



用户手册

版本 3.5.56

文档生成自 2022/12/26 16:51:13

第 1 章 介绍	7
1.1 关于	7
1.2 什么是 ReportLab PDF Library?	7
1.3 ReportLab的商业软件	7
1.4 什么是 Python?	8
1.5 致谢	8
1.6 安装与设定	8
1.7 开始	9
1.8 全局配置	9
1.9 了解有关Python的更多信息	10
1.10 3.x版本系列的目标	10
第 2 章 使用 pdfgen 生成图形和文本	11
2.1 基本概念	11
2.2 更多关于画布的信息	11
2.3 绘图操作	12
2.4 工具："draw"的操作	13
2.5 工具箱：'状态变化'(state change) 操作	15
2.6 其他canvas的方法	17
2.7 坐标（默认用户空间）	17
2.8 颜色	21
2.9 色彩空间检查	24
2.10 彩色套印	24
2.11 标准字体和文本对象	26
2.12 文本对象方法	28
2.13 路径和直线	34
2.14 矩形、圆形、椭圆形。	38
2.15 贝兹尔曲线	39
2.16 路径对象方法	41
2.17 进一步阅读: ReportLab图形库	46
第 3 章 字体和编码	47
3.1 Unicode 和 UTF8 作为默认编码	47
3.2 输出字体自动替换	47
3.3 使用非标准的 Type 1 字体	47
3.4 标准的单字节字体编码	49
3.5 支持TrueType字体	51

3.6 亚洲字体支持	52
3.7 RenderPM 测试	53
第 4 章 PDF 的特殊功能	54
4.1 表单	54
4.2 链接和目的地(书签)	54
4.3 大纲	55
4.4 页面过渡效果	56
4.5 内部文件注释	56
4.6 加密	57
4.7 交互式表单	58
第 5 章 PLATYPUS - 页面布局和排版	66
5.1 设计目标	66
5.2 开始	67
5.3 Flowables	67
5.4 流动定位的准则	68
5.5 Frames	69
5.6 文档和模板	69
第 6 章 Paragraphs - 文本段落	73
6.1 ParagraphStyle	73
6.2 文本段落XML标记标签	79
6.3 段内标记	80
6.4 项目符号和段落编号	83
第 7 章 表格和表格样式	85
7.1 Table	85
7.2 TableStyle	86
7.3 TableStyle 方法	86
7.4 TableStyle Commands	86
第 8 章 Flowables	92
8.1 DocAssign	92
8.2 DocExec	92
8.3 DocPara	92
8.4 DocAssert	92
8.5 DocIf	92

8.6 DocWhile	92
第 9 章 其他有用的Flowables	94
9.1 Preformatted	94
9.2 XPreformatted	94
9.3 Image	95
9.4 Spacer	96
9.5 PageBreak	96
9.6 CondPageBreak	96
9.7 KeepTogether	96
9.8 TableOfContents	96
9.9 SimpleIndex	98
9.10 ListFlowable, ListItem	98
9.11 BalancedColumns	99
第 10 章 编写 Flowable	101
10.1 一个非常简单的 Flowable	101
10.2 修改内置的 Flowable	102
第 11 章 绘制	104
11.1 简介	104
11.2 一般概念	104
11.3 图表	107
11.4 Axes	110
11.5 柱状图	115
11.6 折线图	120
11.7 点线图	122
11.8 饼图	124
11.9 Legends	128
11.10 Shapes 多边形	130
11.11 小部件	134
附录 A ReportLab 示例	140
A.1 奥德赛	140
A.2 标准字体和颜色	140
A.3 Gadflypaper	140
A.4 Pythonpoint	141

附录 B 译者备注	142
B.1 Win Ansi 编码 和 Mac Roman 编码	142
B.2 生成pdf文件，推荐开源字体：思源黑体	142
附录 C 官方示例: Line	143
C.1 Line with markers (serious)	143
C.2 Line with markers (silly)	143
C.3 char with background color	143
C.4 dashed lines and number formats	144
C.5 time serious plot	144
附录 D 官方示例: Pie	145
D.1 Basic pie	145
D.2 Pie with multi-column legend	145
D.3 Exploding pie	145
D.4 Pie with nested legend	146
D.5 Pie with a pie	146
D.6 Legend with text wrapping	147
附录 E 官方示例: Scatter	148
E.1 Scatter plot with legend	148
附录 F 官方示例: bar	149
F.1 Four category eight month	149
F.2 Comparison chart	149
F.3 Multi-line x-axis labels	149
F.4 BarChart with table	150
F.5 Vertical labels	150
F.6 Dual Bar charts on one canvas	151
F.7 Vertical bar chart with mixed stacked & parallel bars	151
F.8 Vertical 3D bar chart with mixed stacked & parallel bars	152
F.9 Horizontal bar with red axis negative labels	152
F.10 Vertical bar with line labels	153
F.11 A Vertical Bar Chart With Line Indicated Bar Labels	153
附录 G 官方示例: quickcharts	155
G.1 面积图	155
G.2 3D条形图	155

G.3 背景柱状图	156
G.4 柱状图	156
G.5 3D背景柱状图	157
G.6 嵌套饼图	157
G.7 分裂饼图	158
G.8 3D分裂饼图	158
G.9 填充雷达图	159
G.10 3D折线图	159
G.11 折线图	160
G.12 标记折线图	160
G.13 直线图	161
G.14 3D直线图	161
G.15 直线标记图	162
G.16 矩形面积图	162
G.17 矩形条形图	163
G.18 矩形柱状图	163
G.19 3D饼图	164
G.20 饼图	164
G.21 雷达图	165
G.22 标记雷达图	165
G.23 堆叠面积图	166
G.24 堆叠条形图	166
G.25 堆叠柱状图	167
附录 H 官方示例: Area	168
H.1 面积折线图	168
H.2 面积动态标签图	168

第 1 章 介绍

1.1 关于

本文档是 ReportLab PDF 库的简介。阅读本文档我们已经假定您熟悉 Python 编程语言是。如果您是 Python 编程的新手，我们将在下一小节告诉您该去哪儿学习 Python 编程。

本文档并未覆盖百分之百的功能，但应该能解释所有概念和帮助您入门，并指出其他学习资源。阅读完本文档后，您应该准备好可以开始编程以生成复杂的 PDF 报告。

在本章, 我们将介绍基础功能:

- 什么是 ReportLab, 为什么要使用它？
- 什么是 Python？
- 我该做哪些准备以便开始？

我们需要您的帮助以确保本手册完整且有用。请将任何反馈发送到我们的用户邮件列表，请参考: www.reportlab.com。

1.2 什么是 ReportLab PDF Library?

这是一个软件库，可让您直接使用 Python 编程语言创建 Adobe 的可移植文档格式 (Portable Document Format) (PDF) 文档。它同样支持创建图表和数据图形各种位图和矢量格式，这就是 PDF。

PDF 是电子文档的全球标准。它支持高质量打印并且完全跨平台支持，这要归功于免费提供的 Acrobat Reader。任何先前生成纸质报告或驱动打印机的应用程序可以从制作 PDF 文档中受益；这些都可以存档通过电子邮件发送，放在网络上或以老式方式打印出来。但是，PDF 文件格式很复杂索引二进制格式，无法直接键入。PDF 格式规范的长度超过 600 页，PDF 文件必须提供精确的字节偏移量- 额外增加一个放置在有效 PDF 文档中任何位置的字符都可以呈现它无效。这使得它比 HTML 难生成。

世界上大多数 PDF 文档都是由 Adobe 的 Acrobat 工具 或 JAWS PDF Creator 等竞争对手产生的，这些工具充当“打印驱动程序”。任何想要自动化 PDF 制作的人通常都会使用 Quark, Word 或 Framemaker 之类的产品，该产品与宏或插件循环连接到 Acrobat，并在其中循环运行。几种语言和产品的管道传输速度可能很慢，而且有些笨拙。

ReportLab 库根据您的图形命令直接创建 PDF。没有干预步骤。您的应用程序可以非常快速地生成报告- 有时比传统的报告编写工具快几个数量级。此方法由其他几个库共享- C 的 PDFlib, Java 的 iText, .NET 的 iTextSharp 等。但是，ReportLab 库的不同之处在于它可以在更高的层次上运行，并具有一个功能齐全的引擎，用于布局包含表格和图表的文档。

此外，由于您正在使用功能强大的通用语言编写程序，因此从何处获取数据，如何转换数据以及输出的类型都没有任何限制。您可以创建。您可以在整个报表系列中重用代码。

预期 ReportLab 库至少在以下情况下有用：

- 网络上动态生成 PDF.
- 大批量公司报告和数据库发布.
- 用于其他应用程序的可嵌入打印引擎，包括“报告语言”，以便用户可以自定义自己的报告。这尤其适用于跨平台应用程序，这些应用程序不能依赖每个操作系统上一致的打印或预览 API。
- 具有图表，表格的复杂文档的“构建系统”和文字，例如管理帐户，统计报告和科学论文
- 从 XML 到 PDF 的一步

1.3 ReportLab 的商业软件

ReportLab 库构成了我们用于生成 PDF 的商业解决方案（报告标记语言（RML））的基础。可以在我们的网站上通过完整的文档进行评估。我们相信 RML 是开发丰富的 PDF 工作流程的最快，最简单的方法。您可以使用最喜欢的模板系统来填充 RML 文档，并使用与 HTML 类似的标记语言。然后调用我们的 `rml2pdf` API 函数以生成 PDF。这就是 ReportLab 员工用来构建您可以在 reportlab.com 上面看到的所有解决方案的东西。主要区别：

- 完整记录了两本手册，一份正式规范（DTD）和大量的自记录测试。（通过对比，我们尝试确保开源文档没有错，但是我们并不总是和代码保持同步）
- 在高级标记中工作，而不是构造Python对象图
- 不需要Python专业知识-您离开后，您的同事可能会感谢您！
- 支持矢量图形并包含其他PDF文档
- 用单个标签表示的更多有用功能，这将需要在开源软件包中进行大量编码
- 包括商业支持

我们要求开源开发人员考虑在适当的地方尝试RML。您可以在我们的网站上注册并尝试购买之前的副本。成本是合理的，并且与项目的规模有关，而收入则帮助我们花费更多的时间来开发此软件。

1.4 什么是 Python?

Python是一种解释型，交互式，面向对象的编程语言。通常将它与Tcl，Perl Scheme或Java进行比较。

Python将非凡的功能与非常清晰的语法结合在一起。它具有模块，类，异常，非常高级的动态数据类型和动态类型。有许多系统调用和库以及各种窗口系统（X11，Motif，Tk，Mac，MFC）的接口。新的内置模块很容易用C或C++编写。Python还可用作需要可编程接口的应用程序的扩展语言。

Python与Java一样古老，并且多年来一直稳步增长。自从我们的ReportLab包首次问世以来，它已经成为主流。许多ReportLab库用户已经是Python的忠实拥护者，但如果您不是Python的忠实拥护者，我们认为该语言是文档生成应用程序的绝佳选择，因为它的表达能力和从任何地方获取数据的能力。

Python受版权保护，但可自由使用和分发，甚至用于商业用途。

1.5 致谢

许多人为ReportLab做出了贡献。我们要特别感谢（按字母顺序）：

Albertas Agejevas, Alex Buck, Andre Reitz, Andrew Cutler, Andrew Mercer, Ben Echols, Benjamin Dumke, Benn B, Chad Miller, Chris Buergi, Chris Lee, Christian Jacobs, Dinu Gherman, Edward Greve, Eric Johnson, Felix Labrecque, Fubu @ bitbucket, Gary Poster, Germán M. Bravo, Guillaume Francois, Hans Brand, Henning Vonbargen, Hosam Aly, Ian Stevens, James Martin-Collar, Jeff Bauer, Jerome Alet, Jerry Casiano, Jorge Godoy, Keven D Smith, Kyle MacFarlane, Magnus Lie Hetland, Marcel Tromp, Marius Gedminas, Mark de Wit, Matthew Duggan, Matthias Kirst, Matthias Klose, Max M, Michael Egorov, Michael Spector, Mike Folwell, Mirko Dziadzka, Moshe Wagner, Nate Silva, Paul McNett, Peter Johnson, PJACock, Publio da Costa Melo, Randolph Bentson, Robert Alsina, Robert Hölzl, Robert Kern, Ron Peleg, Ruby Yocum, Simon King, Stephan Richter, Steve Halasz, Stoneleaf @ bitbucket, T Blatter, Tim Roberts, Tomasz Swiderski, Ty Sarna, Volker Haas, Yoann Roman, and many more.

特别感谢Just van Rossum在字体技术方面的宝贵帮助。

Moshe Wagner和Hosam Aly为RTL补丁做出了巨大的贡献，该补丁尚未发布。

Marius Gedminas在TrueType字体方面的工作值得一臂之力，我们很高兴将其包含在工具包中。最后，我们感谢Michal Kosmulski的DarkGarden字体和Bitstream Inc.的Vera字体。

1.6 安装与设定

为避免重复，安装说明保存在我们发行版的README文件中，可以在以下位置在线查看。<https://hg.reportlab.com/hg-public/reportlab/>

此版本（3.5.56）的ReportLab需要Python版本2.7，3.6+或更高版本。如果您需要使用Python 2.5或2.6，请使用最新的ReportLab 2.7软件包。

1.7 开始

ReportLab是一个开源项目。尽管我们是一家商业公司，但我们免费提供核心PDF生成源，即使出于商业目的，我们也不会直接从这些模块中获得收入。我们也欢迎社区以及其他任何开源项目的帮助。您可以通过多种方式提供帮助：

- 有关核心API的一般反馈。对你起作用吗？有粗糙的边缘吗？有什么感觉笨拙而笨拙的吗？
- 放入报告的新对象，或库的有用工具。我们有一个针对报表对象的开放标准，因此，如果您编写了不错的图表或表类，为什么不贡献它呢？
- 片段和案例研究：如果您产生了不错的输出，请在<http://www.reportlab.com>上在线注册并提交输出片段（带或不带脚本）。如果ReportLab为您解决了工作中的问题，请编写一些“案例研究”并提交。如果您的网站使用我们的工具制作报告，请让我们链接到它。我们很乐意在我们的网站上显示您的作品（并用您的名字和公司来称赞）！
- 处理核心代码：我们有一长串需要完善或实现的事情。如果您缺少某些功能或只是想提供帮助，请告诉我们！

想要了解更多信息或参与其中的任何人的第一步是加入邮件列表。要订阅，请访问<http://two.pairlist.net/mailman/listinfo/reportlab-users>。您还可以从那里浏览小组的档案和贡献。邮件列表是报告错误并获得支持的地方。

他的代码现在位于Mercurial信息库中的我们网站（<http://hg.reportlab.com/hg-public/reportlab/>）上，以及问题跟踪器和Wiki。'每个人都可以随时做出贡献，但是如果您正在积极地进行一些改进，或者想引起人们对问题的关注，请使用邮件列表告知我们。

1.8 全局配置

有许多选项很可能需要为程序进行全局配置。python脚本模块reportlab/rl_config.py汇总了各种设置文件。您可能需要检查文件reportlab/rl_settings.py，其中包含当前使用的变量的默认值。rl_settings模块reportlab.local_rl_settings, reportlab_settings（位于python路径上任何位置的脚本文件）以及文件~/.reportlab_settings（请注意，没有.py）有多个替代项。可以进行临时更改。使用环境变量，这些变量是rl_settings.py中以RL开头的变量，例如RL_verbose=1。

有用的 rl_config 变量

- *verbose*: 设置为整数值以控制诊断输出。
- *shapeChecking*: 将此设置为零可关闭图形模块中的许多错误检查
- *defaultEncoding*: 将此设置为 WinAnsiEncoding 或 MacRomanEncoding。
- *defaultPageSize*: 将其设置为 reportlab/lib/pagesizes.py 中定义的值之一；交付时将其设置为 pagesizes.A4; 其他值是 pagesizes.letter 等。
- *defaultImageCaching*: 设置为零以禁止在硬盘驱动器上创建.a85文件。默认设置是创建这些经过预处理的PDF兼容图像文件，以加快加载速度
- *T1SearchPath*: 这是表示目录的字符串的python列表，可以查询有关类型1字体的信息
- *TTFSearchPath*: 这是表示目录的字符串的python列表，可以查询该目录以获取有关TrueType字体的信息
- *CMapSearchPath*: 这是表示目录的字符串的python列表，可以查询该目录以获取字体代码映射的信息。
- *showBoundary*: 设置为非零以绘制边界线。
- *ZLIB_WARNINGS*: 如果未找到Python压缩扩展，则设置为非零以获得警告。
- *pageCompression*: 设置为非零以尝试获取压缩的PDF。
- *allowtableBoundsErrors*: 设置为0会在非常大的Platypus表元素上强制执行错误
- *emptyTableAction*: 控制空表的行为，可以为“error”（默认），“indicate”或“ignore”
- *TrustedHosts*: 则列出全局模式下受信任的主机列表；这些模式可用于段落文本中的 标签等地方。
- *trustedSchemes*: 与 trustedHosts 一起使用的允许的 URL 方案的列表

有关变量的完整列表，请参见文件 reportlab/rl_settings.py。

其他修改

可以使用以下模块之一对reportlab工具箱环境进行更复杂的修改：`reportlab.local_rl_mods`（reportlab文件夹中的.py脚本），`reportlab_mods`（python路径上的.py文件）或`~/.reportlab_mods`（注意是.py）。

1.9 了解有关Python的更多信息

如果您是Python的初学者，则应该从越来越多的Python编程资源中检出一个或多个。

以下内容可从网上免费获得：

- **Python文档** Python.org网站上的文档列表。 <http://www.python.org/doc/>
- **Python教程** 官方的Python教程，最初由Guido van Rossum亲自编写。 <http://docs.python.org/tutorial/>
- **学习编程** Alan Gauld撰写的编程指南。非常重视Python，但也使用其他语言。 <http://www.freenetpages.co.uk/hp/alan.gauld/>
- **即时Python** Magnus Lie Hetland撰写的长达6页的速成课程。 <http://www.hetland.org/python/instant-python.php>
- **深入Python** 适用于经验丰富的程序员的免费Python教程。 <http://www.diveintopython.net/>

1.10 3.x版本系列的目标

已生成ReportLab 3.0，以帮助迁移到Python3.x。Python 3.x将在将来的Ubuntu版本中成为标准配置，并且越来越受欢迎，并且现在有很大一部分主要的Python程序包都在Python 3上运行。

- Python 3.x兼容性。一行代码应在2.7和3.6上运行
- `__init__.py` 限制为2.7或 ≥ 3.6
- `__init__.py` 允许导入可选的 `reportlab.local_rl_mods`，以允许猴子打补丁等。
- `rl_config` 现在可以导入 `rl_settings`，还可以导入 `local_rl_settings`，`reportlab_settings.py`，最后是 `~/.reportlab_settings`
- ReportLab C扩展现在位于reportlab中。不再需要 `_rl_accel`。现在，所有 `_rl_accel` 导入都通过 `reportlab.lib.rl_accel`
- `xmllib` 以及用于引起HTMLParser问题的 `paraparser` 东西一去不复返了。
- 一些过时的C扩展名（`sgmlop`和`pyHnj`）消失了
- `_rl_accel` C扩展模块对多线程系统的改进支持。
- 删除了 `reportlab/lib/para.py` 和 `pycanvas.py`。这些最好属于第三方案程序包，可以利用上面的 **Monkeypatching** 功能。
- 增加了无需转换为RGB即可输出灰度和1位PIL图像的功能。（由Matthew Duggan贡献）
- 高亮注释（由Ben Echols提供）
- 完全符合 `pip`，`easy_install`，`wheel`等要求

有关详细的发行说明，请访问：<http://www.reportlab.com/software/documentation/release/notes/30/>

第2章 使用 pdfgen 生成图形和文本

2.1 基本概念

pdfgen包是生成PDF文档的最低级别接口。

一个pdfgen引擎本质上是一个将文档"绘制"到页面序列上的指令序列。

提供绘画操作的接口对象是pdfgen canvas。

他的画布应该被认为是一张白纸，白纸上的点用笛卡尔 (x,y) 坐标确定，默认情况下， $(0,0)$ 原点在页面的左下角。此外，第一个坐标 x 往右走，第二个坐标 y 往上走，这是默认的。

下面是一个使用画布的简单示例程序。

```
from reportlab.pdfgen import canvas

c = canvas.Canvas("hello.pdf")
c.drawString(100,100,"Hello World")
c.showPage()
c.save()
```

上面的代码创建了一个canvas对象，它将在当前工作目录下生成一个名为hello.pdf的PDF文件。然后调用hello函数，将canvas作为参数。最后，showPage方法保存canvas的当前页面。

showPage方法会使canvas停止在当前页上绘制，任何进一步的操作都会在随后的页面上绘制（如果有任何进一步的操作--如果没有的话新页面被创建）。在文档构建完成后必须调用save方法--它将生成PDF文档，这也是canvas对象的全部目的。

2.2 更多关于画布的信息

在介绍绘图操作之前，我们先离题万里，介绍一下配置画布可以做的一些事情。

有许多不同的设置可供选择。如果你是Python新手，或者迫不及待地想产生一些输出，你可以跳过前面的内容，但以后再来阅读这个内容吧！

首先，我们来看看画布的构造函数参数：

```
def __init__(self,filename,
             pagesize=(595,799),
             bottomup = 1,
             pageCompression=0,
             encoding=rl_config.defaultEncoding,
             verbosity=0,
             encrypt=None):
```

参数filename控制最终PDF文件的名称。你也可以传入任何开放的二进制流（如sys.stdout，python过程中标准的二进制编码输出），PDF文件将被写入该流。由于PDF是二进制格式，所以在它之前或之后写其他东西的时候要小心，你不能在HTML页面中间内联传递PDF文档！你可以在HTML页面中写一个PDF文件，但不能在HTML页面中写一个PDF文件。

参数pagesize是一个以点为单位的两个数字的元组（1/72英寸）。画布默认为A4（国际标准的页面尺寸，与美国标准的letter页面尺寸不同），但最好明确指定。

大多数常见的页面大小都可以在库模块reportlab.lib.pagesizes中找到，所以你可以使用类似于

```
from reportlab.lib.pagesizes import letter, A4
myCanvas = Canvas('myfile.pdf', pagesize=letter)
width, height = letter #keep for later
```

如果您在打印文件时遇到问题，请确保您使用正确的页面尺寸（通常是A4或letter）。

很多时候，你会想根据页面大小来计算东西。在上面的例子中，我们提取了宽度和高度。

在以后的程序中，我们可能会使用width变量来定义右边距为width - inch，而不是使用一个常数。

通过使用变量，即使页面大小发生变化，页边距也会有意义。

参数bottomup用于切换坐标系。一些图形系统(如PDF和PostScript)将(0,0)置于页面的左下角，其他系统(如许多图形用户界面[GUI's])将原点置于bottomup参数已被废弃，以后可能会被删除。

需要看看它是否真的对所有任务都有效，如果没有，那就把它扔掉吧。

`pageCompression`选项决定是否对每一页的PDF操作流进行压缩。默认情况下，页面流不被压缩，因为压缩会减慢文件的生成过程。如果输出大小很重要，则设置`pageCompression=1`，但请记住，压缩后的文件会更小，但生成速度更慢。请注意，图像总是压缩的，只有当你在每一页上有非常多的文本和矢量图形时，这个选项才会节省空间。

在2.0版本中，`encoding`参数基本上已经过时了，可能99%的用户都可以省略。它的默认值很好，除非你特别需要使用MacRoman中的25个字符之一，而Winansi中没有。这里是一个有用的参考资料：<http://www.alanwood.net/demos/charsetdiffs.html>。该参数决定了该文件的字体编码。标准Type 1字体；这应该与您系统上的编码相对应。请注意，这是字体内部使用的编码；您的文本传递给ReportLab工具包的渲染信息应该总是Python的unicode字符串对象或UTF-8编码的字节字符串(见下一章)！

字体编码目前有两个值：'WinAnsiEncoding'或'MacRomanEncoding'。上面的变量`rl_config.defaultEncoding`指向的是到前者，这是Windows、Mac OS X和许多Unix的标准配置(包括Linux)。如果你是Mac用户，并且没有OS X，你可能会想要进行全局性的修改：修改在`reportlab/pdfbase/pdfdoc.py`来切换它。否则，您可以完全无视这个论点，永远不会通过。对于所有TTF和常用的CID字体，这里传入的编码会被忽略。因为reportlab库本身就知道这些情况下的正确编码。

演示脚本`reportlab/demos/stdfonts.py`将打印出两个测试文档，显示所有字体的所有代码点，这样你就可以查找字符。特殊字符可以用通常的Python转义序列插入到字符串命令中，例如`\101 = 'A'`。

参数`verbosity`决定了打印多少日志信息。默认情况下，它是0，以帮助应用程序从标准输出中捕获PDF。如果值为1，每次生成文档时，您都会得到一条确认信息。更高的数字可能会在将来提供更多的输出。

参数`encrypt`决定了是否以及如何对文档进行加密。默认情况下，文档没有被加密。如果`encrypt`是一个字符串对象，它被用作pdf的用户密码。如果`encrypt`是一个`reportlab.lib.pdfencrypt.StandardEncryption`的实例，那么这个对象就被用来加密pdf。这允许对加密设置进行更精细的控制。加密在第4章有更详细的介绍。

to do -- 所有的信息功能和其他非绘图的东西。

覆盖所有构造函数参数，以及 `setAuthor` 等。

2.3 绘图操作

假设上面提到的`hello`函数实现如下（我们先不详细解释每个操作）。

```
def hello(c):
    from reportlab.lib.units import inch
    # move the origin up and to the left
    c.translate(inch,inch)
    # define a large font
    c.setFont("Helvetica", 14)
    # choose some colors
    c.setStrokeColorRGB(0.2,0.5,0.3)
    c.setFillColorRGB(1,0,1)
    # draw some lines
    c.line(0,0,1.7*inch)
    c.line(0,0,1*inch,0)
    # draw a rectangle
    c.rect(0.2*inch,0.2*inch,1*inch,1.5*inch, fill=1)
    # make text go straight up
    c.rotate(90)
    # change color
    c.setFillColorRGB(0,0,0.77)
    # say hello (note after rotate the y coord needs to be negative!)
    c.drawString(0.3*inch, -inch, "Hello World")
```

检查这段代码时注意到，使用画布进行的操作基本上有两种类型。第一种类型是在页面上画一些东西，如一个文本字符串或一个矩形或一条线。第二种类型改变画布的状态，如改变当前的填充或笔触颜色，或改变当前的字体类型和大小。

如果我们把程序想象成一个在画布上工作的画家，“绘制”操作使用当前的一组工具（颜色、线条样式、字体等）将颜料涂抹到画布上，而“状态改变”操作则改变了当前的一个工具（例如，将填充颜色从原来的任何颜色改为蓝色，或者将当前的字体改为15点的Times-Roman）。

上面列出的 "hello world "程序生成的文档将包含以下图形。

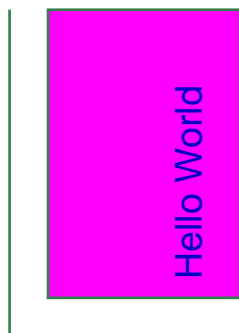


图 2 - 1 : pdfgen 生成 "Hello World"

关于本文档中的演示

本文档包含了所讨论的代码的演示，如上图矩形所示。这些演示是在嵌入指南真实页面中的 "小页 " 上绘制的。这些小页面的宽度为5.5英寸，高度为3英寸。演示显示的是演示代码的实际输出。为了方便起见，输出的大小被稍微缩小了。

2.4 工具："draw"的操作

本节简要列出了该程序可用来使用画布界面在页面上绘制信息的工具。这些工具将在后面的章节中详细讨论。这里列出这些工具是为了便于参考和总结。

绘制直线相关方法

```
canvas.line(x1,y1,x2,y2)
```

```
canvas.lines(linelist)
```

线条方法在画布上绘制直线段。

绘制形状相关方法

```
canvas.grid(xlist, ylist)
```

```
canvas.bezier(x1, y1, x2, y2, x3, y3, x4, y4)
```

```
canvas.arc(x1,y1,x2,y2)
```

```
canvas.rect(x, y, width, height, stroke=1, fill=0)
```

```
canvas.ellipse(x1,y1, x2,y2, stroke=1, fill=0)
```

```
canvas.wedge(x1,y1, x2,y2, startAng, extent, stroke=1, fill=0)
```

```
canvas.circle(x_cen, y_cen, r, stroke=1, fill=0)
```

```
canvas.roundRect(x, y, width, height, radius, stroke=1, fill=0)
```

形状方法在画布上绘制常见的复杂形状。

绘制字符串相关方法

```
canvas.drawString(x, y, text):
```

```
canvas.drawRightString(x, y, text)
```

```
canvas.drawCentredString(x, y, text)
```

绘制字符串方法在画布上绘制单行文字。

文本对象相关方法

```
textobject = canvas.beginText(x, y)
```

```
canvas.drawText(textobject)
```

文本对象用于以`canvas`界面不直接支持的方式来格式化文本。程序使用`beginText`从`canvas`创建一个文本对象，然后通过调用`textobject`方法来格式化文本。最后使用`drawText`将`textobject`绘制到`canvas`上。

路径对象相关方法

```
path = canvas.beginPath()
```

```
canvas.drawPath(path, stroke=1, fill=0, fillMode=None)
```

```
canvas.clipPath(path, stroke=1, fill=0, fillMode=None)
```

路径对象类似于文本对象：它们为执行复杂的图形绘制提供了画布界面无法直接提供的专用控件。程序使用`beginPath`创建一个路径对象，使用路径对象的方法为路径填充图形，然后使用`drawPath`在画布上绘制路径。

也可以使用`clipPath`方法将一个路径作为“剪切区域”--例如，一个圆形的路径可以用来剪切掉一个矩形图像的外部部分，只留下图像的一个圆形部分在页面上可见。

如果指定了`fill=1`，那么`fillMode`参数可以用来设置`0=even-odd`或`1=non-zero`填充模式，这将改变复杂路径的填充方式。如果使用默认的`None`值，则使用`canvas`的`_fillMode`属性值（通常为`0`即`even-odd`）。

图像相关方法



你需要Python Imaging Library (PIL)来使用ReportLab包的图像。通过运行我们的`tests`子目录中的脚本`test_pdfgen_general.py`并查看输出的第7页，可以找到下面的技术示例。

有两种听起来差不多的画像方式。首选的是`drawImage`方法。它实现了一个缓存系统，所以你可以一次定义一个图像，并多次绘制；它将只在PDF文件中存储一次。`drawImage`还公开了一个高级参数，一个透明度掩模，将来还会公开更多。较老的技术，`drawInlineImage`在页面流中存储位图，因此，如果你在文档中不止一次使用相同的图像，效率非常低；但如果图像非常小且不重复，则可以导致PDF更快地渲染。我们先讨论最老的那个。

```
canvas.drawInlineImage(self, image, x,y, width=None,height=None)
```

`drawInlineImage`方法在画布上放置一张图片。参数`image`可以是一个PIL图像对象或图像文件名。许多常见的文件格式都被接受，包括GIF和JPEG。它以(宽, 高)元组的形式返回实际图像的像素大小

。

```
canvas.drawImage(self, image, x,y, width=None,height=None,
                 mask=None)
```

参数和返回值和drawInlinelImage一样。然而，我们使用了一个缓存系统；一个给定的图像将只在第一次使用时被存储，而只是在后续使用时被引用。

如果您提供一个文件名，它假设相同的文件名意味着相同的图像。

如果你提供了一个PIL图片，它在重新嵌入之前会测试内容是否有实际变化。

参数mask可以让你创建透明图像。

它需要6个数字，并定义RGB值的范围，这些值将被屏蔽掉或被视为透明。例如，使用[0,2,40,42,136,139]，它将遮蔽任何红色值为 0 或 1 的像素，绿色值为 40 或 41 的像素，蓝色值为 136、137 或 138 的像素（在 0-255 的范围内）。目前你的工作是知道哪种颜色是 "透明" 的或背景的。

PDF允许许多图像功能，我们将在一段时间内暴露更多的图像功能，可能会用额外的关键字参数来表示drawImage。

结束一个页面

```
canvas.showPage()
```

showPage方法完成了当前页面。所有额外的绘图将在另一个页面上完成。



警告! 当你在pdfgen中前进到一个新的页面时，所有的状态改变（字体改变，颜色设置，几何变换，等等）都会被忘记。任何您希望保留的状态设置必须在程序继续绘制之前重新设置!

