Reportlab Tables – Creating Tables in PDFs with Python

September 21, 2010 Cross-Platform, Python Python, Python PDF Series, Reportlab
Mike

Back in March of this year, I wrote a simple tutorial on Reportlab, a handy 3rd party Python package that allows the developer to create PDFs programmatically. Recently, I received a request to cover how to do tables in Reportlab. Since my Reportlab article is so popular, I figured it was probably worth the trouble to figure out tables. In this article, I will attempt to show you the basics of inserting tables into Reportlab generated PDFs.

One of the few issues I have with Reportlab is their user guide. It shows some good examples, but they're almost always incomplete. Go download the user guide here and start reading chapter seven (the chapter on tables) and you'll quickly see the following code snippet:

```
LIST_STYLE = TableStyle(
    [('LINEABOVE', (0,0), (-1,0), 2, colors.green),
    ('LINEABOVE', (0,1), (-1,-1), 0.25, colors.black),
    ('LINEBELOW', (0,-1), (-1,-1), 2, colors.green),
    ('ALIGN', (1,1), (-1,-1), 'RIGHT')]
)
```

Of course, this snippet is completely un-runnable and the text around it is pretty useless for figuring out how to import **TableStyle** and **colors**. It sounds like you can get all the Table stuff from the Flowable class (pg 76 – first page of chapter 7). Unfortunately, there is no "reportlab.Flowable". You have to know that all the flowables come from something called **platypus**. Let's stop here and talk about what a "flowable" is. Here is what the user guide says:

Flowables are things which can be drawn and which have wrap, draw and perhaps split methods. Flowable is an abstract base class for things to be drawn and an instance knows its size and draws in its own coordinate system (this requires the base API to provide an absolute coordinate system when the Flowable.draw method is called). To get an instance use f=Flowable(). (pg 62, section 5.3). Note that nowhere on that page does it show how to get the Flowable class. I only found out how by doing a text search of their source. Here's the syntax: *from reportlab.platypus import Flowable*

It's this sort of thing that made me almost give up on using Reportlab at all several years ago. Fortunately, my boss made me figure it out and had some real examples for me to follow. Alright, enough ranting. On with the tables tutorial!

Getting Started with Reportlab Tables

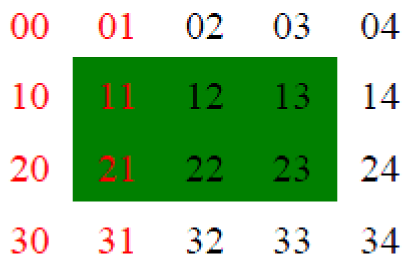
We’re actually going to take some of the table examples from the Reportlab User Guide and make them runnable. I’ll try to explain what’s going on in the code, but you may want to refer to the guide as well. Let’s start with the example on pg 78 entitled *TableStyle Cell Formatting Commands*. This first real example shows how to create a table with multiple colors, but no visible grid. Let’s make it runnable!

```python
from reportlab.lib import colors
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle

doc = SimpleDocTemplate("simple_table.pdf", pagesize=letter)
# container for the 'Flowable' objects
elements = []

data= [['00', '01', '02', '03', '04'],
       ['10', '11', '12', '13', '14'],
       ['20', '21', '22', '23', '24'],
       ['30', '31', '32', '33', '34']]
t=Table(data)
t.setStyle(TableStyle([('BACKGROUND',(1,1),(-2,-2),colors.green),
                       ('TEXTCOLOR',(0,0),(1,-1),colors.red)]))
elements.append(t)
# write the document to disk
doc.build(elements)
```

If you run this code, you should see something like this centered along the top of your PDF:



00	01	02	03	04
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34

If you look at the code, you’ll notice that we have a series of imports. We import **colors** from “reportlab.lib”, **letter** from “reportlab.lib.pagesizes” and **SimpleDocTemplate**, **Table**, and **TableStyle** from “reportlab.platypus”. The imports are pretty self-explanatory, but they also help us familiarize ourselves with the way that Reportlab’s code is laid out. If we need other Flowables, like Paragraph, we can probably assume that it would come from “reportlab.platypus” since Table is a Flowable. If you try it, you’ll see that this assumption is correct.

