmente, o Python possui modelos que facilitam o uso e a interação de documentos em PDFs e Word. Alguns deles são PyPDF2 e Python-Docx.

2 Documentos PDF

PDF utiliza um tipo de formato para o documento e o arquivo de extensão, o .pdf. Embora os PDFs suportem muitos recursos, serão abordadas, apenas as funções que são mais utilizadas: a leitura e a escrita de documentos/textos em PDFs e a criação de novos PDFs a partir de documentos já existentes.

O módulo que você usará para trabalhar com PDFs é PyPDF2. Para instalá-lo, execute pip instalar PyPDF2 a partir da linha de comando. Verifique se está exatamente como citado, o comando é bastante sensível.

Se o módulo foi instalado corretamente, execute a importação PyPDF2 no shell interativo e não deverá exibir erros.

- Formato PDF problemático

Enquanto arquivos PDF são ótimos para serem lidos ou impressos, com facil entendimento, eles não são simples para o software analisar seu conteúdo. Como tal, PyPDF2 pode cometer erros ao extrair texto de um PDF e pode até mesmo ser incapaz de abrir alguns PDFs. O PyPDF2 pode simplesmente não conseguir trabalhar com alguns de seus arquivos PDF específicos. Mas isso é bem raro.

2.1 Extraindo Textos de PDFs

```
>> importPyPDF2
>> pdfFileObj = open('meetingminutes.pdf','rb')  #abrir o arquivo
>> pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
>> pdfReader.numPages  #número total de páginas
>> pageObj = pdfReader.getPage(0)
>> pageObj.extractText()  #retornar uma string do texto da página
```

2.2 Descriptografando PDFs

```
>> importPyPDF2
>> pdfReader = PyPDF2.PdfFileReader(open('encrypted.pdf','rb'))
>> pdfReader.isEncrypted  #retorna se o PDF está criptografado ou não
>> pdfReader.getPage(0)  #lê um PDF criptografado
>> pdfReader.decrypt('rosebud')  #chame a função decrypt()
    #passe a senha como uma string
>> pageObj = pdfReader.getPage(0)
```

2.3 Criando PDFs

- Abrir um ou mais PDFs existentes (de origem): PdfFileReader;
- Criar um novo objeto: PdfFileWriter;
- Copiar páginas dos objetos PdfFileReader para o objeto PdfFileWriter;
- Usar o objeto PdfFileWriter para gravar o PDF de saída.

2.4 Copiando Páginas

```
>> importPyPDF2
>> pdf1File = open('meetingminutes.pdf','rb')
>> pdf2File = open('meetingminutes2.pdf','rb')
>> pdf1Reader = PyPDF2.PdfFileReader(pdf1File)
>> pdf2Reader = PyPDF2.PdfFileReader(pdf2File)
>> pdfWriter = PyPDF2.PdfFileWriter()
>> forpageNuminrange(pdf1Reader.numPages) :
    pageObj = pdf1Reader.getPage(pageNum)
    pdfWriter.addPage(pageObj)
>> forpageNuminrange(pdf2Reader.numPages) :
    pageObj = pdf2Reader.getPage(pageNum)
    pdfWriter.addPage(pageObj)
>> pdfOutputFile = open('combinedminutes.pdf','wb')
>> pdfWriter.write(pdfOutputFile)
>> pdfOutputFile.close()
>> pdf1File.close()
>> pdf2File.close()
```

2.5 Como criptografar PDFs

Um objeto PdfFileWriter também pode adicionar criptografia a um documento PDF.

```
>> importPyPDF2
>> pdfFile = open('meetingminutes.pdf','rb')
>> pdfReader = PyPDF2.PdfFileReader(pdfFile)
>> pdfWriter = PyPDF2.PdfFileWriter()
>> forpageNuminrange(pdfReader.numPages) :
    pdfWriter.addPage(pdfReader.getPage(pageNum))
>> pdfWriter.encrypt('swordfish')
>> resultPdf = open('encryptedminutes.pdf','wb')
>> pdfWriter.write(resultPdf)
>> resultPdf.close()
```

3 Documentos Word

O Python pode criar e/ou modificar documentos do Word, que têm o tipo .docx, com o módulo python-docx. Você pode instalar o módulo executando Pip instalar *python-docx.*

O texto em um documento do Word é mais do que apenas uma sequência de caracteres. Tem fonte, tamanho, cor e outras informações de estilo associados a ele. Um estilo em Word é uma coleção desses atributos.