2.5 工具箱：'状态变化'(state change) 操作

本节简要列举了程序使用canvas界面将信息绘制到页面上的工具的切换方法。这些也将在后面的章节中详细讨论。

改变颜色

```
canvas.setFillColorCMYK(c, m, y, k)
```

```
canvas.setStrokeColorCMYK(c, m, y, k)
```

```
canvas.setFillColorRGB(r, g, b)
```

```
canvas.setStrokeColorRGB(r, g, b)
```

```
canvas.setFillColor(acolor)
```

```
canvas.setStrokeColor(acolor)
```

```
canvas.setFillGray(gray)
```

```
canvas.setStrokeGray(gray)
```

PDF支持三种不同的颜色模型：灰度级、外加剂(red/green/blue或RGB)、和 减去暗度参数（青色/红褐色/黄色/暗度或CMYK）。ReportLab包还提供了命名颜色，如lawngreen。

这里是图形状态下的两个基本颜色参数：Fill的为图形的填充色和Stroke为图形的边框色。

上述两种方法支持使用四种颜色中的任何一种来设置填充或描边颜色。

更改字体

```
canvas.setFont(psfontname, size, leading = None)
```

setFont方法将当前的文本字体改为给定的类型和大小。**leading**参数指定了从一行文字前进到下一行时向下移动的距离。

更改图形线条样式

```
canvas.setLineWidth(width)
```

```
canvas.setLineCap(mode)
```

```
canvas.setLineJoin(mode)
```

```
canvas.setMiterLimit(limit)
```

```
canvas.setDash(self, array=[], phase=0)
```

在PDF中绘制的线条可以以多种图形样式呈现。线条可以有不同的宽度，它们可以以不同的盖子样式结束，它们可以以不同的连接样式相接，它们可以是连续的，也可以是点线或虚线。上述方法可以调整这些不同的参数。

改变几何形状

```
canvas.setPageSize(pair)
```

```
canvas.transform(a,b,c,d,e,f):
```

```
canvas.translate(dx, dy)
```

```
canvas.scale(x, y)
```

```
canvas.rotate(theta)
```

```
canvas.skew(alpha, beta)
```

所有的PDF图纸都适合指定的页面大小。在指定的页面尺寸之外绘制的元素是不可见的。此外，所有绘制的元素都会通过一个可能调整其位置和/或扭曲其外观的仿射变换。

setPageSize方法可以调整当前的页面大小。**transform**, **translate**, **scale**, **rotate**, 和 **skew** 方法会给当前的变换添加额外的变换。重要的是要记住，这些转换是递增的。-- 新的变换会修改当前的变换(但不会取代它)；

状态控制

```
canvas.saveState()
```

```
canvas.restoreState()
```

很多时候，保存当前的字体、图形变换、线条样式和其他图形状态是很重要的，以便以后恢复它们。**saveState**方法标记了当前的图形状态，以便以后通过匹配的**restoreState**进行恢复。请注意，保存和还原方法的调用必须匹配--还原调用会将状态还原到最近保存的状态，而最近的状态还没有被还原。但是，你不能在一个页面上保存状态，然后在下一个页面上还原它 -- 页面之间不会保存状态。

2.6 其他canvas的方法

并非所有canvas对象的方法都适合"tool"或"toolbox"类别。
以下是一些不合群的方法，为了完整起见，在此收录。

```

canvas.setAuthor()
canvas.addOutlineEntry(title, key, level=0, closed=None)
canvas.setTitle(title)
canvas.setSubject(subj)
canvas.pageHasData()
canvas.showOutline()
canvas.bookmarkPage(name)
canvas.bookmarkHorizontalAbsolute(name, yhorizontal)
canvas.doForm()
canvas.beginForm(name, lowerx=0, lowery=0, upperx=None, uppery=None)
canvas.endForm()
canvas.linkAbsolute(contents, destinationname, Rect=None, addtopage=1,
name=None, **kw)
canvas.linkRect(contents, destinationname, Rect=None, addtopage=1,
relative=1, name=None, **kw)
canvas.getPageNumber()
canvas.addLiteral()
canvas.getAvailableFonts()
canvas.stringWidth(self, text, fontName, fontSize, encoding=None)
canvas.setPageCompression(onoff=1)
canvas.setPageTransition(self, effectname=None, duration=1,
direction=0, dimension='H', motion='I')

```

2.7 坐标（默认用户空间）

默认情况下，页面上的位置是由一对数字来标识的，例如，一对(4.5*inch, 1*inch)标识的是页面上的位置。例如，一对(4.5*inch, 1*inch)从左下角开始，向右移动4.5英寸，再向上移动一英寸，来标识页面上的位置。

例如，下面的函数在canvas上绘制一些元素。

```

def coords(canvas):
    from reportlab.lib.units import inch
    from reportlab.lib.colors import pink, black, red, blue, green
    c = canvas
    c.setStrokeColor(pink)
    c.grid([1inch, 2*inch, 3*inch, 4*inch], [0.5*inch, inch, 1.5*inch, 2*inch,
    2.5*inch])
    c.setStrokeColor(black)
    c.setFont("Times-Roman", 20)
    c.drawString(0,0, "(0,0) the Origin")
    c.drawString(2.5*inch, inch, "(2.5,1) in inches")
    c.drawString(4*inch, 2.5*inch, "(4, 2.5)")
    c.setFill-color(red)
    c.rect(0,2*inch,0.2*inch,0.3*inch, fill=1)
    c.setFill-color(green)
    c.circle(4.5*inch, 0.4*inch, 0.2*inch, fill=1)

```

在默认的用户空间中，"原点"(0,0)点在左下角。在默认的用户空间中执行coords函数（针对"演示迷你页"），我们得到以下结果。

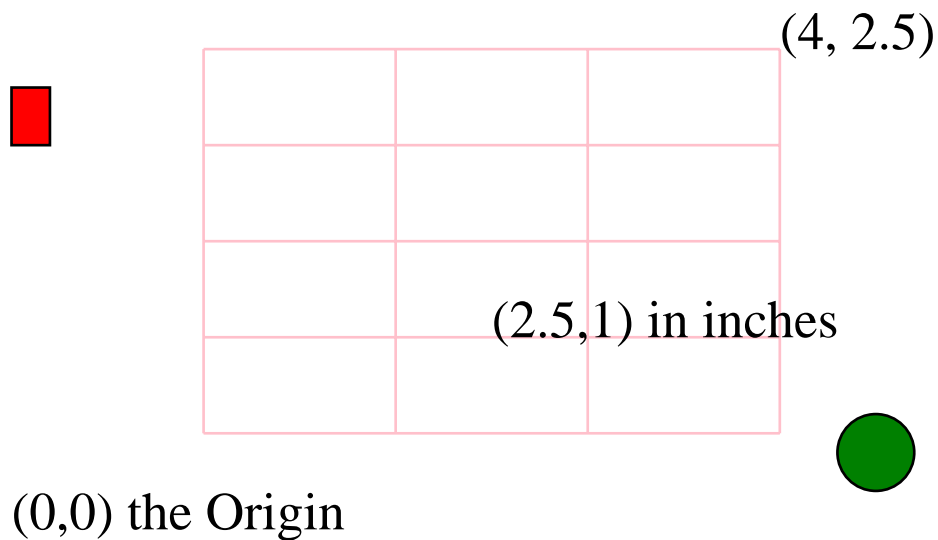


图 2 - 2 : 坐标系统

移动原点：translate方法

通常情况下，"移动原点"到左下角的新点是很有用的。

`canvas.translate(x,y)`方法将当前页面的原点移动到当前由 (x,y) 确定的点。

例如下面的平移函数首先移动原点，然后再绘制如上所示的相同对象。

```
def translate(canvas):
    from reportlab.lib.units import cm
    canvas.translate(2.3*cm, 0.3*cm)
    coords(canvas)
```

这就产生了以下结果：

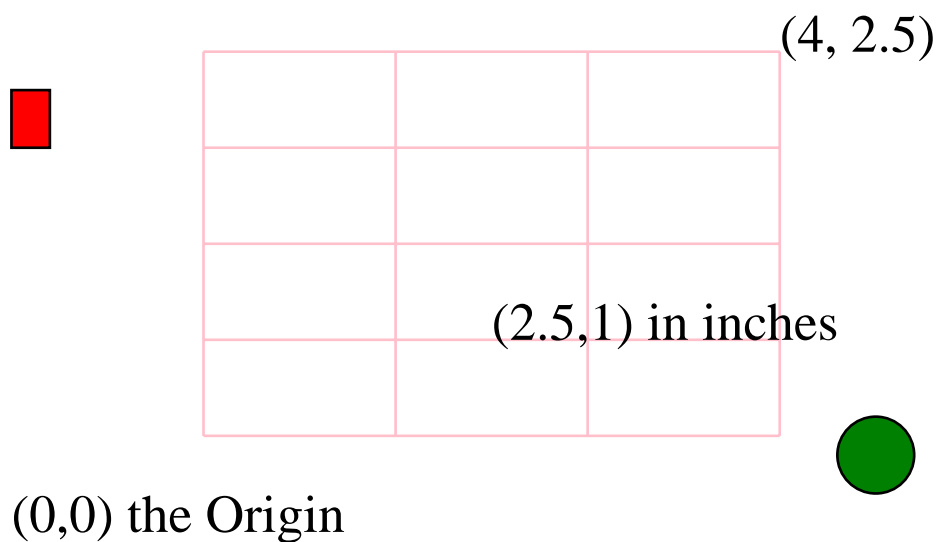


图 2 - 3 : 移动原点：translate方法



注意：如示例中所示，完全可以将对象或对象的一部分绘制到“页面之外”。特别是一个常见的令人困惑的错误是将整个绘图从页面的可见区域翻译出来的翻译操作。如果一个程序产生了一个空白页，那么所有绘制的对象都有可能不在页面上。

缩小与增长：scale操作

另一个重要的操作是缩放操作。缩放操作`canvas.scale(dx, dy)`分别以`dx`, `dy`系数来拉伸或缩小`x`和`y`的尺寸。通常情况下，`dx`和`dy`是相同的 -- 例如，要在所有维度上将图形缩小一半，使用`dx = dy = 0.5`。然而为了说明问题，我们举一个例子，其中`dx`和`dy`是不同的。

```
def scale(canvas):
    canvas.scale(0.75, 0.5)
    coords(canvas)
```

这样就会产生一个“短小精悍”的缩小版的之前显示的操作。

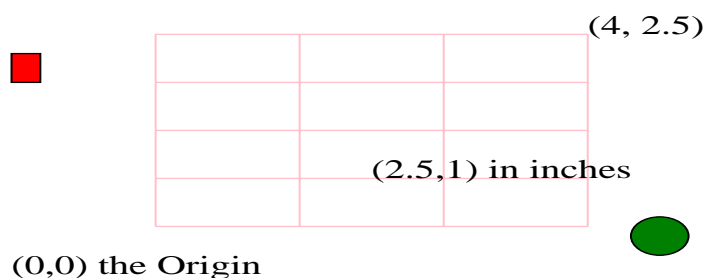


图 2 - 4 : 缩放坐标系统



注意：缩放也可能会将对象或对象的一部分从页面上移开，或者可能会导致对象“缩水为零”。缩放和翻译可以结合起来，但操作的顺序很重要。

```
def scaletranslate(canvas):
    from reportlab.lib.units import inch
    canvas.setFont("Courier-BoldOblique", 12)
    # save the state
    canvas.saveState()
    # scale then translate
    canvas.scale(0.3, 0.5)
    canvas.translate(2.4*inch, 1.5*inch)
    canvas.drawString(0, 2.7*inch, "Scale then translate")
    coords(canvas)
    # forget the scale and translate...
    canvas.restoreState()
    # translate then scale
    canvas.translate(2.4*inch, 1.5*inch)
    canvas.scale(0.3, 0.5)
    canvas.drawString(0, 2.7*inch, "Translate then scale")
    coords(canvas)
```

这个示例函数首先保存当前的`canvas`状态，然后进行`scale`和`translate`操作。之后，函数恢复了状态（有效地消除了缩放和翻译的影响），然后以不同的顺序进行相同的操作。观察下面的效果。

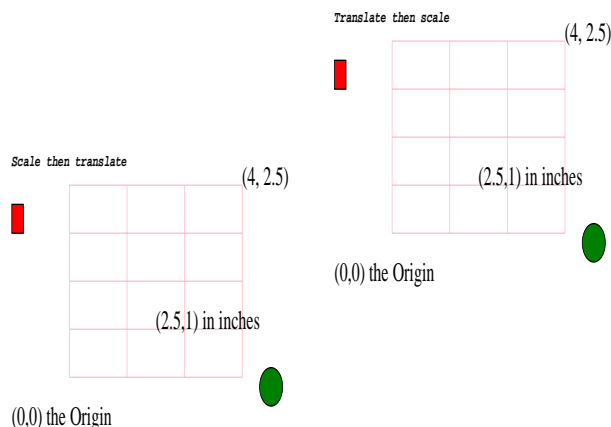


图 2 - 5 : 缩放和翻译



注意：缩放会收缩或增长所有的东西，包括线宽，因此使用`canvas.scale`方法以缩放的微观单位来渲染微观图形可能会产生一个blob（因为所有的线宽都会被大量扩展）。此外，以米为单位渲染飞机翼，缩放为厘米，可能会导致线条收缩到消失的程度。对于工程或科学目的，如这些比例和翻译。在使用画布渲染之前，从外部对单元进行渲染。

保存和恢复 canvas 状态：saveState 和 restoreState。

`scaletranslate`函数使用了`canvas`对象的一个重要特性：能够保存和恢复`canvas`的当前参数。通过在一对匹配的`canvas.saveState()`和`canvas.restoreState()`操作中包含一个操作序列，所有字体、颜色、线条样式、缩放、翻译或`canvas`图形状态的其他方面的变化都可以恢复到`saveState()`点的状态。请记住，保存/还原调用必须匹配：一个杂乱的保存或还原操作可能会导致意外和不理想的行为。另外，请记住，没有`canvas`状态会在页面中断时被保存，保存/还原机制不能跨越分页符工作。

镜像

有趣的是，虽然可能不是非常有用，但注意到比例因子可以是负的。例如下面的函数

```
def mirror(canvas):
    from reportlab.lib.units import inch
    canvas.translate(5.5*inch, 0)
    canvas.scale(-1.0, 1.0)
    coords(canvas)
```

创建`coord`函数绘制的元素的镜像。

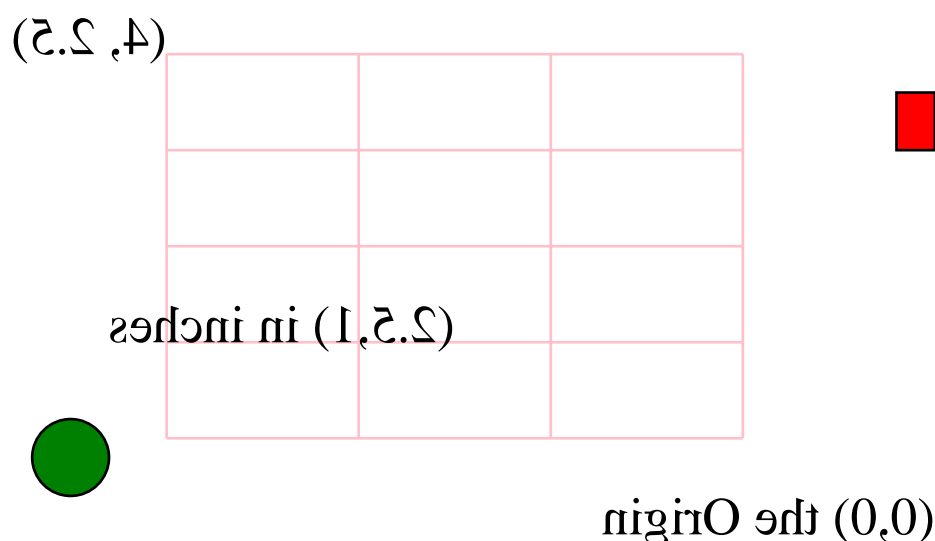


图 2 - 6 : 镜像

注意，文字串是倒着画的。

2.8 颜色

PDF中使用的颜色一般有两种类型，这取决于PDF将被使用的媒体。最常见的屏幕颜色模型RGB可以在PDF中使用，然而在专业印刷中主要使用另一种颜色模型CMYK，它可以对油墨如何应用于纸张进行更多的控制。以下是关于这些颜色模型的更多信息。

RGB颜色

RGB或称加色表示法，遵循电脑屏幕添加不同层次的红、绿、蓝光的方式，使其间的任何颜色，其中白色是通过将三盏灯全开形成的。

在pdfgen中，有三种方法可以指定RGB颜色：通过名称（使用color模块），通过红/绿/蓝（加法，RGB）值，或者通过灰度级别。下面的colors函数对这四种方法分别进行了练习。

```
def colorsRGB(canvas):
    from reportlab.lib import colors
    from reportlab.lib.units import inch
    black = colors.black
    y = x = 0; dy=inch*3/4.0; dx=inch*5.5/5; w=h=dy/2; rdx=(dx-w)/2
    rdy=h/5.0; texty=h+2*rdy
    canvas.setFont("Helvetica",10)
    for [namedcolor, name] in (
        [colors.lavenderblush, "lavenderblush"],
        [colors.lawnegreen, "lawnegreen"],
        [colors.lemonchiffon, "lemonchiffon"],
        [colors.lightblue, "lightblue"],
        [colors.lightcoral, "lightcoral"]):
        canvas.setFillColor(namedcolor)
        canvas.rect(x+rdx, y+rdy, w, h, fill=1)
        canvas.setFillColor(black)
        canvas.drawCentredString(x+dx/2, y+texty, name)
        x = x+dx
    y = y + dy; x = 0
    for rgb in [(1,0,0), (0,1,0), (0,0,1), (0.5,0.3,0.1), (0.4,0.5,0.3)]:
        r,g,b = rgb
        canvas.setFillColorRGB(r,g,b)
        canvas.rect(x+rdx, y+rdy, w, h, fill=1)
        canvas.setFillColor(black)
        canvas.drawCentredString(x+dx/2, y+texty, "r%s g%s b%s"%rgb)
        x = x+dx
    y = y + dy; x = 0
    for gray in (0.0, 0.25, 0.50, 0.75, 1.0):
        canvas.setFillGray(gray)
```

```

canvas.rect(x+rdx, y+rdy, w, h, fill=1)
canvas.setFillColor(black)
canvas.drawCentredString(x+dx/2, y+texty, "gray: %s"%gray)
x = x+dx

```



图 2 - 7 : RGB 颜色模块

RGB颜色透明度

在pdfgen中，可以将对象涂在其他对象上，以达到良好的效果。一般来说，有两种模式可以处理空间中重叠的对象，顶层的默认对象会隐藏掉它下面的其他对象的任何部分。如果你需要透明度，你有两个选择：

1. 如果您的文档打算以专业的方式打印，并且您在CMYK色彩空间中工作，那么您可以使用overPrint。在overPrinting中，颜色会在打印机中物理混合，从而获得一种新的颜色。默认情况下，将应用一个淘汰，只有顶部对象出现。如果您打算使用CMYK，请阅读CMYK部分。
2. 如果您的文档打算用于屏幕输出，并且您使用的是RGB颜色，那么您可以设置一个alpha值，其中alpha是颜色的不透明度值。默认的alpha值是 1（完全不透明），你可以使用任何实数。

Alpha透明度(alpha)类似于overprint，但在RGB色彩空间中工作，下面这个例子演示了alpha功能。请参考我们的网站 <http://www.reportlab.com/snippets/>，并查找overPrint和alpha的片段，以查看生成下面图表的代码。

```

def alpha(canvas):
    from reportlab.graphics.shapes import Rect
    from reportlab.lib.colors import Color, black, blue, red
    red50transparent = Color(100, 0, 0, alpha=0.5)
    c = canvas
    c.setFillColor(black)
    c.setFont('Helvetica', 10)
    c.drawString(25, 180, 'solid')
    c.setFillColor(blue)
    c.rect(25, 25, 100, 100, fill=True, stroke=False)
    c.setFillColor(red)
    c.rect(100, 75, 100, 100, fill=True, stroke=False)
    c.setFillColor(black)
    c.drawString(225, 180, 'transparent')
    c.setFillColor(blue)
    c.rect(225, 25, 100, 100, fill=True, stroke=False)
    c.setFillColor(red50transparent)
    c.rect(300, 75, 100, 100, fill=True, stroke=False)

```

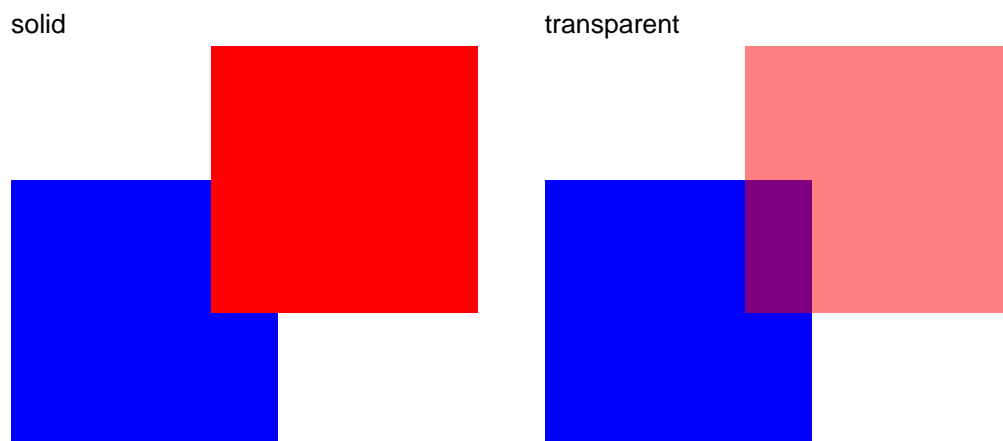


图 2 - 8 : Alpha 例子

CMYK 颜色

CMYK或减法是按照打印机混合三种颜料（青色、品红色和黄色）形成颜色的方式进行的。因为混合化学品比结合光照更难 还有第四个参数是暗度。例如，**CMY**颜料的化学组合通常不会产生完美的黑色--而是产生混浊的颜色--因此，为了得到黑色，打印机不使用**CMY**颜料，而是直接使用黑色墨水。因为**CMYK**更直接地映射到打印机硬件的工作方式，所以在打印时，**CMYK**指定的颜色可能会提供更好的保真度和更好的控制。

CMYK颜色有两种表示方法：每一种颜色都可以用以下方法来表示可以是0到1之间的实值，也可以是0到100之间的整数值。根据您的喜好，您可以使用CMYKColor（对于实值）或PCMYKColor（对于整数值）。0表示 "没有墨水"，所以在白纸上打印会得到白色。1表示 "最大墨水量"(如果使用PCMYKColor，则为100)。例如：CMYKColor(0,0,0,1)是黑色，CMYKColor(0,0,0,0)表示 "没有墨水"。而CMYKColor(0.5,0,0,0)表示50%的青色。

```
def colorsCMYK(canvas):
    from reportlab.lib.colors import CMYKColor, PCMYKColor
    from reportlab.lib.units import inch
    # creates a black CMYK ; CMYKColor use real values
    black = CMYKColor(0,0,0,1)
    # creates a cyan CMYK ; PCMYKColor use integer values
    cyan = PCMYKColor(100,0,0,0)
    y = x = 0; dy=inch*3/4.0; dx=inch*5.5/5; w=h=dy/2; rdx=(dx-w)/2
    rdy=h/5.0; texty=h+2*rdy
    canvas.setFont("Helvetica",10)
    y = y + dy; x = 0
    for cmyk in [(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1), (0,0,0,0)]:
        c,m,y1,k = cmyk
        canvas.setFillColorsCMYK(c,m,y1,k)
        canvas.rect(x+rdx, y+rdy, w, h, fill=1)
        canvas.setFillColors(black)
        canvas.drawCentredString(x+dx/2, y+texty, "c%s m%s y%s k%s"%cmyk)
        x = x+dx
```

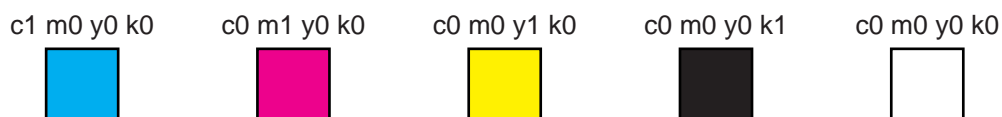


图 2 - 9 : CMYK颜色模型

2.9 色彩空间检查

画布的`enforceColorSpace`参数用于强制执行文档中使用的颜色模型的一致性。它接受这些值。CMYK, RGB, SEP, SEP_BLACK, SEP_CMYK. "SEP"指的是命名的分色, 如Pantone专色--根据使用的参数, 这些颜色可以与CMYK或RGB混合。默认值是'MIXED', 允许你使用任何颜色空间的颜色。如果使用的任何颜色不能转换为指定的模型, 例如`rgb`和`cmYk`, 就会产生一个异常(更多信息请参见`test_pdfgen_general`)。这种方法不检查文档中包含的外部图像。

2.10 彩色套印

当两个CMYK颜色的对象在打印时重叠, 那么"上面"的对象会把下面的对象的颜色打掉, 或者两个对象的颜色会在重叠的区域混合。这种行为可以使用属性`overPrint`来设置。

`overPrint`函数将导致色彩的椭圆区域混合。在下面的例子中, 左边矩形的颜色应该在它们重叠的地方出现混合--如果你看不到这个效果, 那么你可能需要在你的PDF浏览软件中启用"叠印预览"选项。一些PDF浏览器, 如`evince`不支持叠印, 但是Adobe Acrobat Reader支持。

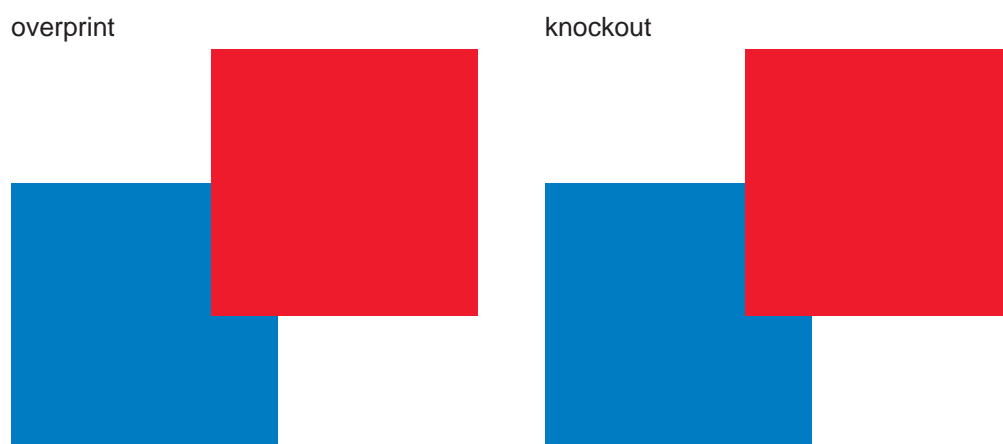


图 2 - 10 : OverPrint示例

其他对象的打印顺序示例

"SPUMONI"字样用白色涂抹在彩色长方形上，有明显的 "去除 "字体内部颜色的效果。

```
def spumoni(canvas):  
    from reportlab.lib.units import inch  
    from reportlab.lib.colors import pink, green, brown, white  
    x = 0; dx = 0.4*inch  
    for i in range(4):  
        for color in (pink, green, brown):  
            canvas.setFillColor(color)  
            canvas.rect(x,0,dx,3*inch,stroke=0,fill=1)  
            x = x+dx  
    canvas.setFillColor(white)  
    canvas.setStrokeColor(white)  
    canvas.setFont("Helvetica-Bold", 85)  
    canvas.drawCentredString(2.75*inch, 1.3*inch, "SPUMONI")
```

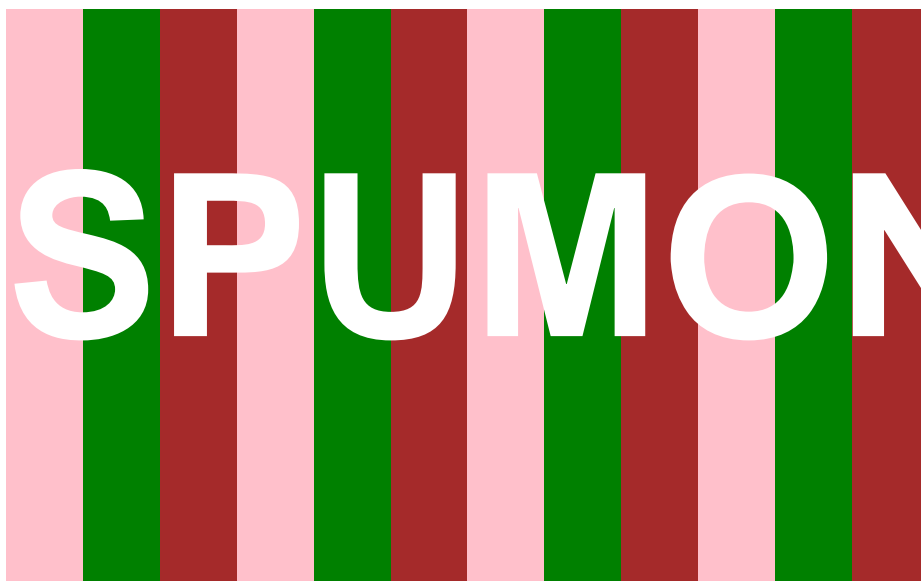


图 2 - 11 : 涂抹颜色

由于默认的`canvas`背景是白色的，在白色背景上画上白色的字母，单词的最后一个字母不可见。

这种分层建立复杂绘画的方法可以在`pdfgen`中完成非常多的层数--比起处理物理颜料时的物理限制要少。

```
def spumoni2(canvas):
    from reportlab.lib.units import inch
    from reportlab.lib.colors import pink, green, brown, white, black
    # draw the previous drawing
    spumoni(canvas)
    # now put an ice cream cone on top of it:
    # first draw a triangle (ice cream cone)
    p = canvas.beginPath()
    xcenter = 2.75*inch
    radius = 0.45*inch
    p.moveTo(xcenter-radius, 1.5*inch)
    p.lineTo(xcenter+radius, 1.5*inch)
    p.lineTo(xcenter, 0)
    canvas.setFill(brown)
    canvas.setStrokeColor(black)
    canvas.drawPath(p, fill=1)
    # draw some circles (scoops)
    y = 1.5*inch
    for color in (pink, green, brown):
        canvas.setFill(color)
        canvas.circle(xcenter, y, radius, fill=1)
        y = y+radius
```

`Spumoni2`函数在`spumoni`图上叠加一个冰淇淋圆锥。
请注意，圆锥和勺子的不同部分也会互相分层。

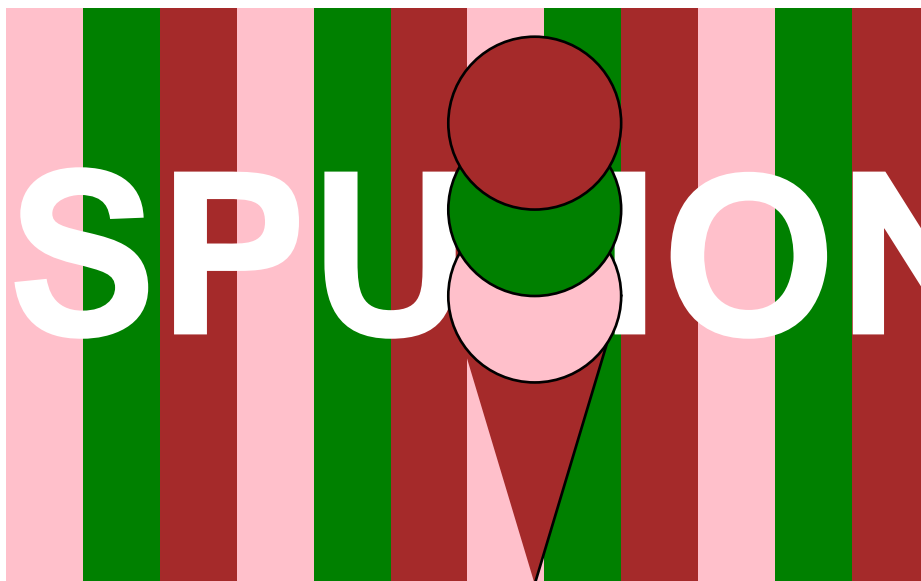


图 2 - 12 : 层层叠加

2.11 标准字体和文本对象

在`pdfgen`中可以用许多不同的颜色、字体和大小来绘制文本。`textsize`函数演示了如何改变文本的颜色、字体和大小，以及如何在页面上放置文本。

```
def textsize(canvas):
    from reportlab.lib.units import inch
    from reportlab.lib.colors import magenta, red
    canvas.setFont("Times-Roman", 20)
    canvas.setFill(red)
    canvas.drawCentredString(2.75*inch, 2.5*inch, "Font size examples")
    canvas.setFill(magenta)
    size = 7
```

```

y = 2.3*inch
x = 1.3*inch
for line in lyrics:
    canvas.setFont("Helvetica", size)
    canvas.drawRightString(x,y,"%s points: " % size)
    canvas.drawString(x,y, line)
    y = y-size*1.2
    size = size+1.5

```

textsize函数生成了以下页面。

Font size examples

7 points: well she hit Net Solutions
 8.5 points: and she registered her own .com site now
 10.0 points: and filled it up with yahoo profile pics
 11.5 points: she snarfed in one night now
 13.0 points: and she made 50 million when Hugh Hefner
 14.5 points: bought up the rights now
 16.0 points: and she'll have fun fun fun
 17.5 points: til her Daddy takes the keyboard away

图 2 - 13 : 不同字体和大小的文字

不同字体和大小的文本在pdfgen中总是有许多不同的字体。

```

def fonts(canvas):
    from reportlab.lib.units import inch
    text = "Now is the time for all good men to..."
    x = 1.8*inch
    y = 2.7*inch
    for font in canvas.getAvailableFonts():
        canvas.setFont(font, 10)
        canvas.drawString(x,y,text)
        canvas.setFont("Helvetica", 10)
        canvas.drawRightString(x-10,y, font+":")
    y = y-13

