xlwings - Make Excel Fly!

Release 0.2.0

Zoomer Analytics LLC

July 29, 2014

CONTENTS

CHAPTER

ONE

WHAT'S NEW

Here are the Release Notes for each version:

1.1 v0.2.0 (July 29, 2014)

1.1.1 Enhancements

- Cross-platform: xlwings is now additionally supporting Microsoft Excel for Mac. The only functionality that is not yet available is the possibility to call the Python code from within Excel via a VBA macro.
- The `clear` and `clear_contents` methods of the `Workbook` object now default to the active sheet (GH5):

```
wb = Workbook()
wb.clear_contents()  # Clears contents of the entire active sheet
```

1.1.2 Bug Fixes

- DataFrames with MultiHeaders were sometimes getting truncated (GH41).

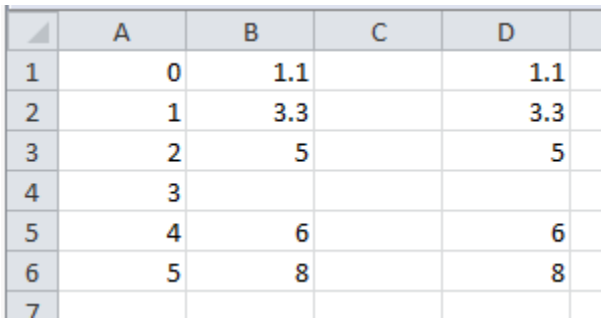
1.2 v0.1.1 (June 27, 2014)

1.2.1 Enhancements

- xlwings is now officially suppported on Python 2.6-2.7 and 3.1-3.4
- Support for Pandas `Series` has been added (GH24):

```
>>> import numpy as np
>>> import pandas as pd
>>> from xlwings import Workbook, Range
>>> wb = Workbook()
>>> s = pd.Series([1.1, 3.3, 5., np.nan, 6., 8.])
>>> s
```

```
0    1.1
1    3.3
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
>>> Range('A1').value = s
>>> Range('D1', index=False).value = s
```



	A	B	C	D
1	0	1.1		1.1
2	1	3.3		3.3
3	2	5		5
4	3			
5	4	6		6
6	5	8		8
7				

- Excel constants have been added under their original Excel name, but categorized under their enum (GH18), e.g.:

```
# Extra long version
import xlwings as xl
xl.constants.ChartType.xlArea

# Long version
from xlwings import constants
constants.ChartType.xlArea

# Short version
from xlwings import ChartType
ChartType.xlArea
```

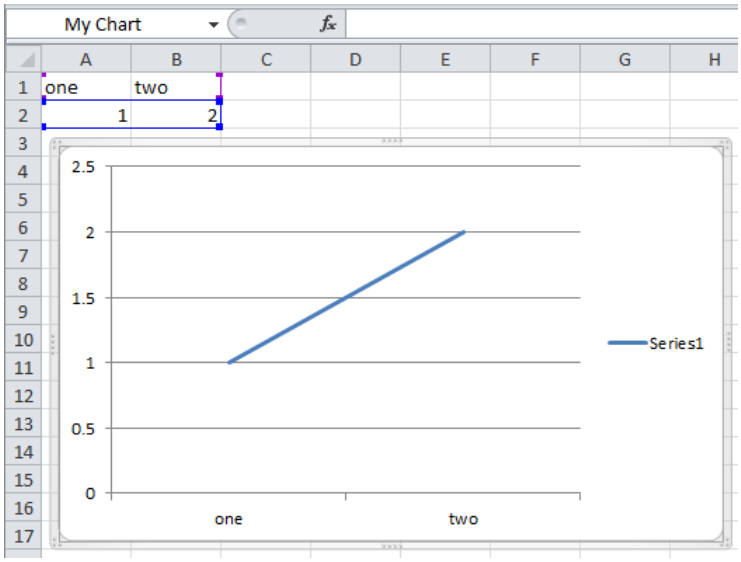
- Slightly enhanced Chart support to control the `ChartType` (GH1):

```
>>> from xlwings import Workbook, Range, Chart, ChartType
>>> wb = Workbook()
>>> Range('A1').value = [['one', 'two'],[10, 20]]
>>> my_chart = Chart().add(chart_type=ChartType.xlLine,
                           name='My Chart',
                           source_data=Range('A1').table)
```

alternatively, the properties can also be set like this:

```
>>> my_chart = Chart().add()  # To work with an existing Chart: my_chart = Chart('My C
>>> my_chart.name = 'My Chart'
>>> my_chart.chart_type = ChartType.xlLine
>>> my_chart.set_source_data(Range('A1').table)
```

- `pytz` is no longer a dependency as `datetime` object are now being read in from Excel as time-zone naive (Excel doesn't know timezones). Before, `datetime` objects got the UTC timezone attached.



