Integrando o Python com Word

Felipe C. R. dos Santos

1 Integrando o Python com o Word

O Word sem dúvidas é o programa mais utilizado na hora de se formalizar um documento, seja ele um contrato, um relatório, um trabalho escolar, uma pesquisa ou até mesmo apenas um lembrete pro fim do dia.

Embora geralmente se pegarmos arquivos distintos de **Word (*.docx*)** eles não terão correlaçao alguma entre um e outro, seja em estilo de fonte, cor, posições, emelementos constintuintes (tabelas, formas e imagens) ou principalmente conteúdo, **no mundo corporativo essa relação é muito comum**!

Imagine você como **responsável pelo relatório semanal** de uma empresa de médio porte, onde possui uma identidade visual fixa e padronizada, além de formatos de arquivos também já planejados. Todos seus arquivos de relatório seriam parecidos, certo? Mesmos tópicos, mesmas cores, mesmas fontes, enfim, mesmo tudo. Apenas o que mudaria aqui seria o conteúdo interno, onde para cada semana novas ações e informações deveriam ser escritas.

Com isso, o interesse por um automatizador que fizesse toda essa estrutura para você, onde a sua única preocupação *configurá-lo uma vez só* e em todas as seguintes apenas dizer para esse automatizar "Olha, aqui na parte de extratos financeiros, na tabela de despesas coleque os seguintes dados...", **fica cada vez mais intenso, não é mesmo?**

Agora imagine você sendo **responsável pela TI** (área de Tecnologia da Informação) de uma empresa de grande porte, onde todos os dados da semana são feitos através de um software já estabelecido e bem confiável, seu trabalho semanal seria apenas extrair esses dados toda semana, analisá-los e gerar um documento com eles, agora o serviço seria bem maior (visto que a empresa é de grande porte e por consequência, seu número de dados é enorme).

Não **seria perfeito automatizar todo o seu trabalho** criando um automatizador que busque todas essas informações direto no softoware, extraindo para você, realizando as devidas operações e em seguida **gerando um arquivo Word totalmente profissional** e bem feito, com todas as estruturas pré-definidas pela sua empresa?

Neste ultimo caso, basicamente o que seria preciso fazer é **configurar uma única vez um software** que faça isso e, em seguida, apenas **receber seu salário para ver o software trabalhando por você** e, em caso de atualizações no programa da empresa, atualizar também o seu automatizador.

É devido a isso que achamos de **extrema importância** criar este curso para **você**, onde ensinaremos tudo o que você precisa saber para, em poucas linhas, criar um arquivo **Word** através do **Python**.

Com esse curso você aprenderá:

1. Criar novos documentos;
2. Abrir documentos existentes;
3. Editar documentos existentes;
4. Inserir títulos;
5. Inserir parágrafos;
6. Customizar parágrafos;
7. Criar estilos de formatação;
8. Trabalhar com imagens;
9. Trabalhar com tabelas;
10. Salvar esses arquivos gerados;
11. E muito mais…

Sem mais delongas, vamos começar?

2 O Python-Docx

O **python-docx** é um pacote Python para criação e editação de arquivos **Microsoft Word (*.docx*)**.

Ele consegue, através de poucas linhas de programação, criar, editar e gerenciar tudo o que o Word tem a te oferecer.

Devido ao seu alto **poder**, **desempenho** incrível e imensa **aceitação** pela comunidade python, ele será utilizado durante nosso curso.

3 Instalação

Para instalar basta utilizarmos o PIP:

pip install ***python-docx***

4 Importando o pacote

O pacote é chamado de **docx**, sendo assim, podemos chamá-lo através de:

import ***docx***

A principal importação que precisamos é da classe **Document**, que gerencia o fluxo de uso (criação, edição e salvamento) do arquivo **.docx**. Portanto, podemos importá-la (e somente ela) através de:

from ***docx*** **import** ***Document***

```
from docx import Document
```

5 Criando nosso arquivo Word (.docx)

Para criarmos um arquivo basta instanciarmos um objeto da classe **Document**, ou seja, uma variável do tipo **Document**.

```
documento = Document()
```