```

函数fonts列出了始终可用的字体。这些字体不需要存储在PDF文档中，因为它们保证在Acrobat Reader中存在。

```

Courier: Now is the time for all good men to...
Courier-Bold: Now is the time for all good men to...
Courier-BoldOblique: Now is the time for all good men to...
Courier-Oblique: Now is the time for all good men to...
Helvetica: Now is the time for all good men to...
Helvetica-Bold: Now is the time for all good men to...
Helvetica-BoldOblique: Now is the time for all good men to...
Helvetica-Oblique: Now is the time for all good men to...
SourceHanSans-ExtraLight: Now is the time for all good men to...
SourceHanSans-Normal: Now is the time for all good men to...
Symbol: ■■■■ ■■ ■■■■ ■■■■ ■■■■ ■■■■ ■■■■ ■■■■ ■■■■ ■■■■...
Times-Bold: Now is the time for all good men to...
Times-BoldItalic: Now is the time for all good men to...
Times-Italic: Now is the time for all good men to...
Times-Roman: Now is the time for all good men to...
ZapfDingbats: ■■■■ ■■ ■■■■ ■■■■ ■■■■ ■■■■ ■■■■ ■■■■ ■■■■ ■■■■

```

图 2 - 14 : 14种标准字体

Symbol和ZapfDingbats字体无法正确显示，因为这些字体中不存在所需的字形。

有关如何使用任意字体的信息，请参阅下一章。

2.12 文本对象方法

对于PDF文档中文本的专用展示，可以使用文本对象。文本对象接口提供了对文本布局参数的详细控制，而这些参数在`canvas`级别是无法直接获得的。

此外，它还可以生成更小的PDF，其渲染速度比许多单独调用`drawString`方法要快。

```

textobject.setTextOrigin(x,y)

textobject.setTextTransform(a,b,c,d,e,f)

textobject.moveCursor(dx, dy) # from start of current LINE

(x,y) = textobject.getCursor()

x = textobject.getX(); y = textobject.getY()

textobject.setFont(psfontname, size, leading = None)

textobject.textOut(text)

textobject.textLine(text='')

textobject.textLines(stuff, trim=1)

```

上面显示的文本对象方法与基本的文本几何体有关。

文本对象维护一个文本光标，当文本被绘制时，这个光标会在页面上移动。例如，`setTextOrigin`将光标放置在一个已知的位置，而`textLine`和`textLines`方法则将文本光标向下移动，经过缺失的线条。

```

def cursormoves1(canvas):
    from reportlab.lib.units import inch
    textobject = canvas.beginText()
    textobject.setTextOrigin(inch, 2.5*inch)
    textobject.setFont("Helvetica-Oblique", 14)
    for line in lyrics:
        textobject.textLine(line)
    textobject.setFillGray(0.4)

```

```

textobject.textLines("""
With many apologies to the Beach Boys
and anyone else who finds this objectionable
""")
canvas.drawText(textobject)

```

函数 `cursormoves` 依靠文本光标的自动移动来放置原点后的文本。

*well she hit Net Solutions
and she registered her own .com site now
and filled it up with yahoo profile pics
she snarfed in one night now
and she made 50 million when Hugh Hefner
bought up the rights now
and she'll have fun fun fun
til her Daddy takes the keyboard away
With many apologies to the Beach Boys
and anyone else who finds this objectionable*

图 2 - 15 : 文本光标的移动方式

也可以通过使用 `moveCursor` 方法更明确地控制光标的移动（该方法将光标移动为从当前线开始的偏移量，而不是当前光标，并且该方法还具有正 `y` 偏移量移动向下）与正常几何形状相反，正 `y` 通常向上移动。

```

def cursormoves2(canvas):
    from reportlab.lib.units import inch
    textobject = canvas.beginPath()
    textobject.setTextOrigin(2, 2.5*inch)
    textobject.setFont("Helvetica-Oblique", 14)
    for line in lyrics:
        textobject.textOut(line)
        textobject.moveCursor(14,14) # POSITIVE Y moves down!!!
    textobject.setFillColors(0.4,0,1)
    textobject.textLines("""
With many apologies to the Beach Boys
and anyone else who finds this objectionable
""")
    canvas.drawText(textobject)

```

在这里，`textOut` 不会向下移动一行，而 `textLine` 函数则向下移动。

*well she hit Net Solutions
 and she registered her own .com site now
 and filled it up with yahoo profile pics
 she snarfed in one night now
 and she made 50 million when Hugh Hefner
 bought up the rights now
 and she'll have fun fun fun
 til her Daddy takes the keyboard away
 With many apologies to the Beach Boys
 and anyone else who finds this objectionable*

图 2 - 16 : 文本光标如何再次移动

字符间距

```
textobject.setCharSpace(charSpace)
```

setCharSpace方法调整文本的一个参数--字符间的间距。

```
def charspace(canvas):
    from reportlab.lib.units import inch
    textobject = canvas.beginText()
    textobject.setTextOrigin(3, 2.5*inch)
    textobject.setFont("Helvetica-Oblique", 10)
    charspace = 0
    for line in lyrics:
        textobject.setCharSpace(charspace)
        textobject.textLine("%s: %s" %(charspace,line))
        charspace = charspace+0.5
    textobject.setFillGray(0.4)
    textobject.textLines("""
    With many apologies to the Beach Boys
    and anyone else who finds this objectionable
    """)
    canvas.drawText(textobject)
```

charspace函数行使各种间距设置。它产生以下页面。

*0: well she hit Net Solutions
 0.5: and she registered her own .com site now
 1.0: and filled it up with yahoo profile pics
 1.5: she snarfed in one night now
 2.0: and she made 50 million when Hugh Hefner
 2.5: bought up the rights now
 3.0: and she'll have fun fun fun
 3.5: til her Daddy takes the keyboard away
 With many apologies to the Beach Boys
 and anyone else who finds this objectionable*

图 2 - 17 : 调整字符间距

字距

`textobject.setWordSpace(wordSpace)`

`setWordSpace`方法调整字与字之间的空间。

```

def wordspace(canvas):
    from reportlab.lib.units import inch
    textobject = canvas.beginText()
    textobject.setTextOrigin(3, 2.5*inch)
    textobject.setFont("Helvetica-Oblique", 12)
    wordspace = 0
    for line in lyrics:
        textobject.setWordSpace(wordspace)
        textobject.textLine("%s: %s" %(wordspace,line))
        wordspace = wordspace+2.5
    textobject.setFillColors(CMYK(0.4,0,0.4,0.2))
    textobject.textLines("""
    With many apologies to the Beach Boys
    and anyone else who finds this objectionable
    """)
    canvas.drawText(textobject)
  
```

`wordspace`函数显示了下面各种文字空间设置的样子。

0: well she hit Net Solutions
 2.5: and she registered her own .com site now
 5.0: and filled it up with yahoo profile pics
 7.5: she snarfed in one night now
 10.0: and she made 50 million when Hugh Hefner
 12.5: bought up the rights now
 15.0: and she'll have fun fun fun
 17.5: til her Daddy takes the keyboard away
 With many apologies to the Beach Boys
 and anyone else who finds this objectionable

图 2 - 18 : 调整字距

水平缩放

```
textobject.setHorizScale(horizScale)
```

文本行可以通过`setHorizScale`方法进行水平拉伸或收缩。

```

def horizontalscale(canvas):
    from reportlab.lib.units import inch
    textobject = canvas.beginText()
    textobject.setTextOrigin(3, 2.5*inch)
    textobject.setFont("Helvetica-Oblique", 12)
    horizontalscale = 80 # 100 is default
    for line in lyrics:
        textobject.setHorizScale(horizontalscale)
        textobject.textLine("%s: %s" %(horizontalscale,line))
        horizontalscale = horizontalscale+10
    textobject.setFillColorsCMYK(0.0,0.4,0.4,0.2)
    textobject.textLines("""
    With many apologies to the Beach Boys
    and anyone else who finds this objectionable
    """)
    canvas.drawText(textobject)
  
```

水平缩放参数`horizScale`是以百分比的形式给出的（默认为100），所以下图所示的80设置看起来很瘦。

80: well she hit Net Solutions
 90: and she registered her own .com site now
 100: and filled it up with yahoo profile pics
 110: she snarfed in one night now
 120: and she made 50 million when Hugh Hefner
 130: bought up the rights now
 140: and she'll have fun fun fun
 150: til her Daddy takes the keyboard away
 With many apologies to the Beach Boys
 and anyone else who finds this objectionable

图 2 - 19 : 调整水平文本的比例

行间间距(领先)

```
textobject.setLeading(leading)
```

一条线的起始点和下一条线的起始点之间的垂直偏移称为前导偏移。
`setLeading`方法调整前导偏移。

```

def leading(canvas):
    from reportlab.lib.units import inch
    textobject = canvas.beginText()
    textobject.setTextOrigin(3, 2.5*inch)
    textobject.setFont("Helvetica-Oblique", 14)
    leading = 8
    for line in lyrics:
        textobject.setLeading(leading)
        textobject.textLine("%s: %s" %(leading,line))
        leading = leading+2.5
    textobject.setFill_color(CMYK(0.8,0,0,0.3))
    textobject.textLines("""
    With many apologies to the Beach Boys
    and anyone else who finds this objectionable
    """)
    canvas.drawText(textobject)
  
```

如下图所示，如果一行的前导偏移量设置得太小，我就会把前一行中的字符的底部部分写在上面。

8: well she hit Net Solutions
 10.5: and she registered her own .com site now
 13.0: and filled it up with yahoo profile pics
 15.5: she snarfed in one night now
 18.0: and she made 50 million when Hugh Hefner
 20.5: bought up the rights now
 23.0: and she'll have fun fun fun
 25.5: til her Daddy takes the keyboard away
 With many apologies to the Beach Boys
 and anyone else who finds this objectionable

图 2 - 20 : 矫枉过正

其他文本对象方法

```
textobject.setTextRenderMode(mode)
```

例如，`setTextRenderMode`方法允许将文本作为剪裁背景图的前景。

```
textobject.setRise(rise)
```

`setRise`方法^{提高或降低}行上的文本（例如，用于创建上标或下标）。

```

textobject.setFill(aColor);
textobject.setStrokeColor(self, aColor)
# and similar

```

这些颜色变化操作会改变文本的颜色，其他方面与`canvas`对象的颜色方法类似。

2.13 路径和直线

正如文本对象被设计成专门用于展示文本一样，路径对象被设计成专门用于构建图形。当路径对象被绘制到`canvas`上时，它们被绘制成一个图形（就像一个矩形），整个图形的绘制模式可以调整：图形的线条可以绘制（笔画），也可以不绘制；图形的内部可以填充，也可以不填充；等等。

例如，`star`函数使用一个路径对象来绘制一个星体

```

def star(canvas, title="Title Here", aka="Comment here.",
        xcenter=None, ycenter=None, nvertices=5):
    from math import pi
    from reportlab.lib.units import inch
    radius=inch/3.0
    if xcenter is None: xcenter=2.75*inch
    if ycenter is None: ycenter=1.5*inch
    canvas.drawCentredString(xcenter, ycenter+1.3*radius, title)
    canvas.drawCentredString(xcenter, ycenter-1.4*radius, aka)
    p = canvas.beginPath()
    p.moveTo(xcenter,ycenter+radius)
    from math import pi, cos, sin
    angle = (2*pi)*2/5.0
    startangle = pi/2.0
    for vertex in range(nvertices-1):
        nextangle = angle*(vertex+1)+startangle
        x = xcenter + radius*cos(nextangle)
        y = ycenter + radius*sin(nextangle)

```

```

p.lineTo(x,y)
if nvertices==5:
    p.close()
canvas.drawPath(p)

```

`star`函数被设计用来说明pdfgen支持的各种行式参数。



图 2 - 21 : 直线样式参数

直线连接设置

通过`setLineJoin`方法，可以调整线段在一个点上相接的是方块还是圆角顶点。

```

def joins(canvas):
    from reportlab.lib.units import inch
    # make lines big
    canvas.setLineWidth(5)
    star(canvas, "Default: mitered join", "0: pointed", xcenter = 1*inch)
    canvas.setLineJoin(1)
    star(canvas, "Round join", "1: rounded")
    canvas.setLineJoin(2)
    star(canvas, "Bevelled join", "2: square", xcenter=4.5*inch)

```

线条连接的设置只有对粗线条才真正有意义，因为对细线条看不清楚。



图 2 - 22 : 不同的直线连接样式

直线帽子设置

使用 `setLineCap` 方法调整的线帽设置，决定了终止线的终点是在顶点处的正方形、顶点上方的正方形还是顶点上方的半圆。

```
def caps(canvas):
    from reportlab.lib.units import inch
    # make lines big
    canvas.setLineWidth(5)
    star(canvas, "Default", "no projection", xcenter = 1*inch,
          nvertices=4)
    canvas.setLineCap(1)
    star(canvas, "Round cap", "1: ends in half circle", nvertices=4)
    canvas.setLineCap(2)
    star(canvas, "Square cap", "2: projects out half a width", xcenter=4.5*inch,
          nvertices=4)
```

线帽设置和线条连接设置一样，只有在线条较粗时才会清晰可见。

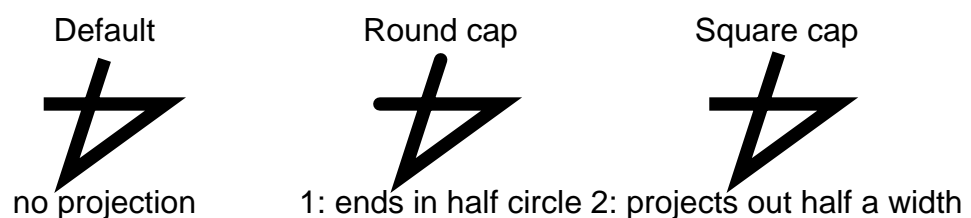


图 2 - 23 : 直线帽子设置

破折号和断线

用`setDash`方法可以将线条分成点或破折号。

```
def dashes(canvas):
    from reportlab.lib.units import inch
    # make lines big
    canvas.setDash(6,3)
    star(canvas, "Simple dashes", "6 points on, 3 off", xcenter = 1*inch)
    canvas.setDash(1,2)
    star(canvas, "Dots", "One on, two off")
    canvas.setDash([1,1,3,3,1,4,4,1], 0)
    star(canvas, "Complex Pattern", "[1,1,3,3,1,4,4,1]", xcenter=4.5*inch)
```

虚线或圆点的图案可以是简单的开/关重复图案，也可以指定为复杂的重复图案。

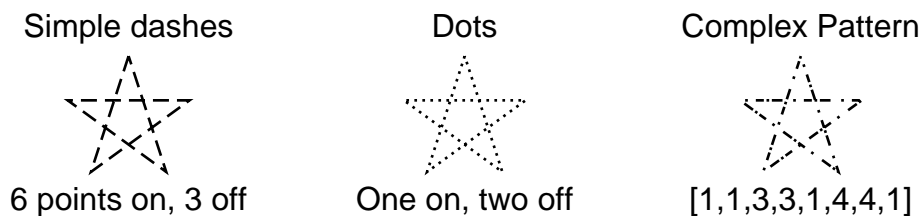


图 2 - 24 : 破折号

用路径对象创建复杂的图形

线条、曲线、弧线等图形的组合可以使用路径对象组合成一个图形。例如下图所示的函数就是利用直线和曲线构造两个路径对象。这个函数将在后面的铅笔图标构造中使用。

```
def penciltip(canvas, debug=1):
    from reportlab.lib.colors import tan, black, green
    from reportlab.lib.units import inch
    u = inch/10.0
    canvas.setLineWidth(4)
    if debug:
        canvas.scale(2.8,2.8) # make it big
        canvas.setLineWidth(1) # small lines
    canvas.setStrokeColor(black)
    canvas.setFillColor(tan)
    p = canvas.beginPath()
    p.moveTo(10*u,0)
    p.lineTo(0,5*u)
    p.lineTo(10*u,10*u)
    p.curveTo(11.5*u,10*u, 11.5*u,7.5*u, 10*u,7.5*u)
    p.curveTo(12*u,7.5*u, 11*u,2.5*u, 9.7*u,2.5*u)
    p.curveTo(10.5*u,2.5*u, 11*u,0, 10*u,0)
    canvas.drawPath(p, stroke=1, fill=1)
    canvas.setFillColor(black)
    p = canvas.beginPath()
    p.moveTo(0,5*u)
    p.lineTo(4*u,3*u)
    p.lineTo(5*u,4.5*u)
    p.lineTo(3*u,6.5*u)
```

```

canvas.drawPath(p, stroke=1, fill=1)
if debug:
    canvas.setStrokeColor(green) # put in a frame of reference
    canvas.grid([0,5*u,10*u,15*u], [0,5*u,10*u])

```

请注意，铅笔头的内部是作为一个对象填充的，即使它是由几条线和曲线构成的。然后使用一个新的路径对象在其上绘制铅笔头。

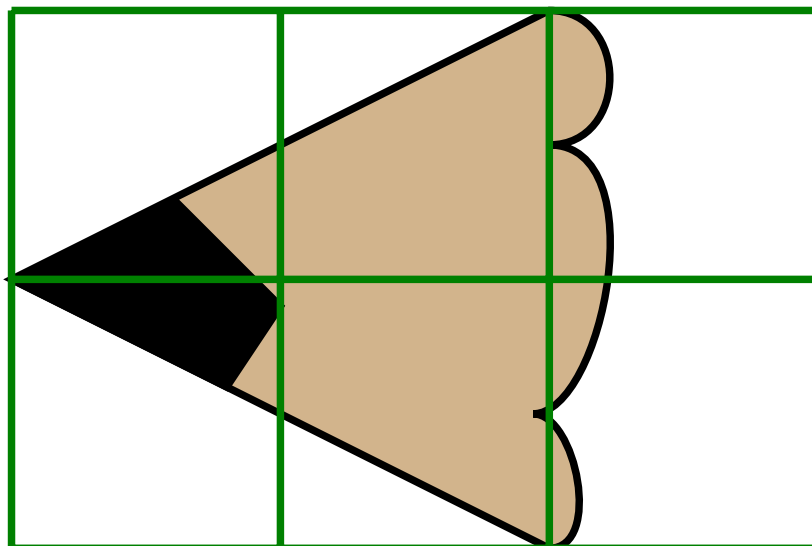


图 2 - 25 : 铅笔头

2.14 矩形、圆形、椭圆形。

pdfgen模块支持许多一般有用的形状，如矩形、圆角矩形、椭圆和圆。这些图形中的每一个都可以在路径对象中使用，也可以直接在canvas上绘制。例如下面的pencil函数使用矩形和圆角矩形绘制了一个铅笔图标，并添加了各种填充颜色和其他一些注释。

```

def pencil(canvas, text="No.2"):
    from reportlab.lib.colors import yellow, red, black, white
    from reportlab.lib.units import inch
    u = inch/10.0
    canvas.setStrokeColor(black)
    canvas.setLineWidth(4)
    # draw eraser
    canvas.setFillColor(red)
    canvas.circle(30*u, 5*u, 5*u, stroke=1, fill=1)
    # draw all else but the tip (mainly rectangles with different fills)
    canvas.setFillColor(yellow)
    canvas.rect(10*u, 0, 20*u, 10*u, stroke=1, fill=1)
    canvas.setFillColor(black)
    canvas.rect(23*u, 0, 8*u, 10*u, fill=1)
    canvas.roundRect(14*u, 3.5*u, 8*u, 3*u, 1.5*u, stroke=1, fill=1)
    canvas.setFillColor(white)
    canvas.rect(25*u, u, 1.2*u, 8*u, fill=1, stroke=0)
    canvas.rect(27.5*u, u, 1.2*u, 8*u, fill=1, stroke=0)
    canvas.setFont("Times-Roman", 3*u)
    canvas.drawCentredString(18*u, 4*u, text)
    # now draw the tip
    penciltip(canvas, debug=0)
    # draw broken lines across the body.
    canvas.setDash([10, 5, 16, 10], 0)
    canvas.line(11*u, 2.5*u, 22*u, 2.5*u)
    canvas.line(22*u, 7.5*u, 12*u, 7.5*u)

```



注意：这个函数是用来创建左边的'边距铅笔'的。还要注意的，元素的绘制顺序很重要，因为，例如，白色矩形'擦掉'了黑色矩形的一部分，而'笔尖'则涂抹了黄色矩形的一部分。

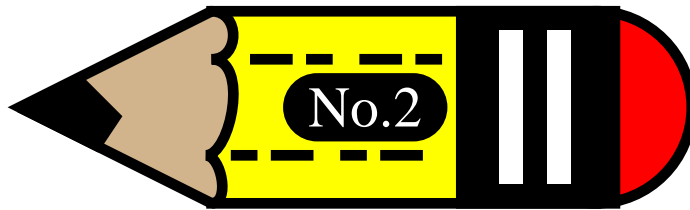


图 2 - 26 : 铅笔

2.15 贝兹尔曲线

想要构造具有弯曲边界的图形的程序，一般使用贝塞尔曲线来形成边界。

```
def bezier(canvas):
    from reportlab.lib.colors import yellow, green, red, black
    from reportlab.lib.units import inch
    i = inch
    d = i/4
    # define the bezier curve control points
    x1,y1, x2,y2, x3,y3, x4,y4 = d,1.5*i, 1.5*i,d, 3*i,d, 5.5*i-d,3*i-d
    # draw a figure enclosing the control points
    canvas.setFillColor(yellow)
    p = canvas.beginPath()
    p.moveTo(x1,y1)
    for (x,y) in [(x2,y2), (x3,y3), (x4,y4)]:
        p.lineTo(x,y)
    canvas.drawPath(p, fill=1, stroke=0)
    # draw the tangent lines
    canvas.setLineWidth(inch*0.1)
    canvas.setStrokeColor(green)
    canvas.line(x1,y1,x2,y2)
    canvas.setStrokeColor(red)
    canvas.line(x3,y3,x4,y4)
    # finally draw the curve
    canvas.setStrokeColor(black)
    canvas.bezier(x1,y1, x2,y2, x3,y3, x4,y4)
```

Bezier曲线由四个控制点 (x_1,y_1) ， (x_2,y_2) ， (x_3,y_3) ， (x_4,y_4) 指定。曲线起于 (x_1,y_1) ，止于 (x_4,y_4) ，从 (x_1,y_1) 到 (x_2,y_2) 的线段和从 (x_3,y_3) 到 (x_4,y_4) 的线段都与曲线形成切线。而且曲线完全包含在凸图形中，顶点在控制点上。



图 2 - 27 : 基本贝塞尔曲线

上图(testbezier的输出)显示了一个bezier曲线、控制点定义的切线和控制点处有顶点的凸图形。

平滑地连接贝塞尔曲线序列

通常情况下，将几条贝塞尔曲线连接成一条平滑曲线是很有用的。要想从几条贝塞尔曲线中构造一条较大的平滑曲线，请确保相邻贝塞尔曲线在控制点连接的切线位于同一直线上。

```
def bezier2(canvas):
    from reportlab.lib.colors import yellow, green, red, black
    from reportlab.lib.units import inch
    # make a sequence of control points
    xd,yd = 5.5*inch/2, 3*inch/2
    xc,yc = xd,yd
    dx,dy = [(0,0.33), (0.33,0.33), (0.75,1), (0.875,0.875),
              (0.875,0.875), (1,0.75), (0.33,0.33), (0.33,0)]
    pointlist = []
    for xoffset in (1,-1):
        yoffset = xoffset
        for (dx,dy) in dx,dy:
            px = xc + xd*xoffset*dx
            py = yc + yd*yoffset*dy
            pointlist.append((px,py))
        yoffset = -xoffset
        for (dy,dx) in dx,dy:
            px = xc + xd*xoffset*dx
            py = yc + yd*yoffset*dy
            pointlist.append((px,py))
    # draw tangent lines and curves
    canvas.setLineWidth(inch*0.1)
    while pointlist:
        [(x1,y1),(x2,y2),(x3,y3),(x4,y4)] = pointlist[:4]
        del pointlist[:4]
        canvas.setLineWidth(inch*0.1)
        canvas.setStrokeColor(green)
        canvas.line(x1,y1,x2,y2)
        canvas.setStrokeColor(red)
        canvas.line(x3,y3,x4,y4)
        # finally draw the curve
        canvas.setStrokeColor(black)
        canvas.bezier(x1,y1, x2,y2, x3,y3, x4,y4)
```

由testbezier2创建的图形描述了一条平滑的复曲线，因为相邻的切线 "排队"，如下图所示。

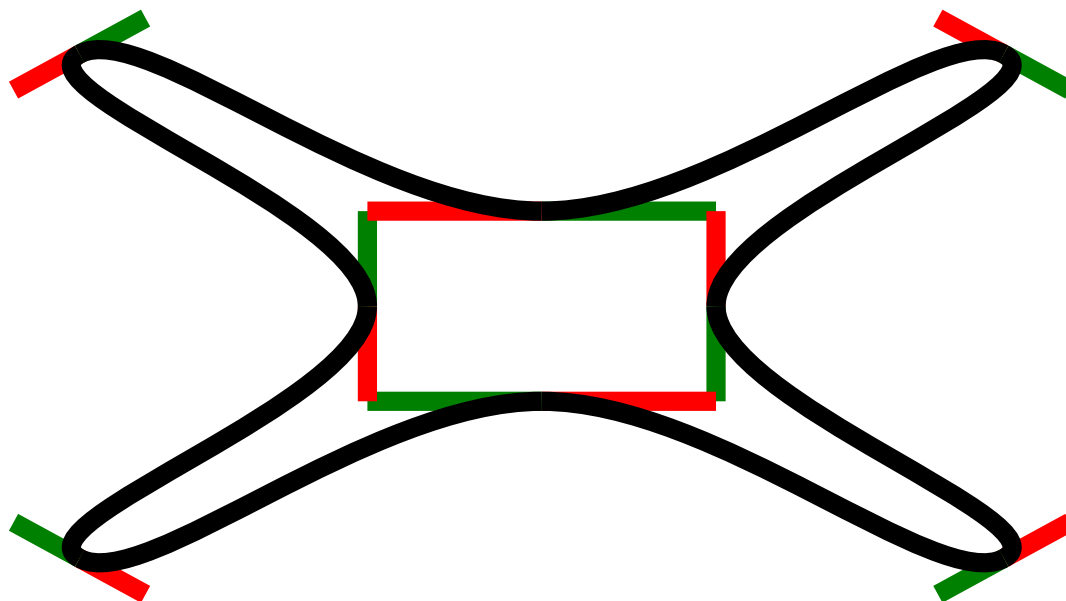


图 2 - 28 : bezier curves

2.16 路径对象方法

路径对象通过在画布上的起始点设置 "笔" 或 "画笔", 并在画布上的附加点上绘制线条或曲线, 从而建立复杂的图形。大多数操作都是从上一次操作的终点开始在画布上涂抹颜料, 并在新的终点留下画笔。

```
pathobject.moveTo(x,y)
```

`moveTo` 方法抬起画笔 (结束任何当前的线条或曲线序列 (如果有的话)), 并在画布上新的 (x,y) 位置替换画笔, 开始一个新的路径序列。

```
pathobject.lineTo(x,y)
```

`lineTo` 方法从当前笔刷位置到新的 (x,y) 位置绘制直线段。

```
pathobject.curveTo(x1, y1, x2, y2, x3, y3)
```

`curveTo` 方法从当前画笔位置开始绘制一条贝塞尔曲线, 使用 $(x1,y1)$ 、 $(x2,y2)$ 和 $(x3,y3)$ 作为其他三个控制点, 将画笔留在 $(x3,y3)$ 上。

```
pathobject.arc(x1,y1, x2,y2, startAng=0, extent=90)
```

```
pathobject.arcTo(x1,y1, x2,y2, startAng=0, extent=90)
```

`arc` 和 `arcTo` 方法可以绘制部分椭圆。`arc` 方法首先 "提起画笔" 并开始一个新的形状序列。而 `arcTo` 方法则是将部分椭圆的起始点与当前的形状序列用线连接起来。在画部分椭圆之前, 先画出部分椭圆的线段。点 $(x1,y1)$ 和 $(x2,y2)$ 定义包围椭圆的矩形的相对角点。`startAng` 是一个角度 (度数), 指定了部分椭圆的开始位置, 其中 0 角是右边界的 midpoint。围成的矩形 (当 $(x1,y1)$ 为左下角, $(x2,y2)$ 为右上角)。 `extent` 是指与椭圆上的横移。

```
def arcs(canvas):
    from reportlab.lib.units import inch
    canvas.setLineWidth(4)
    canvas.setStrokeColorRGB(0.8, 1, 0.6)
    # draw rectangles enclosing the arcs
    canvas.rect(inch, inch, 1.5*inch, inch)
    canvas.rect(3*inch, inch, inch, 1.5*inch)
    canvas.setStrokeColorRGB(0, 0.2, 0.4)
    canvas.setFill-colorRGB(1, 0.6, 0.8)
    p = canvas.beginPath()
    p.moveTo(0.2*inch, 0.2*inch)
```

```
p.arcTo(inch, inch, 2.5*inch, 2*inch, startAng=-30, extent=135)
p.arc(3*inch, inch, 4*inch, 2.5*inch, startAng=-45, extent=270)
canvas.drawPath(p, fill=1, stroke=1)
```

上面的`arcs`函数行使了两种局部椭圆方法。它产生了下面的图形。

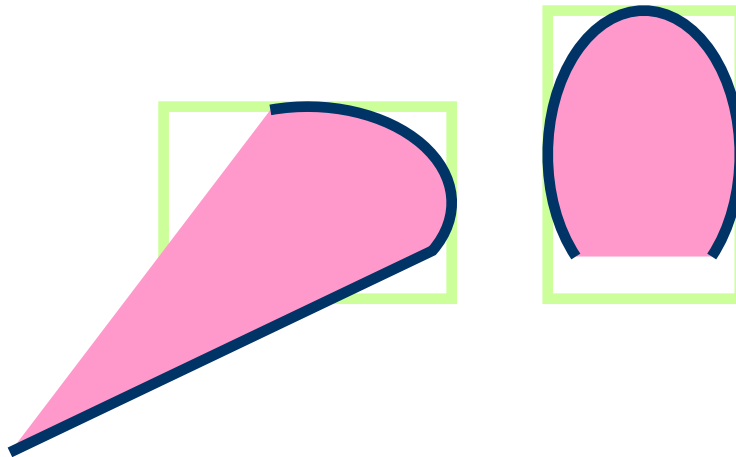


图 2 - 29 : 弧线

```
pathobject.rect(x, y, width, height)
```

`rect`方法在 (x, y) 指定的 `width` 和 `height` 处画一个左下角的矩形。

```
pathobject.ellipse(x, y, width, height)
```

`ellipse`方法在指定的 `width` 和 `height` 的 (x, y) 处绘制一个左下角的矩形包围的椭圆。

```
pathobject.circle(x_cen, y_cen, r)
```

`circle`方法画一个以 (x_cen, y_cen) 为中心，半径 `r` 的圆。

```
def variousshapes(canvas):
    from reportlab.lib.units import inch
    inch = int(inch)
    canvas.setStrokeGray(0.5)
    canvas.grid(range(0, int(11*inch/2), int(inch/2)), range(0, int(7*inch/2),
    int(inch/2)))
    canvas.setLineWidth(4)
    canvas.setStrokeColorRGB(0, 0.2, 0.7)
    canvas.setFillColorsRGB(1, 0.6, 0.8)
    p = canvas.beginPath()
    p.rect(0.5*inch, 0.5*inch, 0.5*inch, 2*inch)
    p.circle(2.75*inch, 1.5*inch, 0.3*inch)
    p.ellipse(3.5*inch, 0.5*inch, 1.2*inch, 2*inch)
    canvas.drawPath(p, fill=1, stroke=1)
```

上面的`variousshapes`函数显示了一个放置在参考网格中的矩形、圆和椭圆。

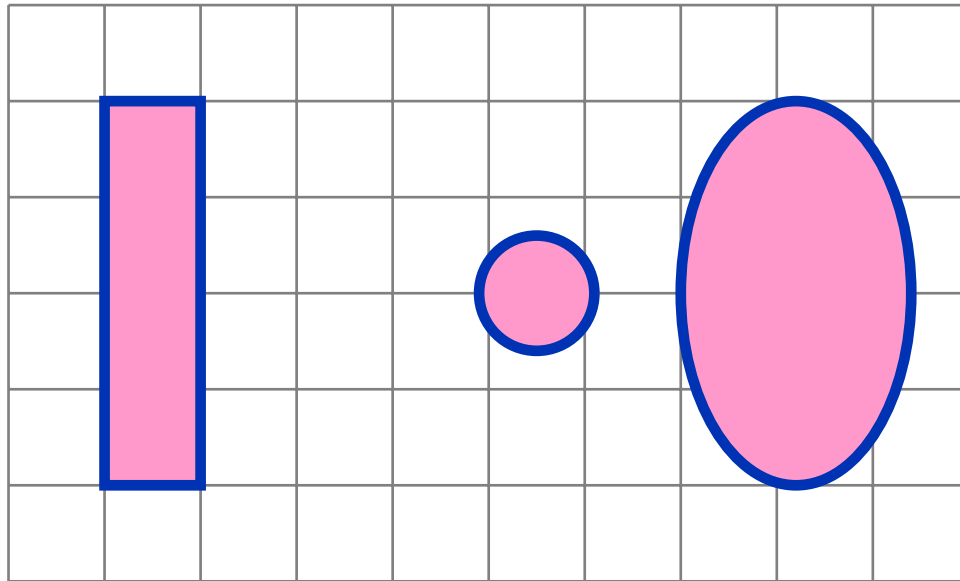


图 2 - 30 : 路径对象中的矩形、圆形、椭圆形。

`pathobject.close()`

`close`方法通过从图形的最后一点到图形的起始点(最近一次通过`moveTo`或`arc`或其他放置操作将画笔放置在纸上的点)画一条线段来关闭当前图形。

```
def closingfigures(canvas):
    from reportlab.lib.units import inch
    h = inch/3.0; k = inch/2.0
    canvas.setStrokeColorRGB(0.2,0.3,0.5)
    canvas.setFillColorsRGB(0.8,0.6,0.2)
    canvas.setLineWidth(4)
    p = canvas.beginPath()
    for i in (1,2,3,4):
        for j in (1,2):
            xc,yc = inch*i, inch*j
            p.moveTo(xc,yc)
            p.arcTo(xc-h, yc-k, xc+h, yc+k, startAng=0, extent=60*i)
            # close only the first one, not the second one
            if j==1:
                p.close()
    canvas.drawPath(p, fill=1, stroke=1)
```