- The `Workbook` object has the following additional methods: `close()`
- The `Range` object has the following additional methods: `is_cell()`, `is_column()`, `is_row()`, `is_table()`

1.2.2 API Changes

- If `asarray=True`, NumPy arrays are now always at least 1d arrays, even in the case of a single cell (GH14):

```
>>> Range('A1', asarray=True).value
array([34.])
```

- Similar to NumPy's logic, 1d Ranges in Excel, i.e. rows or columns, are now being read in as flat lists or 1d arrays. If you want the same behavior as before, you can use the `atleast_2d` keyword (GH13).

Note: The `table` property is also delivering a 1d array/list, if the table Range is really a column or row.

	A	B	C	D	E	F
1	1		1	2	3	4
2	2					
3	3					
4	4					

```
>>> Range('A1').vertical.value
[1.0, 2.0, 3.0, 4.0]
>>> Range('A1', atleast_2d=True).vertical.value
[[1.0], [2.0], [3.0], [4.0]]
```

```
>>> Range('C1').horizontal.value
[1.0, 2.0, 3.0, 4.0]
>>> Range('C1', atleast_2d=True).horizontal.value
[[1.0, 2.0, 3.0, 4.0]]
>>> Range('A1', asarray=True).table.value
array([ 1.,  2.,  3.,  4.])
>>> Range('A1', asarray=True, atleast_2d=True).table.value
array([[ 1.],
       [ 2.],
       [ 3.],
       [ 4.]])
```

- The single file approach has been dropped. xlwings is now a traditional Python package.

1.2.3 Bug Fixes

- Writing `None` or `np.nan` to Excel works now (GH16) & (GH15).
- The import error on Python 3 has been fixed (GH26).
- Python 3 now handles Pandas DataFrames with MultiIndex headers correctly (GH39).
- Sometimes, a Pandas DataFrame was not handling `nan` correctly in Excel or numbers were being truncated (GH31) & (GH35).
- Installation is now putting all files in the correct place (GH20).

1.3 v0.1.0 (March 19, 2014)

Initial release of xlwings.

CHAPTER

TWO

INSTALLATION

The easiest way to install xlwings is via pip:

```
pip install xlwings
```

Alternatively, it can be installed from source. From within the `xlwings` directory, execute:

```
python setup.py install
```

2.1 Dependencies

- **Windows**: pywin32
- **Mac**: psutil, appscript

Note that on Mac, the dependencies are automatically being handled if xlwings is installed with pip. However, the Xcode command line tools need to be available.

On Windows, it is recommended to use one of the scientific Python distributions like Anaconda, WinPython or Canopy as they already include pywin32. Otherwise it needs to be installed from here.

2.2 Python version support

xlwings runs on Python 2.6-2.7 and 3.1-3.4

CHAPTER

THREE

QUICKSTART

This guide assumes you have xlwings already installed. If that's not the case, head over to *Installation*.

3.1 Interact with Excel from Python

Writing/reading values to/from Excel and adding a chart is as easy as:

```
>>> from xlwings import Workbook, Range, Chart
>>> wb = Workbook()  # Creates a connection with a new workbook
>>> Range('A1').value = ['Foo 1', 'Foo 2', 'Foo 3', 'Foo 4']
>>> Range('A2').value = [10, 20, 30, 40]
>>> Range('A1').table.value  # Read the whole table back
[[u'Foo 1', u'Foo 2', u'Foo 3', u'Foo 4'], [10.0, 20.0, 30.0, 40.0]]
>>> chart = Chart().add(source_data=Range('A1').table)
```

The Range object as used above will refer to the active sheet. Include the Sheet name like this:

```
Range('Sheet1', 'A1').value
```

Qualify the Workbook additionally like this:

```
wb.range('Sheet1', 'A1').value
```

The good news is that these commands also work seamlessly with *NumPy arrays* and *Pandas DataFrames*.

3.2 Call Python from Excel (Windows only)

This functionality is currently only available on Windows: If, for example, you want to fill your spreadsheet with standard normally distributed random numbers, your VBA code is just one line:

```
Sub RandomNumbers()
    RunPython ("import mymodule; mymodule.rand_numbers()")
End Sub
```

This essentially hands over control to `mymodule.py`:

```
import numpy as np
from xlwings import Workbook, Range

wb = Workbook()  # Creates a reference to the calling Excel file

def rand_numbers():
    """ produces standard normally distributed random numbers with shape (n,n)"""
    n = Range('Sheet1', 'B1').value  # Write desired dimensions into Cell B1
    rand_num = np.random.randn(n, n)
    Range('Sheet1', 'C3').value = rand_num
```

To make this run, just import de VBA module `xlwings.bas` in the VBA editor (Open the VBA editor with `Alt-F11`, then go to `File > Import File...` and import the `xlwings.bas` file.). It can be found in the directory of your `xlwings` installation.

