xlwt

Release 1.3.0

Aug 22, 2017

Contents

1 API Reference 3

2 Installation Instructions 7

3 Development 9

4 Changes 11

5 Licenses 15

6 Indices and tables 19

Python Module Index 21

xlwt is a library for writing data and formatting information to older Excel files (ie: .xls)

Documentation is sparse, please see the API reference or code for help:

CHAPTER 1

API Reference

class xlwt.Workbook.**Workbook**(*encoding='ascii'*, *style_compression=0*)

This is a class representing a workbook and all its contents. When creating Excel files with xlwt, you will normally start by instantiating an object of this class.

add_sheet(*sheetname*, *cell_overwrite_ok=False*)

This method is used to create Worksheets in a Workbook.

Parameters

- **sheetname** – The name to use for this sheet, as it will appear in the tabs at the bottom of the Excel application.
- **cell_overwrite_ok** – If True, cells in the added worksheet will not raise an exception if written to more than once.

Returns The *Worksheet* that was added.

save(*filename_or_stream*)

This method is used to save the Workbook to a file in native Excel format.

Parameters **filename_or_stream** – This can be a string containing a filename of the file, in which case the excel file is saved to disk using the name provided. It can also be a stream object with a write method, such as a StringIO, in which case the data for the excel file is written to the stream.

class xlwt.Worksheet.**Worksheet**(*sheetname*, *parent_book*, *cell_overwrite_ok=False*)

This is a class representing the contents of a sheet in a workbook.

> **Warning:** You don't normally create instances of this class yourself. They are returned from calls to *add_sheet()*.

write(*r*, *c*, *label=''*, *style=<xlwt.Style.XFStyle object>*)

This method is used to write a cell to a *Worksheet*.

Parameters

- **r** – The zero-relative number of the row in the worksheet to which the cell should be written.
- **c** – The zero-relative number of the column in the worksheet to which the cell should be written.
- **label** – The data value to be written.

  An `int`, `long`, or `Decimal` instance is converted to `float`.

  A `unicode` instance is written as is. A `bytes` instance is converted to `unicode` using the encoding, which defaults to `ascii`, specified when the `Workbook` instance was created.

  A `datetime`, `date` or `time` instance is converted into Excel date format (a float representing the number of days since (typically) `1899-12-31T00:00:00`, under the pretence that 1900 was a leap year).

  A `bool` instance will show up as `TRUE` or `FALSE` in Excel.

  `None` causes the cell to be blank: no data, only formatting.

  An `xlwt.Formula` instance causes an Excel formula to be written.
- **style** – A style, also known as an XF (extended format), is an `XFStyle` object, which encapsulates the formatting applied to the cell and its contents.

  `XFStyle` objects are best set up using the *`easyxf()`* function. They may also be set up by setting attributes in `Alignment`, `Borders`, `Pattern`, `Font` and `Protection` objects then setting those objects and a format string as attributes of an `XFStyle` object.

Formatting

The XF record is able to store explicit cell formatting attributes or the attributes of a cell style. Explicit formatting includes the reference to a cell style XF record. This allows to extend a defined cell style with some explicit attributes. The formatting attributes are divided into 6 groups:

Group	Attributes
Number format	Number format index (index to FORMAT record)
Font	Font index (index to FONT record)
Alignment	Horizontal and vertical alignment, text wrap, indentation, orientation/rotation, text direction
Border	Border line styles and colours
Background	Background area style and colours
Protection	Cell locked, formula hidden

For each group a flag in the cell XF record specifies whether to use the attributes contained in that XF record or in the referenced style XF record. In style XF records, these flags specify whether the attributes will overwrite explicit cell formatting when the style is applied to a cell. Changing a cell style (without applying this style to a cell) will change all cells which already use that style and do not contain explicit cell attributes for the changed style attributes. If a cell XF record does not contain explicit attributes in a group (if the attribute group flag is not set), it repeats the attributes of its style XF record.

`xlwt.Style.`**`easyxf`**(*strg_to_parse=''*, *num_format_str=None*, *field_sep=','*, *line_sep=';'*, *intro_sep=':'*, *esc_char='\\'*, *debug=False*)

This function is used to create and configure `XFStyle` objects for use with (for example) the `Worksheet.write()` method.

It takes a string to be parsed to obtain attribute values for `Alignment`, `Borders`, `Font`, `Pattern` and `Protection` objects.

Refer to the examples in the file *examples/xlwt_easyxf_simple_demo.py* and to the *xf_dict* dictionary in `xlwt.Style`.

Various synonyms including color/colour, center/centre and gray/grey are allowed. Case is irrelevant (except maybe in font names). - may be used instead of _.

Example: `font: bold on; align: wrap on, vert centre, horiz center`

Parameters **num_format_str** – To get the "number format string" of an existing cell whose format you want to reproduce, select the cell and click on Format/Cells/Number/Custom. Otherwise, refer to Excel help.