`closingfigures`函数说明了闭合或不闭合图形的效果，包括一条线段和一个部分椭圆。

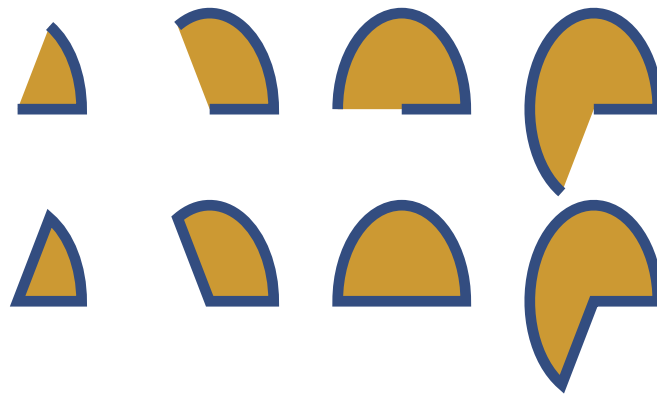


图 2 - 31 : 闭合和不闭合的路径对象数字

关闭或不关闭图形只影响图形的描边轮廓，而不影响图形的填充，如上图所示。

关于使用路径对象绘图的更广泛的例子，请检查**hand**函数。

```
def hand(canvas, debug=1, fill=0):
    (startx, starty) = (0,0)
    curves = [
        (0, 2), (0, 4), (0, 8), # back of hand
        (5, 8), (7, 10), (7, 14),
        (10, 14), (10, 13), (7.5, 8), # thumb
        (13, 8), (14, 8), (17, 8),
        (19, 8), (19, 6), (17, 6),
        (15, 6), (13, 6), (11, 6), # index, pointing
        (12, 6), (13, 6), (14, 6),
        (16, 6), (16, 4), (14, 4),
        (13, 4), (12, 4), (11, 4), # middle
        (11.5, 4), (12, 4), (13, 4),
        (15, 4), (15, 2), (13, 2),
        (12.5, 2), (11.5, 2), (11, 2), # ring
        (11.5, 2), (12, 2), (12.5, 2),
        (14, 2), (14, 0), (12.5, 0),
        (10, 0), (8, 0), (6, 0), # pinky, then close
    ]
    from reportlab.lib.units import inch
    if debug: canvas.setLineWidth(6)
    u = inch*0.2
    p = canvas.beginPath()
    p.moveTo(startx, starty)
    ccopy = list(curves)
    while ccopy:
        [(x1,y1), (x2,y2), (x3,y3)] = ccopy[:3]
        del ccopy[:3]
        p.curveTo(x1*u, y1*u, x2*u, y2*u, x3*u, y3*u)
    p.close()
    canvas.drawPath(p, fill=fill)
    if debug:
        from reportlab.lib.colors import red, green
        (lastx, lasty) = (startx, starty)
        ccopy = list(curves)
        while ccopy:
            [(x1,y1), (x2,y2), (x3,y3)] = ccopy[:3]
            del ccopy[:3]
            canvas.setStrokeColor(red)
            canvas.line(lastx*u, lasty*u, x1*u, y1*u)
            canvas.setStrokeColor(green)
            canvas.line(x2*u, y2*u, x3*u, y3*u)
            (lastx, lasty) = (x3, y3)
```

在调试模式下(默认)，`hand`函数显示了用于构成图形的贝塞尔曲线的切线段。请注意，当线段对齐时，曲线平滑地连接在一起，但当线段不对齐时，曲线显示出一个"尖锐的边缘"。

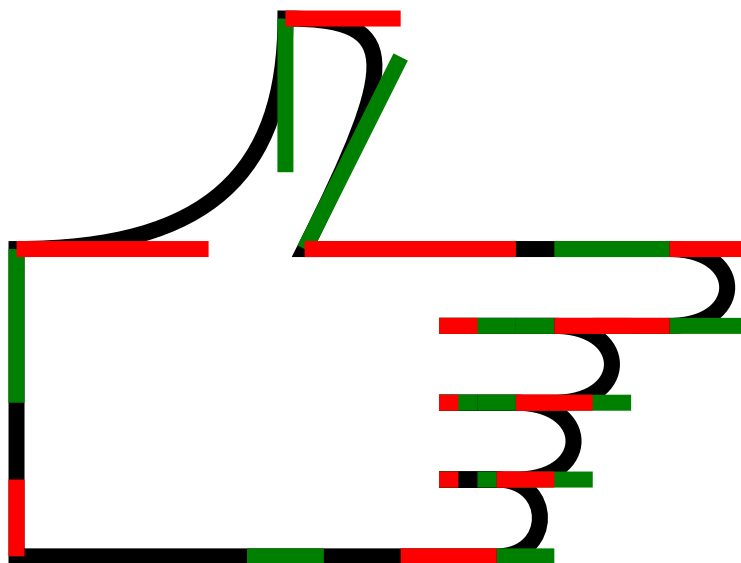


图 2 - 32 : 手形图

在非调试模式下，`hand`函数只显示贝塞尔曲线。如果设置了`fill`参数，则会使用当前的填充颜色填充图形。

```
def hand2(canvas):
    canvas.translate(20,10)
    canvas.setLineWidth(3)
    canvas.setFillColorsRGB(0.1, 0.3, 0.9)
    canvas.setStrokeGray(0.5)
    hand(canvas, debug=0, fill=1)
```

注意：边框的 "描边" 画在它们重叠的内部填充物上。

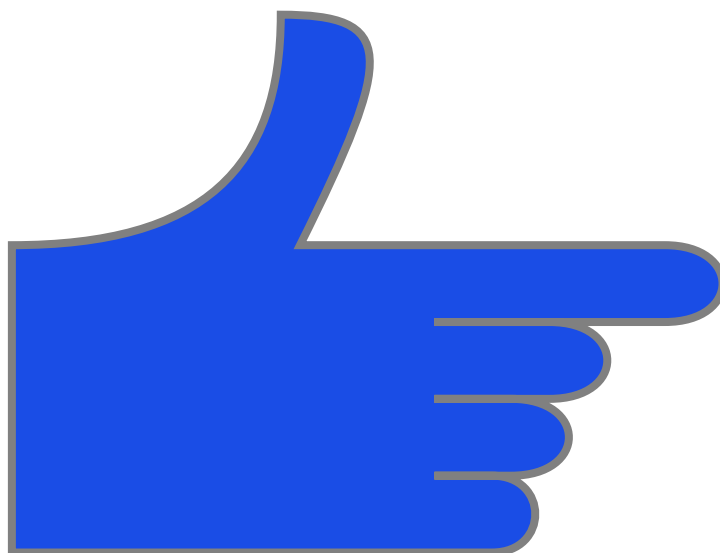


图 2 - 33 : 妙手回春

2.17 进一步阅读: ReportLab图形库

到目前为止，我们所看到的图形是在相当低的水平上创建的。但应该注意的是，还有一种方法可以使用专用的高级ReportLab图形库创建更复杂的图形。

它可以用来为不同的输出格式（矢量图和位图）如PDF、EPS、SVG、JPG和PNG等制作高质量的、独立于平台的、可重复使用的图形。

本文档第11章绘图对其理念和功能进行了更详尽的描述，其中包含了现有组件的信息以及如何创建自定义组件。

第11章还详细介绍了ReportLab图表包及其组件（标签、坐标轴、图例和不同类型的图表，如柱状图、折线图和饼状图），它直接建立在图形库上。

第3章 字体和编码

本章包括字体、编码和亚洲语言功能。如果你只关心生成西欧语言的PDF，你可以只读下面的"Unicode是默认的"部分，并在第一次阅读时跳过其余部分。我们希望随着时间的推移，这部分内容会有很大的增长。我们希望开源能让我们比其他工具更好地支持世界上更多的语言，我们欢迎在这方面的反馈和帮助。

3.1 Unicode 和 UTF8 作为默认编码

从reportlab 2.0版本(2006年5月)开始，您提供给我们API的所有文本输入都应该是UTF8或Python Unicode对象。这适用于`canvas.drawString`和相关API的参数、表格单元格内容、绘图对象参数和段落源文本。

我们曾考虑过让输入编码可配置，甚至依赖于本地，但决定"显式比隐式好"。

这简化了我们以前做的许多关于希腊字母、符号等的事情。要显示任何字符，找出它的unicode码点，并确保你使用的字体能够显示它。

如果您正在改编ReportLab 1.x应用程序，或者从其他包含单字节数据的源头读取数据(例如 latin-1 或 WinAnsi)，您需要进行Unicode转换。Python编解码器包现在包含了所有常用编码的转换器，包括亚洲的编码。

如果你的数据不是UTF8编码，那么一旦你输入一个非ASCII字符，你就会得到一个UnicodeDecodeError。例如，下面这个代码段试图读取并打印一系列名字，包括一个带有法国口音的名字。*Marc -André Lemburg*。标准误差非常有用，它告诉你它不喜欢什么字符。

```
>>> from reportlab.pdfgen.canvas import Canvas
>>> c = Canvas('temp.pdf')
>>> y = 700
>>> for line in file('latin_python_gurus.txt','r'):
...     c.drawString(100, y, line.strip())
...
Traceback (most recent call last):
...
UnicodeDecodeError: 'utf8' codec can't decode bytes in position 9-11: invalid
data
-->é L--emburg
>>>
```

最简单的解决方法就是将你的数据转换为unicode，并说明它来自哪个编码，就像这样。

```
>>> for line in file('latin_input.txt','r'):
...     uniLine = unicode(line, 'latin-1')
...     c.drawString(100, y, uniLine.strip())
>>>
>>> c.save()
```

3.2 输出字体自动替换

在代码中还有很多地方，包括`rl_config.defaultEncoding`参数，以及传递给各种Font构造函数的参数，这些参数都是指编码。在过去，当人们需要使用PDF浏览设备支持的Symbol和ZapfDingbats字体的字形时，这些参数非常有用。默认情况下，标准字体（Helvetica、Courier、Times Roman）将提供Latin-1的字形。然而，如果我们的引擎检测到一个字体中没有的字符，它将尝试切换到Symbol或ZapfDingbats来显示这些字符。例如，如果你在对`drawString`的调用中包含了一对右面剪刀的Unicode字符，`\u2702(✂)`，你应该会看到它们(在`test_pdfgen_general.py/pdf`中有一个例子)。在你的代码中不需要切换字体。

3.3 使用非标准的 Type 1 字体

正如前一章所讨论的那样，每份Acrobat Reader都内置了14种标准字体。因此，ReportLab PDF库只需要通过名称来引用这些字体。如果您想使用其他字体，它们必须对您的代码可用，并将被嵌入到PDF文档中。

您可以使用下面描述的机制在您的文档中包含任意字体。我们有一个名为`DarkGardenMK`的开源字体，我们可以将其用于测试和或文档目的(您也可以使用它)。它与ReportLab发行版捆绑在一起，在`reportlab/fonts`目录下。

目前，字体嵌入依赖于Adobe AFM("Adobe Font Metrics")和PFB("Printer Font Binary")格式的字体描述文件。前者是一个ASCII文件，包含字体中的字符("字形")信息，如高度、宽度、边界框信息和其他"metrics"(指标)，而后者是一个二进制文件，描述字体的形状。在`reportlab/fonts`目录下，包含了`"DarkGardenMK.afm"`和`"DarkGardenMK.pfb"`两个文件，这两个文件被用来作为一个例子字体。

在下面的例子中，找到包含测试字体的文件夹，并用`pdfmetrics`模块注册它，以便将来使用，之后我们可以像其他标准字体一样使用它。

```
import os
import reportlab
folder = os.path.dirname(reportlab.__file__) + os.sep + 'fonts'
afmFile = os.path.join(folder, 'DarkGardenMK.afm')
pfbFile = os.path.join(folder, 'DarkGardenMK.pfb')

from reportlab.pdfbase import pdfmetrics
justFace = pdfmetrics.EmbeddedType1Face(afmFile, pfbFile)
faceName = 'DarkGardenMK' # pulled from AFM file
pdfmetrics.registerTypeFace(justFace)
justFont = pdfmetrics.Font('DarkGardenMK',
                           faceName,
                           'WinAnsiEncoding')
pdfmetrics.registerFont(justFont)

canvas.setFont('DarkGardenMK', 32)
canvas.drawString(10, 150, 'This should be in')
canvas.drawString(10, 100, 'DarkGardenMK')
```

请注意，参数`"WinAnsiEncoding"`与输入无关，它是说字体文件内的哪一组字符将被激活并可用。

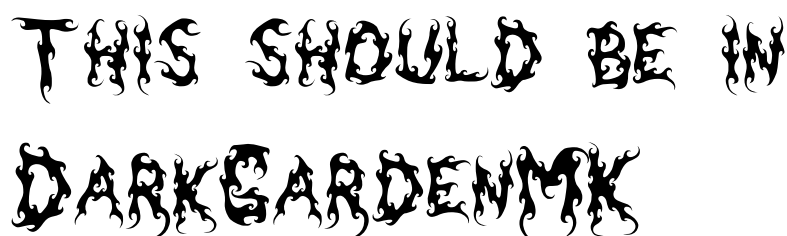


图 3 - 1 : 使用非常不标准的字体

字体的名称来自AFM文件的`FontName`字段。在上面的例子中，我们事先知道了这个名字，但是很多时候字体描述文件的名称是非常神秘的，那么你可能会想从AFM文件中自动检索到这个名字。当缺乏更复杂的方法时，你可以使用一些像这样简单的代码。

```
class FontNameNotFoundError(Exception):
    pass

def findFontName(path):
    "Extract a font name from an AFM file."
```



```
f = open(path)

found = 0
while not found:
    line = f.readline()[:-1]
    if not found and line[:16] == 'StartCharMetrics':
        raise FontNameNotFoundError, path
    if line[:8] == 'FontName':
        fontName = line[9:]
        found = 1

return fontName
```

在DarkGardenMK的例子中，我们明确指定了要加载的字体描述文件的位置。一般来说，你会更倾向于将你的字体存储在一些规范的位置，并让嵌入机制知道它们。使用同样的配置机制，我们已经在本节开头看到了，我们可以为Type-1字体指定一个默认搜索路径。

不幸的是，目前还没有一个可靠的标准来规定这些位置（甚至在同一个平台上也没有），因此，你可能需要编辑`reportlab_settings.py`或者`~/reportlab_settings`来修改`T1SearchPath`标识符的值，以包含额外的目录。我们自己的建议是在开发中使用`reportlab/fonts`文件夹；并且在任何受控服务器部署中，将任何需要的字体作为应用程序的打包部件。这样可以避免字体被其他软件或系统管理员安装和卸载。

关于缺失字形的警告

如果你指定了一个编码，一般会认为字体设计师已经提供了所有需要的字形。然而，事实并非总是如此。在我们的示例字体中，字母表中的字母都存在，但许多符号和重音都没有。默认的行为是，当传递给字体一个它无法绘制的字符时，字体将打印一个“notdef”字符--通常是一个blob、点或空格。然而，你可以要求库警告你；下面的代码（在加载字体之前执行）将导致在你注册字体时，对任何不在字体中的字形产生警告。

```
import reportlab.rl_config
reportlab.rl_config.warnOnMissingFontGlyphs = 0
```

3.4 标准的单字节字体编码

本节为您展示常用编码中可用的字形。

下面的代码表显示了WinAnsiEncoding中的字符，这是Windows和许多美国和西欧Unix系统的标准编码。这是在美国和西欧的Windows和许多Unix系统上的标准编码，也被称为Code Page 1252，实际上与ISO-Latin-1相同（包含一个或两个额外的字符）。这是Reportlab PDF库使用的默认编码。它由reportlab/lib中的一个标准例程codecharts.py生成，可用于显示字体的内容。沿边的索引号是以十六进制表示的。

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00																																
20		!	"	#	\$	%	&	'	()	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{		}	~	•
80	€	•	,	f	„	…	†	‡	^	%	Š	‹	Œ	•	Ž	•	•	‘	’	“	”	•	—	~	™	š	›	œ	•	ž	ÿ	
A0		ı	ç	£	¤	¥	¦	§	¨	©	ª	«	¬	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

图 3 - 2 : WinAnsi Encoding

下面的代码表显示了MacRomanEncoding中的字符，听起来，这是美国和西欧Macintosh电脑上的标准编码。和通常的非unicode编码一样，前128个码点（在本例中，最上面4行）是ASCII标准，并与上面的WinAnsi代码表一致；但下面4行不同。

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00																																
20		!	"	#	\$	%	&	'	()	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{		}	~	■
80	Ä	Å	Ç	É	Ñ	Ö	Ü	á	à	â	ä	ã	å	ç	é	è	ê	ë	í	ì	î	ï	ñ	ó	ò	ô	õ	ö	ú	û	ü	
A0	†	°	¢	£	§	•	¶	ß	®	©	™	´	¨	≠	Æ	Ø	∞	±	≤	≥	¥	μ	ð	Σ	Π	π	∫	ª	º	Ω	æ	ø
C0	¿	¡	¬	√	ƒ	≈	Δ	«	»	…	■	À	Ã	Ö	œ	—	“	”	‘	’	÷	◊	ÿ	ÿ	/	€	‹	›	fi	fl		
E0	‡	·	,	„	‰	Â	Ê	Á	Ë	È	Í	Î	Ì	Ó	Ô	■	Ò	Ú	Û	Ü	ı	ˆ	˜	˘	˙	˚	˛	˜	˘	˙	˚	˛

图 3 - 3 : MacRoman Encoding

这两种编码适用于标准字体（Helvetica、Times-Roman和Courier及其变体），并将适用于大多数商业字体，包括Adobe的字体。然而，有些字体包含非文本字形，这个概念并不真正适用。例如，Zapf Dingbats和Symbol可以被视为各自拥有自己的编码。

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00																																
20		✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	
40	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	
60	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	✂	
80	()	()	()	<	>	()	()	()	()	()	()	()	()	()	()	()	(
A0	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
C0	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫	⑬	⑭	⑮	⑯	⑰	⑱	⑲	⑳	㉑	㉒	㉓	㉔	㉕	㉖	㉗	㉘	㉙	㉚	㉛	
E0	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	➡	

图 3 - 4 : ZapfDingbats和它的唯一编码。

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00																																
20		!	∇	#	∃	%	&	ə	()	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	≡	A	B	X	Δ	E	Φ	Γ	H	I	∅	K	Λ	M	N	O	Π	Θ	P	Σ	T	Y	ζ	Ω	Ξ	Ψ	Z		:		⊥	
60		α	β	γ	δ	ε	φ	γ	η	ι	φ	κ	λ	μ	ν	ο	π	θ	ρ	σ	τ	υ	ϖ	ω	ξ	ψ	ζ	{		}	~	■
80	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
A0	€	Υ	′	≤	/	∞	f	♣	♦	♥	♠	↔	←	↑	→	↓	°	±	″	≥	×	∞	∂	•	÷	≠	≡	≈	...		—	↵
C0	ℵ	ℶ	ℷ	ℸ	⊗	⊕	∅	∩	∪	⊃	⊄	⊆	⊇	∈	∉	∠	∇	®	©	™	Π	√	·	¬	^	∨	⇔	⇐	↑	⇒	↕	
E0	◊	◊	®	©	™	Σ	(}			■	∖	∫	∫												

图 3 - 5 : Symbol及其唯一的编码

3.5 支持TrueType字体

Marius Gedminas (mgedmin@delfi.lt)在Viktorija Zaksiene(vika@pov.lt)的帮助下，为嵌入式TrueType字体提供了支持。TrueType字体可以在Unicode/UTF8中使用，并且不限于256个字符。

我们使用reportlab.pdfbase.ttfonts.TTFont来创建一个真正的字体对象，并使用reportlab.pdfbase.pdfmetrics.registerFont进行注册。在pdfgen直接在画布上绘图时我们可以这样做

```
# we know some glyphs are missing, suppress warnings
import reportlab.rl_config
reportlab.rl_config.warnOnMissingFontGlyphs = 0

from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont

pdfmetrics.registerFont(TTFont('Vera', 'Vera.ttf'))
pdfmetrics.registerFont(TTFont('VeraBd', 'VeraBd.ttf'))
pdfmetrics.registerFont(TTFont('Veralt', 'Veralt.ttf'))
pdfmetrics.registerFont(TTFont('VeraBI', 'VeraBI.ttf'))
canvas.setFont('Vera', 32)
canvas.drawString(10, 150, "Some text encoded in UTF-8")
canvas.drawString(10, 100, "In the Vera TT Font!")
```

Some UTF-8 text encoded
in the Vera TT Font!

图 3 - 6 : 使用Vera TrueType字体

在上面的例子中，True Type字体对象是使用

```
TTFont(name,filename)
```

所以ReportLab的内部名称由第一个参数给出，第二个参数是一个字符串（或类似文件的对象），表示字体的TTF文件。在Marius最初的补丁中，文件名应该是完全正确的，但我们已经修改了，如果文件名是相对的，那么在当前目录下搜索相应的文件，然后在reportlab.rl_config.TTFSearchpath!

在使用Platypus中的TT字体之前，我们应该在和<i>属性下添加一个从家族名称到描述行为的单个字体名称的映射。

```
from reportlab.pdfbase.pdfmetrics import registerFontFamily
registerFontFamily('Vera',normal='Vera',bold='VeraBd',italic='Veralt',
boldItalic='VeraBI')
```

如果我们只有一个Vera常规字体，没有粗体或斜体，那么我们必须将所有字体映射到同一个内部字体名。和<i>标签现在可以安全使用，但没有效果。如上所述注册和映射Vera字体后，我们可以使用段落文本，如

```
<font name="Times-Roman" size="14">This
is in Times-Roman</font> <font
name="Vera" color="magenta"
size="14">and this is in magenta
<b>Vera!</b></font>
```

This is in Times-Roman
and this is in magenta
Vera!

图 3-7: Using TTF fonts in paragraphs

3.6 亚洲字体支持

Reportlab PDF库旨在为亚洲字体提供全面支持。PDF是第一个真正可移植的亚洲文本处理解决方案。有两种主要的方法。Adobe的亚洲语言包，或者TrueType字体。

亚洲语言包

这种方法提供了最好的性能，因为没有任何东西需要嵌入到PDF文件中；与标准字体一样，一切都在阅读器上。

Adobe公司为每种主要语言都提供了附加组件。在Adobe Reader 6.0和7.0中，当您尝试使用它们打开文档时，您会被提示下载并安装这些插件。在早期的版本中，你会在打开亚洲文档时看到一个错误信息，你必须知道该怎么做。

日文、繁体中文(台湾/香港)、简体中文(中国大陆)和韩文都支持，我们的软件知道以下字体。

- chs = Chinese Simplified (mainland): 'SourceHanSansSC'
- cht = Chinese Traditional (Taiwan): 'MSung-Light', 'MHei-Medium'
- kor = Korean: 'HYSMyeongJoStd-Medium', 'HYGothic-Medium'
- jpn = Japanese: 'HeiseiMin-W3', 'HeiseiKakuGo-W5'

由于许多用户不会安装字体包，我们已经包含了一些日文字符的相当颗粒状的**bitmap**。下面我们将讨论生成它们所需要的内容。

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		亜	啞	娃	阿	哀	愛	挨	始	逢	葵	茜	穉	惡	握	渥	旭	葦	芦	膠
20	梓	压	幹	扱	宛	姐	虻	飴	絢	綾	鮎	或	粟	裕	安	庵	按	暗	案	闇
40	鞍	杏	以	伊	位	依	偉	困	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異
60	移	維	緯	胃	萎	衣	謂	違	遣	医	井	亥	域	育	郁	磯	一	壹	溢	逸
80	稻	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭					

在2.0版本之前，当你注册一个CIDFont时，你必须指定许多本地编码之一。在2.0版本中，你应该使用一个新的UnicodeCIDFont类。

```
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.cidfonts import UnicodeCIDFont
pdfmetrics.registerFont(UnicodeCIDFont('HeiseiMin-W3'))
canvas.setFont('HeiseiMin-W3', 16)

# the two unicode characters below are "Tokyo"
msg = u'\u6771\u4eac : Unicode font, unicode input'
canvas.drawString(100, 675, msg)
```

旧的编码风格与显式编码应该仍然有效，但现在只有当你需要构建垂直文本时才有意义。我们的目标是在未来为UnicodeCIDFont构造函数添加更多可读的水平和垂直文本选项。以下四个测试脚本会生成相应语言的样本。

```
tests/test_multibyte_jpn.py
tests/test_multibyte_kor.py
tests/test_multibyte_chs.py
```

tests/test_multibyte_cht.py

在以前版本的ReportLab

PDF库中，我们不得不使用Adobe的CMap文件（如果安装了亚洲语言包，则位于Acrobat Reader附近）。现在我们只需要处理一种编码，字符宽度数据被嵌入到软件包中，生成时不需要CMap文件。在`rl_config.py`中的CMap搜索路径现在已经被废弃，如果你限制自己使用UnicodeCIDFont，则没有效果。

TrueType字体和亚洲字符

这就是简单的方法。在使用亚洲TrueType字体时，完全不需要特殊的处理。例如，在控制面板中安装了日语作为选项的Windows用户，会有一个可以使用的字体 "`msmincho.ttf`"。然而，请注意，解析字体需要时间，而且相当大的子集可能需要嵌入你的PDF中。我们现在也可以解析以`.ttc`结尾的文件，它们是`.ttf`的轻微变化。

To Do

我们预计将在一段时间内开发这个领域的包。accept2dyear这是一个主要优先事项的大纲。

我们欢迎大家的帮助

- 确保我们在横写和竖写中的所有编码都有准确的字符指标。
- 为UnicodeCIDFont添加选项，在字体允许的情况下，允许垂直和比例变体。
- 改进段落中的包字代码，允许竖写。

3.7 RenderPM 测试

这可能也是提及reportlab/graphics/renderPM.py的测试函数的最好地方，它可以被认为是行使renderPM("PixMap Renderer", 相对于renderPDF、renderPS或renderSVG)的测试的规范地方。

如果你从命令行运行这个，你应该会看到很多像下面这样的输出。

```
C:\code\reportlab\graphics>renderPM.py
wrote pmout\renderPM0.gif
wrote pmout\renderPM0.tif
wrote pmout\renderPM0.png
wrote pmout\renderPM0.jpg
wrote pmout\renderPM0.pct
...
wrote pmout\renderPM12.gif
wrote pmout\renderPM12.tif
wrote pmout\renderPM12.png
wrote pmout\renderPM12.jpg
wrote pmout\renderPM12.pct
wrote pmout\index.html
```

它运行了许多测试，从 "Hello World"测试开始，到各种测试，包括：线条；各种尺寸、字体、颜色和对齐方式的文本字符串；基本形状；转换和旋转组；缩放坐标；旋转字符串；嵌套组；锚定和非标准字体。

它创建了一个名为`pmout`的子目录，将图片文件写入其中，并写了一个`index.html`的页面，便于参考所有结果。

与字体相关的测试，你可能会想看看#11（'非标准字体中的文本字符串'）和#12（'测试各种字体'）。

第 4 章 PDF的特殊功能

PDF提供了许多功能，使电子文档的浏览更加高效和舒适，我们的类库就公开了其中的一些功能。

4.1 表单

表单功能使您可以在PDF文件开头附近创建一个图形和文本块，然后在后续页面中简单地引用它。如果要处理5000种重复的业务表单（例如一页发票或工资单），则只需将背景存储一次，并在每页上简单地绘制变化的文本即可。

正确使用表格可以极大地减少文件大小和生产时间，并且显然甚至可以加快打印机上的处理速度。

表单不需要引用整个页面；任何可能经常重复的内容都应以表格的形式放置。

下面的示例显示了使用的基本顺序。

真正的程序可能会预先定义表单，然后从另一个位置引用它们。

```
def forms(canvas):
    #first create a form...
    canvas.beginForm("SpumoniForm")
    #re-use some drawing functions from earlier
    spumoni(canvas)
    canvas.endForm()

    #then draw it
    canvas.doForm("SpumoniForm")
```

4.2 链接和目的地(书签)

PDF支持内部超链接。单击可以触发多种链接类型，目标类型和事件。目前，我们仅支持从文档的一个部分跳转到另一部分并在跳转后控制窗口的缩放级别的基本功能。`bookmarkPage`方法定义一个目标，该目标是跳转的终点。

```
canvas.bookmarkPage(name, fit="Fit", left=None,
                    top=None, bottom=None, right=None, zoom=None
                    )
```

默认情况下，`bookmarkPage`方法将页面本身定义为目标。跳转到由`bookmarkPage`定义的端点之后，PDF浏览器将显示整个页面，并将其缩放以适合屏幕大小：

```
canvas.bookmarkPage(name)
```

通过提供一个`fit`参数，`bookmarkPage`方法可以用多种不同的方式显示页面。

fit	必传参数	描述
Fit		自适应窗口 (默认) Entire page fits in window (the default)
FitH	top	坐标在窗口上方, 自适应宽度 Top coord at top of window, width scaled to fit
FitV	left	坐标在窗口左边, 自适应高度 Left coord at left of window, height scaled to fit
FitR	left bottom right top	缩放窗口以适应指定的矩形 Scale window to fit the specified rectangle
XYZ	left top zoom	细致的控制。如果您省略了一个参数，PDF浏览器会将其解释为 "保持原样"。 Fine grained control. If you omit a parameter the PDF browser interprets it as "leave as is"

表 4-1 - 配合不同类型所需的属性



注意：`fit`的设置是区分大小写的，所以`fit="FIT"`是无效的。

有时你希望跳转的目标是一个页面的某个部分。`fit="FitR"`允许你确定一个特定的矩形，缩放区域以适合整个页面。

要将显示设置为页面的特定x和y坐标，并直接使用`fit="XYZ"`控制缩放。

```
canvas.bookmarkPage('my_bookmark',fit="XYZ",left=0,top=200)
```

这个目标位于页面的最左边，屏幕顶部的位置是200。因为没有设置`zoom`，所以无论用户设置成什么样子，缩放都会保持在这个位置。

```
canvas.bookmarkPage('my_bookmark',fit="XYZ",left=0,top=200,zoom=2)
```

这次的缩放设置为将页面扩大2倍其正常大小。



注意：XYZ和FitR的拟合类型都需要用默认的用户空间来指定它们的位置参数(`top`, `bottom`, `left`, `right`)。它们会忽略画布图形状态下的任何几何变换。

之前有两个书签方法是支持的，但由于`bookmarkPage`的通用性，现在已经废弃了。这两个方法是`bookmarkHorizontalAbsolute`和`bookmarkHorizontal`。

定义内部链接

```
canvas.linkAbsolute(contents, destinationname, Rect=None, addtopage=1,
                    name=None, thickness=0, color=None, dashArray=None, **kw)
```

`linkAbsolute`方法定义了一个跳跃的起点。当用户使用动态查看器(如Acrobat Reader)浏览生成的文档时，当鼠标在`Rect`指定的矩形内点击时，查看器将跳转到与`destinationname`相关联的端点。如同`bookmarkHorizontalAbsolute`一样，矩形`Rect`必须用默认的用户空间来指定。参数`contents`指定了当用户左键点击该区域时在查看器中显示的文本块。

矩形`Rect`必须用元组(`x1`, `y1`, `x2`, `y2`)来指定，以确定在默认用户空间中矩形的左下角和右上角。

示例代码

```
canvas.bookmarkPage("Meaning_of_life")
```

定义了一个位置，作为当前页面的整个标识符`Meaning_of_life`

。为了在绘制一个可能不同的页面时创建一个矩形链接到它，我们将使用以下代码。

```
canvas.linkAbsolute("Find the Meaning of Life", "Meaning_of_life",
                    (inch, inch, 6*inch, 2*inch))
```

默认情况下，在交互式浏览时，链接周围会出现一个矩形。使用关键字参数`Border='[0 0 0]'`来抑制查看链接时周围可见的矩形。例如

```
canvas.linkAbsolute("Meaning of Life", "Meaning_of_life",
                    (inch, inch, 6*inch, 2*inch), Border='[0 0 0]')
```

如果没有指定`Border`参数，`thickness`、`color`和`dashArray`参数可以交替使用来指定边框。如果指定了`Border`参数，它必须是一个PDF数组的字符串表示，或者是一个`PDFArray`(参见`pdfdoc`模块)。`color`参数(应该是一个`Color`实例)相当于一个关键字参数`C`，它应该解析为一个PDF颜色定义(通常是一个三条PDF数组)。

