**Curso Python Web Scraping**

**Evaldo Wolkers**

**Lendo documentos**

Arquivos PDF

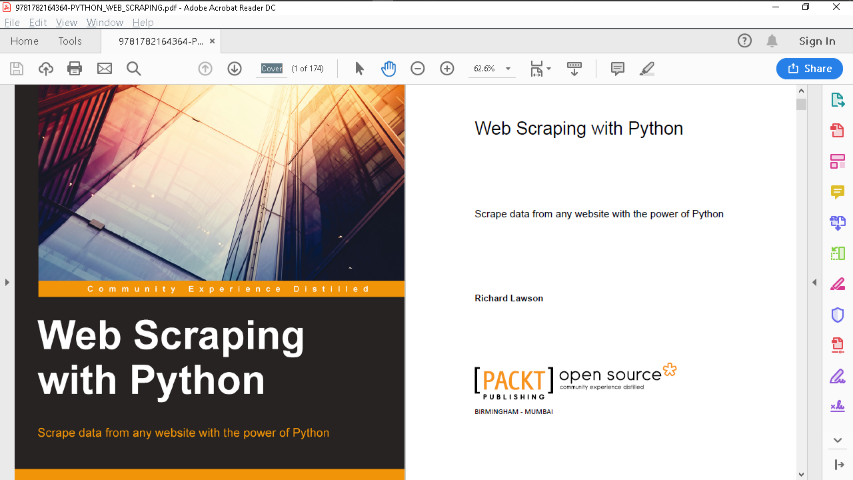
Ao realizar um scraping você pode ter a necessidade de tratar arquivos PDF.

O **PDF** (**Portable Document Format**) é um formato de arquivo, desenvolvido pela Adobe Systems em 1993, para representar documentos de maneira independente do aplicativo, do *hardware* e do sistema operacional usados para criá-los. Um arquivo PDF pode descrever documentos que contenham texto, gráficos e imagens num formato independente de dispositivo e resolução.

O PDF é um padrão aberto, e qualquer pessoa pode escrever aplicativos que leiam ou escrevam neste padrão. Há aplicativos gratuitos para Microsoft Windows, Mac e Linux, alguns deles distribuídos pela própria Adobe e há diversos aplicativos sob licenças livres.



Fonte: <https://pt.wikipedia.org/wiki/Portable_Document_Format>



Um arquivo PDF

aberto no Adobe

Acrobat Reader,

que é um programa

gratuito para leitura de arquivos PDF. Nas minhas aulas tenho disponibilizado materiais em PDF.

Para trabalhar com arquivos PDF podemos usar a biblioteca PDFMiner3k.

O PDFMiner3k é uma implementação do pdfminer para Python.

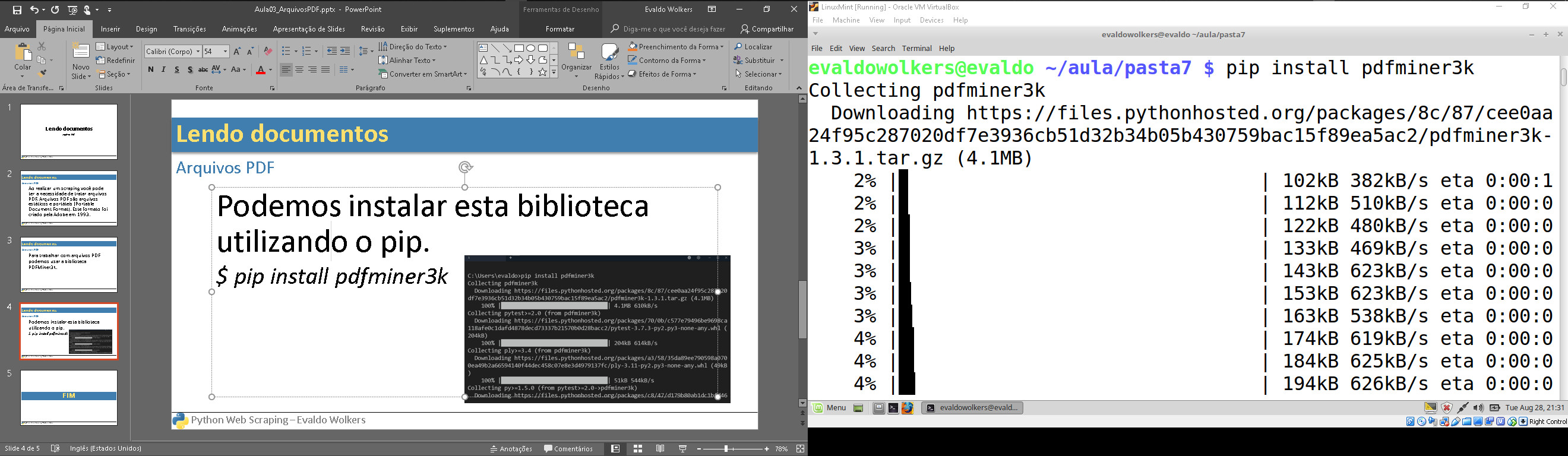
O PDFMiner é uma ferramenta para extrair informações de documentos PDF.