Examples: `"#,##0.00"`, `"dd/mm/yyyy"`

Returns An `XFstyle` object.

Perhaps more useful is to consult the tutorial and the examples in the *examples* folder of the distribution.

For details of how to install the package or get involved in its development, please see the sections below:

CHAPTER 2

Installation Instructions

If you want to experiment with xlwt, the easiest way to install it is to do the following in a virtualenv:

```
pip install xlwt
```

If your package uses setuptools and you decide to use xlwt, then you should add it as a requirement by adding an `install_requires` parameter in your call to `setup` as follows:

```
setup(
    # other stuff here
    install_requires=['xlwt'],
    )
```

CHAPTER 3

Development

This package is developed using continuous integration which can be found here:

https://travis-ci.org/python-excel/xlwt

If you wish to contribute to this project, then you should fork the repository found here:

https://github.com/python-excel/xlwt

Once that has been done and you have a checkout, you can follow these instructions to perform various development tasks:

Setting up a virtualenv

The recommended way to set up a development environment is to turn your checkout into a virtualenv and then install the package in editable form as follows:

```
$ virtualenv .
$ bin/pip install -Ur requirements.txt
$ bin/pip install -e .
```

Running the tests

Once you've set up a virtualenv, the tests can be run as follows:

```
$ bin/nosetests
```

To run tests on all the versions of Python that are supported, you can do:

```
$ bin/tox
```

If you change the supported python versions in `.travis.yml`, please remember to do the following to update `tox.ini`:

```
$ bin/panci --to=tox .travis.yml > tox.ini
```

Building the documentation

The Sphinx documentation is built by doing the following, having activated the virtualenv above, from the directory containing setup.py:

```
$ cd docs
$ make html
```

Making a release

To make a release, just update the version in `xlwt/__init__.py`, update the change log, tag it and push to https://github.com/python-excel/xlwt and Travis CI should take care of the rest.

Once the above is done, make sure to go to https://readthedocs.org/projects/xlwt/versions/ and make sure the new release is marked as an Active Version.

CHAPTER 4

Changes

1.3.0 (22 August 2017)

- Officially support Python 3.6, drop support for 2.6.
- Fix bytes/string type mismatch in `upack2rt()` on python 3.
- Packaging and code style tweaks.
- Use generator expressions to avoid unnecessary lists in memory.

Thanks to the following for their contributions to this release:

- Jon Dufresne
- Bill Adams

1.2.0 (4 January 2017)

- Remove `LOCALE` from regular expression that caused `DeprecationWarning` that become an exception in Python 3.6
- Add `Workbook.sheet_index()` helper.
- `Workbook.get_sheet()` now takes either a string name or an integer index.

1.1.2 (9 June 2016)

- Fix failure in style compression under Python 3.
- Officially support Python 3.5
- Documentation tweaks.

1.1.1 (2 June 2016)

- Fix release problems.

1.1.0 (2 June 2016)

- Fix SST BIFF record in Python 3.
- Fix for writing `ExternSheetRecord` in Python 3.
- Add the ability to insert bitmap images from buffers as well as files.
- Official support for Python 3.5.

Thanks to "thektulu" and Lele Gaifax for the Python 3 fixes. Thanks to Ross Golder for the support for inserting images from buffers.

1.0.0 (15 April 2015)

- Python 3 support.
- Initial set of unit tests.
- An initial set of Sphinx documentation.
- Move to setuptools for packaging.
- Wire up Travis, Coveralls and ReadTheDocs.
- Allow longs as row indexes.

Big thanks to Thomas Kluyver for his work on Python 3 support, Manfred Moitzi for donating his unit tests.

Belated thanks to Landon Jurgens for his help on converting the documentation to Sphinx.

0.7.5 (5 April 2013)

- Fixes a bug that could cause a corrupt SST in .xls files written by a wide-unicode Python build.
- A `ValueError` is now raised immediately if an attempt is made to set column width to other than an int in `range(65536)`
- Added the ability to set a custom RGB colour in the palette to use for colours. Thanks to Alan Rotman for the work, although this could really use an example in the examples folder...
- Fixed an issue trying to set a diagonal border using easyxf. Thanks to Neil Etheridge for the fix.
- Fixed a regression from 0.7.2 when writing sheets with frozen panes.

0.7.4 (13 April 2012)

- Python 2.3 to 2.7 are now the officially supported versions, no Python 3 yet, sorry.

- The `datemode` in an xlwt *`Workbook`* can be set to 1904 by doing `workbook.dates_1904 = 1` and is written to the output file. However the datemode was not being reflected in conversions from `datetime.datetime` and `datetime.date` objects to floats for output, resulting in dates that were 4 years too high when seen in Excel.

