Lecture 9: Working with Word Documents and Spreadsheets

.docx and .xlsx files

Dr Kieran T. Herley
2018-2019

Department of Computer Science
University College Cork
Summary

Reading and writing .docx files. Working with .xlsx or .ods spreadsheets. Simple application: generating exam cover sheets.
Generating exam cover sheets

Generate a set of cover sheets for exam script bundles.

A spreadsheet (.xlsx or .ods) with details of list of exams

A sample cover sheet (as .docx) for a script bundle
The challenges

How to read/write .docx stuff?

python-docx package

How to handle .xlsx or .ods stuff?

pyexcel package

Note: Excel doesn't run on Linux; Linux users use freeware such as LibreOffice to handle .docx or .xlsx; LibreOffice uses different (.ods) format for spreadsheets though; pyexcel works with both.
The challenges

How to read/write .docx stuff? python-docx package

How to handle .xlsx or .ods stuff? pyexcel package

Note: Excel doesn’t run on linux; linux users use freeware such as LibreOffice to handle .docx or .xlsx; LibreOffice uses different (.ods) format for spreadsheets though; pyexcel works with both.
Handling Word Documents
**python-docx**  Python library for creating and updating Microsoft Word (.docx) files

**Usage**

```python
import docx
```

**Documentation**

Document Title

A plain paragraph having some bold and some italic.

* **Heading, level 1**
  
  *Intense quote*

  * first item in unordered list

  1. first item in ordered list

  ![Image](image-url)

<table>
<thead>
<tr>
<th>Qty</th>
<th>Id</th>
<th>Desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>Spam</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>Eggs</td>
</tr>
<tr>
<td>3</td>
<td>631</td>
<td>Spam, spam, eggs, and spam</td>
</tr>
</tbody>
</table>
from docx import Document

.. 

document = Document()

document.add_heading('Document Title', 0)

p = document.add_paragraph('A plain paragraph having some...')
p.add_run('bold').bold = True
p.add_run('and some...')
p.add_run('italic .').italic = True

... 

document.add_page_break()

document.save('demo.docx')
add_paragraph  Return a paragraph newly added to the end of the document, optionally populated with text provided.

add_run  Append a run to this paragraph containing specified text and (optionally) style. (Paragraph composed of runs i.e. contiguous segments of text sharing same style.)

other add_X  Also add_heading, add_picture, add_table etc.

add_page_break  Return newly Paragraph object containing only a page break.

save  Save the document into a file.
Package allows control over document style. See reference for details.
import docx

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

print("Document has %d paragraphs." % len(doc.paragraphs))

for parnum in range(len(doc.paragraphs)):
    print("Paragraph %d:")
    print(doc.paragraphs[parnum].text)
    print()
Handling Spreadsheets
**pyexcel** Python library to facilitate reading, manipulating and writing data in various spreadsheet formats

**Usage**

```python
import pyexcel
```

**Documentation**

http://docs.pyexcel.org/en/latest/
**Workbook** collection of named sheets
**Sheet** two-dimensional array of cells
**Cell** individual addressible unit

We will ignore all complications e.g. merged cells, formula etc.
import pyexcel as pe

FILE_NAME = "exam_list.xlsx format.xlsx"
book = pe.get_book(file_name = FILE_NAME)

# Do stuff . . .

# Only if book modified:
book.save_as("new_" + FILE_NAME)
# Access individual sheet by name.
sheet = book["Sheet1"]

# Access "body" of the sheet.
print(sheet.content)

# Establish number of rows/columns.
print(sheet.number_of_rows(),
      sheet.number_of_columns())
Accessing individual cells within sheet

```python
# Access a random cell by coords (top left is (0, 0)) or by Excel coords (top left is A1).
print(sheet[2, 1])
print(sheet["B3"])
sheet[2, 1] = "Intermediate Nohtyp"
```
# Establish top row as column headers.
sheet.name.columns_by_row(0)
# Establish top row as column headers.

```
sheet.name.columns_by_row(0)
```

# Iterate through rows one by one.

```
records = sheet.to_records()
for r in records:
    print("Module\%s is taught by\%s."
          % (r["Module Code"], r["Lecturer"]))
```

Note: iterator serves up each row as a dictionary, with column labels as keys.
# Change the sheet's name.
sheet.name = "Tuesday"

# Add a fresh row.
sheet.row += ["CS4567", "Small Data", "Prof. Greene"]
Meanwhile Back at the Ranch . . .
Recall the cover sheet application

Generate a set of cover sheets (as .docx) for exam script bundles from spreadsheet (.xlsx or .ods) containing exam details.
import pyexcel
import docx

... 

def write_one(document, modcode, modtitle, modlect):
    # Write a fresh cover sheet with details 'modcode',
    # 'modtitle' and 'modlect' to end of 'document'.
    ...

def write_all(wbname):
    # Generate cover pages for each of the exams listed
    # in workbook 'wbname'.
    ...

write_all(Spreadsheet_NAME)

Refer to handout for more detail.
def write_one(document, modcode, modtitle, modlect):
    # Write a fresh cover sheet with details 'modcode',
    # 'modtitle' and 'modlect' to end of 'document'.
    hdr = document.add_heading(EXAM_BANNER, 0)
    ...
    para = document.add_paragraph()
    ...
    para.add_run(modcode+ "−−−").bold = True
    para.add_run(modtitle). italic = True
    para = document.add_paragraph(modlect)
    ...
    document.add_page_break()
def write_all (wbname):
    # Generate cover pages for each of the exams listed
    # in workbook 'wbname'.
    coverdoc = docx.Document()

    records = pyexcel.iget_records ( file_name=wbname)
    for rec in records :
        ecode = rec[CODE_COL]
        etitle = rec[TITLE_COL]
        elect = rec[LECT_COL]
        write_one(coverdoc, ecode, etitle, elect)

    coverdoc.save(COVER_FILE_NAME)
Back Material
1. Document writing example is taken from