`canvas.linkRect`方法的意图与`linkAbsolute`方法类似，但多了一个参数`relative=1`，所以打算服从本地用户空间转换。

4.3 大纲

Acrobat Reader有一个导航页，它可以容纳一个文档大纲；当您打开本指南时，它通常应该是可见的。我们提供一些简单的方法来添加大纲条目。通常情况下，一个制作文档的程序(如本用户指南)在到达文档中的每个标题时，会调用方法`canvas.addOutlineEntry(self, title, key, level=0, closed=None)`。

`title`是将显示在左侧窗格的标题。`key`必须是一个字符串，它在文档中是唯一的，并且和超链接一样，可以命名一个书签。除非另有说明，否则`level`是0--最上层，而且一次下行超过一层是错误的(例如，在0层标题后加上2层标题)。最后，`closed`参数指定大纲窗格中的节点是默认关闭还是打开。

下面的片段来自于格式化本用户指南的文档模板。中央处理器依次查看每个段落，当出现新的章节时，就会做出一个新的大纲条目，将章节标题文本作为标题文本。键是从章节号中获得的（这里没有显示），所以第2章的键为"ch2"。大纲入口指向的书签是针对整页的，但它也可以很容易地成为一个单独的段落。

```
#abridged code from our document template
if paragraph.style == 'Heading1':
    self.chapter = paragraph.getPlainText()
    key = 'ch%d' % self.chapterNo
    self.canv.bookmarkPage(key)
    self.canv.addOutlineEntry(paragraph.getPlainText(),
                             key, 0, 0)
```

4.4 页面过渡效果

```
canvas.setPageTransition(self, effectname=None, duration=1,
                        direction=0, dimension='H', motion='I')
```

`setPageTransition`方法指定了一个页面如何被下一个页面替换。例如，通过将页面转换效果设置为"溶解"，当当前页面在交互式浏览过程中被下一个页面所取代时，它将显示为融化。这些效果在美化幻灯片演示等地方很有用。关于如何使用此方法，请参阅参考手册。

4.5 内部文件注释

```
canvas.setAuthor(name)
canvas.setTitle(title)
canvas.setSubject(subj)
```

这些方法对文档没有自动可见的效果。它们向文件添加内部注释。这些注释可以使用浏览器的"文档信息"菜单项来查看，它们也可以作为一种简单的标准方式，向不需要解析整个文件的归档软件提供有关文件的基本信息。要找到注释，请使用标准文本编辑器（如MS/Windows上的notepad或unix上的vi或emacs）查看*.pdf输出文件，并在文件内容中查找字符串/Author。

```
def annotations(canvas):
    from reportlab.lib.units import inch
    canvas.drawString(inch, 2.5*inch,
        "setAuthor, setTitle, setSubject have no visible effect")
    canvas.drawString(inch, inch, "But if you are viewing this document dynamically")
    canvas.drawString(inch, 0.5*inch, "please look at File/Document Info")
    canvas.setAuthor("the ReportLab Team")
    canvas.setTitle("ReportLab PDF Generation User Guide")
    canvas.setSubject("How to Generate PDF files using the ReportLab modules")
```

如果您想让主题、标题和作者在查看和打印时自动显示在文档中，您必须像其他文本一样将它们绘制到文档中。

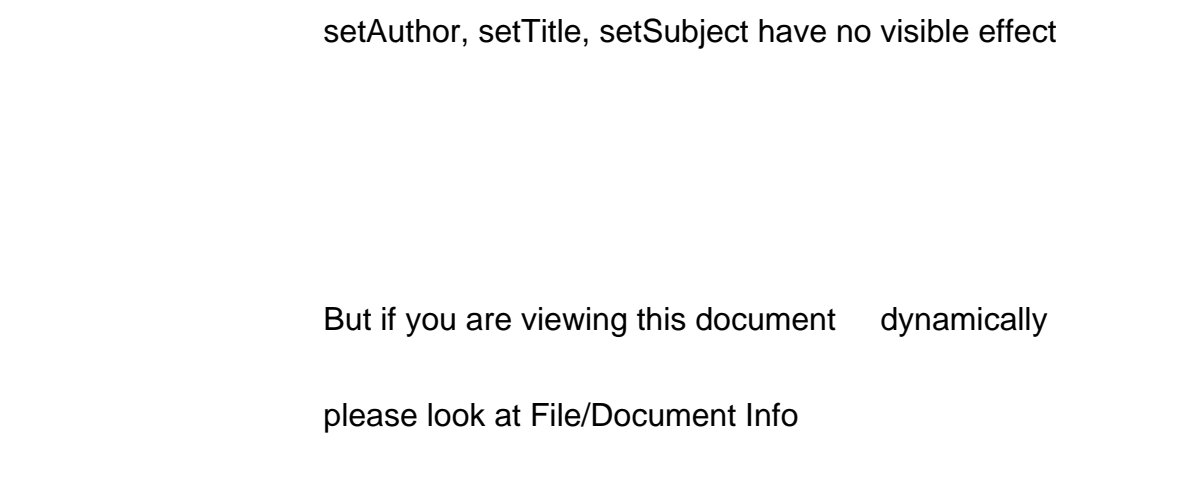


图 4 - 1 : 设置文档内部注释

4.6 加密

关于加密PDF文件

Adobe的PDF标准允许你在对一个PDF文件进行加密时做三件相关的事情。

- 对其进行密码保护，所以用户必须提供有效的密码才能够读取它。
- 对文件的内容进行加密，使其在解密前毫无用处，并对文件进行加密。
- 控制用户在查看文档时是否可以打印、复制、粘贴或修改文档。

PDF安全处理程序允许为一个文档指定两个不同的密码。

- 所有者 "密码"（也就是 "安全密码 "或 "主密码"）。
- 用户 "密码"（也就是 "打开密码"）。

当用户提供其中一个密码时，PDF文件将被打开、解密并显示在屏幕上。

如果提供了所有者密码，那么文件的打开就有了完全的控制权--你可以对它做任何事情，包括改变安全设置和密码，或者用新密码重新加密。

如果用户密码是提供的密码，你就在一个更受限制的模式下打开它。这些限制是在文件加密时设置的，将允许或拒绝用户进行以下操作的权限。

- 修改文件的内容
- 从文档中复制文本和图形
- 添加或修改文本注释和交互式表格字段。
- 打印文件

请注意，所有受密码保护的PDF文件都是加密的，但并不是所有加密的PDF都受密码保护。如果一个文件的用户密码是一个空字符串，当打开文件时将不会有密码提示。如果只用所有者密码来保护文档，那么打开文件时也不会有密码提示。如果在对PDF文件进行加密时，将所有者和用户密码设置为同一字符串，则该文件将始终以用户访问权限打开。这就意味着，可以创建一个文件，比如说，任何人都不能打印出来，即使是创建该文件的人。

所有者密码已设置?	用户密码已设置?	结果
是	-	打开文件时无需密码。适用于所有人。 No password required when opening file. Restrictions apply to everyone.

-	是	打开文件时需要用户密码。适用于所有人。 User password required when opening file. Restrictions apply to everyone.
是	是	打开文件时需要一个密码。 只有在提供用户密码的情况下才有限制。 A password required when opening file. Restrictions apply only if user password supplied.

表 4-2 - PDF加密方式

当一个PDF文件被加密时，加密将应用于文件中的所有字符串和流。这可以防止没有密码的人简单地
从PDF文件中删除密码以获得访问权 - 它使文件无用，除非你真的有密码。

PDF 的标准加密方法使用 MD5 消息摘要算法（如 RFC 1321，MD5 消息摘要算法中所述）和一种称为
RC4 的加密算法。RC4是一种对称流加密算法--加密和解密都使用相同的算法，而且该算法不会改变
数据的长度。

如何使用加密技术

文档可以通过向画布对象传递一个参数来加密。

如果参数是一个字符串对象，它被用作PDF的用户密码。

参数也可以是reportlab.lib.pdfencrypt.StandardEncryption类的实例，它允许对加密设置进行更精
细的控制。

StandardEncryption构造函数接受以下参数。

```
def __init__(self, userPassword,  
              ownerPassword=None,  
              canPrint=1,  
              canModify=1,  
              canCopy=1,  
              canAnnotate=1,  
              strength=40):
```

userPassword和ownerPassword参数在加密的PDF上设置了相关密码。

布尔标志canPrint,canModify,canCopy,canAnnotate决定了当只有用户密码被提供时，用户是否可
以在PDF上执行相应的操作。

如果用户在打开PDF时提供了所有者密码，则无论标志如何，所有的操作都可以执行。

示例

要创建一个名为hello.pdf的文档，用户密码为'rptlab'，不允许打印，可以用以下代码。

```
from reportlab.pdfgen import canvas  
from reportlab.lib import pdfencrypt  
  
enc=pdfencrypt.StandardEncryption("rptlab",canPrint=0)  
  
def hello(c):  
    c.drawString(100,100,"Hello World")  
    c = canvas.Canvas("hello.pdf",encrypt=enc)  
    hello(c)  
    c.showPage()  
    c.save()
```

4.7 交互式表单

交互式表格概述

PDF标准允许各种交互式元素，ReportLab工具包目前只支持一小部分的可能性，应该被认为是一项正在进行中的工作。目前我们允许使用`checkbox`、`radio`、`choice`和`listbox`部件进行选择；文本值可以使用`textfield`部件进行输入。所有的widget都是通过调用`canvas.acroform`属性上的方法创建的。

示例

这显示了在当前页面上创建交互式元素的基本机制。

```
canvas.acroform.checkbox(  
    name='CB0',  
    tooltip='Field CB0',  
    checked=True,  
    x=72,y=72+4*36,  
    buttonStyle='diamond',  
    borderStyle='bevelled',  
    borderWidth=2,  
    borderColor=red,  
    fillColor=green,  
    textColor=blue,  
    forceBorder=True)
```

注意：`acroform`画布属性是根据需求自动创建的，而且一个文档中只允许有一个表单。

复选框用法

`canvas.acroform.checkbox`方法在当前页面上创建了一个`checkbox`小部件。复选框的值是YES或OFF。参数为:

canvas.acroform.checkbox 参数列表		
参数名	描述	默认值
name	表单参数名 the parameter's name	None
x	在页面上的水平位置(绝对坐标) the horizontal position on the page (absolute coordinates)	0
y	在页面上的垂直位置(绝对坐标) the vertical position on the page (absolute coordinates)	0
size	轮廓尺寸：size x size The outline dimensions size x size	20
checked	如果为真，则该复选框被初始选中 if True the checkbox is initially checked	False
buttonStyle	如果为真，则该复选框最初被选中，复选框样式（见下文）。 the checkbox style (see below)	'check'
shape	小组件的轮廓（见下文）。 The outline of the widget (see below)	'square'
fillColor	填充颜色 colour to be used to fill the widget	None
textColor	符号的颜色 the colour of the symbol or text	None
borderWidth	边框宽度 as it says	1
borderColor	边框颜色 the widget's border colour	None
borderStyle	边框样式 The border style name	'solid'
tooltip	悬停在小组件上时要显示的文本。 The text to display when hovering over the widget	None
annotationFlags	空白分隔的注解标志字符串 blank separated string of annotation flags	'print'

canvas.acroform.checkbox 参数列表		
参数名	描述	默认值
fieldFlags	空白分隔的字段标志（见下文）。 Blank separated field flags (see below)	'required'
forceBorder	为 True 的时候会强行画边框 when true a border force a border to be drawn	False
relative	如果为 Tru，服从当前画布的变换 if true obey the current canvas transform	False
dashLen	如果 borderStyle=='dashed'，要使用的折线。 the dashline to be used if the borderStyle=='dashed'	3

单选框使用方法

`canvas.acroform.radio`方法在当前页面上创建一个radio小组件。radio的值是radio组选择的值，如果没有选择，则是OFF。参数是

canvas.acroform.radio 参数列表		
参数名	描述	默认值
name	单选框组的名称(该参数) the radio's group (ie parameter) name	None
value	单选框组的名称 the radio's group name	None
x	在页面上的水平位置(绝对坐标) the horizontal position on the page (absolute coordinates)	0
y	在页面上的垂直位置(绝对坐标) the vertical position on the page (absolute coordinates)	0
size	轮廓尺寸，size x size The outline dimensions size x size	20
selected	如果为 'true'，则该单选机在其组中被选中。 if True this radio is the selected one in its group	False
buttonStyle	按钮样式 the checkbox style (see below)	'check'
shape	小组件的轮廓（见下文）。 The outline of the widget (see below)	'square'
fillColor	用来填充小组件的颜色 colour to be used to fill the widget	None
textColor	符号的颜色 the colour of the symbol or text	None
borderWidth	边框宽度 as it says	1
borderColor	边框颜色 the widget's border colour	None
borderStyle	边框样式 The border style name	'solid'
tooltip	悬停在小组件上时要显示的文本。 The text to display when hovering over the widget	None
annotationFlags	空白分隔的注解标志字符串 blank separated string of annotation flags	'print'
fieldFlags	空白分隔的字段标志（见下文）。 Blank separated field flags (see below)	'noToggleToOff required radio'
forceBorder	当真的时候会强行画边框 when true a border force a border to be drawn	False
relative	如果为true，则遵循当前的画布转换 if true obey the current canvas transform	False
dashLen	如果borderStyle=='dashed'，要使用的折线。 the dashline to be used if the borderStyle=='dashed'	3

列表框用法

`canvas.acroform.listbox`方法在当前页面上创建了一个`listbox`小部件。listbox包含一个选项列表，其中一个或多个选项（取决于`fieldFlags`）可以被选中。

canvas.acroform.listbox 参数列表		
参数	描述	默认值
name	组名 the radio's group (ie parameter) name	None
options	可用选项的列表或元组 List or tuple of avaiable options	[]
value	单个或选定选项的字符串列表。 Singleton or list of strings of selected options	[]
x	在页面上的水平位置(绝对坐标) the horizontal position on the page (absolute coordinates)	0
y	在页面上的垂直位置(绝对坐标) the vertical position on the page (absolute coordinates)	0
width	小部件宽度 The widget width	120
height	小部件高度 The widget height	36
fontName	要使用的第1种字体的名称。 The name of the type 1 font to be used	'Helvetica'
fontSize	要使用的字体大小 The size of font to be used	12
fillColor	用来填充小部件的颜色 colour to be used to fill the widget	None
textColor	符号或文本的颜色 the colour of the symbol or text	None
borderWidth	边框宽度 as it says	1
borderColor	边框颜色 the widget's border colour	None
borderStyle	边框样式 The border style name	'solid'
tooltip	悬停在小部件上时要显示的文本。 The text to display when hovering over the widget	None
annotationFlags	空白分隔的注解标志字符串 blank separated string of annotation flags	'print'
fieldFlags	空白分隔的字段标志（见下文）。 Blank separated field flags (see below)	"
forceBorder	当真的时候会强行画边框 when true a border force a border to be drawn	False
relative	如果为真，服从当前画布的变换 if true obey the current canvas transform	False
dashLen	如果borderStyle=='dashed', 要使用的折线。 the dashline to be used if the borderStyle=='dashed'	3

下拉菜单的使用

`canvas.acroform.choice`方法在当前页面上创建了一个`dropdown`小部件。下拉菜单包含一个选项列表，其中一个或多个选项（取决于`fieldFlags`）可以被选中。如果您在`fieldFlags`中添加了`edit`，那么结果可以被编辑。

canvas.acroform.choice 参数列表		
参数	描述	默认值
name	组名 the radio's group (ie parameter) name	None

canvas.acroform.choice 参数列表		
参数	描述	默认值
options	可用选项的列表或元组 List or tuple of available options	[]
value	单个或选定选项的字符串列表。 Singleton or list of strings of selected options	[]
x	在页面上的水平位置(绝对坐标) the horizontal position on the page (absolute coordinates)	0
y	在页面上的垂直位置(绝对坐标) the vertical position on the page (absolute coordinates)	0
width	小组件宽度 The widget width	120
height	小组件高度 The widget height	36
fontName	要使用的第1种字体的名称。 The name of the type 1 font to be used	'Helvetica'
fontSize	要使用的字体大小 The size of font to be used	12
fillColor	用来填充小组件的颜色 colour to be used to fill the widget	None
textColor	符号的颜色 the colour of the symbol or text	None
borderWidth	边框宽度 as it says	1
borderColor	边框颜色 the widget's border colour	None
borderStyle	边框样式 The border style name	'solid'
tooltip	悬停在小组件上时要显示的文本。 The text to display when hovering over the widget	None
annotationFlags	空白分隔的注解标志字符串 blank separated string of annotation flags	'print'
fieldFlags	空白分隔的字段标志（见下文）。 Blank separated field flags (see below)	'combo'
forceBorder	当真的时候会强行画边框 when true a border force a border to be drawn	False
relative	如果为真，服从当前画布的变换 if true obey the current canvas transform	False
dashLen	如果borderStyle=='dashed'，要使用的破折号。 the dashline to be used if the borderStyle=='dashed'	3
maxlen	无或小组件值的最大长度 None or maximum length of the widget value	None

文本字段用法

`canvas.acroform.textfield`方法在当前页面上创建了一个`textfield`条目部件。文本字段可以被编辑，以改变小组件的值。

canvas.acroform.textfield 参数列表		
参数	描述	默认值
name	组名 the radio's group (ie parameter) name	None
value	文本字段的值 Value of the text field	''
maxlen	无或小组件值的最大长度 None or maximum length of the widget value	100
x	在页面上的水平位置(绝对坐标) the horizontal position on the page (absolute coordinates)	0

canvas.acroform.textfield 参数列表		
参数	描述	默认值
y	在页面上的垂直位置(绝对坐标) the vertical position on the page (absolute coordinates)	0
width	小组件宽度 The widget width	120
height	小组件高度 The widget height	36
fontName	要使用的第1种字体的名称。 The name of the type 1 font to be used	'Helvetica'
fontSize	要使用的字体大小 The size of font to be used	12
fillColor	用来填充小组件的颜色 colour to be used to fill the widget	None
textColor	符号或文本的颜色 the colour of the symbol or text	None
borderWidth	边框宽度 as it says	1
borderColor	边框颜色 the widget's border colour	None
borderStyle	边框样式 The border style name	'solid'
tooltip	悬停在小组件上时要显示的文本。 The text to display when hovering over the widget	None
annotationFlags	空白分隔的注解标志字符串 blank separated string of annotation flags	'print'
fieldFlags	空白分隔的字段标志（见下文）。 Blank separated field flags (see below)	"
forceBorder	当真的时候会强行画边框 when true a border force a border to be drawn	False
relative	如果为真，服从当前画布的变换 if true obey the current canvas transform	False
dashLen	如果borderStyle=='dashed'，要使用破折号。 the dashline to be used if the borderStyle=='dashed'	3

按钮样式

按钮样式参数表示当按钮被选中时，应该在按钮中出现什么样式的符号。有几种选择：**check**、**cross**、**circle**、**star**、**diamond**。

请注意，文档渲染器可能会使这些符号中的某些符号在其预期应用中出现错误。Acrobat阅读器更喜欢在规范规定应该显示的内容上使用自己的渲染（特别是在使用表格高亮功能的时候）

小工具形状

形状参数描述了复选框或单选部件的轮廓应该如何显示，你可以使用:**circle**、**square**。

渲染器可能会自行决定小组件的外观，所以Acrobat Reader更喜欢圆形轮廓的收音机。

边框样式

borderStyle参数会改变页面上小组件的3D外观，你可以使用：**solid**、**dashed**、**inset**、**bevelled**、**underlined**。

fieldFlags 参数

fieldFlags参数可以是一个整数或一个包含空白的独立标记的字符串，其值如下表所示。更多信息请参考PDF规范。

字段标志 Tokens 和 values		
Token	描述	值
readOnly	小部件只读 The widget is read only	1<<0
required	小部件是必需的 the widget is required	1<<1
noExport	不要导出小组件的值 don't export the widget value	1<<2
noToggleToOff	单选框组必选选择一个 radios one only must be on	1<<14
radio	单选法 added by the radio method	1<<15
pushButton	当按钮为公共按钮时 if the button is a push button	1<<16
radiosInUnison	单选框拥有一样的值时一起切换 radios with the same value toggle together	1<<25
multiline	用于多行文本小部件 for multiline text widget	1<<12
password	密码文本域 password textfield	1<<13
fileSelect	文件选择小部件 file selection widget	1<<20
doNotSpellCheck	不拼写检查 as it says	1<<22
doNotScroll	文本框不滚动 text fields do not scroll	1<<23
comb	根据最大长度值制作 comb 样式的文字 make a comb style text based on the maxlen value	1<<24
richText	如果使用富文本 if rich text is used	1<<25
combo	针对下拉框 for choice fields	1<<17
edit	如果选择是可编辑的 if the choice is editable	1<<18
sort	是否要对数值进行排序 if the values should be sorted	1<<19
multiSelect	如果选择允许多选 if the choice allows multi-select	1<<21
commitOnSelChange	reportlab 没有使用 not used by reportlab	1<<26

annotationFlags 参数

PDF 小部件是注释，并具有注释属性，这些属性显示在下面的表格中。

字段标志 Tokens 和 values		
Token	描述	值
readOnly	小部件只读 The widget is read only	1<<0
required	小部件是必需的 the widget is required	1<<1
noExport	不要导出小组件的值 don't export the widget value	1<<2
noToggleToOff	单选框组必选选择一个 radios one only must be on	1<<14
radio	单选法 added by the radio method	1<<15

字段标志 Tokens 和 values		
Token	描述	值
pushButton	当按钮为公共按钮时 if the button is a push button	1<<16
radiosInUnison	单选框拥有一样的值时一起切换 radios with the same value toggle together	1<<25
multiline	用于多行文本小组件 for multiline text widget	1<<12
password	密码文本域 password textfield	1<<13
fileSelect	文件选择小组件 file selection widget	1<<20
doNotSpellCheck	不拼写检查 as it says	1<<22
doNotScroll	文本框不滚动 text fields do not scroll	1<<23
comb	根据最大长度值制作 comb 样式的文字 make a comb style text based on the maxlen value	1<<24
richText	如果使用富文本 if rich text is used	1<<25
combo	针对下拉框 for choice fields	1<<17
edit	如果选择是可编辑的 if the choice is editable	1<<18
sort	是否要对数值进行排序 if the values should be sorted	1<<19
multiSelect	如果选择允许多选 if the choice allows multi-select	1<<21
commitOnSelChange	reportlab 没有使用 not used by reportlab	1<<26

第 5 章 PLATYPUS - 页面布局和排版

5.1 设计目标

Platypus是"Page Layout and Typography Using Scripts"的缩写。它是一个高水平的页面布局库，让你可以用最少的努力以编程方式创建复杂的文档。

Platypus的设计力求将 "高层次" 的布局决定与文档内容尽可能分开。例如，段落使用段落样式，页面使用页面模板，目的是让数百个有数千页的文件可以按照不同的样式规格重新格式化，只需在一个包含段落样式和页面布局规格的共享文件中修改几行即可。

Platypus的整体设计可以认为有几个层次，自上而下，这些是：

- DocTemplates作为文档的最外层容器。
- PageTemplates作为各种页面布局的规格。
- Frames页面中可包含流动文本或图形的区域规格。
- Flowables对应"flowed into the document"流入文档的文本或图形元素（即图像、段落和表格等内容，但不包括页脚或固定页面图形等内容）。

pdfgen.Canvas为最终从其他图层接收文档绘画的最低层。

DocTemplate

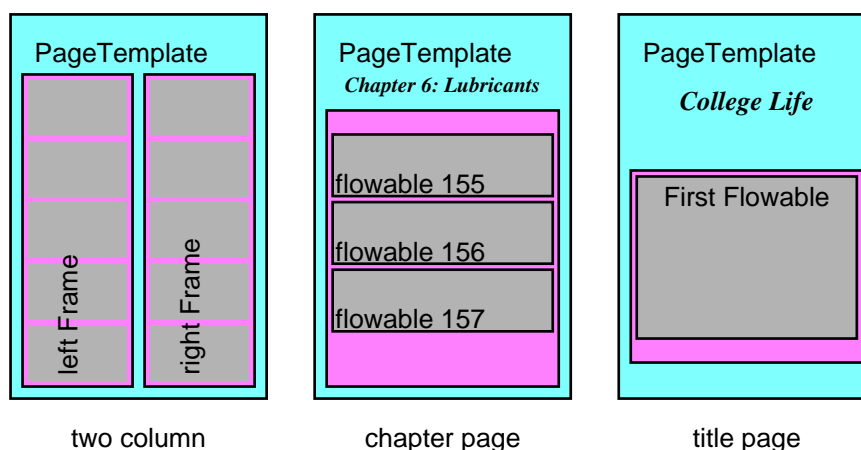


图 5 - 1 : DocTemplate 结构说明

上面的插图形象地说明了DocTemplate、PageTemplate和Flowables的概念。然而，它具有欺骗性，因为每一个PageTemplate实际上可以指定任何数量的页面的格式（而不是像从图中推断的那样只指定一个）。

DocTemplate包含一个或多个PageTemplate，每个PageTemplate包含一个或多个Frame。Flowables是指可以flowed(流入)Frame的东西，例如Paragraph或Table。

要使用platypus，你需要从DocTemplate类中创建一个文档，并向其build方法传递一个Flowables列表。document的build方法知道如何将flowable列表处理成合理的东西。

在内部，DocTemplate类使用各种事件来实现页面布局和格式化。每个事件都有一个对应的处理方法，称为handle_XXX，其中XXX是事件名称。一个典型的事件是frameBegin，它发生在机械开始第一次使用一个框架的时候。

Platypus故事由一系列基本元素组成，这些元素被称为Flowables，它们驱动着数据驱动的Platypus格式化引擎。为了修改引擎的行为，一种特殊的可流式元素ActionFlowables告诉布局引擎，例如，跳到下一列或者换成另一个PageTemplate。

5.2 开始

考虑以下代码序列，它为Platypus提供了一个非常简单的 "hello world" 例子。

```
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.rl_config import defaultPageSize
from reportlab.lib.units import inch
PAGE_HEIGHT=defaultPageSize[1]; PAGE_WIDTH=defaultPageSize[0]
styles = getSampleStyleSheet()
```

首先，我们从其他模块中导入一些构造函数、一些段落样式和其他方便。

```
Title = "Hello world"
pageinfo = "platypus example"
def myFirstPage(canvas, doc):
    canvas.saveState()
    canvas.setFont('Times-Bold',16)
    canvas.drawCentredString(PAGE_WIDTH/2.0, PAGE_HEIGHT-108, Title)
    canvas.setFont('Times-Roman',9)
    canvas.drawString(inch, 0.75 * inch, "First Page / %s" % pageinfo)
    canvas.restoreState()
```

我们用上面的函数定义文档首页的固定特征。

```
def myLaterPages(canvas, doc):
    canvas.saveState()
    canvas.setFont('Times-Roman',9)
    canvas.drawString(inch, 0.75 * inch, "Page %d %s" % (doc.page, pageinfo))
    canvas.restoreState()
```

由于我们希望第一个页面之后的页面看起来与第一个页面不同，我们为其他页面的固定特征定义了一个备用布局。请注意，上面的两个函数使用 **pdfgen** 级别的画布操作来为页面绘制注释。

```
def go():
    doc = SimpleDocTemplate("phello.pdf")
    Story = [Spacer(1,2*inch)]
    style = styles["Normal"]
    for i in range(100):
        bogustext = ("This is Paragraph number %s. " % i) * 20
        p = Paragraph(bogustext, style)
        Story.append(p)
        Story.append(Spacer(1,0.2*inch))
    doc.build(Story, onFirstPage=myFirstPage, onLaterPages=myLaterPages)
```

最后，我们创建一个"store"并构建文档。请注意，我们在这里使用的是" canned"(罐头)文档模板，它是预建的页面模板。我们还使用了预建的段落样式。我们在这里只使用了两种类型的"flowables"--Spacers和Paragraphs。第一个Spacer确保段落跳过标题字符串。

要查看这个示例程序的输出，请以"顶层脚本"的形式运行模块docs/userguide/doc_examples.py（来自ReportLab docs发行版）。脚本解释python doc_examples.py将生成Platypus输出phello.pdf。

5.3 Flowables

Flowables是可以被绘制的东西，它有wrap, draw和可能的split方法。Flowable是一个抽象的基类，用于绘制事物，一个实例知道它的大小，并在它自己的坐标系中绘制(这需要基API在调用Flowable.draw方法时提供一个绝对坐标系)。要获得一个实例，使用 **f=Flowable()**。

注意：Flowable类是一个抽象类，通常只作为基类使用。

为了说明使用Flowables的一般方式，我们将展示如何在画布上使用和绘制衍生类Paragraph。Paragraph是如此重要，它们将有一整章的篇幅来介绍。

```
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus import Paragraph
from reportlab.pdfgen.canvas import Canvas
styleSheet = getSampleStyleSheet()
style = styleSheet['BodyText']
P=Paragraph('This is a very silly example',style)
canv = Canvas('doc.pdf')
aW = 460 # available width and height
aH = 800
w,h = P.wrap(aW, aH) # find required space
if w<=aW and h<=aH:
    P.drawOn(canv,0,aH)
    aH = aH - h # reduce the available height
    canv.save()
else:
    raise ValueError, "Not enough room"
```

Flowable 方法

Flowable.draw()

这将被调用来要求 flowable 实际渲染自己。Flowable类没有实现draw。调用代码应该确保 flowable 有一个属性canv，它是pdfgen.Canvas，它应该被绘制到Canvas上，并且Canvas处于一个适当的状态(就翻译、旋转等而言)。通常这个方法只在内部被drawOn方法调用，派生类必须实现这个方法。派生类必须实现这个方法。

Flowable.drawOn(canvas,x,y)

这是控制引擎用来将flowable渲染到特定画布的方法。它处理转换为画布坐标(x,y)，并确保flowable有一个canv属性，这样draw方法(在基类中没有实现)就可以在一个绝对坐标框架中渲染。

Flowable.wrap(availWidth, availHeight)

在询问对象的大小、绘制或其他什么之前，这个函数将被包围的框架调用。它返回实际使用的尺寸。

Flowable.split(self, availWidth, availheight)

当wrap失败时，更复杂的框架会调用这个函数。愚蠢的flowables应该返回[]，这意味着它们无法拆分。聪明的flowables应该自己拆分并返回一个flowables列表。客户端代码要确保避免重复尝试拆分。如果空间足够，拆分方法应该返回[self]。否则，flowable应该重新排列，并返回一个按顺序考虑的flowable列表[f0,...]。实现的拆分方法应该避免改变self，因为这将允许复杂的布局机制在一个可流动的列表上进行多次传递。

5.4 流动定位的准则

有两种方法，默认情况下返回零，为可流动物的垂直间距提供指导。

```
Flowable.getSpaceAfter(self):
Flowable.getSpaceBefore(self):
```

这些方法会返回flowable后面或前面应该有多少空间。这些空间不属于flowable本身，也就是说，flowable的draw方法在渲染时不应该考虑它。控制程序将使用返回的值来确定上下文中特定flowable需要多少空间。

所有的flowables都有一个hAlign属性：('LEFT','RIGHT','CENTER'或'CENTRE')。对于占满整个框架宽度的段落，这个属性没有影响。对于小于框架宽度的表格、图像或其他对象，这决定了它们的水位置。

下面的章节将涵盖最重要的特定类型的可流动文件，段落和表格。

5.5 Frames

Frames是活动的容器，它本身就包含在PageTemplate中，Frames有一个位置和大小，并保持一个剩余可绘制空间的概念。如：

```
Frame(x1, y1, width,height, leftPadding=6, bottomPadding=6,
      rightPadding=6, topPadding=6, id=None, showBoundary=0)
```

创建一个左下角坐标为(x1,y1)的Frame实例(在使用时相对于画布)，尺寸为 width x height。Padding参数是用于减少绘画空间的正量。参数id是运行时使用的标识符，例如"LeftColumn"或"RightColumn"等。如果showBoundary参数是非零，那么框架的边界将在运行时被绘制出来（这有时很有用）。

Frame 方法

```
Frame.addFromList(drawlist, canvas)
```

消耗drawlist前面的Flowables，直到帧满为止。如果不能容纳一个对象，则引发一个异常。

```
Frame.split(flowable,canv)
```

要求flowable使用可用空间进行分割，并返回flowable的列表。

```
Frame.drawBoundary(canvas)
```

将框架边界画成一个矩形（主要用于调试）。

使用 Frames

Frames可以直接与canvases和flowables一起使用来创建文档。Frame.addFromList方法为你处理wrap 和 drawOn调用。你不需要所有的Platypus引擎来获得有用的东西到PDF中。

```
from reportlab.pdfgen.canvas import Canvas
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib.units import inch
from reportlab.platypus import Paragraph, Frame
styles = getSampleStyleSheet()
styleN = styles['Normal']
styleH = styles['Heading1']
story = []

#add some flowables
story.append(Paragraph("This is a Heading",styleH))
story.append(Paragraph("This is a paragraph in <i>Normal</i> style.",
    styleN))
c = Canvas('mydoc.pdf')
f = Frame(inch, inch, 6*inch, 9*inch, showBoundary=1)
f.addFromList(story,c)
c.save()
```

5.6 文档和模板

BaseDocTemplate类实现了文档格式化的基本机制。该类的一个实例包含了一个或多个PageTemplate的列表，这些PageTemplate可用于描述单页信息的布局。build方法可用于处理Flowables列表，以生成一个PDF文档。