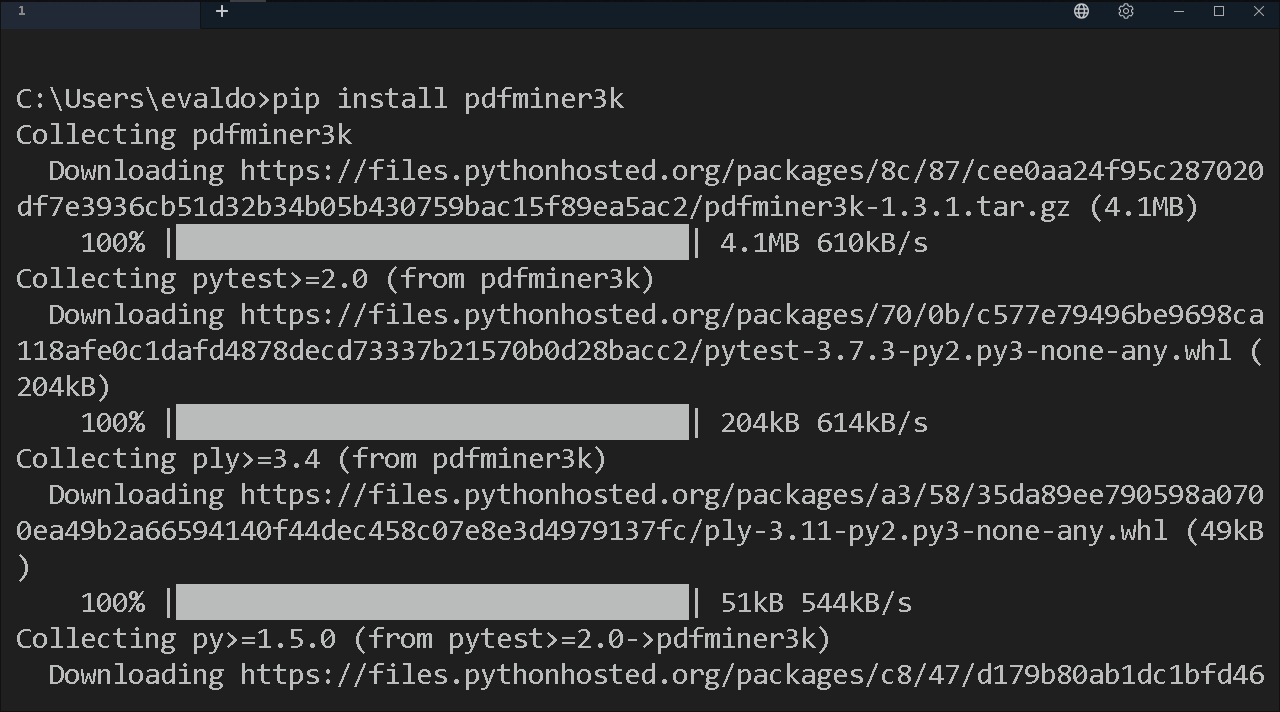
Ao contrário de outras ferramentas relacionadas a PDF, ele se concentra inteiramente em obter e analisar dados de texto. O PDFMiner permite obter a localização exata dos textos em uma página, bem como outras informações, como fontes ou linhas.

Inclui um conversor de PDF que pode transformar arquivos PDF em outros formatos de texto (como HTML).

Podemos instalar esta biblioteca utilizando o pip.

*$ pip install pdfminer3k*





Realizando as importações necessárias.

