RML Example 17: Graphics

This should be overwritten by included pdf

RML (Report Markup Language) is ReportLab's own language for specifying the appearance of a printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of ouput and are distributed within our commercial package as test cases. Each should be self explanatory and stand alone.

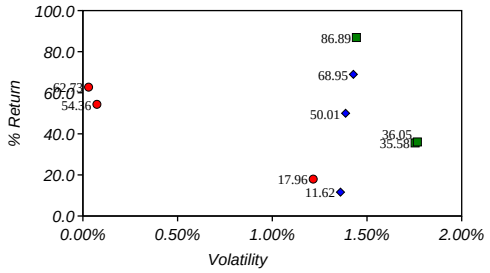
Diagra Integration

Diagra is ReportLab's charting and data graphics product. It allows charts and data graphics to be prepared visually in a Drawing Editor, and used in a variety of contexts including within RML, as bitmaps on the web, and for generating batches of EPS files.

Referring to a Drawing Module

The first stage is to use the drawing editor to create a drawing module. Take note of the class name as you generate it. You can then refer to it directly with a drawing tag. The drawing tag takes at least two parameters. The `module`attribute holds the name of the module the drawing is defined in. This is a normal python module reference as used with the import statement, and may contain dots to refer to items anywhere on the path e.g. `reportlab.graphics.charts.barcharts`. The directory where the RML document lives will always be on the path, so if your graphic is in the same directory, you can just use the filename (minus extension). The second attribute, `function`, is the name of the constructor to call. If you used the Drawing Editor, insert the class name of the drawing here. You could also call an arbitrary Python function which returns a Python object; often it is convenient to write helper functions to set up your drawings outside of RML. Finally, there is a third optional argument baseDir, which contains a directory name to look underThis must be escaped with double slashes on Windows e.g. C:\\mycharts. We included a chart module scatterplot.py in this test directory containing a class ScatterPlotDrawing, so for our example we will just refer so let's refer to it now:

```
<drawing module="test_014_scatterplot" function="ScatterPlotDrawing" />
```



It's important to recognize that the Diagra framework is completely general and not necessarily for charts. We use it to crank out ReportLab logos with variable sizes and colors! So, let's show one more example, which is about the simplest data graphic we have: a 'slidebox' which accepts one numeric parameter:

```
<drawing module="test_014_slidebox" function="SlideBoxDrawing" />
```

Source: ReportLab

Making it dynamic

Static charts are not much use to anyone. In most cases, you will want to pass in the numeric data at runtime, and perhaps change the title. The Diagra framework is completely general - not a charting framework per se - and lets you set any attribute of any object within the chart; so you could set the height of a bar or something similar. Let's start with an ultra-simple example. The above SlideBox takes a single numeric parameter. In general you should use the drawing editor to explore the available parameters.

x=72 y=88: raw-post Canvas ShowPdfFlowable ('test_014_graphics.pdf', 1) A

x=72 y=72: raw-pre Canvas ShowPdfFlowable ('test_014_graphics.pdf', 1) A

```
<drawing module="test_014_slidebox" function="SlideBoxDrawing" >
```

```
    <param name="SlideBox.trianglePosition">4</param>
</drawing>
```

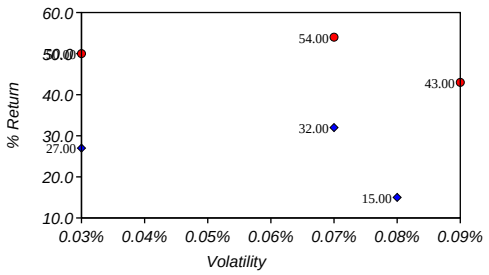
Source: ReportLab

x=72 y=120:raw-post Canvas ShowPdfFlowable ('test_014_graphics.pdf', 2) A
x=72 y=104:raw-pre Canvas ShowPdfFlowable ('test_014_graphics.pdf', 2) A

RML Example 17: Graphics

The most common use of the param tag will be to set the dynamic data for a chart. For example, let's take the preceding one and pass in some data. The Drawing Editor will reveal that the data parameter for scatter plot is a list of sequences of x-y pairs, so we make up our data nested this way (you can use square or round brackets, it doesn't matter):

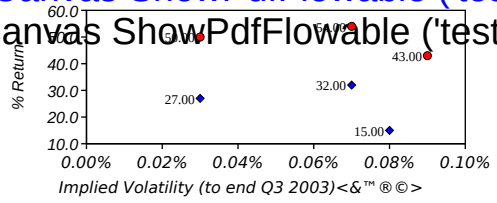
```
<drawing module="test_014_scatterplot" function="ScatterPlotDrawing">
  <param name="ScatterPlot.data">[((0.03, 50), (0.04, 54), (0.05, 43)),
                                  ((0.03, 27), (0.04, 32), (0.055, 15))]</param>
  </drawing>
```



The content of each param tag is evaluated literally. This is more compact and easier to generate and parse than generating thousands of series and data-point tags.
You can also modify other parameters which are nothing to do with numeric data, just as you do in the Drawing Editor. Let's force the x axis to include the zero, so the leftmost points do not dangle in the margin, and change the title below the y axis:
Parameters passed through to charts may now contain the standard XML escapes for '&','<' and '>'. However, unicode font handling for charts and graphics is not yet complete; non-ASCII characters such as copyright and trademark which are passed in through RML may be displayed as multiple bytes of garbage when displayed in a Type 1 Font. We believe that the graphics are the last remaining area of our framework that needs unicode-enabling and hope to complete this in a release next week. The PDF rendering for the standard numeric escapes eg ࡊ is carried out, but will only work in param tags if the relevant objects font understands them. This should be the case for TTF fonts if the document encoding is "utf-8". See the example below where you should see <&™ ® ©> in the x axis label.

```
<drawing module="test_014_scatterplot" function="ScatterPlotDrawing">
  <param name="ScatterPlot.height">50</param>
  <param name="ScatterPlot.xValueAxis.labels.fontName">VeraItalic</param>
  <param name="ScatterPlot.xLabel">Implied Volatility (to end Q3 2003)&lt;&amp;&#x2122;&#174;&#169;&gt;
  <param name="ScatterPlot.xValueAxis.forceZero">1</param>
  <param name="ScatterPlot.data">
    [((0.03, 50), (0.07, 54), (0.09, 43)), ((0.03, 27), (0.07, 32), (0.08, 15))]
  </param>
</drawing>
```

