Python Powered Excel: Focus on Data Instead of File Formats

Jennifer Watt
Human Genome Sequencing Center
Baylor College of Medicine, Houston, TX

Abstract

Programmatic manipulation of Excel spreadsheets is a common and useful capability. (Examples)

There are many file formats in use, and many Python libraries with varying capabilities.

This poster is decision-making roadmap in this problem space.

Do you face the common usability problems when an Excel file driven web application is delivered for non-developer users, or need to process different Excel formats for your program?

Not everyone knows (or cares) about the differences between various Excel formats. (i.e. CSV, XLS, XLSX)

Instead of training Excel users about file formats, the various Python libraries help developers to handle most of the Excel file formats by providing a common programming interface.

Packages

Working with Excel files in Python

Key
- GitHub Repo
- Bitbucket Repo
- Integration with Pandas

- xlrd
- xlwt
- xlutils
- openpyxl
- xlsxwriter
- pyexcel
- pylightxl

Format/Style

Examples: format with xlsxwriter and style with openpyxl

```
# Sample data to run the conditional formatting against
data = [
    ['sepal_length', 'sepal_width',
     'petal_length', 'petal_width', 'species'],
    [5.1, 3.5, 1.4, 0.2, 'setosa'],
    [4.9, 3.0, 1.4, 0.2, 'setosa'],
    [4.7, 3.2, 1.3, 0.2, 'setosa'],
    [4.6, 3.1, 1.5, 0.2, 'setosa'],
    [5.0, 3.6, 1.4, 0.2, 'setosa']]
]
```

Example of how to add conditional formatting to a range of cells (i.e. A4:A8)

Better for formulas and formatting → xlsxwriter

	A	B	C	D	E	F
1	Cells with values >= 5 are in light red. Values < 5 are in light green.					
2						
3	sepal_lengt	sepal_widt	petal_lengt	petal_widt	species	
4	5.1	3.5	1.4	0.2	setosa	
5	4.9	3	1.4	0.2	setosa	
6	4.7	3.2	1.3	0.2	setosa	
7	4.6	3.1	1.5	0.2	setosa	
8	5	3.6	1.4	0.2	setosa	

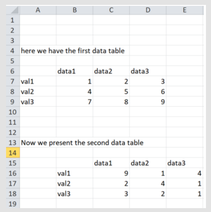
Input iris dataset (n=5)

	A	B	C	D	E
1	sepal_length	sepal_width	petal_length	petal_width	species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa

openpyxl → Better for style

	A	B	C	D	E
1	sepal_length	sepal_width	petal_length	petal_width	species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa

Example: pylightxl



Input semi-structured data

```
here we have the first data table

        data1   data2   data3
val1        1       2       3
val2        4       5       6
val3        7       8       9


Now we present the second data table

        data1   data2   data3
val1        9       1       4
val2        2       4       1
val3        3       2       1
```

```
import pylightxl as xl

db = xl.readxl("../data/Book1.xlsx")

keyrows = [
    rowID
    for rowID, row in enumerate(db.ws("Sheet1").rows, 1)
    if "val1" in row
]

keycols = []
for keyrow in keyrows:

    keycols.append(
        db.ws("Sheet1").row(keyrow).index("val1") + 1
    )
```

```
datagroups = {}

for tableIndex, keyrow in enumerate(keyrows, 1):
    i = 0
    datagroups.update({tableIndex: []})
    while True:
        datarow = db.ws("Sheet1").row(keyrow + i)[
            keycols[tableIndex - 1] :
        ]
        if datarow[0] == "":
            break
        datagroups[tableIndex].append(datarow)
        i += 1
```

Output datagroups

```
print(datagroups[1])
>>> [[1, 2, 3, ''], [4, 5, 6, ''], [7, 8, 9, '']]
print(datagroups[2])
>>> [[9, 1, 4], [2, 4, 1], [3, 2, 1]]
```