**from** pdfminer.pdfinterp **import** PDFResourceManager, process\_pdf  
**from** pdfminer.converter **import** TextConverter  
**from** pdfminer.layout **import** LAParams  
**from** io **import** StringIO  
*# Para abrir um PDF localmente***from** io **import** open  
*# Para abrir um PDF online***from** urllib.request **import** urlopen

PDFResourceManager

Repositório de recursos compartilhados.

ResourceManager facilita a reutilização de recursos compartilhados tais como fontes e imagens, para que objetos não sejam alocados várias vezes ocupando muito espaço em memória.

LAParams

Define os parâmetros que serão passados para a função TextConverter.

line\_overlap=0.5 (sobreposição de linha)

char\_margin=2.0 (margem do caracter)

line\_margin=0.5 (margem da linha)

word\_margin=0.1 (margem da palavra)

paragraph\_indent=None (indentação de parágrafo)

TextConverter

Converte o conteúdo do PDF em texto.

HTMLConverter

Converte o conteúdo do PDF em HTML.

XMLConverter

Converte o conteúdo do PDF em XML.

Process\_pdf

process\_pdf é uma função que executa as seguintes tarefas:

Cria um objeto analisador de PDF associado ao objeto de arquivo.

Cria um objeto de documento PDF que armazena a estrutura do documento.

Conecta o analisador e os objetos do documento.

Fornece a senha ao documento para inicialização (caso você tenha informado uma senha).

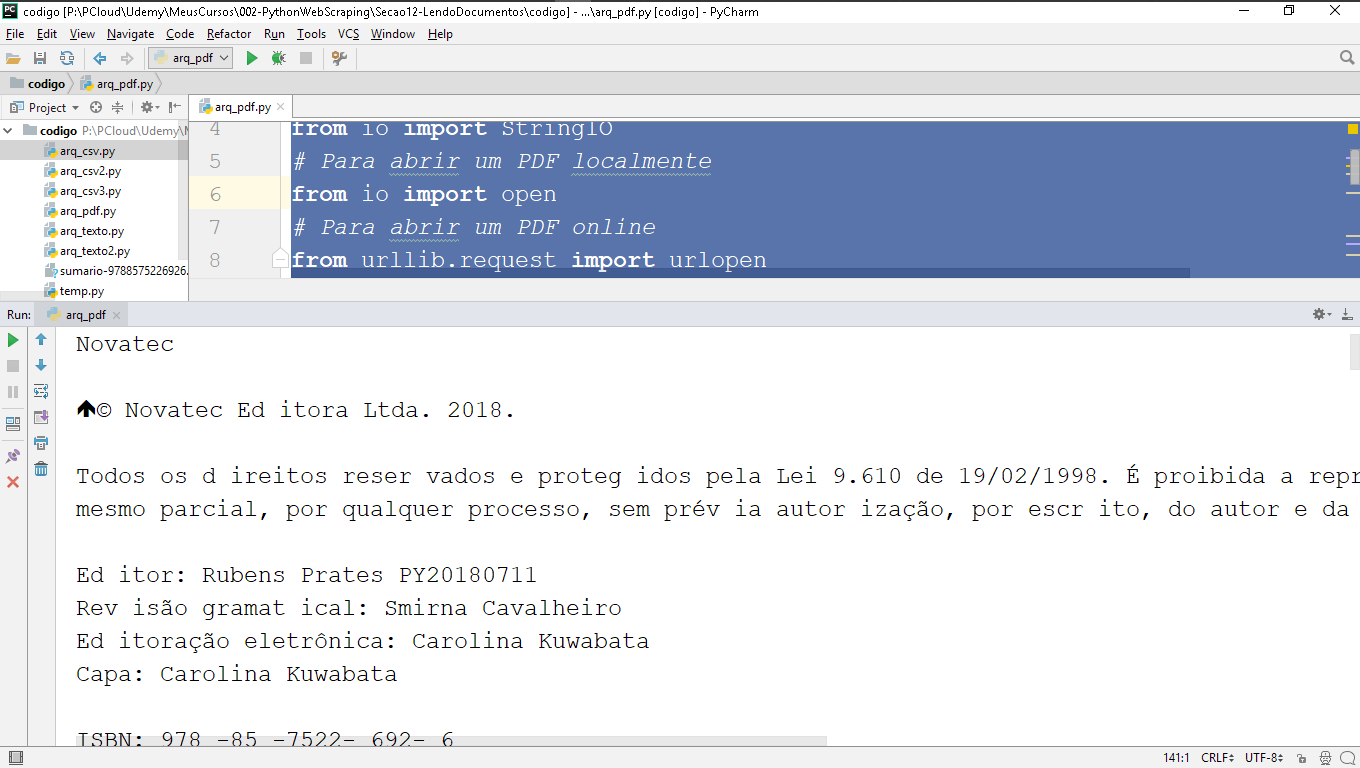
Verifica se o documento permite a extração de texto. Se não permite, aborta.