x=72 y=152:raw-post Canvas ShowPdfFlowable ('test_014_graphics.pdf', 3) A
x=72 y=136:raw-pre Canvas ShowPdfFlowable ('test_014_graphics.pdf', 3) A



RML Example 6: Paragraphs

RML (Report Markup Language) is ReportLab's own language for specifying the appearance
printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of oupu
are distributed within our commercial package as test cases. Each should be self explanator
stand alone.

Paragraph 1: About this page

This page tests out a number of attributes of the **paraStyle** tag. This paragraph is
style we have called "style1". It should be a normal paragraph, set in Courier 12
should be a normal paragraph, set in Courier (not bold). It should be a normal par
graph, set in Courier 12 pt. This should be red.

Paragraph 2: Indent Left

This paragraph is in a style we have called "style2". It should be ind
on the left. It should be indented on the left by 1 inch. It should be
dented on the left. This should be ~~struck out~~.

Paragraph 3: Indent Right

This paragraph is in a style we have called "style3". It should be indented
on the right. It should be indented on the right by 1 inch. It should be in-
dented on the right.

Paragraph 4: Space Before

This paragraph is in a style we have called "style4". It should be have a space be
it. It should be have a space before it of 2 centimeters. It should be have a spac
fore it.

Paragraph 5: Space After

This paragraph is in a style we have called "style5". It should be have a space af
it. It should be have a space after it of 2 centimeters. It should be have a space
it.

x=72 y=120: raw-post Canvas.ShowPdfFlowable ('test_002_paras.pdf', 1) B

Paragraph 6: First Line Indent

This paragraph is in a style we have called "style6".It should be have a
dented first line. It should be have an first line indented by 2 centimeters. It s
be have an indented first line.

x=72 y=72: raw-pre Canvas ShowPdfFlowable ('test_002_paras.pdf', 1) B

Paragraph 7: Leading

This paragraph is in a style we have called "style7". It should be using leading.
should have a gap of 5 points between each line. It should be using leading. It sh
have a gap of 5 pt between each line. It should be using leading. The gap between
should be half of the height of a line. This paragraph should look like it has a l

x=72 y=104: transformed-post Canvas ShowPdfFlowable ('test_002_paras

x=72 y=88: transformed-pre Canvas ShowPdfFlowable ('test_002_paras

RML Example 5: Paragraphs

Paragraphs 8-12: Simple Bullet Points

- **Parastyle name="style8" parent="style1" bulletFontName = "ZapfDingbats" bulletFo**
= "5"
- These paragraphs are in a style we have called "style8"
- These five lines should have bullet points.
- The bullet font is ZapfDingbats.
- The bullet size is 5 points
- This is a long line to see how multi-line bullets look: These paragraphs are in a
we have called "style8". These four lines should have bullet points. The bullet fo
ZapfDingbats. The bullet size is 5 points

Paragraphs 13-18: Indented Bullet Points

- **bulletFontName = "ZapfDingbats" bulletFontSize = "10" bulletIndent = "20"**
- These paragraphs are in a style we have called "style9"
- These five lines should have *indented* bullet points.
- Bullet points should look like a pointing hand.
- Bullet font is still ZapfDingbats, and bullet size is 10 points.
- The bullet indent is 20 points
- This is a long line to see how multi-line bullets look: These paragraphs are
style we have called "style9". These four lines should have *indented* bullet points
let points should look like a pointing hand. Bullet font is still ZapfDingbats, an
let size is 10 points. The bullet indent is 20 points

Paragraph 19-24: Indented Bullet Points with a Left Indent for the T

- **bulletFontName = "ZapfDingbats" bulletFontSize = "10" bulletIndent = "20"**
leftIndent = "35"
- These paragraphs are in a style we have called "style10"
- These four lines should have *indented* bullet points, with the text indented
well.
- Bullet points should look like a pointing hand.
- Bullet font is still ZapfDingbats, and bullet size is 10 points.
- The bullet indent is 20 points, and the text indent is 35 points
- This is a long line to see how multi-line bullets look: These paragraphs are
style we have called "style10". These four lines should have *indented* bullet
points, with the text indented as well. Bullet points should look like a poi
hand. Bullet font is still ZapfDingbats, and bullet size is 10 points.

Paragraph 25: Left Justified Paragraphs

x=72 y=184:raw-post Canvas ShowPdfFlowable ('test_002_paras.pdf', 2) B

This paragraph is in a style we have called "style11". It should be left justified
has an argument which states 'alignment = "left"'. It should be left justified. It
should be aligned to the left.

x=72 y=136:raw-pre Canvas ShowPdfFlowable ('test_002_paras.pdf', 2) B

Paragraph 26: Right Justified Paragraphs

This paragraph is in a style we have called "style12". It should be right justifi
has an argument which states 'alignment = "right"'. It should be right justifi
should be aligned to the

x=72 y=168:transformed-post Canvas ShowPdfFlowable ('test_002_paras

x=72 y=152:transformed-pre Canvas ShowPdfFlowable ('test_002_paras

Paragraph 27: Centered Paragraphs

This paragraph is in a style we have called "style13".It should be center justifi
has an argument which states 'alignment = "center"'. It should be centered. It sho
aligned to the center.