3.3 Easy deployment

Deployment is really the part where xlwings shines:

- Just zip-up your Spreadsheet with your Python code and send it around. The receiver only needs to have an installation of Python with xlwings (and obviously all the other packages you're using).
- There is no need to install any Excel add-in.
- If this still sounds too complicated, just freeze your Python code into an executable and use `RunFrozenPython` instead of `RunPython`. This gives you a standalone version of your Spreadsheet tool without any dependencies.

CHAPTER

FOUR

WORKBOOK OBJECT

In order to use xlwings, creating a workbook object is always the first thing to do:

class `xlwings.`**`Workbook`**(*fullname=None*)

Workbook connects an Excel Workbook with Python. You can create a new connection from Python with

- a new workbook: `wb = Workbook()`
- an existing workbook: `wb = Workbook(r'C:\path\to\file.xlsx')`

If you want to create the connection from Excel through the xlwings VBA module, use:

```
wb = Workbook()
```

Parameters **fullname** (*string, default None*) – If you want to connect to an existing Excel file from Python, use the fullname, e.g: `r'C:\path\to\file.xlsx'`

`activate`(*sheet*)

Activates the given sheet.

Parameters **sheet** (*string or integer*) – Sheet name or index.

`get_selection`(*asarray=False*)

Returns the currently selected Range from Excel as xlwings Range object.

Parameters **asarray** (*boolean, default False*) – returns a NumPy array where empty cells are shown as nan

Return type xlwings Range object

`range`(**args*, ***kwargs*)

The range method gets and sets the Range object with the following arguments:

```
range('A1')           range('Sheet1', 'A1')           range(1, 'A1')
range('A1:C3')        range('Sheet1', 'A1:C3')        range(1, 'A1:C3')
range((1,2))          range('Sheet1', (1,2))          range(1, (1,2))
range((1,1), (3,3))   range('Sheet1', (1,1), (3,3))   range(1, (1,1), (3,3))
range('NamedRange')   range('Sheet1', 'NamedRange')   range(1, 'NamedRange')
```

If no worksheet name is provided as first argument (as name or index), it will take the range from the active sheet.

Please check the available methods/properties directly under the Range object.

Parameters

- **asarray** (*boolean, default False*) – returns a NumPy array where empty cells are shown as nan
- **index** (*boolean, default True*) – Includes the index when setting a Pandas DataFrame
- **header** (*boolean, default True*) – Includes the column headers when setting a Pandas DataFrame

Returns xlwings Range object

Return type Range

chart(**args*, ***kwargs*)

The chart method gives access to the chart object and can be called with the following arguments:

```
chart(1)              chart('Sheet1', 1)              chart(1, 1)
chart('Chart 1')      chart('Sheet1', 'Chart 1')      chart(1, 'Chart 1')
```

If no worksheet name is provided as first argument (as name or index), it will take the Chart from the active sheet.

To insert a new Chart into Excel, create it as follows:

wb.chart().add()

Parameters ***args** – Definition of sheet (optional) and chart in the above described combinations.

clear_contents(*sheet=None*)

Clears the content of a whole sheet but leaves the formatting.

Parameters **sheet** (*string or integer, default None*) – Sheet name or index. If sheet is None, the active sheet is used.

clear(*sheet=None*)

Clears the content and formatting of a whole sheet.

Parameters **sheet** (*string or integer, default None*) – Sheet name or index. If sheet is None, the active sheet is used.

close()

Closes the Workbook without saving it

CHAPTER

FIVE

RANGE OBJECT

Analogous to its counterpart in Excel, the xlwings Range object represents a selection of cells containing one or more contiguous blocks of cells in Excel.

class xlwings.**Range**(**args*, ***kwargs*)

A Range object can be created with the following arguments:

```
Range('A1')           Range('Sheet1', 'A1')           Range(1, 'A1')
Range('A1:C3')        Range('Sheet1', 'A1:C3')        Range(1, 'A1:C3')
Range((1,2))          Range('Sheet1, (1,2))           Range(1, (1,2))
Range((1,1), (3,3))   Range('Sheet1', (1,1), (3,3))   Range(1, (1,1), (3,3))
Range('NamedRange')   Range('Sheet1', 'NamedRange')   Range(1, 'NamedRange')
```

If no worksheet name is provided as first argument (as name or index), it will take the Range from the active sheet.