0.7.3 (21 February 2012)

- Added user_set and best_fit attributes to Column class.
- Fixed an `[Errno 0] Error` raised when `Worksheet.flush_row_data()` was called after `Workbook.save()`
- Fixed an error on Windows that occurred when writing large blocks to files.
- Added the ability to write rich text cells
- Fixed a bug when writing `MULBLANK` records on big-endian platforms.
- allow the `active_pane` on worksheets to be specified
- added support for zoom (magn) factors and improved possibilities when generating split panes

0.7.2 (1 June 2009)

- Added function Utils.rowcol_pair_to_cellrange. `(0, 0, 65535, 255) -> "A1:IV65536"`
- Removed *`Worksheet`* property `show_empty_as_zero`, and added attribute `show_zero_values` (default: `1 == True`).
- Fixed formula code generation problem with formulas including MAX/SUM/etc functions with arguments like A1+123.
- Added .pattern_examples.xls and put a pointer to it in the easyxf part of Style.py.
- Fixed Row.set_cell_formula() bug introduced in 0.7.1.
- Fixed bug(?) with SCL/magnification handling causing(?) Excel to raise a dialogue box if sheet is set to open in page preview mode and user then switches to normal view.
- Added color and colour as synonyms for font.colour_index in easyxf.
- Removed unused attribute Row.__has_default_format.

0.7.1 (4 March 2009)

See source control for changes made.

0.7.0 (19 September 2008)

- Fixed more bugs and added more various new bits of functionality

0.7.0a4 (8 October 2007)

- fork of pyExcelerator, released to python-excel.
- Fixed various bugs in pyExcelerator and added various new bits of functionality

CHAPTER 5

Licenses

xlwt has various licenses that apply to the different parts of it, they are listed below:

The license for the work John Machin has done since xlwt was created:

```
Portions copyright (c) 2007, Stephen John Machin, Lingfo Pty Ltd
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

3. None of the names of Stephen John Machin, Lingfo Pty Ltd and any
contributors may be used to endorse or promote products derived from this
software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.
```

The licensing for the unit tests added as part of the work for Python 3 compatibility is as follows:

```
Author:  mozman --<mozman@gmx.at>
Purpose: test_mini
Created: 03.12.2010
Copyright (C) 2010, Manfred Moitzi
License: BSD licence
```

The license for pyExcelerator, from which xlwt was forked:

```
  Copyright (C) 2005 Roman V. Kiseliov
  All rights reserved.

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions
  are met:

  1. Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.

  2. Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in
     the documentation and/or other materials provided with the
     distribution.

  3. All advertising materials mentioning features or use of this
     software must display the following acknowledgment:
     "This product includes software developed by
      Roman V. Kiseliov <roman@kiseliov.ru>."

  4. Redistributions of any form whatsoever must retain the following
     acknowledgment:
     "This product includes software developed by
      Roman V. Kiseliov <roman@kiseliov.ru>."

  THIS SOFTWARE IS PROVIDED BY Roman V. Kiseliov ``AS IS'' AND ANY
  EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
  PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL Roman V. Kiseliov OR
  ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
  NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
  OF THE POSSIBILITY OF SUCH DAMAGE.

Roman V. Kiseliov
Russia
Kursk
Libknecht St., 4

+7(0712)56-09-83

<roman@kiseliov.ru>
```

Portions of xlwt.Utils are based on pyXLWriter which is licensed as follows:

```
Copyright (c) 2004 Evgeny Filatov <fufff@users.sourceforge.net>
Copyright (c) 2002-2004 John McNamara (Perl Spreadsheet::WriteExcel)

This library is free software; you can redistribute it and/or modify it
under the terms of the GNU Lesser General Public License as published by
the Free Software Foundation; either version 2.1 of the License, or
(at your option) any later version.

This library is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU Lesser
General Public License for more details:

https://www.gnu.org/licenses/lgpl.html
```

pyXLWriter also makes reference to the PERL Spreadsheet::WriteExcel as follows:

```
This module was written/ported from PERL Spreadsheet::WriteExcel module
The author of the PERL Spreadsheet::WriteExcel module is John McNamara
<jmcnamara@cpan.org>
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

X

xlwt.Bitmap, 5
xlwt.Cell, 4
xlwt.Column, 4
xlwt.CompoundDoc, 5
xlwt.Formatting, 4
xlwt.Row, 4
xlwt.Style, 4
xlwt.Workbook, 3
xlwt.Worksheet, 3

Index

A

add_sheet() (xlwt.Workbook.Workbook method), 3

E

easyxf() (in module xlwt.Style), 4

S

save() (xlwt.Workbook.Workbook method), 3

W

Workbook (class in xlwt.Workbook), 3
Worksheet (class in xlwt.Worksheet), 3
write() (xlwt.Worksheet.Worksheet method), 3

X

xlwt.Bitmap (module), 5
xlwt.Cell (module), 4
xlwt.Column (module), 4
xlwt.CompoundDoc (module), 5
xlwt.Formatting (module), 4
xlwt.Row (module), 4
xlwt.Style (module), 4
xlwt.Workbook (module), 3
xlwt.Worksheet (module), 3