Paragraph 28: Justified Paragraphs

RML Example 5: Paragraphs

doesn't contain any bold text though.

Paragraph 28.1: Justified Paragraphs With Bold Text

This paragraph is in a style we have called "style14". It should be **justified**. It argument which states **'alignment = "justify"'**. It should be justified. This par doesn't contain any bold text though.

Paragraphs 29-32: Bullets using left align, right align, centered an justify.

- **bulletFontName = "ZapfDingbats" bulletFontSize = "5" bulletIndent = "20" leftIndent = "35" alignment = "left"**
- This is "style15", bullets with a left alignment. (The bullets in this style based on "style10")
- **bulletFontName = "ZapfDingbats" bulletFontSize = "5" bulletIndent leftIndent = "35" alignment = "**
- This is "style16", bullets with a right alignment.(The bullets in this sty based on "sty
- **bulletFontName = "ZapfDingbats" bulletFontSize = "5" bulletIndent = "20 leftIndent = "35" alignment = "center"**
- This is "style17", bullets with a center alignment.(The bullets in this sty based on "style10")
- **bulletFontName = "ZapfDingbats" bulletFontSize = "5" bulletIndent = leftIndent = "35" alignment = "justify"**
- This is "style18", bullets with a justified paragraph.(The bullets in this are based on "style10")

These all look wierd, but most people do not actually use these styles because the so wrong.

Paragraph 33-35: Using Colours by Colour Name

This text should be **RED**

x=72 y=248:raw-post Canvas ShowPdfFlowable ('test_002_paras.pdf', 3) B

x=72 y=200:raw-pre Canvas ShowPdfFlowable ('test_002_paras.pdf', 3) B

This text should be **GREEN**

x=72 y=232:transformed-post Canvas ShowPdfFlowable ('test_002_paras

x=72 y=216:transformed-pre Canvas ShowPdfFlowable ('test_002_paras

This text should be **BLUE**

RML Example 17: Graphics

RML (Report Markup Language) is ReportLab's own language for specifying the appearance of a printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of ouput and are distributed within our commercial package as test cases. Each should be self explanatory and stand alone.

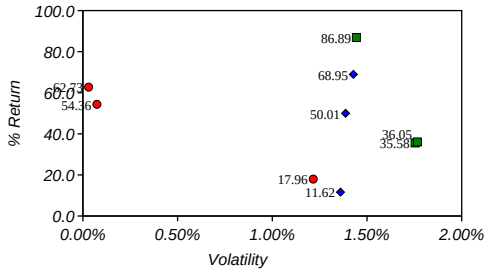
Diagra Integration

Diagra is ReportLab's charting and data graphics product. It allows charts and data graphics to be prepared visually in a Drawing Editor, and used in a variety of contexts including within RML, as bitmaps on the web, and for generating batches of EPS files.

Referring to a Drawing Module

The first stage is to use the drawing editor to create a drawing module. Take note of the class name as you generate it. You can then refer to it directly with a drawing tag. The drawing tag takes at least two parameters. The `module`attribute holds the name of the module the drawing is defined in. This is a normal python module reference as used with the import statement, and may contain dots to refer to items anywhere on the path e.g. `reportlab.graphics.charts.barcharts`. The directory where the RML document lives will always be on the path, so if your graphic is in the same directory, you can just use the filename (minus extension). The second attribute, `function`, is the name of the constructor to call. If you used the Drawing Editor, insert the class name of the drawing here. You could also call an arbitrary Python function which returns a Python object; often it is convenient to write helper functions to set up your drawings outside of RML. Finally, there is a third optional argument baseDir, which contains a directory name to look underThis must be escaped with double slashes on Windows e.g. C:\\mycharts. We included a chart module scatterplot.py in this test directory containing a class ScatterPlotDrawing, so for our example we will just refer so let's refer to it now:

```
<drawing module="test_014_scatterplot" function="ScatterPlotDrawing" />
```



It's important to recognize that the Diagra framework is completely general and not necessarily for charts. We use it to crank out ReportLab logos with variable sizes and colors! So, let's show one more example, which is about the simplest data graphic we have: a 'slidebox' which accepts one numeric parameter:

```
<drawing module="test_014_slidebox" function="SlideBoxDrawing" />
```

Source: ReportLab

Making it dynamic

Static charts are not much use to anyone. In most cases, you will want to pass in the numeric data at runtime, and perhaps change the title. The Diagra framework is completely general - not a charting framework per se - and lets you set any attribute of any object within the chart; so you could set the height of a bar or something similar. Let's start with an ultra-simple example. The above SlideBox takes a single numeric parameter. In general you should use the drawing editor to explore the available parameters.

x=72 y=88: raw-post Canvas ShowPdfFlowable ('test_014_graphics.pdf', 1) C
x=72 y=72: raw-pre Canvas ShowPdfFlowable ('test_014_graphics.pdf', 1) C

```
<drawing module="test_014_slidebox" function="SlideBoxDrawing" >
```

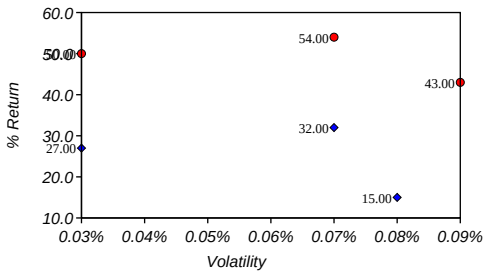
```
    <param name="SlideBox.trianglePosition">4</param>
</drawing>
```

