RML Example 56: Preformatted text

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.

Preformatted text test

Our pre tag can be used for printing simple blocks of code. It respects whitespace and newlines, and will not normally attempt to wrap your code. However, if your individual lines are too long, this can overflow the width of the column and even run off the page. Three optional attributes - maximumLineLength, splitCharacters and newLineCharacter - can be used to do simple wrapping. maximumLineLength will force the text to wrap.
Note that this simply counts characters - it takes no account of actual width on the page. The examples below wrap lines above a certain length and add a '> ' to the start of the following line.

```
class placePara(MapNode):
    SUPER = LazyPlaceParagraph
    def evaluate(self, tagname, sdict, pcontent, extra, context):
        localdict = sdict.copy()
        stylename = "para.defaultStyle"
        if localdict.has_key("style"):
            stylename = localdict["style"]
            del localdict["style"]
        bulletText = None
        if localdict.has_key("bulletText"):
            bulletText = localdict["bulletText"]
            del localdict["bulletText"]
        extraAttrs = dict(
                        placeX = readLength(localdict.pop('x'),context),
                        placeY = readLength(localdict.pop('y'),context),
                        placeWidth = readLength(localdict.pop('width','0'),
> context),
                        placeOrigin = localdict.pop('origin','page'),
                        placeAnchor = localdict.pop('anchor','sw'),
                        placeContext = context,
                        )
        return DeferredInitializationEx(self.SUPER, extraAttrs, ('para',
> localdict,pcontent,context), context[stylename], bulletText=bulletText)

    def MyProcessContent(self, content, controller, context, overrides):
        return content  #tuple tree parser now in place
Controller["placePara"] = placePara()
```