Example: pyexcel

```
import json
import pyexcel as pe

fn = "../data/iris_short.xlsx"
fn2 = "../data/iris_short_2.xlsx"

# Get array (list of lists) from Excel
pe.get_array(file_name=fn)
```

```
[['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
'species'],
 [5.1, 3.5, 1.4, 0.2, 'setosa'],
 [4.9, 3, 1.4, 0.2, 'setosa'],
 [4.7, 3.2, 1.3, 0.2, 'setosa'],
 [4.6, 3.1, 1.5, 0.2, 'setosa']]
```

```
# Get dict from single sheet
ws_dict = pe.get_dict(
    file_name=fn,
    name_columns_by_row=0,
)

for k, v in ws_dict.items():
    print({str(k): v})
```

```
{'sepal_length': [5.1, 4.9, 4.7, 4.6]}
{'sepal_width': [3.5, 3, 3.2, 3.1]}
{'petal_length': [1.4, 1.4, 1.3, 1.5]}
{'petal_width': [0.2, 0.2, 0.2, 0.2]}
{'species': ['setosa', 'setosa', 'setosa', 'setosa']}
```

```
# Get dict from multiple sheet book
wb_dict = pe.get_book_dict(file_name=fn2)

for k, item in wb_dict.items():
    print(json.dumps({k: item}))
```

```
{"Sheet1": [["sepal_length", "sepal_width", "petal_length", "petal_width",
"species"], [5.1, 3.5, 1.4, 0.2, "setosa"], [4.9, 3, 1.4, 0.2, "setosa"],
[4.7, 3.2, 1.3, 0.2, "setosa"], [4.6, 3.1, 1.5, 0.2, "setosa"]]}
{"Sheet2": [["sepal_length", "sepal_width", "petal_length", "petal_width",
"species"], [5.1, 3.5, 1.4, 0.2, "setosa"], [4.9, 3, 1.4, 0.2, "setosa"],
[4.7, 3.2, 1.3, 0.2, "setosa"], [4.6, 3.1, 1.5, 0.2, "setosa"]]}
```

Library Comparison

Information on python packages available to work with Excel files on Python platform

xlutils collects utilities that require both xlrd and xlwt; the use cases are now covered by openpyxl. (omitted in table)

package	xlrd	xlwt	openpyxl	xlsxwriter	pyexcel	pylightxl
Description	Reading data and formatting information from Excel files.	Writing data and formatting to older Excel files	Most mature Python library for reading and writing Excel files.	Writing data, formatting information and, in particular, charts in later Excel format	Providing one API for reading, manipulating and writing data in various excel formats.	A light weight Excel reader/writer with zero non-standard library dependencies.
Python Version	27, 34+	27, 3+	36+	27, 34+	27, 3+	27, 3+
File Formats	R: XLS, XLSX	W: XLS	R,W: XLSX, XLSM, XLTX, XLTM	W: XLSX	R,W: CSV, ODS, XLS, XLSX, XLSM...	R: XLSX, XLSM; W: XLSX
Supports Formats/Style			✓	✓		
Pros	Read older Excel file (XLS). Loading worksheets on demand.	Write to older Excel files (XLS).	Able to stream dataframes straight to file, adding formulas is simple.	Possible to write more than one dataframe to a worksheet or to multiple worksheets.	The data in excel files can be turned into array or dict with minimal code.	Small library, pythonic and easy to use. Can read semi-structured data.
Cons	This library currently has no active maintainers. You are advised to use openpyxl	Poor documentation	Can write functions to generate dict, but will not be as concise as pyexcel.			Does not support worksheet cell data more than 536,870,912 cells (32-bit list limitation)
Dependancies			Support for lxml library if installed.		Need to install extra pyexcel plugins.	
Development Status	Inactive	Inactive	Production/Stable	Production/Stable	Production/Stable	Active

Recommendations

Based on the Library Comparison table, here are the recommendations:

- In general, if you are using older Excel files (XLS), use xlrd/xlwt (reader/writer).
- xlsxwriter supports full formatting and more.
- If you want to add style to output files, openpyxl can generate nicely formatted spreadsheets (powerful feature of traditional spreadsheet). It supports format styles, add comment, add charts and tables, parse formulas, protect workbook from modification.
- If output of the data in Excel files is array or dict, you can use pyexcel with minimal code.
- If input is semi-structured data (single sheet that can begin with any row/col and has any number of rows/col per data group), use pylightxl.

References

The best place to start when working with Excel files in Python is this website:
http://www.python-excel.org

pyexcel provides one application programming interface to read, manipulate and write data in various excel formats:
http://docs.pyexcel.org/en/latest/

pylightxl is my more recent discovery (the writer part was being written as I was exploring):
https://pylightxl.readthedocs.io/en/latest/

Contact

Contact via:

: provocateurtx@yahoo.com

: https://github.com/BCM-HGSC/pycon2020_excel_poster

Acknowledgements

Mahitha Rajendran for constructive feedback and Walker Hale for being the source of encouragement, both at Baylor College of Medicine, Houston, TX.
Viktor Kis for allowing me to use the pylightxl logo and code snippets.
Autumn Watt at Intel, Inc. for her honest critique and brilliant style editing.
Kevin Burleigh for his critical yet thoughtful review.
MakeSigns for the Scientific Poster PowerPoint template and helpful chat.