Source: ReportLab

x=72 y=120:raw-post Canvas ShowPdfFlowable ('test_014_graphics.pdf', 2) C
x=72 y=104:raw-pre Canvas ShowPdfFlowable ('test_014_graphics.pdf', 2) C

The most common use of the param tag will be to set the dynamic data for a chart. For example, let's take the preceding one and pass in some data. The Drawing Editor will reveal that the data parameter for scatter plot is a list of sequences of x-y pairs, so we make up our data nested this way (you can use square or round brackets, it doesn't matter):

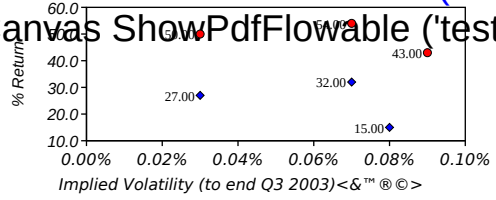
```
<drawing module="test_014_scatterplot" function="ScatterPlotDrawing">
  <param name="ScatterPlot.data">[((0.03, 50), (0.04, 54), (0.05, 43)),
                                  ((0.03, 27), (0.04, 32), (0.055, 15))]</param>
  </drawing>
```



The content of each param tag is evaluated literally. This is more compact and easier to generate and parse than generating thousands of series and data-point tags.
You can also modify other parameters which are nothing to do with numeric data, just as you do in the Drawing Editor. Let's force the x axis to include the zero, so the leftmost points do not dangle in the margin, and change the title below the y axis:
Parameters passed through to charts may now contain the standard XML escapes for '&','<' and '>'. However, unicode font handling for charts and graphics is not yet complete; non-ASCII characters such as copyright and trademark which are passed in through RML may be displayed as multiple bytes of garbage when displayed in a Type 1 Font. We believe that the graphics are the last remaining area of our framework that needs unicode-enabling and hope to complete this in a release next week. The PDF rendering for the standard numeric escapes eg ࡊ is carried out, but will only work in param tags if the relevant objects font understands them. This should be the case for TTF fonts if the document encoding is "utf-8". See the example below where you should see <&™ ® ©> in the x axis label.

```
<drawing module="test_014_scatterplot" function="ScatterPlotDrawing">
  <param name="ScatterPlot.height">50</param>
  <param name="ScatterPlot.xValueAxis.labels.fontName">VeraItalic</param>
  <param name="ScatterPlot.xLabel">Implied Volatility (to end Q3 2003)&lt;&amp;&#x2122;&#174;&#169;&gt;
  <param name="ScatterPlot.xValueAxis.forceZero">1</param>
  <param name="ScatterPlot.data">
    [((0.03, 50), (0.07, 54), (0.09, 43)), ((0.03, 27), (0.07, 32), (0.08, 15))]
  </param>
</drawing>
```

x=72 y=152:raw-post Canvas ShowPdfFlowable ('test_014_graphics.pdf', 3) C
x=72 y=136:raw-pre Canvas ShowPdfFlowable ('test_014_graphics.pdf', 3) C



Paragraphs with anchored bullets: green line is the global indent, blue the bullet indent

1.1 bullet anchor absent
1.22 bullet anchor absent
1.3 bullet anchor absent
1.1 bullet anchor start
1.22 bullet anchor start
1.3 bullet anchor start
1.1 bullet anchor middle
1.22 bullet anchor middle
1.3 bullet anchor middle
1.1 bullet anchor end
1.22 bullet anchor end
1.3 bullet anchor end
1.1 bullet anchor numeric
1.22 bullet anchor numeric
1.3 bullet anchor numeric

Here is another example, demonstrating bulletAnchor, note the alignment of the text when we get to double figures:

Numbers not aligned	Numbers aligned
9.0 bulletAnchor absent	9.0 bulletAnchor="numeric"
10.0 bulletAnchor absent	10.0 bulletAnchor="numeric"

x=72 y=88: raw-post Canvas ShowPdfFlowable ('test_002_paras.pdf', 4) D
x=72 y=72: raw-pre Canvas ShowPdfFlowable ('test_002_paras.pdf', 4) D

RML Example 5: Paragraphs

You SHOULD be able to specify colours by all the means available to reportlab.lib.colours. Currently, you cannot use RGB or HEX values...

Last Paragraph: Para Tags and Paragraph Content

This should *not* have any extra spaces at the start of **this** line (though there should be at the start of the heading). RML should ignore additional whitespace, and you should be able to format the actual paragraphs as you like. This should be underlined. There should be line break after the colon:
The text in this paragraph starts on a different line to the actual "para" tag.

