xlwt 1.2.0

Library to create spreadsheet files compatible with MS Excel 97/2000/XP/2003 XLS files, on any platform, with Python 2.6, 2.7, 3.3+

Pypi.python.org

xlwt

This is a library for developers to use to generate spreadsheet files compatible with Microsoft Excel versions 95 to 2003.

The package itself is pure Python with no dependencies on modules or packages outside the standard Python distribution.

Installation

Do the following in your virtualenv:

```
pip install xlwt
```

Quick start

```
import xlwt
from datetime import datetime

style0 = xlwt.easyxf('font: name Times New Roman, color-index red, bold
on',
    num_format_str='#,##0.00')
style1 = xlwt.easyxf(num_format_str='D-MMM-YY')

wb = xlwt.Workbook()
ws = wb.add_sheet('A Test Sheet')

ws.write(0, 0, 1234.56, style0)
ws.write(1, 0, datetime.now(), style1)
ws.write(2, 0, 1)
ws.write(2, 1, 1)
ws.write(2, 2, xlwt.Formula("A3+B3"))

wb.save('example.xls')
```

Documentation

Documentation can be found in the `docs` directory of the xlwt package. If these aren't sufficient, please consult the code in the examples directory and the source code itself.

The latest documentation can also be found at: http://xlwt.readthedocs.org/en/latest/

Xlwt.readthedocs.io

http://xlwt.readthedocs.io/en/latest/index.html

xlwt documentation

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:

- API Reference

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:

- Installation Instructions
- Development
- Changes
- Licenses

Indices and tables

- Index
- Module Index
- Search Page

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.

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'],
    )
```

Python version requirements

This package has been tested with Python 2.6, 2.7, 3.3+ on Linux, and is also expected to work on Mac OS X and Windows.