BaseDocTemplate

```
BaseDocTemplate(self, filename,
                 pagesize=defaultPageSize,
                 pageTemplates=[],
                 showBoundary=0,
                 leftMargin=inch,
                 rightMargin=inch,
                 topMargin=inch,
                 bottomMargin=inch,
                 allowSplitting=1,
                 title=None,
                 author=None,
                 _pageBreakQuick=1,
                 encrypt=None)
```

创建一个适合创建基本文档的文档模板。它带有相当多的内部机制，但没有默认的面模板。所需的 **filename** 可以是一个字符串，一个用于接收创建的PDF文档的文件名；也可以是一个有 **write** 方法的对象，如 **BytesIO** 或 **file** 或 **socket**。

允许的参数应该是不言自明的，但是 **showBoundary** 控制是否绘制 **Frame** 的边界，这对于调试来说是很有用的。**allowSplitting** 参数决定了内置方法是否应该尝试 **split** 单个 **Flowables** 跨越 **Frame**。**_pageBreakQuick** 参数决定了在结束页面之前，是否应该尝试结束页面上的所有框架。**encrypt** 参数决定了是否对文档进行加密，以及如何加密。默认情况下，文档是不加密的。如果 **encrypt** 是一个字符串对象，那么它将作为pdf的用户密码。如果 **encrypt** 是一个 **reportlab.lib.pdfencrypt.StandardEncryption** 的实例，那么这个对象就被用来加密pdf。这允许对加密设置进行更精细的控制。

BaseDocTemplate 方法

这些都是客户程序员直接关心的问题，因为他们通常会被使用。

```
BaseDocTemplate.addPageTemplates(self, pageTemplates)
```

此方法用于在现有文档中添加一个或一系列 **PageTemplate**。

```
BaseDocTemplate.build(self, flowables, filename=None,
                      canvasmaker=canvas.Canvas)
```

这是应用程序程序员感兴趣的主要方法。假设文档实例被正确设置，**build** 方法将 **story** 以 **flowables** 列表的形式接收（**flowables** 参数），并在列表中循环，将 **flowables** 列表一次一个地强制通过格式化机制。实际上，这使得 **BaseDocTemplate** 实例发出对实例 **handle_XXX** 方法的调用来处理各种事件。

BaseDocTemplate 抽象方法

这些在基类中根本没有语义。它们的目的是作为布局机制的纯虚拟钩子。紧接派生类的创建者可以覆盖这些，而不用担心影响布局引擎的属性。

```
BaseDocTemplate.afterInit(self)
```

这个方法在基类初始化后被调用；派生类可以覆盖该方法来添加默认的 **PageTemplates**。

```
BaseDocTemplate.afterPage(self)
```

这是在页面处理后，紧接着当前页面模板的 **afterDrawPage** 方法被调用。一个派生类可以使用这个方法来做一些依赖于页面信息的事情，比如字典页面上的首字和尾字。

```
BaseDocTemplate.beforeDocument(self)
```

在对文档进行任何处理之前，但在处理机制准备好之后，就会调用这个函数，因此它可以用来对实例的 **pdfgen.canvas** 等进行处理。因此，它可以用来对实例的 **pdfgen.canvas** 等进行操作。

```
BaseDocTemplate.beforePage(self)
```

这是在页面处理开始时，在当前页面模板的 **beforeDrawPage** 方法之前调用的。它可以用来重置页面特定的信息持有者。

```
BaseDocTemplate.filterFlowables(self,flowables)
```

在主 `handle_flowable` 方法开始时，调用这个函数来过滤`flowables`。在返回时，如果`flowables[0]`被设置为`None`，则会被丢弃，主方法立即返回。

```
BaseDocTemplate.afterFlowable(self, flowable)
```

在`flowable`被渲染后调用。有兴趣的类可以使用这个钩子来收集特定页面或框架上存在的信息。

BaseDocTemplate 事件处理

这些方法构成了布局引擎的主要部分。程序员不应该直接调用或覆盖这些方法，除非他们试图修改布局引擎。当然，有经验的程序员如果想在某个特定的事件，即XXX处进行干预，而这个事件并不对应于其中的一个虚拟方法，那么总是可以覆盖并调用`drived`类版本中的基方法。我们为每个处理方法提供了一个基类同义词，名称相同，前缀为下划线"`_`"，这样就很容易了。

```
def handle_pageBegin(self):
    doStuff()
    BaseDocTemplate.handle_pageBegin(self)
    doMoreStuff()

#using the synonym
def handle_pageEnd(self):
    doStuff()
    self._handle_pageEnd()
    doMoreStuff()
```

在这里我们列出这些方法只是为了说明正在处理的事件。有兴趣的程序员可以看一下源码。

```
handle_currentFrame(self,fx)
handle_documentBegin(self)
handle_flowable(self,flowables)
handle_frameBegin(self,*args)
handle_frameEnd(self)
handle_nextFrame(self,fx)
handle_nextPageTemplate(self,pt)
handle_pageBegin(self)
handle_pageBreak(self)
handle_pageEnd(self)
```

使用文档模板可以非常简单，`SimpleDocTemplate`是由`BaseDocTemplate`派生出来的一个类，它提供了自己的`PageTemplate`和`Frame`设置。

```
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib.pagesizes import letter
from reportlab.platypus import Paragraph, SimpleDocTemplate
styles = getSampleStyleSheet()
styleN = styles['Normal']
styleH = styles['Heading1']
story = []

#add some flowables
story.append(Paragraph("This is a Heading",styleH))
story.append(Paragraph("This is a paragraph in <i>Normal</i> style.",
    styleN))
doc = SimpleDocTemplate('mydoc.pdf',pagesize = letter)
doc.build(story)
```

PageTemplates

`PageTemplate`类是一个语义相当简单的容器类。每个实例都包含一个`Frames`的列表，并且有一些方法应该在每个页面的开始和结束时被调用。

```
PageTemplate(id=None,frames=[],onPage=_doNothing,onPageEnd=_doNothing)
```

用于初始化一个实例，`frames`参数应该是一个`Frames`的列表，而可选的`onPage`和`onPageEnd`参数是可调用的，它们的签名应该是 `def XXX(canvas,document)`，其中`canvas`和`document`是正在绘制的画布和文档。这些例程的目的是用来绘制页面的非流动（即标准）部分。这些属性函数与纯虚拟方法 `PageTemplate.beforePage` 和 `PageTemplate.afterPage`完全平行，这两个方法的签名是 `beforePage(self,canvas,document)`。这些方法允许使用类派生来定义标准行为，而属性则允许改变实例

。在运行时，`id` 参数用于执行 `PageTemplate` 的切换，所以 `id='FirstPage'` 或 `id='TwoColumns'` 是典型的。

第 6 章 Paragraphs - 文本段落

`reportlab.platypus.Paragraph`类是Platypus Flowables中最有用的一个；它可以格式化相当任意的文本，并提供了使用XML样式标记的内联字体样式和颜色变化。格式化后的文本的整体形状可以是公正的，左右粗细的，或者居中的。XML标记甚至可以用来插入希腊字符或做下标。

下面的文字创建了一个Paragraph类的实例。

```
Paragraph(text, style, bulletText=None)
```

参数`text`包含了段落的文本；在文本的结尾和换行后，多余的空白会被删除。这允许在Python脚本中轻松使用缩进的三引号文本。`bulletText`参数提供了段落的默认子弹文本。段落文本和子弹的字体和其他属性可以使用样式参数来设置。

参数 `style` 应该是一个 `ParagraphStyle` 类的实例，通常使用

```
from reportlab.lib.styles import ParagraphStyle
```

这个容器类以结构化的方式提供了多个默认段落属性的设置。这些样式被安排在一个名为`stylesheet`的字典样式对象中，它允许以`stylesheet['BodyText']`的形式访问这些样式。我们提供了一个示例样式表。

```
from reportlab.lib.styles import getSampleStyleSheet
stylesheet=getSampleStyleSheet()
normalStyle = stylesheet['Normal']
```

可以为Paragraph设置的选项可以从ParagraphStyle默认值中看出。前面带下划线('_')的值来自于`reportlab.rl_config`模块中的默认值，这些值来自于`reportlab.rl_settings`模块。

6.1 ParagraphStyle

```
class ParagraphStyle(PropertySet):
    defaults = {
        'fontName': _baseFontName,
        'fontSize': 10,
        'leading': 12,
        'leftIndent': 0,
        'rightIndent': 0,
        'firstLineIndent': 0,
        'alignment': TA_LEFT,
        'spaceBefore': 0,
        'spaceAfter': 0,
        'bulletFontName': _baseFontName,
        'bulletFontSize': 10,
        'bulletIndent': 0,
        'textColor': black,
        'backColor': None,
        'wordWrap': None,
        'borderWidth': 0,
        'borderPadding': 0,
        'borderColor': None,
        'borderRadius': None,
        'allowWidows': 1,
        'allowOrphans': 0,
        'textTransform': None,
        'endDots': None,
        'splitLongWords': 1,
        'underlineWidth': _baseUnderlineWidth,
        'bulletAnchor': 'start',
        'justifyLastLine': 0,
        'justifyBreaks': 0,
        'spaceShrinkage': _spaceShrinkage,
        'strikeWidth': _baseStrikeWidth, #stroke width
        'underlineOffset': _baseUnderlineOffset, #fraction of fontsize
        to
        offset underlines
        'underlineGap': _baseUnderlineGap, #gap for double/triple
        underline
        'strikeOffset': _baseStrikeOffset, #fraction of fontsize to offset
```

```

strikethrough
'strikeGap': _baseStrikeGap,    #gap for double/triple strike
'linkUnderline': _platypus_link_underline,
# 'underlineColor': None,
# 'strikeColor': None,
'hyphenationLang': _hyphenationLang,
'uriWasteReduce': _uriWasteReduce,
'embeddedHyphenation': _embeddedHyphenation,
}

```

使用文字段落样式: ParagraphStyle

Paragraph和ParagraphStyle类一起处理大多数常见的格式化需求。下面的示例以不同的样式绘制段落，并添加了一个边界框，这样你就可以看到确切的空间被占用了。

```

alignment = 0
allowOrphans = 0
allowWidows = 1
backColor = None
borderColor = None
borderPadding = 0
borderRadius = None
borderWidth = 0
bulletAnchor = start
bulletFontName = Helvetica
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 0
fontName = SourceHanSans-ExtraLight
fontSize = 10
hyphenationLang =
justifyBreaks = 0
justifyLastLine = 0
leading = 12
leftIndent = 0
linkUnderline = 0
rightIndent = 0
spaceAfter = 0
spaceBefore = 6
spaceShrinkage = 0.05
splitLongWords = 1
strikeGap = 1
strikeOffset = 0.25*F
strikeWidth =
textColor = Color(0,0,0,1)
textTransform = None
underlineGap = 1
underlineOffset = -0.125*F
underlineWidth =
uriWasteReduce = 0
wordWrap = None

```

你在此被指控在1970年5月28日，
你故意，非法，并与恶意的预想，
出版一个所谓的英语 -
匈牙利短语书，意图造成破坏和平。
你如何辩护？

图 6-1: 默认 ParagraphStyle

spaceBefore和**spaceAfter**这两个属性如它们所说的那样，除了在一个框架的顶部或底部。在一个框架的顶部，**spaceBefore**被忽略，而在底部，**spaceAfter**被忽略。这意味着你可以指定一个'Heading 2'样式在页面中间出现时，它之前有两英寸的空间，但不会在页面顶部得到数英亩的空白。这两个属性应该被认为是被Frame的"请求"，而不是段落本身所占空间的一部分。

fontSize和**fontName**标签是显而易见的，但重要的是设置**leading**。

这是相邻文本行之间的间距；一个好的经验法则是让这个间距比点的大小大 20%。要获得双倍行距的文本，请使用较高的**leading**。如果您将**autoLeading**(默认为"off")设置为"min"(使用观察到的前导，即使比指定的小)或"max"(使用观察到的和指定的较大值)，那么就会尝试逐行确定前导。如果行

中包含不同的字体大小等，这可能是有用的。

下图为前后空间和增加的引导。

```
alignment = 0
allowOrphans = 0
allowWidows = 1
backColor = None
borderColor = None
borderPadding = 0
borderRadius = None
borderWidth = 0
bulletAnchor = start
bulletFontName = Helvetica
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 0
fontName = SourceHanSans-ExtraLight
fontSize = 10
hyphenationLang =
justifyBreaks = 0
justifyLastLine = 0
leading = 16
leftIndent = 0
linkUnderline = 0
rightIndent = 0
spaceAfter = 6
spaceBefore = 6
spaceShrinkage = 0.05
splitLongWords = 1
strikeGap = 1
strikeOffset = 0.25*F
strikeWidth =
textColor = Color(0,0,0,1)
textTransform = None
underlineGap = 1
underlineOffset = -0.125*F
underlineWidth =
uriWasteReduce = 0
wordWrap = None
```

你在此被指控在1970年5月28日，
你故意，非法，并与恶意的预想，
出版一个所谓的英语 -
匈牙利短语书，意图造成破坏和平。
你如何辩护？

图 6-2: 前后空间和增加 leading

属性 **borderPadding** 调整段落与背景边框之间的 **padding**。它可以是一个单一的值，也可以是一个包含2到4个值的元组。这些值的应用方式与层叠样式表（CSS）相同。如果给定一个值，该值将应用于所有四条边框。如果给了一个以上的值，则从顶部开始按顺时针顺序应用到边上。如果给了两个或三个值，则缺失的值将从相反的边开始应用。请注意，在下面的例子中，黄色方框是由段落本身绘制的。

```
alignment = 0
allowOrphans = 0
allowWidows = 1
backColor = #FFFF00
borderColor = #000000
borderPadding = (7, 2, 20)
borderRadius = None
borderWidth = 1
bulletAnchor = start
bulletFontName = Helvetica
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 0
fontName = SourceHanSans-ExtraLight
fontSize = 10
hyphenationLang =
justifyBreaks = 0
justifyLastLine = 0
leading = 12
leftIndent = 0
linkUnderline = 0
rightIndent = 0
spaceAfter = 0
spaceBefore = 0
spaceShrinkage = 0.05
splitLongWords = 1
strikeGap = 1
strikeOffset = 0.25*F
strikeWidth =
textColor = Color(0,0,0,1)
textTransform = None
underlineGap = 1
underlineOffset = -0.125*F
underlineWidth =
uriWasteReduce = 0
wordWrap = None
```

你在此被指控在1970年5月28日，
你故意，非法，并与恶意的预想，
出版一个所谓的英语 -
匈牙利短语书，意图造成破坏和平。
你如何辩护？

图 6-3: 可变 padding

`leftIndent` 和 `rightIndent` 属性的作用正是你所期望的；`firstLineIndent` 被添加到第一行的 `leftIndent` 中。如果你想要一个笔直的左边缘，记得将 `firstLineIndent` 等于 0。

```
alignment = 0
allowOrphans = 0
allowWidows = 1
backColor = None
borderColor = None
borderPadding = 0
borderRadius = None
borderWidth = 0
bulletAnchor = start
bulletFontName = Helvetica
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 24
fontName = SourceHanSans-ExtraLight
fontSize = 10
hyphenationLang =
justifyBreaks = 0
justifyLastLine = 0
leading = 12
leftIndent = 24
linkUnderline = 0
rightIndent = 24
spaceAfter = 0
spaceBefore = 0
spaceShrinkage = 0.05
splitLongWords = 1
strikeGap = 1
strikeOffset = 0.25*F
strikeWidth =
textColor = Color(0,0,0,1)
textTransform = None
underlineGap = 1
underlineOffset = -0.125*F
underlineWidth =
uriWasteReduce = 0
wordWrap = None
```

你在此被指控在1970年
5月28日，你故意，非法，
并与恶意的预想，
出版一个所谓的英语 - 匈牙利
短语书，意图造成破坏和平
。你如何辩护？

图 6-4: 左右缩进三分一，首行缩进三分二

将`firstLineIndent`设置为负数，`leftIndent`则高得多，并使用不同的字体（我们稍后会告诉你怎么做！）可以给你一个定义列表：

```

alignment = 0
allowOrphans = 0
allowWidows = 1
backColor = None
borderColor = None
borderPadding = 0
borderRadius = None
borderWidth = 0
bulletAnchor = start
bulletFontName = Helvetica
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 0
fontName = SourceHanSans-ExtraLight
fontSize = 10
hyphenationLang =
justifyBreaks = 0
justifyLastLine = 0
leading = 12
leftIndent = 36
linkUnderline = 0
rightIndent = 0
spaceAfter = 0
spaceBefore = 0
spaceShrinkage = 0.05
splitLongWords = 1
strikeGap = 1
strikeOffset = 0.25 * F
strikeWidth =
textColor = Color(0,0,0,1)
textTransform = None
underlineGap = 1
underlineOffset = -0.125 * F
underlineWidth =
uriWasteReduce = 0
wordWrap = None

```

皮克尔斯法官: 你在此被指控在
1970年5月28日, 你故意, 非法
, 并与恶意的预想,
出版一个所谓的英语 - 匈牙利短
语书, 意图造成破坏和平。
你如何辩护?

图 6-5: 左右缩进三分一, 首行缩进三分二

在模块 `reportlab.lib.enums` 中, `alignment` 有四个可能的值, 定义为常量。这些值是 `TA_LEFT`, `TA_CENTER` 或 `TA_CENTRE`, `TA_RIGHT` 和 `TA_JUSTIFY`, 值分别为 0, 1, 2 和 4。这些都和你所期望的一样。

将 `wordWrap` 设置为 'CJK' 来获得亚洲语言的换行。对于普通的西方文本, 你可以通过 `allowWidows` 和 `allowOrphans` 来改变断行算法处理 *widows* 和 *orphans* 的方式。这两个值通常都应该设置为 0, 但由于历史原因, 我们允许 *widows*。文本的默认颜色可以用 `textColor` 设置, 段落的背景颜色可以用 `backColor` 设置。段落的边框属性可以使用 `borderWidth`, `borderPadding`, `borderColor` 和 `borderRadius` 来改变。

`textTransform` 属性可以是 *None*, `uppercase` 或 `lowercase` 得到明显的结果, `capitalize` 得到初始字母大写。

属性 `endDots` 可以是 *None*, 一个字符串, 或者一个对象, 其属性为 `text` 和可选的 `fontName`、`fontSize`、`textColor`、`backColor` 和 `dy(y offset)`, 用于指定左/右对齐段落最后一行的尾部内容。

`splitLongWords` 属性可以设置为假值, 以避免拆分非常长的单词。

属性 `bulletAnchor` 可以是 '*start*', '*middle*', '*end*' 或 '*numeric*' 来控制子弹的锚定位置。

`justifyBreaks` 属性控制了是否应该用 `
` 标签故意断行。

属性 `spaceShrinkage` 是一个小数, 指定段落行的空间可以缩小多少以使其合适; 通常是 0.05 左右。

当使用 `<u>` 或链接标签时，`underlineWidth`、`underlineOffset`、`underlineGap` 和 `underlineColor`属性控制了下划线行为。这些标签可以有这些属性的覆盖值。`width` 和 `offset` 的属性值是一个 `fraction * Letter`，其中 `letter` 可以是 `P`、`L`、`f` 或 `F` 中的一个，代表字体大小比例。`P` 使用标签处的字体大小，`F` 是标签中的最大字体大小，`f` 是标签内的初始字体大小。`L` 表示全局（`ParagraphStyle`）字体大小。`strikeWidth`、`strikeOffset`、`strikeGap` 和 `strikeColor`属性对删除线有同样的作用。

属性 `linkUnderline` 控制链接标签是否自动下划线。

如果安装了 `pyphen` python 模块，则属性 `hyphenationLang` 控制哪种语言将被用于在没有明确嵌入连字符的情况下连字符。

如果设置了 `embeddedHyphenation`，那么就会尝试拆分带有嵌入式连字符的单词。

属性 `uriWasteReduce` 控制我们如何尝试分割长的 `uri`。它是我们认为太过浪费的行的分数，在模块 `reportlab.rl_settings` 中的默认值是0.5。这意味着如果我们将浪费至少一半的行数，我们将尝试拆分一个看起来像 `uri` 的单词。

目前，连字符 和 `uri` 分割是默认关闭的。您需要通过使用 `~/rl_settings` 文件或在 python 路径中添加 `reportlab_settings.py` 模块来修改默认设置。合适的值是

```
hyphenationLanguage='en_GB'
embeddedHyphenation=1
uriWasteReduce=0.3
```

6.2 文本段落XML标记标签

XML标记可以用来修改或指定整体段落样式，也可以指定段落内的标记。

最外层 `< para >` 标签

段落文本可以选择由 `<para attributes...> </para>` 标签包围。开头的 `<para>` 标签的任何属性都会影响 `Paragraph text` 和/或 `bulletText`所使用的样式。

属性	同义词
<code>alignment</code>	<code>align</code> , <code>alignment</code>
<code>allowOrphans</code>	<code>allowOrphans</code> , <code>alloworphans</code>
<code>allowWidows</code>	<code>allowWidows</code> , <code>allowwidows</code>
<code>autoLeading</code>	<code>autoLeading</code> , <code>autoleading</code>
<code>backColor</code>	<code>backColor</code> , <code>backcolor</code> , <code>bg</code> , <code>bgcolor</code>
<code>borderColor</code>	<code>borderColor</code> , <code>bordercolor</code>
<code>borderRadius</code>	<code>borderRadius</code> , <code>borderradius</code>
<code>borderWidth</code>	<code>borderWidth</code> , <code>borderwidth</code>
<code>borderpadding</code>	<code>borderpadding</code>
<code>bulletAnchor</code>	<code>banchor</code> , <code>bulletAnchor</code> , <code>bulletanchor</code>
<code>bulletColor</code>	<code>bcolor</code> , <code>bulletColor</code> , <code>bulletcolor</code>
<code>bulletFontName</code>	<code>bfont</code> , <code>bulletFontName</code> , <code>bulletfontname</code>
<code>bulletFontSize</code>	<code>bfontsize</code> , <code>bulletFontSize</code> , <code>bulletfontsize</code>
<code>bulletIndent</code>	<code>bindent</code> , <code>bulletIndent</code> , <code>bulletindent</code>
<code>bulletOffsetY</code>	<code>boffsety</code> , <code>bulletOffsetY</code> , <code>bulletoffsety</code>
<code>embeddedHyphenation</code>	<code>embeddedHyphenation</code> , <code>embeddedhyphenation</code>
<code>endDots</code>	<code>endDots</code> , <code>enddots</code>
<code>firstLineIndent</code>	<code>findent</code> , <code>firstLineIndent</code> , <code>firstlineindent</code>
<code>fontName</code>	<code>face</code> , <code>font</code> , <code>fontName</code> , <code>fontname</code>
<code>fontSize</code>	<code>fontSize</code> , <code>fontsize</code> , <code>size</code>
<code>hyphenationLang</code>	<code>hyphenationLang</code> , <code>hyphenationLanguage</code> , <code>hyphenationlang</code>
<code>hyphenationMinWordLength</code>	<code>hyphenationMinWordLength</code> , <code>hyphenationminwordlength</code>

hyphenationOverflow	hyphenationOverflow, hyphenationoverflow
justifyBreaks	justifyBreaks, justifybreaks
justifyLastLine	justifyLastLine, justifylastline
leading	leading
leftIndent	leftIndent, leftindent, lindent
rightIndent	rightIndent, rightindent, rindent
spaceAfter	spaceAfter, spacea, spaceafter
spaceBefore	spaceBefore, spaceb, spacebefore
spaceShrinkage	spaceShrinkage, spaceshrinkage
splitLongWords	splitLongWords, splitlongwords
strikeColor	strikeColor, strikecolor
strikeGap	strikeGap, strikegap
strikeOffset	strikeOffset, strikeoffset
strikeWidth	strikeWidth, strikewidth
textColor	color, fg, textColor, textcolor
textTransform	textTransform, texttransform
underlineColor	underlineColor, underlinecolor
underlineGap	underlineGap, underlinegap
underlineOffset	underlineOffset, underlineoffset
underlineWidth	underlineWidth, underlinewidth
uriWasteReduce	uriWasteReduce, uriwastereduce
wordWrap	wordWrap, wordwrap

表 6-3 - 样式属性的同义词

我们为我们的 Python 属性名提供了一些有用的同义词，包括小写版本，以及 HTML 标准中存在的等价属性。这些增加的内容使构建 XML 打印应用程序变得更加容易，因为许多段内标记可能不需要翻译。下表显示了最外层段落标签中允许的属性和同义词。

6.3 段内标记

< ! [CDATA[在每个段落中，我们使用一组基本的XML标签来提供标记。其中最基本的是粗体 (...)、斜体 (<i>...</i>) 和下划线 (<u>...</u>)。其他允许的标签有强调 (..)，和删除线(<strike>...</strike>)。<link>和<a>标签可用于引用当前文档中的URI、文档或书签。<a> 标签的 **a** 变体可用于标记文档中的一个位置。也允许使用 **break** (
)标签。]]>。

```
<font name=SourceHanSans-Normal><nobr>  
>兹指控</nobr></font>你于1970年5月28日  
故意、非法和恶意地预谋出版一本所谓的英  
匈短语书，意图破坏和平。  
<u>你如何辩护</u>？
```

兹指控你于1970年5月28日故意、非法和恶意地预谋出版一本所谓的英匈短语书，意图破坏和平。
你如何辩护？

图 6-6: 简单的黑体和斜体标签

```
这是一个锚标签的<a href="#MYANCHOR"  
color="blue">链接</a>，即<a  
name="MYANCHOR"/><font color="green">  
这里</font>。这是另一个指向同一锚标签的<  
link href="#MYANCHOR" color="blue" fontNa  
me="SourceHanSans-ExtraLight">链接</link  
>。
```

这是一个锚标签的链接，即这里。
这是另一个指向同一锚标签的链接。

图 6-7: 锚和链接

link标签可以用作参考，但不能用作锚。a和link超链接标签有附加属性fontName、fontSize、color和 backgroundColor属性。超链接引用可以有http:（外部网页）、pdf:（不同的pdf文档）或docume

nt: (相同的pdf文档) 等方案; 缺少的方案将被视为document, 就像引用以#开头时的情况一样 (在这种情况下, 锚应该省略它)。任何其他方案都会被视为某种URI。

<pre>兹指控你于1970年5月28日 故意、非法和<strike>恶意地预谋</strike>
出版一本所谓的英匈短语书, 你怎么辩解 ?</pre>	<pre>兹指控你于1970年5月28日故意、非 法和恶意地预谋 出版一本所谓的英匈短语书, 你怎 么辩解?</pre>
---	---

图 6-8: 强调, 删除线, 和换行标签

 标签

标签可以用来改变段落中任何子串的字体名称、大小和文本颜色。法定属性有size、face、name (与face相同)、color和fg (与color相同)。name是字体家族的名称, 没有任何'bold'或'italic'的后缀。颜色可以是HTML的颜色名称, 也可以是以各种方式编码的十六进制字符串; 请参见`reportlab.lib.colors`了解允许的格式。

<pre>你在此被控于1970年5月28日 故意、非法和怀着预谋的恶意 出版一本所谓的英匈短语书, 意图破 坏和平。你如何辩护?</pre>	<pre>你在此被控于1970年5月28日故意、 非法和怀着预谋的恶意出 版一本所谓的英匈短语书, 意图破 坏和平。你如何辩护?</pre>
--	--

图 6-9: font 标签

上标和下标

上标和下标是由<![CDATA[<sup>和<sub>标签支持的, 它们的工作原理和你所期望的完全一样。另外这三个标签还有属性 rise 和 size, 可以选择设置上标/下标文本的上升/下降和字体大小。此外, 大多数希腊字母可以通过使用<greek>和</greek>标签, 或者使用mathML实体名来访问。]]>。

<pre>等式 (&alpha;) 。 <greek>e</greek> <sup> rise=9 size=6><greek>ip</greek></sup>=-1。</pre>	<pre>等式 (α) 。 ε^π = -1。</pre>
---	---

图 6-10: 希腊字母和上标

内联图片


我们可以用标签在段落中嵌入图片, 该标签有src、width、height等属性, 其含义很明显。v align属性可以设置为类似css的值, 从 "baseline"、"sub"、"super"、"top"、"text-top"、"middle"、"bottom"、"text-bottom"; 该值也可以是一个数字百分比或绝对值。


```

<para autoLeading="off" fontSize=12>This
&lt;img/&gt;  is aligned
<b>top</b>.<br/><br/>This &lt;img/&gt;
 is aligned
<b>bottom</b>.<br/><br/> This &lt;img/&gt;
 is aligned
<b>middle</b>.<br/><br/>This &lt;img/&gt;
 is aligned
<b>-4</b>.<br/><br/>This &lt;img/&gt;  is aligned
<b>+4</b>.<br/><br/>This &lt;img/&gt; 
has width<b>10</b>.<br/></para>

```

This  is aligned top.

This  is aligned bottom.

This  is aligned middle.

This  is aligned -4.

This  is aligned +4.

This  has width10.

图 6-11: 内联图片

`src`属性可以指向一个远程位置，例如`src="https://www.reportlab.com/images/logo.gif"`。默认情况下，我们将`rl_config.trustedSchemes` 设置为`['https','http','file','data','ftp']`和`rl_config.trustedHosts=None`，后者意味着没有限制。你可以使用覆盖文件来修改这些变量，例如`reportlab_settings.py`或`~/.reportlab_settings`。或者在环境变量`RL_trustedSchemes` & `RL_trustedHosts`中以逗号分隔。请注意，`trustedHosts`值可能包含glob野车，因此`*.reportlab.com`将匹配明显的域。

NB使用`trustedHosts`和/或`trustedSchemes`可能无法控制行为，以及当URI模式被查看器应用程序检测到时的操作。

<u> 和 <strike> 标签

这些标签可用于执行明确的下划线或删除线。这些标签的属性有`width`、`offset`、`color`、`gap` 和 `kind`。`kind`属性控制将绘制多少行(默认`kind=1`)，当`kind>1`时，`gap`属性控制行与行之间的间隔。

<nobr> 标签

如果使用连字符，`<nobr>`标签会抑制它，所以`<nobr>`一个非常长的单词不会被打断`</nobr>`不会被打断。

文字段落和列表的编号

`<seq>`标签为编号列表、章节标题等提供了全面的支持，它作为`reportlab.lib.sequencer`中`Sequencer`类的接口。它是`reportlab.lib.sequencer`中`Sequencer`类的接口。这些都是用来对整个文档中的标题和数字进行编号的。您可以创建任意多的独立的

"计数器"，并通过`id`属性进行访问；每次访问时，这些计数器都将以1为单位递增。

`seqreset`标签可以重置计数器。如果你想让它从1以外的数字开始恢复，使用语法`<seqreset id="mycounter" base="42">`。让我们开始吧。

```

<seq id="spam"/>, <seq id="spam"/>, <seq
id="spam"/>. 重置<seqreset id="spam"/>.
<seq id="spam"/>, <seq id="spam"/>, <seq
id="spam"/>.

```

1, 2, 3. 重置. 1, 2, 3.

图 6-12: 基本序列

你可以通过使用`<seqdefault id="Counter">`

标签指定一个计数器ID作为default来节省指定ID的时间；然后每当没有指定计数器ID时就会使用它。这样可以节省一些输入，特别是在做多级列表的时候；您只需在进入或退出一个级别时更改计数器ID。

```
<seqdefault id="spam"/>Continued...
<seq/><seq/>, <seq/>, <seq/>, <seq/>,
<seq/>, <seq/>.
```

Continued... 4,5, 6, 7, 8, 9, 10.