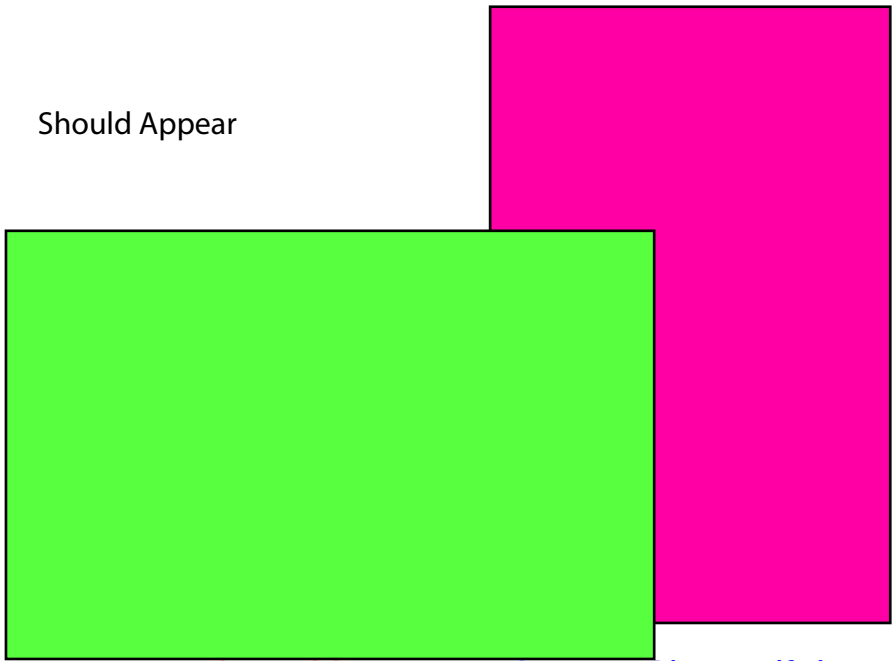
Quoting and escaping

This checks for all the possible quotes: & = &, < = <, > = >, ' = ', " = ", £ = £.
If this is not italic, **and this is not bold**, even normal angle brackets are broken.

x=72 y=120:raw-post Canvas ShowPdfFlowable ('test_002_paras.pdf', 5) D
x=72 y=104:raw-pre Canvas ShowPdfFlowable ('test_002_paras.pdf', 5) D

x=72 y=88: raw-post Canvas ShowPdfFlowable ('cropped-media.pdf', 1) no box uncr
x=72 y=72: raw-pre Canvas ShowPdfFlowable ('cropped-media.pdf', 1) no box uncro

Should Appear

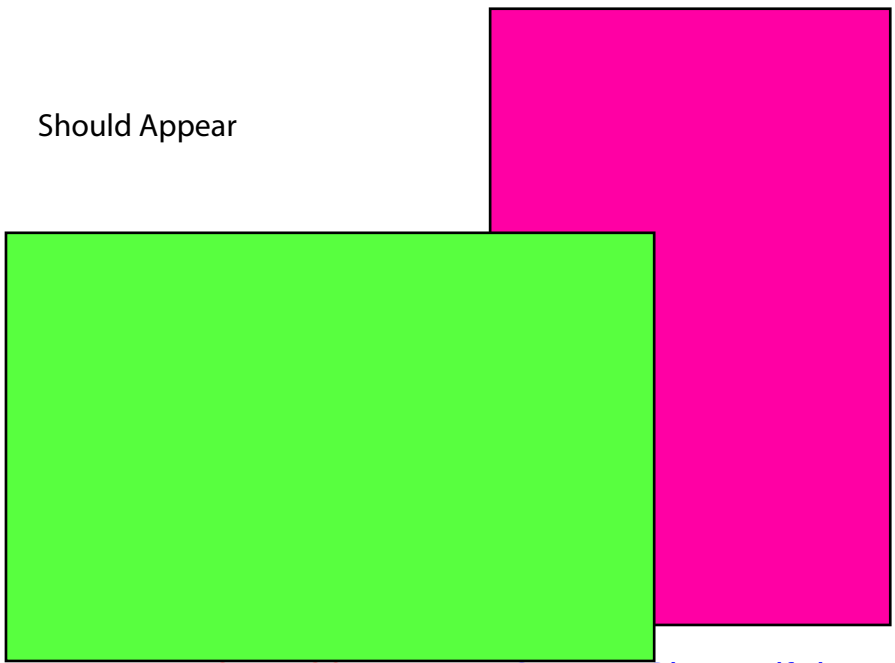


This should noyt appe

x=72 y=120:raw-post Canvas ShowPdfFlowable ('cropped-media.pdf', 1) CropBox

x=72 y=72: raw-pre Canvas ShowPdfFlowable ('cropped-media.pdf', 1) CropBox

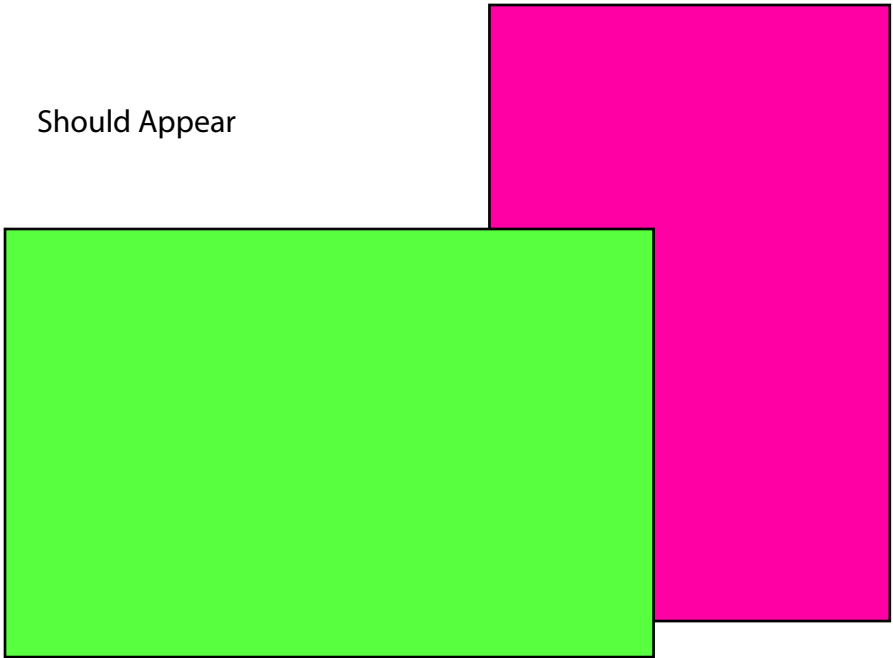
Should Appear



x=72 y=120:raw-post Canvas ShowPdfFlowable ('cropped-media.pdf', 1) CropBox cr

x=72 y=72: raw-pre Canvas ShowPdfFlowable ('cropped-media.pdf', 1) CropBox cro

Should Appear



x=72 y=120:raw-post Canvas ShowPdfFlowable

x=72 y=72: raw-pre Canvas ShowPdfFlowable ("

RML Example 6: Paragraphs

RML (Report Markup Language) is ReportLab's own language for specifying the appearance of a printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of ouput and are distributed within our commercial package as test cases. Each should be self explanatory and stand alone.