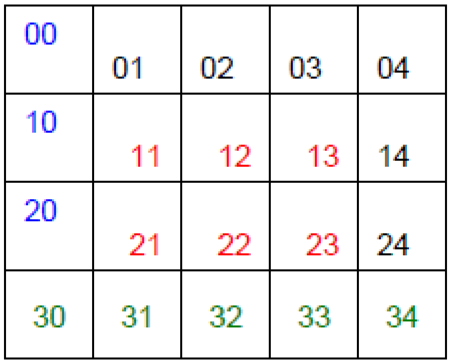
Next we create a document template using the **SimpleDocTemplate** class. The first argument is the path to the PDF that we want to create and the second argument is the page size. In this example, we just put in the name of the document. This will cause the script to put the PDF in the same folder that it’s run from. For reasons that I’ve never seen explained, you use a list to hold the flowables. In this code, we call our list “elements”. In the user guide, they call it “story”.

The **data** list holds the data that will go in our table. The list is a list of lists. The table will 5 columns wide (the length of the nested lists) and 4 rows tall (the number of nested lists). We pass this list into our **Table** class to create the Table in memory and then call our Table instance's setStyle method to change the style. To do this, we pass it a **TableStyle** class, which contains the styles we want to apply. In this case, we want to apply a green background color from the cells in column 2, row 2 to column 3, row 2. Note that the columns and rows are zero-based so 0 = 1, 1 = 2, etc. Also notice that the example uses (-2, -2) rather than the much easier to understand (3, 2). This illustrates that you can also specify settings from the lower right corner instead of just doing everything from the top left. Unfortunately, the coordinate system starting on the lower right starts at (-1,-1) and that makes it a little harder to get one's mind around.

Then we set the text color to red in the first two columns. I admit that I'm not quite sure how this works as the ending coordinates just don't seem to fit with the ones for the background color. I'll leave that to my readers to explain.

Finally, we append the table to our elements list and then call our document instance's **build** method with the elements list as its only argument. This causes the document to be build and the PDF created. Now you have a PDF table!

Adding a Grid to the Table



00	01	02	03	04
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34

Let's take the next example from the Reportlab documentation and see if we can get it to run too. This next example shows how to add a visible grid to the table as well as how to position the text within the cells.

```
from reportlab.lib import colors
from reportlab.lib.pagesizes import letter, inch
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle

doc = SimpleDocTemplate("simple_table_grid.pdf", pagesize=letter)
# container for the 'Flowable' objects
elements = []

data= [['00', '01', '02', '03', '04'],
       ['10', '11', '12', '13', '14'],
        ['20', '21', '22', '23', '24'],
        ['30', '31', '32', '33', '34']]
t=Table(data,5*[0.4*inch], 4*[0.4*inch])
t.setStyle(TableStyle([('ALIGN',(1,1),(-2,-2),'RIGHT'),
                       ('TEXTCOLOR',(1,1),(-2,-2),colors.red),
                       ('VALIGN',(0,0),(0,-1),'TOP'),
                       ('TEXTCOLOR',(0,0),(0,-1),colors.blue),
                       ('ALIGN',(0,-1),(-1,-1),'CENTER'),
                       ('VALIGN',(0,-1),(-1,-1),'MIDDLE'),
                       ('TEXTCOLOR',(0,-1),(-1,-1),colors.green),
                       ('INNERGRID', (0,0), (-1,-1), 0.25,
colors.black),
                       ('BOX', (0,0), (-1,-1), 0.25, colors.black),
                       ]))

elements.append(t)
# write the document to disk
doc.build(elements)
```

We have one new import here from the **pagesizes** library: **inch**. The "inch" just gives us a simple way to set sizes or margins in our PDF. In this case, we use it to set the table's column widths and row heights. The next change is in *setStyle* section of the code and this is really the key to getting tables to look the way you want. The *TableStyle* class controls the look of the table completely. For example, the first two lines align the middle six cells to the right and color them red. The next two lines set the four cells in the first column to the color blue and to the text to the top of the cell. You can figure out the three lines on your own.

The last two lines of code draw the inner grid and the box around the grid. Then we append the table to our elements list and finally, build the document.

