pyExcelerator/xlwt Cheatsheet

```
#!/bin/env python
"""Executable cheatsheet illustrating use of pyExcelerator and its
fork, xlwt

I recommend using xlwt which is a somewhat unknown fork of
pyExcelerator.  There are examples shipped with both projects, use
them if necessary, but the source is usually your best friend.  The
libraries are quite capable but don't support charting.  xlwt also has
a mailing list that is active here:
http://groups.google.com.au/group/python-excel

Another good link is here:
http://ntalikeris.blogspot.com/2007/10/create-excel-file-with-python-my-sort.html

This illustrates common usage for .xls generation, but also uses
factories to limit object creation of styles (which can crash Excel)
It's meant to show example, but for more details, I recommend the
sources I mention above.

Please send comments/suggestions my way

author: matthewharrison@gmail.com
"""

#import pyExcelerator as pycel
import xlwt as pycel

# Excel has issues when creating too many styles/fonts, hence use
# a factory to reuse instances (see FAQ#13 http://poi.apache.org/faq.html )
STYLE_FACTORY = {}
FONT_FACTORY = {}

def create_spreadsheet():
    # Create a workbook
    wb = pycel.Workbook()

    # Add a sheet
    ws = wb.add_sheet("Example Sheet")

    # Tweak printer settings
    # following makes a landscape layout on Letter paper
    # the width of the columns
    ws.fit_num_pages = 1
    ws.fit_height_to_pages = 0
    ws.fit_width_to_pages = 1
    # Set to Letter paper
    # See BiffRecords.SetupPageRecord for paper types/orientation
    ws.paper_size_code = 1
    # Set to landscape
    ws.portrait = 0

    # Write some stuff using our helper function

    # Formatting - hint, look at Format code in OOo
    #              format cells... Numbers tab
    # Write a percent
    write(ws, 0, 0, .495, {"format":"0%"})
    # Write a percent with negatives red
    write(ws, 1, 0, -.495, {"format":"0%;[RED]-0%"})
    # Dollar amounts
    write(ws, 2, 0, 10.99, {"format":'$#,##0'})

    # Font
    # Size
    write(ws, 0, 1, "Size 160(8pt)", {"font": (("height", 160),)})
    write(ws, 1, 1, "Size 200(10pt)", {"font": (("height", 200),)})
    # Bold
    write(ws, 2, 1, "Bold text", {"font": (("bold", True),)})

    # Background color
    # See http://groups.google.com.au/group/python-excel/attach/936214066dddf464/palette_trial.xls?part=2
    # for colour indices
    YELLOW = 5
    write(ws, 3, 1, "Yellow (5) Background",
          {"background": (("pattern", pycel.Pattern.SOLID_PATTERN),
                          ("pattern_fore_colour", YELLOW) )})

    # Border
    write(ws, 0, 2, "Border",
          {"border": (("bottom",pycel.Formatting.Borders.THIN),
                      ("bottom_colour", YELLOW))})


    # Wrapping
    write(ws, 0, 3, "A bunch of long text to wrap",
          {"alignment":(("wrap", pycel.Alignment.WRAP_AT_RIGHT),)})

    # Set column width
    # (see pycel.BIFFRecords.ColInfoRecord for details, width in
    # 1/256th of zero character)
    write(ws, 0, 4, "A bunch of longer text not wrapped")
    ws.col(4).width = len("A bunch of longer text not wrapped")*256

    # Freeze/split headers when scrolling
    write(ws, 0, 5, "Header")
    ws.panes_frozen = True
    ws.horz_split_pos = 1
    for row in range(1, 200):
        write(ws, row, 5, row)

    # Save the workbook
    wb.save("out.xls")

def write(ws, row, col, data, style=None):
    """
    Write data to row, col of worksheet (ws) using the style
    information.

    Again, I'm wrapping this because you'll have to do it if you
    create large amounts of formatted entries in your spreadsheet
    (else Excel, but probably not OOo will crash).
    """
    if style:
        s = get_style(style)
        ws.write(row, col, data, s)
    else:
        ws.write(row, col, data)

def get_style(style):
    """
    Style is a dict maping key to values.
    Valid keys are: background, format, alignment, border

    The values for keys are lists of tuples containing (attribute,
    value) pairs to set on model instances...
    """
    print "KEY", style
    style_key = tuple(style.items())
    s = STYLE_FACTORY.get(style_key, None)
    if s is None:
        s = pycel.XFStyle()
        for key, values in style.items():
            if key == "background":
                p = pycel.Pattern()
                for attr, value in values:
                    p.__setattr__(attr, value)
                s.pattern = p
            elif key == "format":
                s.num_format_str = values
            elif key == "alignment":
                a = pycel.Alignment()
                for attr, value in values:
                    a.__setattr__(attr, value)
                s.alignment = a
            elif key == "border":
                b = pycel.Formatting.Borders()
                for attr, value in values:
                    b.__setattr__(attr, value)
                s.borders = b
            elif key == "font":
                f = get_font(values)
                s.font = f
        STYLE_FACTORY[style_key] = s
    return s

def get_font(values):
    """
    'height' 10pt = 200, 8pt = 160
    """
    font_key = values
    f = FONT_FACTORY.get(font_key, None)
    if f is None:
        f = pycel.Font()
        for attr, value in values:
            f.__setattr__(attr, value)
        FONT_FACTORY[font_key] = f
    return f

if __name__ == "__main__":
    create_spreadsheet()
```

matthewharrison@gmail.com