You usually want to go for `Range(...).value` to get the values (as list of lists).

Parameters

- ***args** – Definition of sheet (optional) and Range in the above described combinations.
- **asarray** (*boolean, default False*) – Returns a NumPy array (atleast_1d) where empty cells are transformed into nan.
- **index** (*boolean, default True*) – Includes the index when setting a Pandas DataFrame or Series.
- **header** (*boolean, default True*) – Includes the column headers when setting a Pandas DataFrame.
- **atleast_2d** (*boolean, default False*) – Returns 2d lists/arrays even if the Range is a Row or Column.

is_cell()

Returns True if the Range consists of a single Cell otherwise False

is_row()

Returns True if the Range consists of a single Row otherwise False

is_column()

Returns True if the Range consists of a single Column otherwise False

is_table()
Returns True if the Range consists of a 2d array otherwise False

value
Gets and sets the values for the given Range.

Returns Empty cells are set to None. If `asarray=True`, a numpy array is returned where empty cells are set to nan.

Return type list or numpy array

formula
Gets or sets the formula for the given Range.

table
Returns a contiguous Range starting with the indicated cell as top-left corner and going down and right as long as no empty cell is hit.

Parameters **strict** (*boolean, default False*) – strict stops the table at empty cells even if they contain a formula. Less efficient than if set to False.

Return type xlwings Range object

Examples

To get the values of a contiguous range or clear its contents use:

```
Range('A1').table.value
Range('A1').table.clear_contents()
```

vertical
Returns a contiguous Range starting with the indicated cell and going down as long as no empty cell is hit. This corresponds to `Ctrl + Shift + Down Arrow` in Excel.

Parameters **strict** (*bool, default False*) – strict stops the table at empty cells even if they contain a formula. Less efficient than if set to False.

Return type xlwings Range object

Examples

To get the values of a contiguous range or clear its contents use:

```
Range('A1').vertical.value
Range('A1').vertical.clear_contents()
```

horizontal
Returns a contiguous Range starting with the indicated cell and going right as long as no empty cell is hit.

Parameters **strict** (*bool, default False*) – strict stops the table at empty cells even if they contain a formula. Less efficient than if set to False.

Return type xlwings Range object

Examples

To get the values of a contiguous range or clear its contents use:

```
Range('A1').horizontal.value
Range('A1').horizontal.clear_contents()
```

current_region

The current_region property returns a Range object representing a range bounded by (but not including) any combination of blank rows and blank columns or the edges of the worksheet. It corresponds to `Ctrl + *`.

Returns

Return type xlwings Range object

clear()

Clears the content and the formatting of a Range.

clear_contents()

Clears the content of a Range but leaves the formatting.

CHAPTER

SIX

CHART OBJECT

Note: The chart object is currently still lacking a lot of important methods/attributes.

Note: Check out the What's New regarding the latest changes to the Chart object.

class xlwings.**Chart**(**args*, ***kwargs*)

A chart object that represents an existing Excel chart can be created with the following arguments:

```
Chart(1)              Chart('Sheet1', 1)              Chart(1, 1)
Chart('Chart 1')      Chart('Sheet1', 'Chart 1')      Chart(1, 'Chart 1')
```

If no worksheet name is provided as first argument (as name or index), it will take the chart from the active sheet.

To insert a new chart into Excel, create it as follows:

```
Chart().add()
```

Parameters

- ***args** – Definition of sheet (optional) and chart in the above described combinations.
- **chart_type** (*Member of ChartType, default xlColumnClustered*) – Chart type, can also be set using the chart_type property

add(*sheet=None*, *left=168*, *top=217*, *width=355*, *height=211*, ***kwargs*)

Inserts a new chart in Excel.

Parameters

- **sheet** (*string or integer, default None*) – Name or Index of the sheet, defaults to the active sheet
- **left** (*float, default 100*) – left position in points
- **top** (*float, default 75*) – top position in points
- **width** (*float, default 375*) – width in points
- **height** (*float, default 225*) – height in points

- **chart_type** (*xlwings.ChartType member, default xlColumnClustered*) – Excel chart type. E.g. xlwings.ChartType.xlLine
- **name** (*str, default None*) – Excel chart name. Defaults to Excel standard name if not provided, e.g. 'Chart 1'
- **source_data** (*xlwings Range*) – e.g. Range('A1').table

name
Gets and sets the name of a chart

chart_type
Gets and sets the chart type of a chart

activate()
Makes the chart the active chart.

set_source_data(*source*)
Sets the source for the chart

Parameters **source** (*Range*) – xlwings Range object, e.g. `Range('A1')`