RML Example 45: Codesnippets

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.

Code snippets test

Python has a code-highlighting package named 'Pygments'. Pygments is installed, you can generate colorized code snippets. If the sample below is in colour, you have it installed; otherwise, you can obtain it from pypi.python.org, or from the `python-pygments` package in Ubuntu. Pygments supports many, many different lexers.
Here is the code to insert a Python snippet...

```
<codesnippet language="python">...</codesnippet>
```

...and here is some sample outout...

```
        class code(MapNode):
            def evaluate(self, tagname, sdict, pcontent, extra, context):
                stylename = "pre.defaultStyle"
                if sdict.has_key("style"):
                    stylename = sdict["style"]
                if sdict.has_key("syntax"):
                    lang = sdict["syntax"]


                #clean up the block of code prior to display.

                src = ''.join(map(str, pcontent))
                #split line ends, strip trailing space


                lines = map(lambda x: x.rstrip(), src.split('\n'))
                #generally we trim off up to one leading and trailing blank lines
                #that's probably from indenting the XML
                if lines[0] == '':
                    lines = lines[1:]
                if lines[-1] == '':
                    lines = lines[:-1]
```

Now we'll show colourised XML:

```
        <tag attr="value">
            <content>Foo bar!</content>
        </tag>
```

If you don't specify a language parameter (or if pygments cannot be imported), it won't get colourised.

```
        <tag attr="value">
            <content>Foo bar!</content>
        </tag>
```

Because your XML might be indented, by default we remove an initial or final blank line, and we also 'dedent' to remove any whitespace on the left of your code block. However, the default paragraph style selected adds an indent on the left side.
The one below specifies a different paragraph style, defined in our document, to add a coloured backdrop...

```
def my_function(arg):
    foo, bar = do_stuff_to(arg)
    return [bar, foo[0]]
```