6 Abrir um arquivo existente

O construtor da classe **Document** também aceita um atributo, no caso, uma *stream binária*.
O legal disso é que podemos ler um arquivo existente em modo *binário* (**rb**) e enviá-lo como parâmetro ao construtor, dessa forma, podemos abrir um documento word já existente e lê-lo ou editá-lo.

```
arquivo = open('arquivo_existente.docx', 'rb')
documento = Document(arquivo)
arquivo.close()
```

Como alternativa também podemos apenas passar o nome do arquivo por parâmetro

```
documento = Document('teste.docx')
```

7 Salvando nosso arquivo

Já para salvarmos nosso arquivo, podemos usar o método **save()** também da classe **Document**.
Essa função exige por parâmetro o nome do nosso arquivo, como por exemplo *'teste.docx'*.

```
documento.save('teste.docx')
```

8 Adicionando um cabeçalho (Título)

Podemos adicinar títulos/cabeçalhos (*heading*) através do método **add_heading()**.
Tal método aceita dois parâmetros, sendo eles (respectivamente):
1. **text:** O texto a ser inserido como cabeçalho;
2. **level:** O nível de identação do cabeçalho, indo de 0 até 9, tendo como padrão o valor 1.

```
documento.add_heading('Título do meu␣
 ↪documento', 0)
```

9 Adicionando um parágrafo (Texto)

Podemos adicionar parágrafos (textos) através do método **add_paragraph()**.
Tal método aceita dois parâmetros, sendo eles (respectivamente):
1. **text**: O texto a ser inserido como parágrafo;
2. **style**: Estilo do texto a ser adicionado:
- Normal;
- Citação;
- Listas.

Para mais exemplos de estilos: https://bit.ly/2jVtFyl.

9.1 Criando parágrafos normais

```
documento.add_paragraph('Eu sou um exemplo de␣
 ↪texto')
```

9.2 Criando listas

```
documento.add_paragraph('Item 1',style =␣
 ↪'List Number')
documento.add_paragraph('Item 2',style =␣
 ↪'List Number')
documento.add_paragraph('Item 3',style =␣
 ↪'List Number')
documento.add_paragraph('Item 4',style =␣
 ↪'List Number')
```

9.3 Criando citações

```
documento.add_paragraph('Minha Citação␣
 ↪Intensa','Intense Quote')
documento.add_paragraph('Minha␣
 ↪Citação','Quote')
```

10 O Objeto Paragraph

Sempre que adicionamos um texto no nosso documento, seja ele um título (*add_heading()*) ou parágrafo (*add_paragraph()*) nós na verdade adicionamos um objeto do tipo **Paragraph**:

```
paragrafo = documento.add_paragraph("Eu sou␣
 ↪um parágrafo")
titulo = documento.add_heading("Eu sou um␣
 ↪título")
print(type(paragrafo))
print(type(titulo))
```

Os objetos do tipo **Paragraph** possuem diversas funções e propriedades, entre elas:

1. **text**: O texto desse nosso parágrafo;
2. **paragraph_format**: A formatação do nosso parágrafo;
3. **style**: O estilo do nosso parágrafo;
4. **font**: A fonte atual do parágrafo;
5. **runs**: Lista com todas as 'partes' (*run*) desse nosso parágrafo.

10.1 Estilizando com Negrito e Itálico

Se olharmos mais atentamente ao 5º item, podemos ver que o nosso texto pode ser feito em 'partes' (runs).
O interessante disso é que **cada uma dessa parte pode ter seu próprio estilo de letra** (negrito e/ou itálico).

Para isso pode usar o método ***add_run()***, que aceita como parâmetro o texto para se adicionar e definirmos as propriedades **bold** e **italic** para *True* ou *False*.

```
titulo = documento.add_heading("Eu sou um")
titulo.add_run("título").bold = True

paragrafo = documento.add_paragraph("Eu sou␣
 ↪um ")
paragrafo.add_run("paragrafo ").italic = True
```

Podemos também ter um objeto do tipo **Run** o que nos permite uma melhor edição:

```
titulo = documento.add_heading("Eu sou um")
run_titulo = titulo.add_run("titulo")
run_titulo.bold = True
run_titulo.italic = True
```

10.2 Estilizando a Fonte do Parágrafo

Podemos também editar a fonte do parágrafo, usando a propriedade **font**.

