Section 14

Excel, Word, and PDF Documents

Mohammed Asir Shahid

2021-08-05

Contents

1 Reading Excel Spreadsheets

The openpyxl module lets us modify Excel files using Python. It is a third party module that we'll need to install ourselves.

```
pip install openpyxl
```

The Excel document is called a workbook that is saved by .xlsx file extension. Each workbook contains sheets/worksheets. Inside each sheet there are columns (letters) and rows (numbers). The intersection of a column and row is called a cell.

```
import openpyxl,os

workbook=openpyxl.load_workbook("example.xlsx")

print(type(workbook))
print(workbook.get_sheet_names())

sheet=workbook.get_sheet_by_name("Sheet1")
print(type(sheet))

cell=sheet["A1"]

print(type(cell))
print(cell.value)

cell=sheet["B1"]

print(type(cell))
print(cell.value)

cell=sheet["C1"]

print(type(cell))
print(cell.value)


print(type(cell))
print(cell.value)

for i in range(1,8):
    print(i, sheet.cell(row=i, column=2).value)
```

```
<class 'openpyxl.workbook.workbook.Workbook'>
['Sheet1', 'Sheet2', 'Sheet3']
<class 'openpyxl.worksheet.worksheet.Worksheet'>
<class 'openpyxl.cell.cell.Cell'>
2015-04-05 13:34:02
<class 'openpyxl.cell.cell.Cell'>
Apples
<class 'openpyxl.cell.cell.Cell'>
73
<class 'openpyxl.cell.cell.Cell'>
73
1 Apples
```

```
2 Cherries
3 Pears
4 Oranges
5 Apples
6 Bananas
7 Strawberries
```

2 Editing Excel Spreadsheets

In the last lesson, we learned how to read .xlsx files. Now we will learn to create and modify them.

```
import openpyxl,os

wb=openpyxl.Workbook()

print(type(wb))

print(wb.get_sheet_names())

sheet=wb.get_sheet_by_name("Sheet")

print(sheet)
print(sheet["A1"].value)
sheet["A1"]=42
sheet["A2"]="Hello"
print(sheet["A1"].value)

wb.save("example1.xlsx")

sheet2=wb.create_sheet()

print(wb.get_sheet_names())

sheet2.title="My New Sheet Name"

print(wb.get_sheet_names())

wb.save("example2.xlsx")
wb.create_sheet(index=0, title="My Other Sheet")
# This changes the position of the new sheet

wb.save("example3.xlsx")
```

```
<class 'openpyxl.workbook.workbook.Workbook'>
['Sheet']
<Worksheet "Sheet">
None
42
['Sheet', 'Sheet1']
['Sheet', 'My New Sheet Name']
```

3 Reading and Editing PDFs

PDF files are binary files which make them far more complicated than plain text files such as .org or .py files. They store far more information than plain text files.

There are some Python modules we can use to interact with PDFs, however it isn't that straightforward. We will be looking at a third party module called PyPDF2.

```
pip install PyPDF2
```

```
import PyPDF2, os

pdfFile=open("meetingminutes1.pdf", "rb")
# The "rb" is since this is a binary file

PyPDF2.PdfFileReader(pdfFile)

reader=PyPDF2.PdfFileReader(pdfFile)

print(reader.numPages)

page=reader.getPage(0)
```

```
print(page.extractText())

#for pageNum in range(reader.numPages):
#    print(reader.getPage(pageNum).extractText())
```

```
19
OOFFFFIICCIIAALL  BBOOAARRDD  MMIINNUUTTEESS   Meeting of
March 7
, 2014

     The Board of Elementary and Secondary Education shall provide leadership and
create policies for education that expand opportunities for children, empower
families and communities, and advance Louisiana in an increasingly
competitive glob
al market.
 BOARD
 of ELEMENTARY
 and
 SECONDARY
 EDUCATION
```

Due to the complexity of PDF documents, Python can't add text arbitrarily. PDF Writer's functionality is limited to editing at the page level. So lets say we want to combine our two meeting minute fies

```
import PyPDF2, os

pdf1File=open("meetingminutes1.pdf", "rb")
pdf2File=open("meetingminutes2.pdf", "rb")
# The "rb" is since this is a binary file

reader1=PyPDF2.PdfFileReader(pdf1File)
reader2=PyPDF2.PdfFileReader(pdf2File)

writer=PyPDF2.PdfFileWriter()
```

```
for pageNum in range(reader1.numPages):
    page=reader1.getPage(pageNum)
    writer.addPage(page)

for pageNum in range(reader2.numPages):
    page=reader2.getPage(pageNum)
    writer.addPage(page)

outputFile=open("combinedminutes.pdf","wb")
writer.write(outputFile)
outputFile.close()
pdf1File.close()
pdf2File.close()
```

4 Reading and Editing Word Documents

Python can also create and modify .docx documents. Python can do this by using the third party python-docx module

```
pip install python-docx
```

Compared to plain text files, .docx files have a lot of structure. This is represented by 3 different data types in python-docx.

The document object contains paragraph objects. Each of these paragraph objects contains run objects.

```
import docx

d=docx.Document("demo.docx")

print(d.paragraphs)

print(d.paragraphs[0].text)
print(d.paragraphs[1].text)

p=d.paragraphs[1]
```

```
print(p.runs)

for i in range(4):
    print(p.runs[i].text)

for i in range(4):
    print(p.runs[i].bold)

for i in range(4):
    p.runs[i].underline=True
    print(p.runs[i].text)

d.save("demo2.docx")
```

```
[<docx.text.paragraph.Paragraph object at 0x7f6f203ea190>, <docx.text.paragraph.Paragra
Document Title
A plain paragraph having some bold and some italic.
[<docx.text.run.Run object at 0x7f6f203ea2b0>, <docx.text.run.Run object at 0x7f6f203ea
A plain paragraph having some
bold
 and some
italic.
None
True
None
None
A plain paragraph having some
bold
 and some
italic.
```

In .docx documents, each paragrah and run has its own style. For example, Normal, Heading 1, Heading 2, Title, etc.

```
import docx

d=docx.Document("demo2.docx")
```

```
p=d.paragraphs[1]

print(p.runs)

print(p.style)

p.style="Title"

print(p.style)

d.save("demo3.docx")
```

```
[<docx.text.run.Run object at 0x7f495d4c0910>, <docx.text.run.Run object at 0x7f495d4c
_ParagraphStyle('Normal') id: 139953074604112
_ParagraphStyle('Title') id: 139953074679424
```

Now let's create a brand new word document.

```
import docx

d=docx.Document()

d.add_paragraph("Hello this is a paragraph.")

d.add_paragraph("This is another paragraph")

d.save("demo4.docx")

p=d.paragraphs[0]

p.add_run("This is a new run")

p.runs[1].bold=True

d.save("demo5.docx")
```

There is now way to add paragraph and run objects anywhere except at the end. So if we want to add something in the middle, we will need to create a new .docx document and then copy over paragraphs and runs from the original document to the new document and then making changes along the way.

How could we get all the text in a word document as a string?

```
import docx

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

print(getText("demo.docx"))
```

```
Document Title
A plain paragraph having some bold and some italic.
Heading, level 1
Intense quote
first item in unordered list
first item in ordered list
```