Cria um objeto de interpretação de PDF.

Processa cada página contida no documento.

**def** lerPDF(arquivoPDF):  
 *# PDFResourceManager Usado para armazenar recursos compartilhados  
 # como fontes e imagens* recursos = PDFResourceManager()  
 buffer = StringIO()  
 layoutParams = LAParams()  
 dispositivo = TextConverter(recursos, buffer, laparams=layoutParams)  
  
 process\_pdf(recursos, dispositivo, arquivoPDF)  
 dispositivo.close()  
  
 conteudo = buffer.getvalue()  
 buffer.close()  
 **return** conteúdo

*# Arquivo PDF online  
#arquivoPDF = urlopen("https://s3.novatec.com.br/sumarios/sumario-9788575226926.pdf")  
# Arquivo PDF local (Abrindo modo leitura e binário)*arquivoPDF = open(**"sumario-9788575226926.pdf"**, **"rb"**)  
stringSaida = lerPDF(arquivoPDF)  
print(stringSaida)  
arquivoPDF.close()



Convertendo

em HTML

**from** pdfminer.pdfinterp **import** PDFResourceManager, process\_pdf  
**from** pdfminer.converter **import** HTMLConverter  
**from** pdfminer.layout **import** LAParams  
**from** io **import** StringIO  
**from** urllib.request **import** urlopen  
  
**def** lerPDF(arquivo):  
 recursos = PDFResourceManager()  
 buffer = StringIO()  
 layoutParams = LAParams()  
 disp = HTMLConverter(recursos, buffer, laparams=layoutParams)  
  
 process\_pdf(recursos, disp, arquivo)  
 disp.close()  
  
 conteudo = buffer.getvalue()  
 buffer.close()  
 **return** conteudo  
  
arquivoPDF = urlopen(**"https://s3.novatec.com.br/sumarios/sumario-9788575226926.pdf"**)  
saida = lerPDF(arquivoPDF)  
print(saida)  
arquivoPDF.close()

Convertendo

em XML

**from** pdfminer.pdfinterp **import** PDFResourceManager, process\_pdf  
**from** pdfminer.converter **import** XMLConverter  
**from** pdfminer.layout **import** LAParams  
**from** io **import** StringIO  
**from** urllib.request **import** urlopen  
  
**def** lerPDF(arquivo):  
 recursos = PDFResourceManager()  
 buffer = StringIO()  
 layoutParams = LAParams()  
 disp = XMLConverter(recursos, buffer, laparams=layoutParams)  
  
 process\_pdf(recursos, disp, arquivo)  
 disp.close()  
  
 conteudo = buffer.getvalue()  
 buffer.close()  
 **return** conteudo  
  
arquivoPDF = urlopen(**"https://s3.novatec.com.br/sumarios/sumario-9788575226926.pdf"**)  
saida = lerPDF(arquivoPDF)  
print(saida)  
arquivoPDF.close()

Obs.: Para exportar para XML e HTML tive que alterar o arquivo

“Python36\Lib\site-packages\pdfminer\utils.py”

da biblioteca pdfminer e estou disponibilizando em anexo à aula.

A função htmlescape tem que ser corrigida.

def htmlescape(s, encoding=None):

"""Escapes a string for SGML/XML/HTML"""

s = s.replace('&','&amp;').replace('>','&gt;').replace('<','&lt;').replace('"','&quot;')

# Additionally to basic replaces, we also make sure that all characters are convertible to our

# target encoding. If they're not, they're replaced by XML entities.

if not encoding:

encoding = 'ascii'

encoded = s.encode(encoding, errors='xmlcharrefreplace')

return encoded.decode(encoding)