Dentre as propriedades da fonte temos:
1. **size**: O tamanho atual da fonte;
2. **name**: O nome da fonte atual (ex: Arial);
3. **color**: A cor da fonte.

```
paragrafo = documento.add_paragraph()
run = paragrafo.add_run("Eu sou um paragrafo")
fonte = run.font
fonte.name = 'Corbel'
```

Para editarmos o tamanho, precisamos importar o método **Pt** de **docx.shared**, sendo assim:

from *docx.shared* import *Pt*

```
from docx.shared import Pt
fonte.size = Pt(20)
```

Para editarmos a cor, precisamos importar o método **RGBColor** de **docx.shared**, sendo assim:

from *docx.shared* import *RGBColor*

```
from docx.shared import RGBColor
fonte.color.rgb = RGBColor(255,0,0)# (R,G,B)
```

```
from docx import Document
from docx.shared import Pt
from docx.shared import RGBColor
documento = Document()
paragrafo = documento.add_paragraph()
run = paragrafo.add_run("Eu sou um paragrafo")
fonte = run.font
fonte.name = 'Corbel'
fonte.size = Pt(20)
fonte.color.rgb = RGBColor(255,0,0)
documento.save('teste2.docx')
```

10.3 Editando o alinhamento do nosso parágrafo

Para alinharmos o nosso parágrafo podemos usar a propriedade **paragraph_format.alignment**.

O alinhamento pode ser feito através do *WD_PARAGRAPH_ALIGNMENT*, que por sua vez precisa ser importado de **docx.enum.text**:

> **from** ***docx.enum.text*** **import** ***WD_PARAGRAPH_ALIGNMENT***

Podemos ter os seguintes alinhamentos:

1. CENTER;
2. DISTRIBUTE;
3. JUSTIFY;
4. JUSTIFY_HI;
5. JUSTIFY_LOW;
6. JUSTIFY_MED;
7. LEFT;
8. RIGHT;
9. THAI_JUSTIFY.

```
from docx.enum.text import␣
 ↪WD_PARAGRAPH_ALIGNMENT

paragrafo = documento.add_paragraph()
run = paragrafo.add_run("Eu sou um paragrafo")
paragrafo.paragraph_format.alignment =␣
 ↪WD_PARAGRAPH_ALIGNMENT.CENTER
```

11 Criando estilos nossos

Também podemos criar estilos nossos e inserí-los em parágrafos, dessa forma, garantimos que todos os parágrafos com esse estilo tenham a mesma configuração de fonte, tamanho, itálico, negrito e etc...

O bom de criar estilos é que não precisamos configurar tudo a cada run, podemos basicamente criar uma vez e apenas atribuir esse estilo nos parágrafos que queremos.

Para isso, precimamos usar a função **styles.add_styles()** que ficam no nosso **documento** (objeto do tipo *Document*).
Como parâmetros temos 2, sendo o primeiro o **nome** do nosso estilo e como segundo o tipo do nosso estilo (como por exemplo, se é para um parágrafo).

Para o estilo, precisamos usar o *WD_STYLE_TYPE*, e para isso precisamos importá-lo de **docx.enum.style**:

from *docx.enum.style* import *WD_STYLE_TYPE*

```
from docx.enum.style import WD_STYLE_TYPE
style = documento.styles.
 ↪add_style('EstiloEditado', WD_STYLE_TYPE.
 ↪PARAGRAPH)

font = style.font
font.name = 'Corbel'
font.size = Pt(15)
font.color.rgb = RGBColor(127,255,127)
style.paragraph_format.alignment =␣
 ↪WD_PARAGRAPH_ALIGNMENT.CENTER
font.bold = True
font.italic = True
```

11.1 Atribuindo estilos

Para atribuir estilos em um parágrafos podemos usar a propriedade **style** do nosso parágrafo e atribuir o nosso estilo configurado previamente através do dicionário **styles** do nosso documento, passando o nome do nosso estilo como chave de busca, ex:

*documento.styles[‘**NossoEstilo**’]*

```
paragrafo = documento.add_paragraph()
paragrafo.style = documento.
 ↪styles['EstiloEditado']
paragrafo.add_run("Eu sou um parágrafo␣
 ↪editado")
```

12 Inserindo Imagens

Podemos inserir imagens através do método **add_picture()**.
Este método aceita três parâmetros sendo eles:

1. **image_path_or_stream** => Parâmetro obrigatório contendo nossa imagem (caminho ou *stream* da imagem);

2. **width** => Largura, sendo um parâmetro opcional;

3. **height** => Altura, sendo também um parâmetro opcional.

Algumas informações importantes são:

1. Tanto *width* quanto *height* são do tipo **Inches** que precisam ser importados do namespace **docx.shared**:
   > **from *docx.shared* import *Inches***

2. Caso nenhum dos dois seja enviado então será utilizado o tamanho original da imagem;

3. Já no caso de um deles (apenas) seja passado, a imagem será redimensionada de forma a preservar o aspecto original da imagem:
   > Ex: Caso seja 1920x1080 e passemos a altura como sendo 108, então a largura vai ser redimensionada junta para poder manter o aspécto original, ou seja, irá valor 192.

4. Caso você envie os dois, a imagem será redimensionada para aquela proporção, podendo ser diferente do aspécto original, o que pode ocasionar distorção.

```
from docx.shared import Inches

documento.add_picture('C:\\logo-bylearn.jpg')␣
 ↪# Tamanho original

documento.add_picture('C:\\logo-bylearn.jpg',␣
 ↪width=Inches(2)) # Mudamos apenas a largura
documento.add_picture('C:\\logo-bylearn.jpg',␣
 ↪ height=Inches(4)) # Mudamos a altura

documento.add_picture('C:\\logo-bylearn.jpg',␣
 ↪width=Inches(2), height=Inches(4)) #␣
 ↪Forçamos o tamanho da imagem
```

13 Inserindo Tabelas

Podemos também criar tabelas completas através do método **add_table()**, que possui como parâmetros **rows** (linhas) e **cols** (colunas) que existirão na tabela.

Também podemos adicionar novas linhas dinamicamente, através do método **add_row()** da sua tabela.

Para pegarmos todas as células da linha, podemos usar a propriedade **cells** da sua linha.

```
cursos = [
    ['Felipe','Python',5],
    ['Alisson','Android',4],
    ['Haynes','Python',4]
#   (Instrutor, Categoria, Quantidade)
]

tabela = documento.add_table(rows=1,cols=3)
# Adicionamos apenas 1 linha (o titulo)
titulo_tabela = tabela.rows[0].cells
 # Pegamos as células da tabela
titulo_tabela[0].text = "Nome do Instrutor"
titulo_tabela[1].text = "Categoria do Curso"
titulo_tabela[2].text = "Quantidade de Cursos"

for nome, categoria, quantidade in cursos:
    linha = tabela.add_row().cells
    linha[0].text = nome
    linha[1].text = categoria
    linha[2].text = str(quantidade)
```

14 Quebra de página (inserir nova página)

Podemos inserir novas páginas através do método **add_page_break()**.

```
documento.add_paragraph("Primeira Página")
documento.add_page_break()
documento.add_paragraph("Segunda Página")
```