Paragraph 1: About this page

This page tests out a number of attributes of the **paraStyle** tag. This paragraph is in a style we have called "style1". It should be a normal paragraph, set in Courier 12 pt. It should be a normal paragraph, set in Courier (not bold). It should be a normal paragraph, set in Courier 12 pt. This should be red.

Paragraph 2: Indent Left

This paragraph is in a style we have called "style2". It should be indented on the left. It should be indented on the left by 1 inch. It should be indented on the left. This should be ~~struck out~~.

Paragraph 3: Indent Right

This paragraph is in a style we have called "style3". It should be indented on the right. It should be indented on the right by 1 inch. It should be indented on the right.

Paragraph 4: Space Before

This paragraph is in a style we have called "style4". It should be have a space before it. It should be have a space before it of 2 centimeters. It should be have a space before it.

Paragraph 5: Space After

This paragraph is in a style we have called "style5". It should be have a space after it. It should be have a space after it of 2 centimeters. It should be have a space after it.

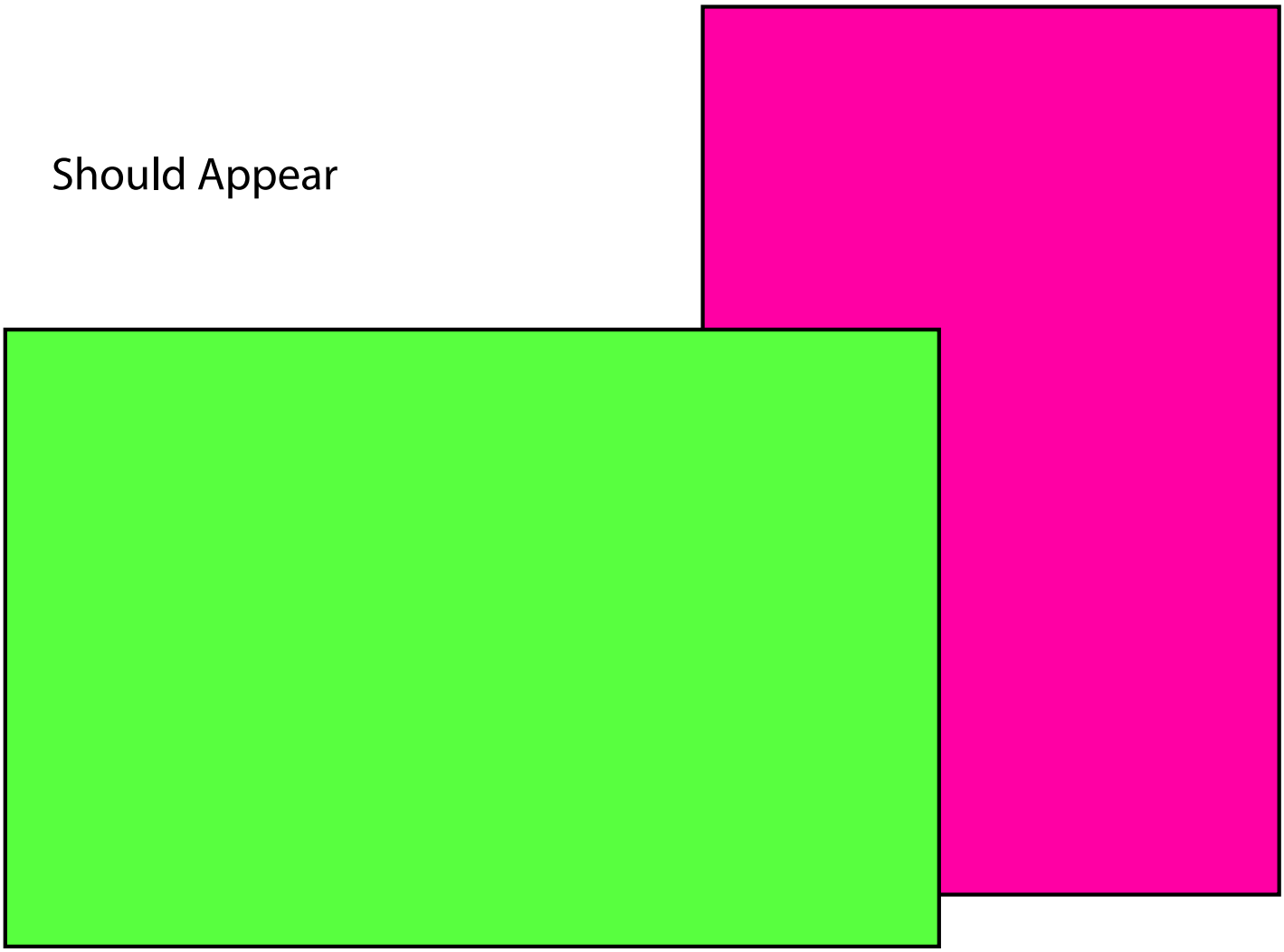
Paragraph 6: First Line Indent

This paragraph is in a style we have called "style6".It should be have an indented first line. It should be have an first line indented by 2 centimeters. It should be have an indented first line.

