xlrd

Library for developers to extract data from Microsoft Excel (tm) spreadsheet files

# Pypi.python.org

<https://pypi.python.org/pypi/xlrd>

Extract data from Excel spreadsheets (.xls and .xlsx, versions 2.0 onwards) on any platform. Pure Python (2.6, 2.7, 3.2+). Strong support for Excel dates. Unicode-aware.

# Portail SIG

<http://www.portailsig.org/content/python-lire-et-ecrire-des-fichiers-microsoft-excel-application-quantum-gis>

## Python : lire et écrire des fichiers Microsoft Excel, application à Quantum GIS

ven 13-05-2011 [Martin Laloux](http://www.portailsig.org/users/gene)

* [SIG OpenSource](http://www.portailsig.org/category/actudossier/sig-opensource)

|  |
| --- |
| http://www.portailsig.org/sites/default/files/images/icone/python-powered-h-140x182.png |
| **Niveau** | Intermédiaire |
| **Logiciels utilisés** | [**Python**](http://www.python.org/)  [**xlrd**](http://pypi.python.org/pypi/xlrd/0.7.1)  [**xlwt**](http://pypi.python.org/pypi/xlwt/0.7.2) |
| **Plateforme** | Windows | Mac | Linux | FreeBSD |

S'il y a une question récurrente sur les forums [SIG](http://www.portailsig.org/glossary/4/letters#term28), c'est comment traiter des données contenues dans des fichiers Microsoft Excel dans des SIGs comme Quantum GIS et autres (voir par exemple [www.forumsig.org/showthread.php](http://www.forumsig.org/showthread.php?t=30831))

Jusqu'à peu, les utilisateurs de Windows étaient favorisés car le module Pywin (qui utilise com) pouvait être utilisé pour cette tâche (voir  « Python for Windows, Ressources and examples »,  par exemple). Rien pour les autres...

Heureusement, deux modules « universels » sont sortis , **xlrd** pour lire les données et **xlwt** pour écrire des données (classeur, feuilles etc.).

## Lecture d'un fichier Excel

1. **import** xlrd
2. *# ouverture du fichier Excel*
3. wb = xlrd.open\_workbook('testxy.xls')
5. *# feuilles dans le classeur*
6. **print** wb.sheet\_names()
7. [u'Feuil1', u'Feuil2', u'Feuil3']
9. *# lecture des données dans la première feuille*
10. sh = wb.sheet\_by\_name(u'Feuil1')
11. **for** rownum **in** range(sh.nrows):
12. **print** sh.row\_values(rownum)
13. [u'id', u'x', u'y', u'test']
14. [1.0, 235.0, 424.0, u'a']
15. [2.0, 245.0, 444.0, u'b']
16. [3.0, 255.0, 464.0, u'c']
17. [4.0, 265.0, 484.0, u'd']
18. [5.0, 275.0, 504.0, u'e']
19. [6.0, 285.0, 524.0, u'f']
20. [7.0, 295.0, 544.0, u'g']
22. *# lecture par colonne*
23. colonne1 = sh.col\_values(0)
24. **print** colonne1
25. [u'id', 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0]
27. colonne2=sh.col\_values(1)
28. **print** colonne2
29. [u'x', 235.0, 245.0, 255.0, 265.0, 275.0, 285.0, 295.0]
31. *# extraction d'un élément particulier*
32. **print** colonne1[1],colonne2[1]
33. 1.0 235.0
35. etc...

## Création d'un fichier Excel

1. **from** xlwt **import** Workbook
3. *# création*
4. book = Workbook()
6. *# création de la feuille 1*
7. feuil1 = book.add\_sheet('feuille 1')
9. *# ajout des en-têtes*
10. feuil1.write(0,0,'id')
11. feuil1.write(0,1,'x')
12. feuil1.write(0,2,'y')
13. feuil1.write(0,3,'test')
15. *# ajout des valeurs dans la ligne suivante*
16. ligne1 = feuil1.row(1)
17. ligne1.write(0,'1')
18. ligne1.write(1,'235.0')
19. ligne1.write(2,'424.0')
20. ligne1.write(3,'a')
21. etc...
23. *# ajustement éventuel de la largeur d'une colonne*
24. feuil1.col(0).width = 10000
26. *# éventuellement ajout d'une autre feuille 2*
27. feuil2 = book.add\_sheet('feuille 2')
29. etc...

32. *# création matérielle du fichier résultant*
33. book.save('monsimple.xls')

## Principes

Les modules sont basés sur la documentation d'OpenOffice sur le format des fichiers Microsoft Excel ( [sc.openoffice.org/excelfileformat.pdf](http://sc.openoffice.org/excelfileformat.pdf" \t "_blank) ) et sont écrits en pur Python, c'est-à-dire disponibles pour toutes les plateformes. Ils ne peuvent traiter que les fichiers de type .xls. Pour traiter les fichiers de type .xlsx, il existe le module **tablib**, « Format agnostic tabular data library (XLS, JSON, YAML, CSV) » ([pypi.python.org/pypi/tablib/](http://pypi.python.org/pypi/tablib/)).

## Pour aller plus loin

Le site [www.python-excel.org/](http://www.python-excel.org/) fournit tous les éléments pour aller plus loin, en particulier un tutoriel détaillé ([python-excel.googlegroups.com/web/python-excel.pdf](http://python-excel.googlegroups.com/web/python-excel.pdf?gda=y44Uc0IAAAAVzH7yYteCZsiPEIYpN2jAGLIElKAcGA3QWlOTd4LDldXEtWX7y7b6ByPnawQ3OxpV4u3aa4iAIyYQIqbG9naPgh6o8ccLBvP6Chud5KMzIQ), toutes les possibilités d'OpenOffice/Libre Office sont quasi disponibles). Un module offrant des fonctions supplémentaires est aussi disponible, **xlutils**. Il permet de copier, de modifier et/ou de filtrer des fichiers .xls existants.

## Commentaires

### erreur dans le code:

il y a une erreur lors de la creation du fichier excel :

# création

test = Workbook()

je remplacerais par 'book'

Pareil pour la fin : book.save

Lorsque vous créer les feuilles dans le fichier vous appelez bien :

'book.xxxx'

il faut que tout les objets correspondent.

### merci pour signaler une

merci de signaler une erreur mais elle n'est pas où vous le soulignez:

la fonction = workbook() est obligatoire pour créer un classeur

ensuite vous ajoutez une feuille au classeur avec la fonction .add\_sheet()

Donc le script correct est:

# création du classeur book

book = Workbook()

# ajout d'une feuille au classeur book

feuil1 = book.add\_sheet('feuille 1')

### Bonjour, Python dev a

Anonyme (non vérifié) - 19/06/2014 - 10:15

Python dev a exactement raison, il ya une erreur. Je crois qu'il ne s'est pas bien expliqué.

A la ligne 4 vous mettez 'book = Workbook()' et à la ligne 33 vous mettez 'test.save('monsimple.xls')' et non 'book.save('monsimple.xls')'.

### suite

merci, cela vient de la correction demandée par Python dev (book à la place de test) et d'un oubli de ma part pour la seconde correction.