15 Desafio: Criando um documento completo do Zero

Agora você já é um **ByLearner** experiente!
Que tal criarmos um documento completo totalmente do zero usando tudo o que aprendemos?

```
from docx import Document

from docx.shared import Pt
from docx.shared import RGBColor
from docx.shared import Inches

from docx.enum.text import␣
 ↪WD_PARAGRAPH_ALIGNMENT
from docx.enum.style import WD_STYLE_TYPE

documento = Document()

titulo = documento.add_heading(level= 0)
titulo.add_run("Integrando o ")
titulo.add_run("Python ").bold = True
titulo.add_run("com o ")
titulo.add_run("Word").bold = True

documento.add_paragraph("Nós aprendemos:")
documento.add_paragraph('Inserir␣
 ↪Títulos',style = 'List Number')
documento.add_paragraph('Inserir␣
 ↪Parágrafos',style = 'List Number')
documento.add_paragraph('Customizar␣
 ↪Parágrafos',style = 'List Number')
documento.add_paragraph('Criar Estilos',style␣
 ↪= 'List Number')
documento.add_paragraph('Trabalhar com␣
 ↪imagens',style = 'List Number')
documento.add_paragraph('Trabalhar com␣
 ↪Tabelas',style = 'List Number')
```

```
documento.add_paragraph('E muito mais',style␣
 ↪= 'List Number')

documento.add_page_break()

titulo = documento.add_heading("Caso você␣
 ↪tenha alguma dificuldade, ")
run_titulo = titulo.add_run("lembre-se sempre:
 ↪")
run_titulo.bold = True
run_titulo.italic = True

documento.add_paragraph('A prática leva a␣
 ↪perfeição, com treino e força de vontade,␣
 ↪nada te impedirá.','Quote')

documento.add_paragraph('Mas caso tenha␣
 ↪alguma dúvida neste processo,\
nós da ByLearn sempre estaremos aqui para te␣
 ↪ajudar','Intense Quote')

documento.add_page_break()

paragrafo = documento.
 ↪add_paragraph("Geralmente usamos o Word␣
 ↪para escrever relatórios, e muitas vezes é␣
 ↪algo padronizado.")
run = paragrafo.add_run("\nNesses casos, o␣
 ↪Python pode te ajudar a automatizar esse␣
 ↪trabalho")
fonte = run.font
fonte.name = 'Corbel'
fonte.size = Pt(20)
fonte.color.rgb = RGBColor(255,0,0) #␣
 ↪(Vermelho, Verde, Azul)

paragrafo = documento.add_paragraph()
```

```
run = paragrafo.add_run("Mas e quanto aos␣
 ↪relatórios com padrões de formatação␣
 ↪(identidade visual)?")
paragrafo.paragraph_format.alignment =␣
 ↪WD_PARAGRAPH_ALIGNMENT.CENTER

style = documento.styles.
 ↪add_style('EstiloEditado', WD_STYLE_TYPE.
 ↪PARAGRAPH)
font = style.font
font.name = 'Corbel'
font.size = Pt(15)
font.color.rgb = RGBColor(127,255,127)
style.paragraph_format.alignment =␣
 ↪WD_PARAGRAPH_ALIGNMENT.CENTER
font.bold = True
font.italic = True

paragrafo = documento.add_paragraph()
paragrafo.style = documento.
 ↪styles['EstiloEditado']
paragrafo.add_run("Sem problemas, podemos ter␣
 ↪estilos próprios para padronizarmos nossa␣
 ↪identidade visual!")

documento.add_page_break()

titulo = documento.add_heading("Gostou do␣
 ↪curso? Espero que sim :)")
titulo.add_run("\nNós temos vários outros␣
 ↪cursos, sendo alguns deles:")

cursos = [
    ['Felipe','Python',5],
    ['Alisson','Android',4],
    ['Haynes','Python',4] ]

tabela = documento.add_table(rows=1,cols=3)
```

```
titulo_tabela = tabela.rows[0].cells
titulo_tabela[0].text = "Nome do Instrutor"
titulo_tabela[1].text = "Categoria do Curso"
titulo_tabela[2].text = "Quantidade de Cursos"

for nome, categoria, quantidade in cursos:
    linha = tabela.add_row().cells
    linha[0].text = nome
    linha[1].text = categoria
    linha[2].text = str(quantidade)

documento.add_page_break()

documento.add_picture('c:\\logo-bylearn.jpg')

documento.save('documento_final.docx')
```

16 Lendo um arquivo

Como já vimos, podemos abrir um arquivo existente com:

*Document(‘***nome_do_arquivo.docx***’)*

Onde, como mostrado acima, basta chamar o construtor de um documento normal (**Document()**) e enviar por parâmetro o nome do arquivo.

```
documento = Document('documento_final.docx')
```