Paragraph 7: Leading

This paragraph is in a style we have called "style7". It should be using leading. It should have a gap of 5 points between each line. It should be using leading. It should have a gap of 5 pt between each line. It should be using leading. The gap between lines should be half of the height of a line. This paragraph should look like it has a line spacing of "1.5 lines"

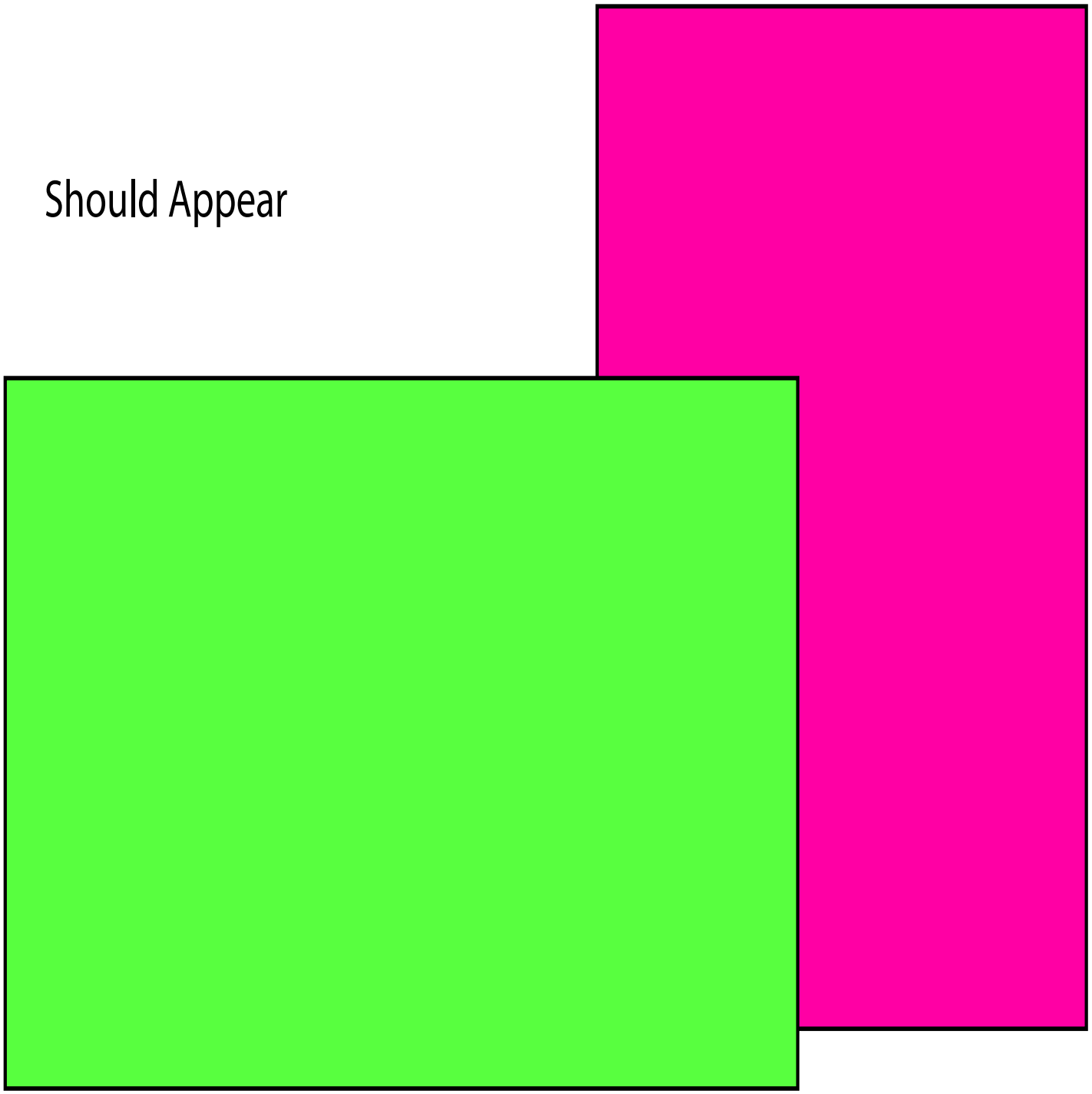
Should Appear



x=72 y=120:raw-post Canvas ShowPdfFlowable ('cropped-media.pdf', 1) cropped & c

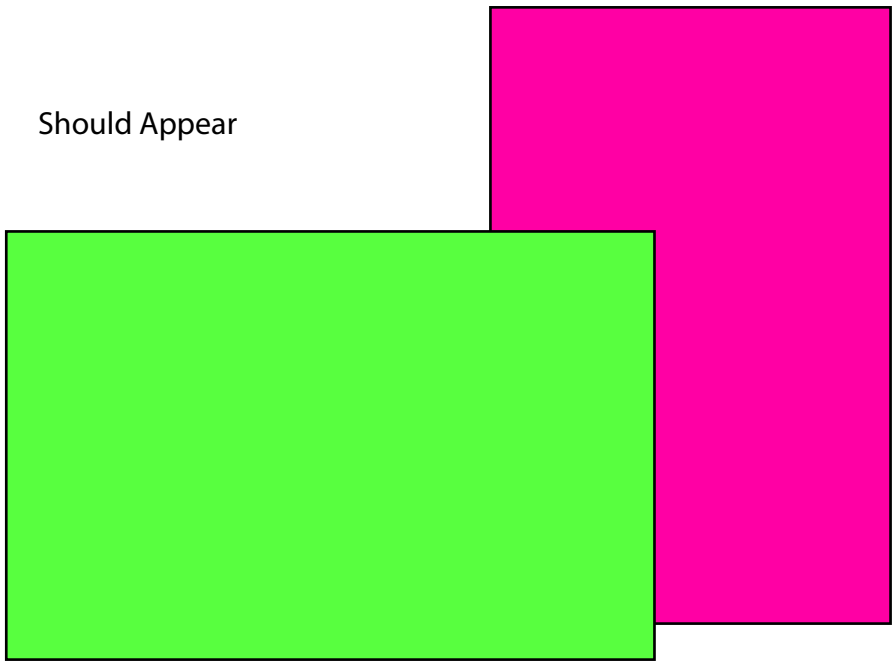
x=72 y=72: raw-pre Canvas ShowPdfFlowable ('cropped-media.pdf', 1) cropped & or