3.1 Lendo Documentos Word

Exemplo:

```
>> importdocx
>> doc = docx.Document('demo.docx')  #abrindo o documento
>> len(doc.paragraphs)  #retorna a quantidade de paragráfos
>> doc.paragraphs[0].text
>> doc.paragraphs[1].text
>> len(doc.paragraphs[1].runs)
>> doc.paragraphs[1].runs[0].text  #os objetos 'run' têm um atributo de texto
>> doc.paragraphs[1].runs[1].text
>> doc.paragraphs[1].runs[2].text
>> doc.paragraphs[1].runs[3].text
```

- Com o *Python-Docx*, seus programas Python agora poderão ler o texto de um arquivo *.docx* e usá-lo como qualquer outro valor de sequência de caracteres.

3.2 Obtendo o texto completo de um arquivo .docx

Se você se preocupa apenas com o texto, não com as informações do estilo, você pode usar a função getText(). Ela abre o documento e anexa o texto em *fullText*.

```
>> importdocx
    def getText(filename):
    doc = docx.Document(filename)
    fullText = []
    for para in doc.paragraphs:
    fullText.append(para.text)
  return '\.n'.join(fullText)
```

3.3 Criando documentos do Word com estilos não padrão

Se você quiser criar documentos do Word que usam estilos além do padrão, você precisará abrir o Word e um documento em branco do Word e criar um

estilo você mesmoclicando no botão Novo Estilo na parte inferior dos Estilos *(isso no Windows).*

Isso abrirá a caixa de diálogo 'Criar novo estilo de formatação', onde você pode usar o novo estilo. Em seguida, volte para o shell interativo e abra este documento em branco com *docx.Document()*. O nome que você deu a esse estilo agora estará disponível para uso com Python-Docx.

3.4 Escrevendo documentos em Word

$>> importdocx$
$>> doc = docx.Document()$ $\#retorna\ um\ novo\ documento$
$>> doc.add\_paragraph('Helloworld!')$ $\#adiciona\ um\ paragráfo$
$>> doc.save('helloworld.docx')$

Você pode adicionar parágrafos chamando o método add_paragraph () novamente com o novo texto do parágrafo, ou para facilitar, você pode chamar o método add_run () do parágrafo e passar uma string. Digite o seguinte no shell interativo:

$>> importdocx$
$>> doc = docx.Document()$
$>> doc.add\_paragraph('Helloworld!')$
$>> paraObj1 = doc.add\_paragraph('Segundoparagráfo')$
$>> paraObj2 = doc.add\_paragraph('Aindaumoutrosegundoparagráfo.')$
$>> paraObj1.add\_run('Otextoéadicionadoaosegundoparagráfo.')$
$>> doc.save('multipleParagraphs.docx')$

3.5 Adicionando Cabeçalhos

Chamar add_heading () adiciona um parágrafo com um dos estilos de título.

$>> doc = docx.Document()$
$>> doc.add\_heading('Header0', 0)$
$>> doc.add\_heading('Header1', 1)$
$>> doc.add\_heading('Header2', 2)$
$>> doc.add\_heading('Header3', 3)$
$>> doc.save('headings.docx')$

Os argumentos para add_heading() são uma string do texto do título e um inteiro de 0 a 4. O inteiro 0 torna o estilo de título e os números inteiros

1 a 4 são para vários níveis de cabeçalho, sendo 1 o título principal e 4 o sub-título mais baixo.

3.6 Adicionando Linhas e Quebras de Página

$>> doc = docx.Document()$
$>> doc.add\_paragraph('Thisisonthefirstpage!')$ $\#primeira\ página$
$>> doc.paragraphs[0].runs[0].add\_break(docx.text.WD\_BREAK.PAGE)$
$>> doc.add\_paragraph('Thisisonthesecondpage!')$ $\#segunda\ página$
$>> doc.save('twoPage.docx')$

3.7 Adicionando Figuras

O método add_picture() permite adicionar imagem para o final do documento.

$>> doc.add\_picture('picture.png', width = docx.shared.Inches(x), height = docx.shared.Cm(y))$

O primeiro argumento é uma string do nome do arquivo da imagem.É opcional a escolha da width (largura - polegadas) e heigth (altura -centímetros), caso não seja escolhida, será mantido o tamanho normal da imagem.

4 Projeto Combinando páginas selecionadas de muitos PDFs

```
import PyPDF2
import os

pdfFiles = []
for filename in os.listdir('.'):
      if filename.endswith('.pdf'):
         pdfFiles.append(filename)
pdfFiles.sort(key = str.lower)

pdfWriter = PyPDF2.PdfFileWriter()
```

```
for filename in pdfFiles:
      pdfFileObj = open(filename, 'rb')
      pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
      for pageNum in range(1, pdfReader.numPages):
        pageObj = pdfReader.getPage(pageNum)
        pdfWriter.addPage(pageObj)

pdfOutput = open('allminutes.pdf', 'wb')
pdfWriter.write(pdfOutput)
pdfOutput.close()
```