16.1 Lendo os parágrafos

Para ler todos os parágrafos podemos usar a propriedade **paragraphs** do nosso documento.

```
for paragrafo in documento.paragraphs:
    print(paragrafo.text)
```

Podemos até mesmo **recuperar os estilos** de cada parágrafo, bastando acessar a propriedade ***style*** do nosso parágrafo.

```
for paragrafo in documento.paragraphs:
    if paragrafo.style.font.bold:
        print("\nO parágrafo a seguir está␣
 ↪escrito em negrito:")
        print(paragrafo.text)

for paragrafo in documento.paragraphs:
    if paragrafo.style.font.italic:
        print("\nO parágrafo a seguir está␣
 ↪escrito em itálico:")
        print(paragrafo.text)
```

PS: Por padrão, caso não tenha sido definido nenhum estilo, o valor dele será ***None*** e não **False** ou algum outro valor padrão.
Já no caso do valor ser ***None***, então ele não possui nenhuma modificação de estilo naquela propriedade, portanto, ele utilizará o padrão do Word.

```
for paragrafo in documento.paragraphs:
    if paragrafo.style.font.size != None:
        print("O parágrafo a seguir teve o␣
 ↪tamanho de sua fonte modificada:")
        print(paragrafo.text)
```

16.2 Lendo as tabelas

Para ler todos os parágrafos podemos usar a propriedade **tables** do nosso documento.

```
data = []

for tabela in documento.tables:
    for i, linha in enumerate(tabela.rows):
        texto = (cell.text for cell in linha.
 ↪cells)
        if i == 0:
            keys = tuple(texto)
            continue

        linha_dict = dict(zip(keys, texto))
        data.append(linha_dict)

print(data)
```

16.3 Lendo as imagens

Para ler todos os parágrafos podemos usar a propriedade **inline_shapes** do nosso documento.

```
for imagem in documento.inline_shapes:
    print ("Nome do arquivo",imagem._inline.
 ↪graphic.graphicData.pic.nvPicPr.cNvPr.name)
    print ("Tamanho (em centímetros):␣
 ↪",{imagem.height.cm},"x",{imagem.width.
 ↪cm},"cms")
```

17 Documentação Oficial

O aprendizado nunca deve terminar e uma grande fonte de estudo a partir de agora será a documentação oficial. Com todo o ensino que você aprendeu até agora no curso, você entenderá facilmente tudo o que estiver lá na documentação.

Link: https://python-docx.readthedocs.io/en/latest/

18 Extra: Conteúdos diários gratuitos

Por fim, antes de terminarmos nosso livro, precisamos te falar sobre os conteúdos diários que criamos para você, um **ByLearner**!

Temos vídeos, blogs, explicações de códigos, curiosidades, lives, minicursos, enquetes e muito mais... Inclusive os nossos *Drops*.

Caso você ainda não saiba o que é um ***Drops da ByLearn***, é simpes:
- Através da nossa didática rápida e direto ao ponto preparamos um conteúdo bem legal de uma forma bem interessante para você assistir no tempo livre.
- Sabe quando está no face/insta e surge um vídeo legal de 1 minutinho no seu feed? Ou quando surge uma imagem bem chamativa e criativa?
- Então... Agora imagine se esse video bem supimpa fosse educativo, ou se essa imagem criativa te ensinasse a programar... - É exatamente isso que é um *Drops*.

O nosso foco é conseguir o interesse do aluno e proporcionar um conhecimento novo qualquer hora do dia!

18.1 Para isso você pode nos achar em:

Facebook: @ByLearn
Instagram: @ByLearn
Youtube: @Bylearn
Twitter: @ByLearn
LinkedIn: @ByLearn
Github: @ByLearn

Bom... Você entendeu né? É tudo ***@ByLearn***, procura lá.

18.2 Para encontrar nossos cursos:

Acesse nosso site em: http://bylearn.com.br.

19 Muito obrigado por tudo!

Foi um prazer enorme criar esse livro para vocês e ensiná-lo durante todo esse curso.
Esperamos que tenha gostado e que o conteúdo tenha sido útil para você.
Nos vemos no próximo livro!