Should Appear



x=72 y=120:raw-post Canvas ShowPdfFlowable ('cropped-media.pdf', 1) cropped & f

x=72 y=72: raw-pre Canvas ShowPdfFlowable ('cropped-media.pdf', 1) cropped & fit



x=72 y=120:raw-post Canvas ShowPdfFlowable ('cropped-media.pdf', 1) cropped & c

x=72 y=72: raw-pre Canvas ShowPdfFlowable ('cropped-media.pdf', 1) cropped & ce

RML Example 46: Cmyk

RML (Report Markup Language) is ReportLab's own language for specifying the appearance of a printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of ouput and are distributed within our commercial package as test cases. Each should be self explanatory and stand alone.

Tests of colour separated PDF output..

On screen, you'll see various blacks and shades of blue. The on-screen appearance depends on CMYK equivalents you supply.
If you open this document in Illustrator, Quark, Acrobat etc and check the separations, there should be plates for black and for Pantone 288. The black/greys should be converted into cmyk automatically.

The swatches above should fade from black to grey - 100% down to 25%.

The swatches above should be tints of Pantone 288 - 100% down to 25%.
Now we'll check the colours in a table

Black	Blue

We can now pass colours through into substrings in paragraphs even though that's handled by a different parser.

RML Example 47: Cmyk_sep

RML (Report Markup Language) is ReportLab's own language for specifying the appearance of a printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of ouput and are distributed within our commercial package as test cases. Each should be self explanatory and stand alone.

Tests of colour separated PDF output..

On screen, you'll see various blacks and shades of blue. The on-screen appearance depends on CMYK equivalents you supply.
If you open this document in Illustrator, Quark, Acrobat etc and check the separations, there should be plates for black and for Pantone 288.

The swatches above should fade from black to grey - 100% down to 25%.

The swatches above should be tints of Pantone 288 - 100% down to 25%.
Now we'll check the colours in a table

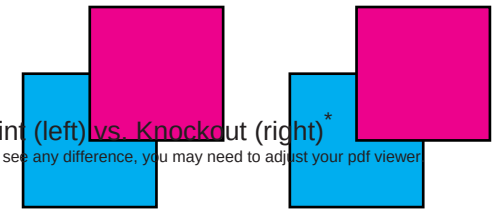
Black	Blue

Now we'll check the colours in a table
We can now pass colours through into substrings in paragraphs even though that's handled by a different parser.

RML Example 48: Overprint

RML (Report Markup Language) is ReportLab's own language for specifying the appearance of a printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of ouput and are distributed within our commercial package as test cases. Each should be self explanatory and stand alone.



Overprint (left) vs. Knockout (right)*

*If you can't see any difference, you may need to adjust your pdf viewer

RML Example 50: Separations

RML (Report Markup Language) is ReportLab's own language for specifying the appearance of a printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of ouput and are distributed within our commercial package as test cases. Each should be self explanatory and stand alone.

Tests of colour separated PDF output..

On screen, you'll see various blacks and shades of blue. The on-screen appearance depends on CMYK equivalents you supply.
If you open this document in Illustrator, Quark, Acrobat etc and check the separations, there should be plates for MY-BLACK and for Pantone 288.

The swatches above should fade from black to grey - 100% down to 25%.

The swatches above should be tints of Pantone 288 - 100% down to 25%.
Now we'll check the colours in a table

Black	Blue

We can now pass colours through into substrings in paragraphs even though that's handled by a different parser.

RML Example 52: Sep_black

RML (Report Markup Language) is ReportLab's own language for specifying the appearance of a printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of ouput and are distributed within our commercial package as test cases. Each should be self explanatory and stand alone.

Tests of colour separated PDF output..

On screen, you'll see various blacks and shades of blue. The on-screen appearance depends on CMYK equivalents you supply.
If you open this document in Illustrator, Quark, Acrobat etc and check the separations, there should be plates for MY-BLACK and for Pantone 288.

The swatches above should fade from black to grey - 100% down to 25%.

The swatches above should be tints of Pantone 288 - 100% down to 25%.
Now we'll check the colours in a table, RGB black is OK here

Black	Blue

We can now pass colours through into substrings in paragraphs even though that's handled by a different parser.

RML Example 51: Sep_cmyk

RML (Report Markup Language) is ReportLab's own language for specifying the appearance of a printed page, which is converted into PDF by the utility rml2pdf.

These RML samples showcase techniques and features for generating various types of ouput and are distributed within our commercial package as test cases. Each should be self explanatory and stand alone.

Tests of colour separated PDF output..

On screen, you'll see various blacks and shades of blue. The on-screen appearance depends on CMYK equivalents you supply.
If you open this document in Illustrator, Quark, Acrobat etc and check the separations, there should be plates for MY-BLACK and for Pantone 288.

The swatches above should fade from black to grey - 100% down to 25%.

The swatches above should be tints of Pantone 288 - 100% down to 25%.
Now we'll check the colours in a table, RGB black is OK here

Cyan	Blue

We can now pass colours through into substrings in paragraphs even though that's handled by a different parser.