Creating Complex Cell Values

The last major example we're going to look at is the Complex Cell Values example from the Reportlab user guide. This one shows you to insert other Reportlab Flowables in the cells themselves. Let's take a quick look!

```
from reportlab.lib import colors
from reportlab.lib.pagesizes import letter, inch
from reportlab.platypus import Image, Paragraph, SimpleDocTemplate,
Table
from reportlab.lib.styles import getSampleStyleSheet

doc = SimpleDocTemplate("complex_cell_values.pdf", pagesize=letter)
# container for the 'Flowable' objects
elements = []
```

```
styleSheet = getSampleStyleSheet()

I = Image('replogo.gif')
I.drawHeight = 1.25*inch*I.drawHeight / I.drawWidth
I.drawWidth = 1.25*inch
P0 = Paragraph('''
               <b>A pa<font color=red>r</font>a<i>graph</i></b>
               <super><font color=yellow>1</font></super>''',
               styleSheet["BodyText"])
P = Paragraph('''
    <para align=center spaceb=3>The <b>ReportLab Left
    <font color=red>Logo</font></b>
    Image</para>''',
    styleSheet["BodyText"])
data= [['A', 'B', 'C', P0, 'D'],
       ['00', '01', '02', [I,P], '04'],
       ['10', '11', '12', [P,I], '14'],
       ['20', '21', '22', '23', '24'],
       ['30', '31', '32', '33', '34']]

t=Table(data,style=[('GRID',(1,1),(-2,-2),1,colors.green),
                    ('BOX',(0,0),(1,-1),2,colors.red),
                    ('LINEABOVE',(1,2),(-2,2),1,colors.blue),
                    ('LINEBEFORE',(2,1),(2,-2),1,colors.pink),
                    ('BACKGROUND', (0, 0), (0, 1), colors.pink),
                    ('BACKGROUND', (1, 1), (1, 2), colors.lavender),
                    ('BACKGROUND', (2, 2), (2, 3), colors.orange),
                    ('BOX',(0,0),(-1,-1),2,colors.black),
                    ('GRID',(0,0),(-1,-1),0.5,colors.black),
                    ('VALIGN',(3,0),(3,0),'BOTTOM'),
                    ('BACKGROUND',(3,0),(3,0),colors.limegreen),
                    ('BACKGROUND',(3,1),(3,1),colors.khaki),
                    ('ALIGN',(3,1),(3,1),'CENTER'),
                    ('BACKGROUND',(3,2),(3,2),colors.beige),
                    ('ALIGN',(3,2),(3,2),'LEFT'),
])
t._argW[3]=1.5*inch

elements.append(t)
# write the document to disk
doc.build(elements)
```

This snippet has a couple new imports. First off, we need access to **Image** and **Paragraph**, which are part of the **platypus** lib. We also need to be able to use **getSampleStyleSheet**, so we import that from the **styles** lib. Paragraphs are flowables that allow us to use HTML-like syntax to style our text and insert images, although we don't do the image insertion in this manner in this particular example.

Anyway, once again, the style is where the action is. In this example, we take a shortcut and pass the style directly into the Table object using its **style** parameter instead of using the Table's "setStyle" method in combination with the TableStyle class. The result would have been the same if we had done it that way though. You should be able to parse the style yourself at this point. The only real difference was where we added the Paragraph and Image instances to the data list.

Other Odds and Ends

There are just a few other things to mention about Tables that didn't really fit with the previous examples. Reportlab provides the ability to split tables when there are too many rows to fit on the page. This is enabled by default. According to the documentation, there is no way to split by column currently, so don't bother trying to insert an extra-wide column.

The Table also has **repeatRows** and **repeatCols** parameters. The repeatRows parameter controls how many rows to repeat on a split. The repeatCols parameter is not currently implemented. To be honest, I'm not even sure why this parameter is even included. It's been there for quite a while and it still does nothing.

Wrapping Up

Now you should have a good foundation for PDF table creation using the Reportlab toolkit. This article just scratches the surface of what Reportlab can do, so be sure to read the documentation, the tests and the various other examples on their website to truly understand the power within this handy toolkit.

Further Reading

- Reportlab Open Source
- A Reportlab Tutorial

Downloads

- reportlab_tables.zip
- reportlab_tables.tar