图 6-13: 默认序列

最后，我们可以使用Python字符串格式化的变体和<seq>标签中的**template**属性来访问多级序列。这是用来做所有数字中的标题，以及二级标题。子串 **%(counter)s** 提取了一个计数器的当前值，但不递增；在 **%(counter)s** 中添加一个加号，使计数器递增。数字标题使用了类似下面的模式。

```
图 <seq
template="%(Chapter)s-%(FigureNo+)s"/> -
多级模板
```

图 6-1 - 多级模板

图 6-14: 多级模板

我们做了点小手脚--真正的文档用的是 "Figure"，但上面的文字用的是 "FigureNo"
--否则我们会把编号弄乱的

6.4 项目符号和段落编号

除了三个缩进属性之外，还需要一些其他参数来正确处理带项目符号和编号的列表。我们在这里讨论这个问题，因为你现在已经看到了如何处理编号。一个段落可以有一个可选的**bulletText**参数传递给它的构造函数；或者，项目符号文本可以放在它头部的标签中。这段文字将被绘制在段落的第一行，其X原点由样式的**bulletIndent**属性决定，字体由**bulletFontName**属性给出。"项目符号"可以是一个单一的字符，如（啵！）一个项目符号，或者是一个文本片段，如一些编号序列中的数字，甚至是定义列表中使用的简短标题。字体可能提供各种项目编号字符，但我们建议首先尝试Unicode项目编号(•)，可以写成**•**，和**#x2022;**或(utf8中) **\xe2\x80\xa2**：

属性	同义词
bulletAnchor	anchor, bulletAnchor, bulletanchor
bulletColor	bulletColor, bulletcolor, color, fg
bulletFontName	bulletFontName, bulletfontname, face, font
bulletFontSize	bulletFontSize, bulletfontsize, fontsize, size
bulletIndent	bulletIndent, bulletindent, indent
bulletOffsetY	bulletOffsetY, bulletoffsety, offsety

表 6-4 - <bullet> 属性和同义词

在一个给定的段落中，<bullet>标签只允许使用一次，】它的使用会覆盖在**Paragraph**创建中指定的隐含的项目符号样式和 **bulletText**。（项目符号文本）

```
alignment = 0
allowOrphans = 0
allowWidows = 1
backColor = None
borderColor = None
borderPadding = 0
borderRadius = None
borderWidth = 0
bulletAnchor = start
bulletFontName = Symbol
bulletFontSize = 10
bulletIndent = 18
embeddedHyphenation = 0
endDots = None
firstLineIndent = 0
fontName = SourceHanSans-ExtraLight
fontSize = 10
hyphenationLang =
justifyBreaks = 0
justifyLastLine = 0
leading = 12
leftIndent = 32
linkUnderline = 0
rightIndent = 0
spaceAfter = 0
spaceBefore = 0
spaceShrinkage = 0.05
splitLongWords = 1
strikeGap = 1
strikeOffset = 0.25*F
strikeWidth =
textColor = Color(0,0,0,1)
textTransform = None
underlineGap = 1
underlineOffset = -0.125*F
underlineWidth =
uriWasteReduce = 0
wordWrap = None
```

- 这是一个要点。Spam spam
spam spam spam spam spam
spam spam spam spam
spamspace spam spam spam
spam spam spam spam spam
spam

图 6-15: 项目点的基本使用

除了使用序列标签外，数字也使用了完全相同的技术。也可以在项目符号中放入一个多字符的字符串；用深缩进和粗体子弹字体，你可以制作一个紧凑的定义列表。

第 7 章 表格和表格样式

Table和**LongTable**类来源于**Flowable**类，是一个简单的文本网格机制。当计算列宽时，**longTable**类使用了一种贪婪的算法，它是为速度至上的长表设计的。**Table**单元格可以容纳任何可以转换为Python **string**或**Flowables** (或**Flowables**列表)的内容。

我们现在的表格是在高效绘图和规范与功能之间的权衡。

我们假设读者对HTML表格有一定的熟悉。简而言之，它们具有以下特点。

- 它们可以包含任何可转换为字符串的东西；可流动的对象，如其他表格；或整个子集。
- 如果你不提供行高，他们可以计算出行高来适应数据。
(他们也可以计算出宽度，但一般来说，设计师最好手动设置宽度，这样画得更快)。
- 如果需要的话，它们可以在不同的页面上进行分割（参见**canSplit**属性）。你可以指定在分割后，顶部和底部的若干行应该重复显示（例如，在第2,3,4页再次显示页眉...）。
- 他们有一个简单而强大的符号来指定阴影和网格线，这对于财务或数据库表来说非常适用，因为你不知道前面有多少行。你可以很容易的说 "把最后一行加粗，并在上面加一条线"。
- 样式和数据是分开的，所以你可以声明少量的表格样式，并将它们用于一个报表系列。
样式也可以 "继承"，就像段落一样。

然而，与HTML表格相比，有一个主要的限制。它们定义了一个简单的矩形网格。没有简单的行或列跨度；如果你需要跨度单元格，你必须将表格嵌套在表格单元格内，或者使用更复杂的方案，其中跨度的前导单元格包含实际内容。

通过向构造函数传递列宽的可选序列、行高的可选序列以及行序的数据来创建**Tables**。表的绘制可以通过使用**TableStyle**实例来控制。这允许控制行的颜色和权重（如果有的话），以及文本的字体、对齐和填充。提供了一个原始的自动行高和列宽计算机制。

7.1 Table

这些是客户端程序员感兴趣的主要方法。

Table 初始化

```
Table(
    data,
    colWidths=None,
    rowHeights=None,
    style=None,
    splitByRow=1,
    repeatRows=0,
    repeatCols=0,
    rowSplitRange=None,
    spaceBefore=None,
    spaceAfter=None,
)
```

参数**data**是单元格值的序列，每个单元格值都应该使用**str**函数转换为字符串值，或者应该是一个**Flowable**实例(如**Paragraph**)或此类实例的列表(或元组)。如果一个单元格值是一个**Flowable**或**Flowable**的列表，这些单元格必须有一个确定的宽度，或者包含的列必须有一个固定的宽度。单元格值的第一行在 **data[0]** 中，也就是说，单元格值是按行顺序排列的。**i, jth** 单元格值在 **data[i][j]** 中。单元格值中的新行字符 **\n** 被视为行分割字符，并在draw时用于将单元格格式化为行。

其他参数是相当明显的，**colWidths** 参数是一个数字序列，也可能是**None**，代表列的宽度。在**colWidths** 中的元素数决定了表中的列数。值为**None**意味着相应的列宽应该自动计算。

参数 **rowHeights** 是一个数字序列，也可能是**None**，代表行的高度。**rowHeights** 中的元素数决定了表中的行数。值为**None**意味着相应的行高应该自动计算。

参数 **style** 可以是表的初始样式。

splitByRow 参数只适用于太高和太宽而无法适应当前上下文的表格。

在这种情况下，你必须决定是向下和横向 "平铺"，还是横向然后向下。这个参数是一个布尔值，表示当当前绘图区域可用空间太小，而调用者希望**Table**进行分割时，**Table**应该先按行进行分割，再按列进行分割。目前还没有实现按列分割**Table**，所以将**splitByRow**设置为**False**将导致**NotImplem**

entedError。

参数repeatRows指定了当Table被要求拆分时应该重复的前导行的数量或元组。如果它是一个元组，它应该指定哪些前导行应该被重复；这允许表的第一次出现比后来的分割部分有更多的前导行。目前，repeatCols参数被忽略，因为Table不能按列进行拆分。

当在platypus故事中重新编排时，spaceBefore 和 spaceAfter 参数可以用来在表格之前或之后放置额外的空间。

rowSplitRange参数可以用来控制表的分割，将表分割成它的行的子集；这可以防止分割太接近表的开始或结束。

Table.setStyle(tblStyle)

这个方法将类TableStyle(下面讨论)的一个特定实例应用到Table实例中。这是让tables以一种很好的格式化方式出现的唯一方法。

对setStyle方法的连续使用以加法的方式应用这些样式。也就是说，后面的应用会覆盖前面重叠的应用。

7.2 TableStyle

这个类是通过传递给它一个commands序列来创建的，每个 command 是一个元组，由它的第一个元素识别，它是一个字符串；command 元组的其余元素代表命令的起始和停止单元格坐标，可能还有厚度和颜色等。

7.3 TableStyle 方法

TableStyle(commandSequence)

创建方法以参数命令序列为例初始化TableStyle。

```
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')])
```

TableStyle.add(commandSequence)

此方法允许你向现有的TableStyle添加命令，即你可以在多个语句中建立TableStyles。

```
LIST_STYLE.add('BACKGROUND', (0,0), (-1,0), colors.Color(0,0.7,0.7))
```

TableStyle.getCommands()

此方法返回实例的命令序列。

```
cmds = LIST_STYLE.getCommands()
```

7.4 TableStyle Commands

传递给TableStyles的命令主要有三组，分别影响表格背景、绘制线条或设置单元格样式。

每个命令的第一个元素是它的标识符，第二个和第三个参数决定了受影响的单元格的单元格坐标，负坐标从极限值向后数，就像Python索引一样。坐标是以(列，行)的形式给出的，它遵循电子表格的"A1"模型，但不是更自然的(对数学家来说)"RC"排序。左上角的单元格是(0，0)，右下角的单元格是(-1，-1)。根据命令的不同，各种额外的(???)发生在3开始的指数上。



译者注：对表格坐标系理解不够透彻的参考: [这里](#)

TableStyle Cell Formatting Commands

单元格格式化命令都以一个标识符开头，后面是单元格定义的开始和结束，也许还有其他参数。

Command	同义词	描述
FONT		字体名称, 可选字体大小和字符间距(leading).
FONTNAME	FACE	字体名称.
FONTSIZE	SIZE	以点为单位的字体大小; 字符间距(leading)可能会不同步.
LEADING		以点为单位的字符间距(leading).
TEXTCOLOR		颜色名称或 (R,G,B) 元组.
ALIGNMENT	ALIGN	LEFT, RIGHT 和 CENTRE (或 CENTER) 或 DECIMAL 中的一个.
LEFTPADDING		整数左边距, 默认为6.
RIGHTPADDING		整数右边距, 默认为6.
BOTTOMPADDING		整数下边距, 默认为3.
TOPPADDING		整数上边距, 默认为3.
BACKGROUND		取一个由对象、字符串名称或数字元组/列表定义的颜色, 或者取一个描述所需渐变填充的列表/元组, 该列表/元组应包含[DIRECTION, startColor, endColor]三个元素, 其中 DIRECTION 是 VERTICAL 或 HORIZONTAL。
ROWBACKGROUNDS		一个要循环使用的颜色列表。
COLBACKGROUNDS		一个要循环使用的颜色列表。
VALIGN		取 TOP, MIDDLE 或默认的 BOTTOM 中的一个。

这将设置相关单元格的背景颜色。下面的例子显示了 **BACKGROUND** 和 **TEXTCOLOR** 命令的作用。

```
from reportlab.platypus.tables import Table, TableStyle
from reportlab.lib import colors

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)]))
```

结果:

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

为了看到对齐样式的效果，我们需要一些宽度和网格，但应该很容易看到样式的来源。

```
from reportlab.platypus.tables import Table, TableStyle
from reportlab.lib import colors
from reportlab.lib.units import inch

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),
]))
```

结果:

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

TableStyle Line Commands

线条命令以标识符、起始和终止单元格坐标开始，并始终以所需线条的厚度（以点为单位）和颜色跟随。颜色可以是名称，也可以指定为(R, G, B)元组，其中R,G和B是浮动数，(0, 0, 0)是黑色。行命令名称为 GRID, BOX, OUTLINE, INNERGRID, LINEBELOW, LINEABOVE, LINEBEFORE 和 LINEAFTER。BOX和OUTLINE相等，GRID相当于同时应用 BOX 和 INNERGRID。我们可以通过下面的例子看到一些行命令的作用。

```
from reportlab.platypus.tables import Table
from reportlab.lib import colors

data= [['00', '01', '02', '03', '04'],
        ['10', '11', '12', '13', '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),
                    ])
```

结果:

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

行命令在拆分表格时，会给表格带来问题；下面的例子显示了一个表格在不同位置被拆分的情况

```
from reportlab.platypus.tables import Table
from reportlab.lib import colors

data= [['00', '01', '02', '03', '04'],
        ['10', '11', '12', '13', '14'],
        ['20', '21', '22', '23', '24'],
        ['30', '31', '32', '33', '34']]

t=Table(data,style=[('GRID',(0,0),(-1,-1),0.5,colors.grey),
                    ('GRID',(1,1),(-2,-2),1,colors.green),
                    ('BOX',(0,0),(1,-1),2,colors.red),
                    ('BOX',(0,0),(-1,-1),2,colors.black),
                    ('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),
```


))

结果:

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

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

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

当在第一行或第二行进行拆分和分割时。

复杂的单元格值

如上所述，我们可以有复杂的单元格值，包括Paragraphs, Images和其他Flowables或相同的列表。要看到这个操作，请考虑下面的代码和它产生的表格。请注意，Image的背景是白色的，这将会遮挡住您为单元格选择的任何背景。为了得到更好的效果，你应该使用透明的背景。

```

from reportlab.platypus.flowables import Image
from reportlab.platypus import Paragraph
from reportlab.platypus.tables import Table
from reportlab.lib import colors
from reportlab.lib.units import inch
from reportlab.lib.styles import ParagraphStyle

I = Image('report\images\replogo.gif')
I.drawHeight = 1.25*inch*I.drawHeight / I.drawWidth
I.drawWidth = 1.25*inch

normal_style = ParagraphStyle(
    name='Normal',
    fontName='SourceHanSans-Light',
    fontSize=10,
    leading=12,
    spaceBefore=6,
)
_body_text = ParagraphStyle(
    name='BodyText',
    parent=normal_style,
    spaceBefore=6,
)



P0 = Paragraph(
    <b>A pa<font color=red>r</font>a<i>graph</i></b>
    <sup><font color=yellow>1</font></sup>',
    _body_text)
P = Paragraph(
    <para align=center spaceb=3>The <b>ReportLab Left
    <font color=red>Logo</font></b>
    Image</para>',
    _body_text)
  
```

```
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
```

结果:

A	B	C	A paragraph ¹	D
00	01	02	 The ReportLab Left Logo Image	04
10	11	12	The ReportLab Left Logo Image 	14
20	21	22	23	24
30	31	32	33	34

TableStyle 跨单元格命令

我们的Table类支持跨度的概念， 但它的指定方式与html不同。样式规范

```
SPAN, (sc,sr), (ec,er)
```

表示列 `sc - ec` 和行 `sr - er` 中的单元格应该合并成一个超级单元格， 其内容由单元格 `(sc, sr)` 决定。其他单元格应该存在， 但应该包含空字符串， 否则可能会得到意想不到的结果。

```
from reportlab.platypus.tables import Table
from reportlab.lib import colors

data= [['Top\nLeft', '', '02', '03', '04'],
        ['', '', '12', '13', '14'],
        ['20', '21', '22', 'Bottom\nRight', ''],
        ['30', '31', '32', '', '']]

t=Table(data,style=[
    ('GRID',(0,0),(-1,-1),0.5,colors.grey),
    ('BACKGROUND',(0,0),(1,1),colors.palegreen),
    ('SPAN',(0,0),(1,1)),
```

```
('BACKGROUND',(-2,-2),(-1,-1), colors.pink),
('SPAN',(-2,-2),(-1,-1)),
])
```

结果:

Top Left		02	03	04
		12	13	14
20	21	22	Bottom Right	
30	31	32		

注意到我们的GRID命令不需要太保守。跨越的单元格不会被画穿。

TableStyle 其他命令

要控制 Table 的拆分，可以使用 NOSPLIT 命令。

```
NOSPLIT, (sc,sr), (ec,er)
```

要求列 **sc-ec** 和行 **sr-er** 中的单元格不能被拆分。

TableStyle 特殊的指标

在任何样式命令中，第一行索引可以设置为特殊字符串*splitlast*或*splitfirst*中的一个，以表明该样式只用于拆分表的最后一行或延续表的第一行。这样可以使拆分表在拆分后有更好的效果。

第 8 章 Flowables

下列flowable使您可以在包装文本时有条件地求值并执行表达式和语句：

8.1 DocAssign

定义：

```
DocAssign(self, var, expr, life='forever')
```

为表达式`expr`指定一个名称为`var`的变量。例如：

```
DocAssign('i',3)
```

8.2 DocExec

定义：

```
DocExec(self, stmt, lifetime='forever')
```

执行语句`stmt`。例如：

```
DocExec('i-=1')
```

8.3 DocPara

定义：

```
DocPara(self, expr, format=None, style=None, klass=None, escape=True)
```

创建一个以 `expr` 的值为文本的段落。如果指定了格式，它应该使用 `%(__expr__)s` 对表达式 `expr` (如果有的话) 进行字符串插值。它也可以使用`%(name)s`对命名空间中的其他变量进行插值。例如

```
DocPara('i',format='The value of i is %(__expr__)d',style=normal)
```

8.4 DocAssert

定义：

```
DocAssert(self, cond, format=None)
```

如果`cond`评价为`False`，则引发包含`format`字符串的`AssertionError`。

```
DocAssert(val, 'val is False')
```

8.5 DocIf

定义：

```
DocIf(self, cond, thenBlock, elseBlock=[])
```

如果`cond`的值为`True`，那么这个flowable就会被`thenBlock` elsethe `elseBlock`取代。

```
DocIf('i>3',Paragraph('The value of i is larger than 3',normal),\
    Paragraph('The value of i is not larger than 3',normal))
```

8.6 DocWhile

定义：

```
DocWhile(self, cond, whileBlock)
```

当`cond`值为`True`时，运行`whileBlock`。例如：

```
DocAssign('i',5)
DocWhile('i',[DocPara('i',format='The value of i is %(__expr__)d',style=normal),DocExec('i-=1')])
```

这个例子产生的一组段落的形式是：

```
The value of i is 5
The value of i is 4
The value of i is 3
The value of i is 2
The value of i is 1
```

第 9 章 其他有用的Flowables

9.1 Preformatted

定义:

```
Preformatted(text, style, bulletText=None, dedent=0, maxLineLength=None, splitChars=None, newLineChars=None)
```

创建一个预格式化的段落，不进行任何包装、分行或其他操作。在文本中不考虑XML样式标签。如果dedent是非零，dedent的公共前导空格将从每行前面移除。

定义最大行长

您可以使用属性 `maxLineLength` 来定义最大行长。如果行长超过这个最大值，行将被自动分割。

行将被分割成`splitChars`中定义的任何一个字符。如果没有为该属性提供值，则行将在以下任何标准字符上进行分割：空格、冒号、句号、分号、逗号、连字符、前斜线、后斜线、左括号、左方括号和左大括号。

字符可以自动插入到已创建的每一行的开头。你可以设置属性`newLineChars`为你想使用的字符。

```
from reportlab.platypus.flowables import Preformatted
from reportlab.lib.styles import getSampleStyleSheet

stylesheet=getSampleStyleSheet()
normalStyle = stylesheet['Code']
text=""
class XPreformatted(Paragraph):
    def __init__(self, text, style, bulletText = None, frags=None, caseSensitive=1):
        self.caseSensitive = caseSensitive
        if maximumLineLength and text:
            text = self.stopLine(text, maximumLineLength, splitCharacters)
        cleaner = lambda text, dedent=dedent: ".join(_dedenter(text or '', dedent))"
        self._setup(text, style, bulletText, frags, cleaner)
    ""
t=Preformatted(text,normalStyle,maxLineLength=60, newLineChars='>')
```

结果:

```
class XPreformatted(Paragraph):
    def __init__(self, text, style, bulletText = None,
> frags=None, caseSensitive=1):
        self.caseSensitive = caseSensitive
        if maximumLineLength and text:
            text = self.stopLine(text, maximumLineLength,
> splitCharacters)
        cleaner = lambda text, dedent=dedent: ''.join(
> _dedenter(text or '', dedent))
        self._setup(text, style, bulletText, frags, cleaner)
```

9.2 XPreformatted

定义:

```
$XPreformatted(text, style, bulletText=None, dedent=0, frags=None)
```

这是`Paragraph`类的一种非重排形式；`text`中允许使用XML标签，其含义与`Paragraph`类相同。至于`Preformatted`，如果dedent是非零，dedent普通的前导空格将从每行的前面移除。

```
from reportlab.platypus import XPreformatted
from reportlab.lib.styles import getSampleStyleSheet

stylesheet=getSampleStyleSheet()
normalStyle = stylesheet['Code']
text=""

This is a non rearranging form of the <b>Paragraph</b> class;
<b><font color=red>XML</font></b> tags are allowed in <i>text</i> and have the same
```

meanings as for the `Paragraph` class.
 As for `Preformatted`, if `dedent` is non zero `dedent`
 common leading spaces will be removed from the
 front of each line.
 You can have `&` style entities as well for `<`, `>` and `"`;

```
'''
t=XPreformatted(text,normalStyle,dedent=3)
```

结果:

```
This is a non rearranging form of the Paragraph class;
XML tags are allowed in text and have the same

meanings as for the Paragraph class.
As for Preformatted, if dedent is non zero dedent
common leading spaces will be removed from the
front of each line.
You can have &amp; style entities as well for &lt; &gt; and ".
```

9.3 Image

定义:

```
Image(filename, width=None, height=None)
```

创建一个 **flowable**，它将包含由文件 **filename** 中的数据定义的图像，该文件可以是文件路径、类似文件的对象或 **reportlab.graphics.shapes.Drawing** 的实例。默认的 PDF 图像类型 **jpeg** 被支持，如果安装了 PIL 扩展到 Python，其他图像类型也可以被处理。如果指定了 **width** 和 **height**，那么它们决定了显示图像的尺寸，单位是 **points**。如果没有指定任何一个尺寸(或者指定为 **None**)，那么图像的相应像素尺寸被假定为 **points** 并使用。

```
Image("report\images\lj8100.jpg")
```

将显示为



然而

```
im = Image("report\images\lj8100.jpg", width=2*inch, height=2*inch)
im.hAlign = 'CENTER'
```

产出



9.4 Spacer

定义:

```
Spacer(width, height)
```

这与预期的一样，它为story增加了一定的空间。目前，这只对垂直空间有效。

9.5 PageBreak

这个 **Flowable** 代表了一个页面中断。它的工作原理是有效地消耗所有给它的垂直空间。这对于单个 **Frame** 文档来说已经足够了，但对于多个框架来说，这只是一个框架中断，所以 **BaseDocTemplate** 机制会在内部检测到 **pageBreaks** 并进行特殊处理。

9.6 CondPageBreak

定义:

```
CondPageBreak(height)
```

如果当前的**Frame**中没有足够的垂直空间，那么这个**Flowable**试图强制**Frame**断裂。因此，它的命名可能是错误的，也许应该重新命名为**CondFrameBreak**。

9.7 KeepTogether

定义:

```
KeepTogether(flowables)
```

这个复合的 **Flowable** 接收一个 **Flowables** 的列表，并试图将它们放在同一个 **Frame** 中。如果列表中 **flowables** 中的 **Flowables** 的总高度超过了当前框架的可用空间，那么所有的空间都会被使用，并且会强制中断框架。

9.8 TableOfContents

定义:

```
TableOfContents()
```

通过使用**TableOfContents** flowable 可以生成一个目录。下面的步骤是在您的文档中添加一个目录表所需要的。

创建一个 **TableOfContents** 的实例。覆盖关卡样式（可选）并将对象添加到**story**中。

```
toc = TableOfContents()
PS = ParagraphStyle
toc.levelStyles = [
    PS(fontName='Times-Bold', fontSize=14, name='TOCHeading1',
```



```

        leftIndent=20, firstLineIndent=-20, spaceBefore=5, leading=16),
    PS(fontSize=12, name='TOCHeading2',
       leftIndent=40, firstLineIndent=-20, spaceBefore=0, leading=12),
    PS(fontSize=10, name='TOCHeading3',
       leftIndent=60, firstLineIndent=-20, spaceBefore=0, leading=12),
    PS(fontSize=10, name='TOCHeading4',
       leftIndent=100, firstLineIndent=-20, spaceBefore=0, leading=12),
]
story.append(toc)

```

对目录的输入可以通过调用 `TableOfContents` 对象的 `addEntry` 方法手动完成，也可以通过在 `DocTemplate` 的 `afterFlowable` 方法中自动发送一个 `'TOCEntry'` 通知。传递给 `notify` 的数据是一个由三个或四个项目组成的列表，其中包括一个级别号，条目文本，页码和一个可选的目标键，该条目应该指向。这个列表通常会在文档模板的方法中创建，比如 `afterFlowable()`，使用 `notify()` 方法调用通知，并提供适当的数据，比如这样。

```

def afterFlowable(self, flowable):
    """Detect Level 1 and 2 headings, build outline,
    and track chapter title."""
    if isinstance(flowable, Paragraph):
        txt = flowable.getPlainText()
        if style == 'Heading1':
            # ...
            self.notify('TOCEntry', (0, txt, self.page))
        elif style == 'Heading2':
            # ...
            key = 'h2-%s' % self.seq.nextf('heading2')
            self.canv.bookmarkPage(key)
            self.notify('TOCEntry', (1, txt, self.page, key))
        # ...

```

这样，每当一个样式为 `'Heading1'` 或 `'Heading2'` 的段落被添加到故事中，它就会出现在目录中。`Heading2` 条目将可点击，因为已经提供了一个书签键。

最后你需要使用 `DocTemplate` 的 `multiBuild` 方法，因为内容表需要多次传递才能生成。

```
doc.multiBuild(story)
```

下面是一个简单但可行的带目录的文档例子。

```

from reportlab.lib.styles import ParagraphStyle as PS
from reportlab.platypus import PageBreak
from reportlab.platypus.paragraph import Paragraph
from reportlab.platypus.doctemplate import PageTemplate, BaseDocTemplate
from reportlab.platypus.tableofcontents import TableOfContents
from reportlab.platypus.frames import Frame
from reportlab.lib.units import cm

class MyDocTemplate(BaseDocTemplate):

    def __init__(self, filename, **kw):
        self.allowSplitting = 0
        BaseDocTemplate.__init__(self, filename, **kw)
        template = PageTemplate('normal', [Frame(2.5*cm, 2.5*cm, 15*cm, 25*cm, id='F1')])
        self.addPageTemplates(template)

    def afterFlowable(self, flowable):
        "Registers TOC entries."
        if flowable.__class__.__name__ == 'Paragraph':
            text = flowable.getPlainText()
            style = flowable.style.name
            if style == 'Heading1':
                self.notify('TOCEntry', (0, text, self.page))
            if style == 'Heading2':
                self.notify('TOCEntry', (1, text, self.page))

    h1 = PS(name = 'Heading1',
            fontSize = 14,
            leading = 16)

    h2 = PS(name = 'Heading2',
            fontSize = 12,
            leading = 14,

```

```

        leftIndent = delta)

# Build story.
story = []
toc = TableOfContents()
# For conciseness we use the same styles for headings and TOC entries
toc.levelStyles = [h1, h2]
story.append(toc)
story.append(PageBreak())
story.append(Paragraph('First heading', h1))
story.append(Paragraph('Text in first heading', PS('body')))
story.append(Paragraph('First sub heading', h2))
story.append(Paragraph('Text in first sub heading', PS('body')))
story.append(PageBreak())
story.append(Paragraph('Second sub heading', h2))
story.append(Paragraph('Text in second sub heading', PS('body')))
story.append(Paragraph('Last heading', h1))

doc = MyDocTemplate('mintoc.pdf')
doc.multiBuild(story)

```

9.9 SimpleIndex

定义:

```
SimpleIndex()
```

可以通过使用 **SimpleIndex** flowable 生成一个索引。下面的步骤是为您文档添加索引所需要的。在段落中使用索引标签来索引术语。

```

story = []

...

story.append('The third <index item="word" />word of this paragraph is indexed.')

```

创建一个 **SimpleIndex** 的实例，并将其添加到你希望它出现的故事中。

```

index = SimpleIndex(dot='.', headers=headers)
story.append(index)

```

你可以传入 **SimpleIndex** 构造函数的参数在 **reportlab** 参考资料中解释。现在，使用 **SimpleIndex.getCanvasMaker()** 返回的 **canvasmaker** 来构建文档。

```
doc.build(story, canvasmaker=index.getCanvasMaker())
```

要建立一个多级索引，请将一个以逗号分隔的项目列表传递给索引标签的 **item** 属性。

```

<index item="terma,termb,termc" />
<index item="terma,termd" />

```

terma 将表示最顶层的术语，**termc** 将表示最具体的术语，**terd** 和 **termb** 将出现在 **terma** 的同一层次。

如果您需要为一个包含逗号的术语编制索引，您需要将其加倍转义。为了避免三个连续逗号的歧义（一个转义逗号后面是一个列表分隔符或一个列表分隔符后面是一个转义逗号？）在正确的位置引入一个空格。术语开头或结尾的空格将被删除。

```
<index item="comma(,),,,... " />
```

此处索引了 "逗号 (,) "、", "和" "等术语。

9.10 ListFlowable, ListItem

使用这些类来制作有序和无序的列表。列表可以被嵌套。

ListFlowable() 将创建一个有序的列表，它可以包含任何可流动的。该类有许多参数可以改变列表编号的字体、颜色、大小、样式和位置，或者无序列表中的项目符号。还可以使用 **bulletType**

属性将编号类型设置为使用小写或大写字母 ('A,B,C'等) 或罗马数字 (大写或小写) 。要将列表改为无序类型, 请设置 `bulletType='bullet'`。

在 `ListFlowable()` 列表中的项目可以通过将它们包装在 `ListItem()` 类中并设置其属性来改变它们的默认外观。

下面将创建一个有序列表, 并将第三项设置为无序子列表。

```
from reportlab.platypus import ListFlowable, ListItem
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus.paragraph import Paragraph

styles = getSampleStyleSheet()
style = styles["Normal"]

t = ListFlowable(
    [
        Paragraph("Item no.1", style),
        ListItem(
            Paragraph("Item no. 2", style), bulletColor="green", value=7
        ),
        ListFlowable(
            [
                Paragraph("sublist item 1", style),
                ListItem(
                    Paragraph('sublist item 2', style),
                    bulletColor='red',
                    value='square',
                ),
            ],
            bulletType='bullet',
            start='square',
        ),
        Paragraph("Item no.4", style),
    ],
    bulletType='i',
)
```

结果:

```
i   Item no.1
vii Item no. 2
viii ■ sublist item 1
      ■ sublist item 2
ix  Item no.4
```

为了应对嵌套, `start` 参数可以设置为一个可能的起始列表; 对于 `ul` 来说, 可接受的起始是任何 `unicode` 字符或 `flowables.py` 已知的特定名称, 例如 `bulletchar`、`circle`、`square`、`disc`、`diamond`、`diamondwx`、`rarrowhead`、`sparkle`、`squarelrs` 或 `blackstar`。对于 `ol` 来说, `start` 可以是 '1iaAI' 中的任何字符, 以表示不同的数字风格。

9.11 BalancedColumns

使用 `BalancedColumns` 类来制作一个 `flowable`, 将其内容 `flowable` 分割成两个或更多大小大致相等的列。实际上, `n` 框架被合成为内容, 而 `flowable` 试图在它们之间平衡内容。当创建的框架总高度过大时, 会被拆分, 拆分后会保持平衡。

```
from reportlab.platypus.flowables import BalancedColumns
from reportlab.platypus.frames import ShowBoundaryValue
F = [
    list of flowables.....
]
story.append(
    Balanced(
        F,      #the flowables we are balancing
        nCols = 2, #the number of columns
        needed = 72, #the minimum space needed by the flowable
        spacBefore = 0,
        spaceAfter = 0,
        showBoundary = None, #optional boundary showing
    )
)
```

```
leftPadding=None,    #these override the created frame
rightPadding=None,   #paddings if specified else the
topPadding=None,     #default frame paddings
bottomPadding=None,  #are used
innerPadding=None,   #the gap between frames if specified else
                    #use max(leftPadding,rightPadding)
name="",             #for identification purposes when stuff goes awry
endSlack=0.1,        #height disparity allowance ie 10% of available height
)
)
```

第 10 章 编写 Flowable

Flowables旨在成为创建可重复使用的报表内容的开放标准，您可以轻松创建自己的对象。我们希望随着时间的推移，我们将建立一个贡献库，为 **reportlab** 用户提供丰富的图表、图形和其他 "report widgets" 选择，他们可以在自己的报表中使用。本节将向您展示如何创建您自己的 flowables。

我们应该把Figure类放在标准库中，因为它是一个非常有用的基础。

10.1 一个非常简单的 Flowable

回想一下本用户指南中 **pdfgen** 一节中的 **hand** 函数，它生成了一个由贝塞尔曲线构成的闭合图形的手图。

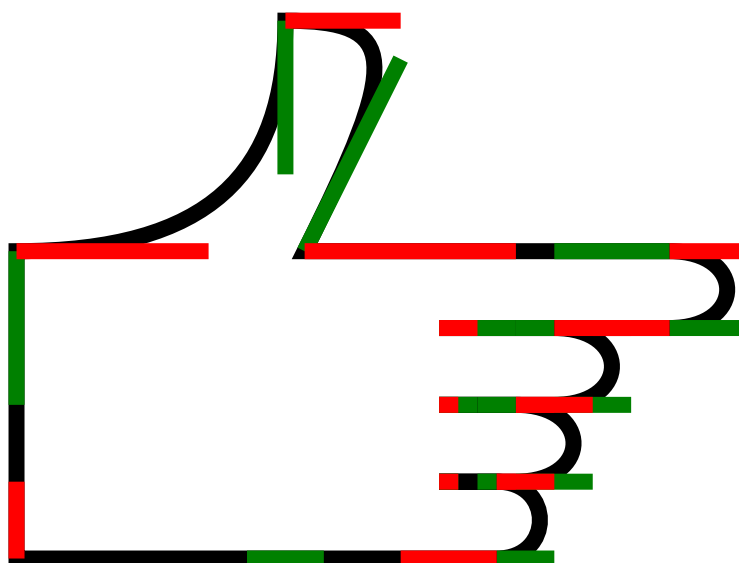


图 10 - 1: 一只手

为了在 **Platypus flowable** 中嵌入这个或其他绘图，我们必须定义一个 **Flowable** 的子类，至少有一个 **wrap** 方法和一个 **draw** 方法。

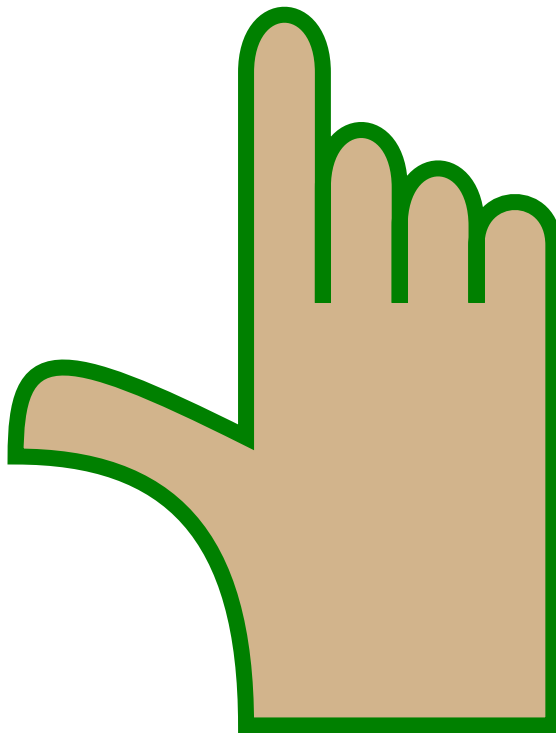
```
from reportlab.platypus.flowables import Flowable
from reportlab.lib.colors import tan, green
class HandAnnotation(Flowable):
    """A hand flowable."""
    def __init__(self, xoffset=0, size=None, fillcolor=tan, strokecolor=green):
        from reportlab.lib.units import inch
        if size is None: size=4*inch
        self.fillcolor, self.strokecolor = fillcolor, strokecolor
        self.xoffset = xoffset
        self.size = size
        # normal size is 4 inches
        self.scale = size/(4.0*inch)
    def wrap(self, *args):
        return (self.xoffset, self.size)
    def draw(self):
        canvas = self.canv
        canvas.setLineWidth(6)
        canvas.setFillColors(self.fillcolor)
        canvas.setStrokeColor(self.strokecolor)
        canvas.translate(self.xoffset+self.size,0)
        canvas.rotate(90)
        canvas.scale(self.scale, self.scale)
        hand(canvas, debug=0, fill=1)
```

wrap 方法必须提供绘图的大小-- **Platypus**

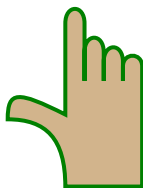
主循环用它来决定这个元素是否适合当前框架的剩余空间。在 **Platypus** 主循环将 (0,0)

原点转换到适当的框架中的适当位置后，`draw`方法将执行对象的绘制。

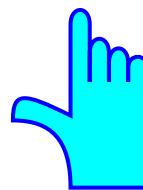
以下是 `HandAnnotation flowable` 的一些使用示例。



默认情况下



只是一寸高。



高1英寸，向左偏蓝和青色。

10.2 修改内置的 Flowable

要修改现有的可流动对象，您应该创建一个派生类并覆盖需要更改以获得所需行为的方法。

作为创建旋转图像的示例，您需要覆盖现有 `Image` 类的 `wrap` 和 `draw` 方法

```
from reportlab.platypus.flowables import Image
class RotatedImage(Image):
    def wrap(self, availWidth, availHeight):
        h, w = Image.wrap(self, availHeight, availWidth)
        return w, h
    def draw(self):
        self.canv.rotate(90)
        Image.draw(self)
I = RotatedImage('report/images/replologo.gif')
I.hAlign = 'CENTER'
```

结果:



第 11 章 绘制

11.1 简介

ReportLab Graphics是ReportLab库的一个子包。它最初是作为一个独立的程序集，但现在是ReportLab工具包的一个完整的集成部分，它允许您使用其强大的图表和图形功能来改进您的PDF表格和报表。

11.2 一般概念

在本节中，我们将介绍图形库的一些比较基本的原理，这些原理会在后面的各个地方出现。

图纸和渲染器

*Drawing*是一个独立于平台的形状集合的描述。它不直接与PDF, Postscript或任何其他输出格式相关联。幸运的是，大多数矢量图形系统都遵循Postscript模型，因此可以毫不含糊地描述形状。

一张图中包含了许多原始的*Shape*

(形状)。普通形状是那些广为人知的矩形、圆形、线条等。一个特殊的（逻辑）形状是（组），它可以容纳其他形状并对它们进行变换。*Group*代表了*Shape*的组合，并允许将*Group*合当作一个单一的*Shape*来处理。几乎所有的东西都可以从少量的基本*Shape* (形状)建立起来。

该包提供了几个*Renderers* (渲染器)，它们知道如何将图纸绘制成不同的格式。其中包括 PDF (*renderPDF*)、Postscript (*renderPS*) 和位图输出 (*renderPM*)。位图渲染器使用 Raph Levien 的 *libart* 栅格化器和 Fredrik Lundh 的 *Python Imaging Library* (PIL)。SVG 渲染器使用了 Python 的标准库 XML 模块，所以你不需要安装 XML-SIG 的附加包 PyXML。如果你安装了正确的扩展，你可以为网络生成位图形式的图画，也可以为 PDF 文档生成矢量形式的图画，并得到 "相同的输出"。

PDF 渲染器具有特殊的 "特权" -- 一个 *Drawing* 对象也是一个 *Flowable*，因此，可以直接放在任何 *Platypus* 文档的故事中，或者直接用一行代码在 *Canvas* 上绘制。此外，PDF 渲染器还有一个实用功能，可以快速制作一页PDF文档。

SVG 渲染器很特别，因为它仍然是相当的实验性的。它所生成的 SVG 代码并没有经过任何优化，只是将 ReportLab Graphics (RLG) 中可用的功能映射到 SVG 中。这意味着不支持 SVG 动画、交互性、脚本或更复杂的剪切、遮罩或渐变形状。因此，请小心，并请报告您发现的任何错误。

坐标系统

在我们的X-Y坐标系中，Y方向从底部向上。这与PDF、Postscript和数学符号一致。对人们来说，这似乎也更自然，尤其是在处理图表时。请注意，在其他图形模型中（如SVG），Y坐标点向下。对于SVG

渲染器来说，这实际上是没有问题的，因为它将根据需要对你的图纸进行翻转，因此您的SVG输出看起来和预期的一样。

X坐标一如既往地从左到右。到目前为止，似乎没有任何模型主张反方向 -- 至少目前还没有（似乎有一些有趣的例外，阿拉伯人在看时间序列图时...）。

开始

让我们创建一个简单的图形，包含字符串 "Hello World" 和一些特殊的字符，显示在一个彩色的矩形之上。创建完成后，我们将把图画保存到一个独立的PDF文件中。

```
from reportlab.lib import colors
from reportlab.graphics.shapes import *

d = Drawing(400, 200)
d.add(Rect(50, 50, 300, 100, fillColor=colors.yellow))
d.add(String(150, 100, 'Hello World', fontSize=18, fillColor=colors.red))
d.add(String(180, 86, 'Special characters \
```



```
\xc2\xa2\xc2\xa9\xc2\xae\xc2\xa3\xce\xb1\xce\xb2',
fillColor=colors.red))
```

```
from reportlab.graphics import renderPDF
renderPDF.drawToFile(d, 'example1.pdf', 'My First Drawing')
```

这将产生一个包含以下图形的PDF文件。



图 11 - 1 : 'Hello World'

每个渲染器都可以做任何适合其格式的事情，并且可以有任意需要的API。如果它指的是一种文件格式，它通常有一个 `drawToFile` 函数，这就是你需要知道的关于渲染器的所有信息。让我们以 Encapsulated Postscript 格式保存同一张图。

```
from reportlab.graphics import renderPS
renderPS.drawToFile(d, 'example1.eps')
```

这将生成一个具有相同图形的EPS文件，可以导入到Quark Express等出版工具中。如果我们想生成同样的图形作为网站的位图文件，比如说，我们需要做的就是写这样的代码。

```
from reportlab.graphics import renderPM
renderPM.drawToFile(d, 'example1.png', 'PNG')
```

许多其他的位图格式，如GIF、JPG、TIFF、BMP和PPN都是真正可用的，这使得你不太可能需要添加外部的后处理步骤来转换为你需要的最终格式。

要生成一个包含相同图形的SVG文件，并将其导入到 **Illustrator** 等图形编辑工具中，我们需要做的就是编写这样的代码。

```
from reportlab.graphics import renderSVG
renderSVG.drawToFile(d, 'example1.svg')
```

属性验证

Python是非常动态的，让我们在运行时执行的语句很容易成为意外行为的来源。一个微妙的“错误”是当分配到一个框架不知道的属性时，因为使用的属性的名字包含了一个错别字。Python 让你可以逃过一劫(比如说，给一个对象添加一个新的属性)，但图形框架在不采取特殊对策的情况下，是不会检测到这个“错别字”的。

有两种验证技术可以避免这种情况。默认情况下，每个对象在运行时都会检查每一次赋值，这样你就只能赋值给“合法”的属性。这就是默认情况。由于这样做会带来很小的性能损失，所以在你需要的时候可以关闭这种行为。

```
>>> r = Rect(10,10,200,100, fillColor=colors.red)
>>>
```

```
>>> r.fullColor = colors.green # note the typo
>>> r.x = 'not a number'      # illegal argument type
>>> del r.width                # that should confuse it
```

这些语句在静态类型的语言中会被编译器捕获，但是 Python 让你摆脱了它。第一个错误可能会让你盯着图片想弄清楚为什么颜色是错的。第二个错误可能只有在以后，当一些后端试图绘制矩形时才会变得清晰。第三种错误，虽然可能性较小，但会导致一个不知道如何绘制的无效对象。

```
>>> r = shapes.Rect(10,10,200,80)
>>> r.fullColor = colors.green
Traceback (most recent call last):
  File "<interactive input>", line 1, in ?
  File "C:\code\users\andy\graphics\shapes.py", line 254, in __setattr__
    validateSetattr(self,attr,value) #from reportlab.lib.attrmap
  File "C:\code\users\andy\lib\attrmap.py", line 74, in validateSetattr
    raise AttributeError, "Illegal attribute '%s' in class '%s'" % (name,
    obj.__class__.__name__)
AttributeError: Illegal attribute 'fullColor' in class Rect
>>>
```

这将带来性能上的惩罚，所以当你需要这种行为时，可以将其关闭。要做到这一点，您应该在第一次导入 `reportlab.graphics.shapes` 之前使用以下代码行。

```
>>> import reportlab.rl_config
>>> reportlab.rl_config.shapeChecking = 0
>>> from reportlab.graphics import shapes
>>>
```

一旦你关闭了 `reportlab.rl_config.shapeChecking`，类实际上是在没有验证钩子的情况下构建的；那么，代码应该会变得更快。目前，对成批图表的惩罚似乎是25%，所以几乎不值得禁用。然而，如果我们将渲染器转移到C语言上（这是很有可能的），剩下的75%就会缩减到几乎没有，而且从验证中节省的成本也会很可观。

每个对象，包括绘图本身，都有一个`verify()`方法。这个方法要么成功，要么引发一个异常。如果你关闭了自动验证，那么你应该在开发代码时，在测试中明确地调用`verify()`，或者在一个批处理中调用一次。

属性编辑

我们将在下面介绍的 `reportlab/graphics` 的一个基石是，你可以自动记录 widget。这意味着你可以掌握它们所有的可编辑属性，包括它们的子组件。

另一个目标是能够为图纸创建GUI和配置文件。可以建立一个通用的GUI来显示图纸的所有可编辑的属性，并让你修改这些属性并查看结果。**Visual Basic** 或 **Delphi** 开发环境是这类东西的好例子。在批处理图表应用程序中，一个文件可以列出图表中所有组件的所有属性，并与数据库查询合并，以制作一批图表。另一个目标是能够为图纸创建GUI和配置文件。可以建立一个通用的GUI来显示图纸的所有可编辑的属性，并让你修改这些属性并查看结果。**Visual Basic** 或 **Delphi** 开发环境是这类东西的好例子。在批处理图表应用程序中，一个文件可以列出图表中所有组件的所有属性，并与数据库查询合并，以制作一批图表。

为了支持这些应用，我们有两个接口，`getProperties` 和 `setProperties`，以及一个方便的方法`dumpProperties`。第一个方法返回一个对象的可编辑属性的字典；第二个方法则集体设置这些属性。如果一个对象有公开的“子对象”，那么我们也可以递归地设置和获取它们的属性。当我们稍后看`Widgets`时，这将更有意义，但我们需要将支持放到框架的基础上。

```
>>> r = shapes.Rect(0,0,200,100)
>>> import pprint
>>> pprint.pprint(r.getProperties())
{'fillColor': Color(0.00,0.00,0.00),
 'height': 100,
 'rx': 0,
 'ry': 0,
 'strokeColor': Color(0.00,0.00,0.00),
 'strokeDashArray': None,
 'strokeLineCap': 0,
 'strokeLineJoin': 0,
 'strokeMiterLimit': 0,
 'strokeWidth': 1,
```

```

'width': 200,
'x': 0,
'y': 0}
>>> r.setProperties({'x':20, 'y':30, 'strokeColor': colors.red})
>>> r.dumpProperties()
fillColor = Color(0.00,0.00,0.00)
height = 100
rx = 0
ry = 0
strokeColor = Color(1.00,0.00,0.00)
strokeDashArray = None
strokeLineCap = 0
strokeLineJoin = 0
strokeMiterLimit = 0
strokeWidth = 1
width = 200
x = 20
y = 30
>>>

```



注意：pprint 是标准的Python库模块，它允许您在多行上“漂亮地打印”输出，而不是只有一行很长的内容。

这三种方法在这里似乎并没有什么作用，但正如我们将看到的那样，它们使我们的 **widgets** 框架在处理非原始对象时更加强大。

Children 命名

您可以将对象添加到 **Drawing** 和 **Group** 对象中。这些对象通常会进入一个内容列表中。然而，你也可以在添加对象时给它们一个名字。这允许您在构造一个绘图后引用并可能改变它的任何元素。

```

>>> d = shapes.Drawing(400, 200)
>>> s = shapes.String(10, 10, 'Hello World')
>>> d.add(s, 'caption')
>>> s.caption.text
'Hello World'
>>>

```

请注意，您可以在绘图中的多个上下文中使用相同的形状实例；如果您选择在许多位置（如散点图【scatter plot】）使用相同的 **Circle** 对象，并使用不同的名称来访问它，它仍然是一个共享对象，并且更改将是全局的。

这为创建和修改交互式图纸提供了一种范式。

11.3 图表

其中大部分的动机是为了创建一个灵活的图表包。本节将介绍我们的图表模型背后的想法，设计目标是什么，以及图表包的哪些组件已经存在。

设计目标

以下是一些设计目标。

让简单的顶层使用变得真正简单

应该可以用最少的代码行来创建一个简单的图表，并通过合理的自动设置让它“做正确的事情”。上面的饼图片段就是这样做的。如果一个真正的图表有许多子组件，您仍然不需要与它们交互，除非您想自定义它们的功能。

允许精确定位

在出版和平面设计中，一个绝对的要求是控制每个元素的位置和风格。我们会尽量让属性以固定的尺寸和绘图比例来指定事物，而不是有自动调整大小的功能。因此，当你把y标签的字体大小变大时，“内图矩形”不会神奇地改变，即使这意味着你的标签可以溢出图表矩形的左边缘。你的工作是预览图表，并选择能用的大小和空间。

有些事情确实需要自动完成。例如，如果你想在 200 点的空间里放进 N 个条形图，而事先又不知道 N，我们就把条形图的分隔指定为条形图宽度的百分比，而不是一个点的大小，让

图表来计算。这样做还是具有确定性和可控性的。

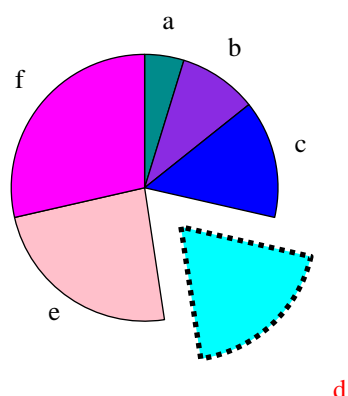
单独或分组控制子元素

我们使用智能集合类，让你自定义一组东西，或者只是其中的一个。例如你可以在我们的实验饼图中这样做。

```
from reportlab.graphics.shapes import Drawing, Circle
from reportlab.graphics.charts.piecharts import Pie
from reportlab.graphics.charts.textlabels import Label
from reportlab.lib import colors
```

```
d = Drawing(400,200)
pc = Pie()
pc.x = 150
pc.y = 50
pc.data = [10,20,30,40,50,60]
pc.labels = ['a','b','c','d','e','f']
pc.slices.strokeWidth=0.5
pc.slices[3].popout = 20
pc.slices[3].strokeWidth = 2
pc.slices[3].strokeDashArray = [2,2]
pc.slices[3].labelRadius = 1.75
pc.slices[3].fontColor = colors.red
d.add(pc, "")
```

结果:



`pc.slices[3]` 实际上是懒惰地创建了一个小对象，它保存了有关片子的信息；如果有第四个片子的话，这个对象将在绘制时被用来格式化。

只揭露你应该改变的事情

从统计学的角度来看，让你直接调整上面例子中的一个饼片的角度是错误的，因为这是由数据决定的。所以并不是所有的东西都会通过公共属性暴露出来。可能会有“后门”让你在真正需要的时候违反这一点，或者提供高级功能的方法，但一般来说，属性会是正交的。

组成和成分

图表是由可重复使用的儿童部件构建的。图例是一个易于掌握的例子。如果你需要一个特殊类型的图例（例如圆形色板），你应该将标准的图例部件子类化。然后你可以做一些像...

```
c = MyChartWithLegend()
c.legend = MyNewLegendClass() # just change it
c.legend.swatchRadius = 5 # set a property only relevant to the new one
c.data = [10,20,30] # and then configure as usual...
```

...或者创建/修改你自己的图表或绘图类，默认创建其中一个。这对于时间序列图来说也是非常重要的，因为在时间序列图中，X轴可以有很多样式。

顶层图表类会创建一些这样的组件，然后调用方法或设置私有属性来告诉它们的高度和位置-所有这些都应该为你做，而你无法定制。我们正在努力模拟这些组件应该是什么，并将在达成共识后在这里发布它们的API。

倍数

组件方法的一个必然结果是，你可以用多个图表或自定义数据图形来创建图表。我们最喜欢的例子是我们图库中由用户贡献的天气报告；我们希望能够轻松创建这样的图，将构件与它们的图例挂钩，并以一致的方式提供这些数据。

(如果你想看图片，可以在我们的网站上找到。[这儿](#))

概述

图表或图是一个放置在图纸上的对象，它本身不是一个图纸。因此，你可以控制它的位置，在同一张图上放几个，或者添加注释。

图表有两个轴，轴可以是 **Value** 轴或 **Category** 轴。轴又有一个 Labels 属性，可以让你配置所有的文本标签或单独配置每个标签。不同图表的大多数配置细节都与轴属性或轴标签有关。

对象通过上一节中讨论的接口暴露出属性；这些都是可选的，是为了让最终用户配置外观。为了使图表工作而必须设置的东西，以及图表和它的组件之间的基本通信，都是通过方法来处理的。

你可以对任何图表组件进行子类化，并使用你的替代品来代替原来的组件，只要你实现了基本的方法和属性。

Labels 属性

标签是附加在某些图表元素上的一串文本。它们用于坐标轴、标题或坐标轴旁，或附加到单个数据点上。标签可以包含换行符，但只能用一种字体。

标签的文本和 "origin"

通常由其父对象设置。它们是通过方法而不是属性来访问的。因此，X轴决定了每个刻度线标签的 "reference point"（参考点）和每个标签的数字或日期文本。然而，最终用户可以直接设置标签（或标签集合）的属性，以影响其相对于该原点的位置及其所有格式化。

```
from reportlab.graphics.shapes import Drawing, Circle
from reportlab.graphics.charts.textlabels import Label
from reportlab.lib import colors

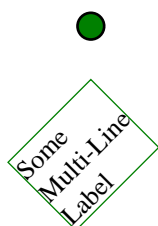
d = Drawing(200, 100)

# mark the origin of the label
d.add(Circle(100,90, 5, fillColor=colors.green))

lab = Label()
lab.setOrigin(100,90)
lab.boxAnchor = 'ne'
lab.angle = 45
lab.dx = 0
lab.dy = -20
lab.boxStrokeColor = colors.green
lab.setText('Some\nMulti-Line\nLabel')

d.add(lab)
```

结果:



在上图中，标签是相对于绿色小球定义的。文本框的东北角应在原点向下10点，并围绕该角旋转45度。

目前，标签具有以下特性，我们认为这些特性对我们迄今所看到的所有图表都是足够的。

属性	描述
<code>dx</code>	标签的 X 位移。 The label's x displacement.
<code>dy</code>	标签的 Y 位移。 The label's y displacement.
<code>angle</code>	标签的旋转角度（逆时针）。 The angle of rotation (counterclockwise) applied to the label.
<code>boxAnchor</code>	标签的框锚，'n'、'e'、'w'、's'、'ne'、'nw'、'se'、'sw'中的一种。 The label's box anchor, one of 'n', 'e', 'w', 's', 'ne', 'nw', 'se', 'sw'.
<code>textAnchor</code>	标签文字的固定位置，'start'、'middle'、'end'中的一个。 The place where to anchor the label's text, one of 'start', 'middle', 'end'.
<code>boxFillColor</code>	标签框中使用的填充颜色。 The fill color used in the label's box.
<code>boxStrokeColor</code>	标签框中使用的笔触颜色。 The stroke color used in the label's box.
<code>boxStrokeWidth</code>	标签框的线宽。 The line width of the label's box.
<code>fontName</code>	标签的字体名称。 The label's font name.
<code>fontSize</code>	标签的字体大小。 The label's font size.
<code>leading</code>	标签文字行的前导值。(leading) The leading value of the label's text lines.
<code>x</code>	参考点的X坐标。 The X-coordinate of the reference point.
<code>y</code>	参考点的Y坐标。 The Y-coordinate of the reference point.
<code>width</code>	标签的宽度。 The label's width.
<code>height</code>	标签的高度。 The label's height.

表 11-5 - Label 属性

要查看更多的具有不同属性组合的 **Label** 对象的例子，请查看文件夹 **tests** 中的ReportLab测试套件，运行脚本 **test_charts_textlabels.py** 并查看它所生成的PDF文档。

11.4 Axes

我们确定了两种基本的轴 --**Value**和**Category**。这两种轴都有水平和垂直的味道。两者都可以被子类化来制作非常特殊类型的轴。例如，如果您有复杂的规则，在时间序列应用程序中显示哪些日期，或者想要不规则的缩放，您可以覆盖该轴并创建一个新的轴。

轴负责确定从数据到图像坐标的映射；根据图表的要求变换点；绘制自己及其刻度线、网格线和轴标签。

这张图显示了两个轴，每一种都有一个，它们是在没有参考任何图表的情况下直接创建的。

```
from reportlab.graphics.shapes import Drawing, Circle
from reportlab.graphics.charts.textlabels import Label
from reportlab.graphics.charts.axes import XCategoryAxis, YValueAxis, XValueAxis
from reportlab.lib import colors

drawing = Drawing(400, 200)

data = [(10, 20, 30, 40), (15, 22, 37, 42)]

xAxis = XCategoryAxis()
xAxis.setPosition(75, 75, 300)
xAxis.configure(data)
# xAxis.categoryNames = ['Beer', 'Wine', 'Meat', 'Cannelloni']
xAxis.categoryNames = ['啤酒', '葡萄酒', '牛肉', '碎肉卷']
xAxis.labels.boxAnchor = 'n'
xAxis.labels[3].dy = -15
xAxis.labels[3].angle = 30
# xAxis.labels[3].fontName = 'Times-Bold'
xAxis.labels.fontName = 'SourceHanSans-ExtraLight'

yAxis = YValueAxis()
yAxis.setPosition(50, 50, 125)
yAxis.configure(data)

drawing.add(xAxis)
drawing.add(yAxis)
```

结果:

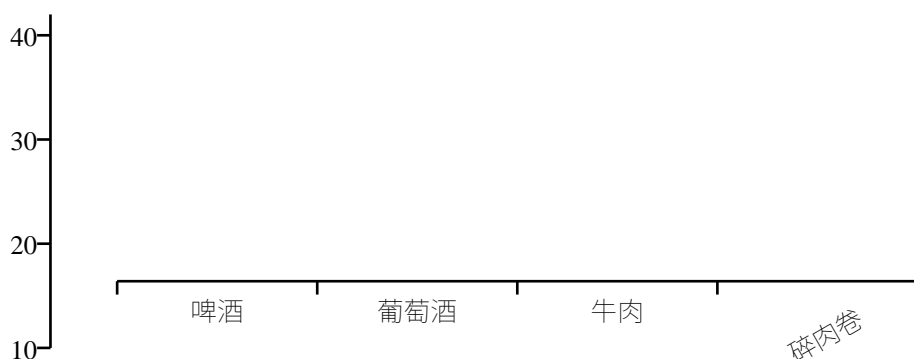


图 11-2 - 两个孤立的轴

请记住，通常情况下，你不必直接创建轴；当使用标准图表时，它自带现成的轴。这些方法是图表用来配置它和处理几何体的。不过，下面我们将详细介绍它们。正交双轴到我们描述的那些轴，除了指的是刻度线外，其他的属性基本相同。

XCategoryAxis

类别轴并没有真正的刻度，它只是把自己分成大小相等的 **buckets**。它比值轴更简单。图表（或程序员）用方法 **setPosition(x, y, length)** 设置它的位置。下一个阶段是向它显示数据，以便它能够自行配置。对于类别轴来说，这很容易--它只是计算其中一个数据系列中的数据点数量。**reversed** 属性（如果是1）表示类别应该反过来。绘制时，该轴可以通过其 **scale()** 方法为图表提供一些帮助，该方法告诉图表一个给定类别在页面上的开始和结束位置。我们还没有看到任何让人们覆盖类别的宽度或位置的需求。

一个 **XCategoryAxis** 类有以下可编辑的属性。

属性	描述
visible	轴是否应该被绘制？有时你不想显示一个或两个轴，但它们仍然需要存在，因为它们管理着点的缩放。 Should the axis be drawn at all? Sometimes you don't want to display one or both axes, but they still need to be there as they manage the scaling of points.
strokeColor	轴的颜色 Color of the axis
strokeDashArray	是否要用破折号画轴，如果要画，画什么样的轴。默认值为"None"。 Whether to draw axis with a dash and, if so, what kind.Defaults to None
strokeWidth	轴的宽度，以点为单位（points） Width of axis in points
tickUp	刻度线应突出到轴上方多远？ （请注意，使其等于图表高度会给您一条网格线） How far above the axis should the tick marks protrude? (Note that making this equal to chart height gives you a gridline)
tickDown	刻度线应突出到轴下方多远？ How far below the axis should the tick mark protrude?
categoryNames	\$None\$ 或字符串列表。该长度应与每个数据系列的长度相同。 Either None, or a list of strings. This should have the same length as each data series.
labels	刻度线标签的集合。 默认情况下，每个文本标签的“north”（北方）（例如:顶部中心）位于轴上每个类别的中心下方5点处。 您可以重新定义整个标签组或任何一个标签的任何属性。 如果 categoryNames=None，则不会绘制标签。 A collection of labels for the tick marks. By default the 'north' of each text label (i.e top centre) is positioned 5 points down from the centre of each category on the axis. You may redefine any property of the whole label group or of any one label. If categoryNames=None, no labels are drawn.
title	尚未实现。这需要像一个标签，但也可以让您直接设置文本。 它在轴下方将具有默认位置。 Not Implemented Yet. This needs to be like a label, but also lets you set the text directly. It would have a default location below the axis.

表 11-6 - XCategoryAxis 属性

YValueAxis

图中的左轴是 YValueAxis。

值轴与类别轴的不同之处在于，沿其长度的每个点都对应于图表空间中的 y 值。

轴的工作是配置自身，并将 Y 值从图表空间转换为按需点，以辅助父图表进行绘制。

setPosition(x, y, length)和configure(data)的作用与类别轴完全相同。

如果尚未完全指定最大，最小和刻度间隔，则 configure() 会导致轴选择合适的值。配置完成后，值轴可以使用 scale() 方法将 y 个数据值转换为绘图空间。

从而：

```
>>> yAxis = YValueAxis()
>>> yAxis.setPosition(50, 50, 125)
>>> data = [(10, 20, 30, 40),(15, 22, 37, 42)]
```



```
>>> yAxis.configure(data)
>>> yAxis.scale(10) # should be bottom of chart
50.0
>>> yAxis.scale(40) # should be near the top
167.1875
>>>
```

默认情况下，最高的数据点与轴的顶部对齐，最低的数据点与轴的底部对齐，轴为其刻度线点选择 'nice round numbers' (漂亮的整数)。您可以用下面的属性覆盖这些设置。

属性	描述
visible	轴是否应该被绘制？有时你不想显示一个或两个轴，但它们仍然需要存在，因为它们管理着点的缩放。 Should the axis be drawn at all? Sometimes you don't want to display one or both axes, but they still need to be there as manage the scaling of points.
strokeColor	轴的颜色 Color of the axis
strokeDashArray	是否要用破折号画轴，如果要画，画什么样的轴。默认值为 "None"。 Whether to draw axis with a dash and, if so, what kind. Defaults to None
strokeWidth	轴的宽度，以点为单位 (points) Width of axis in points
tickLeft	刻度线应突出到轴的左侧多远？ (请注意，使其等于图表宽度会给您一条网格线) How far to the left of the axis should the tick marks protrude? (Note that making this equal to chart height gives you a gridline)
tickRight	刻度线应突出到轴的右侧多远？ How far to the right of the axis should the tick mark protrude?
valueMin	轴底部应对应的 y 值。默认值为 None，在这种情况下，轴会将其设置为最低的实际数据点（例如，上例中为 10）。通常将其设置为零以避免误导眼睛。 The y value to which the bottom of the axis should correspond. Default value is None in which case the axis sets it to the lowest actual data point (e.g. 10 in the example above). It is common to set this to zero to avoid misleading the eye.
valueMax	轴顶部应对应的 y 值。默认值为 None，在这种情况下，轴会将其设置为最高实际数据点（例如，上例中为 42）。通常将其设置为“整数”，这样数据条就不会到达顶部。 The y value to which the top of the axis should correspond. Default value is None in which case the axis sets it to the highest actual data point (e.g. 42 in the example above). It is common to set this to a 'round number' so data bars do not quite reach the top.
valueStep	y 在刻度间隔之间变化。默认情况下，此值为 "None"，图表尝试选择比下面的最小刻度间距更宽的“很好的整数”。 The y change between tick intervals. By default this is None, and the chart tries to pick 'nice round numbers' which are just wider than the minimumTickSpacing below.
valueSteps	放置刻度的数字列表。 A list of numbers at which to place ticks.

<code>minimumTickSpacing</code>	<p>当<code>valueStep</code>设置为<code>None</code>时使用，否则忽略。 设计人员指定刻度线之间的距离不应小于X点（大概是基于标签字体大小和角度的考虑）。 图表尝试使用1,2,5,10,20,50,100 ...（如有必要，请减小到1以下）类型的值，直到找到大于所需间隔的间隔，并将其用于 <code>step</code>。 This is used when <code>valueStep</code> is set to <code>None</code>, and ignored otherwise. The designer specified that tick marks should be no closer than X points apart (based, presumably, on considerations of the label font size and angle). The chart tries values of the type 1,2,5,10,20,50,100... (going down below 1 if necessary) until it finds an interval which is greater than the desired spacing, and uses this for the step.</p>
<code>labelTextFormat</code>	<p>这确定了标签中的内容。与接受固定字符串的类别轴不同，<code>ValueAxis</code> 上的标签应为数字。 您可以提供“<code>%.2f</code>”之类的“格式字符串”（显示两位小数），也可以提供一个接受数字并返回字符串的任意函数。 后者的一种用法是将时间戳转换为可读的年月日格式。 This determines what goes in the labels. Unlike a category axis which accepts fixed strings, the labels on a <code>ValueAxis</code> are supposed to be numbers. You may provide either a 'format string' like <code>'%.2f'</code> (show two decimal places), or an arbitrary function which accepts a number and returns a string. One use for the latter is to convert a timestamp to a readable year-month-day format.</p>
<code>title</code>	<p>尚未实现。这需要像一个标签，但也可以让您直接设置文本。 它在轴下方将具有默认位置。 Not Implemented Yet. This needs to be like a label, but also lets you set the text directly. It would have a default location below the axis.</p>

表 11-7 - `YValueAxis` 属性

`valueSteps`属性让您明确指定刻度线的位置，因此您不必遵循常规的时间间隔。因此，您可以通过几个辅助函数来绘制月末和月末日期，而且不需要特殊的时间序列图表类。下面的代码显示了如何创建一个简单的 `XValueAxis` 与特殊的刻度间隔。请确保在调用配置方法之前设置 `valueSteps` 属性！

```
from reportlab.graphics.shapes import Drawing
from reportlab.graphics.charts.axes import XValueAxis

drawing = Drawing(400, 100)

data = [(10, 20, 30, 40)]

xAxis = XValueAxis()
xAxis.setPosition(75, 50, 300)
xAxis.valueSteps = [10, 15, 20, 30, 35, 40]
xAxis.configure(data)
xAxis.labels.boxAnchor = 'n'

drawing.add(xAxis)
```

结果:

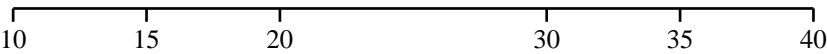


图 11-3 - 带非等距刻度线的轴

除了这些属性之外，所有的轴类都有三个属性，描述如何将其中的两个轴相互连接。再次强调，只有当你定义你自己的图表或者想使用这些坐标轴修改现有图表的外观时，这才是有趣的。这些属性在这里只是非常简单的列出，但你可以在`reportlab/graphics/axes.py`模块中找到大量的示例函数，你可以检查.....

通过在第一个轴上调用方法`joinToAxis(otherAxis, mode, pos)`将一个轴连接到另一个轴，`mode`和`pos`分别是`joinAxisMode`和`joinAxisPos`描述的属性。`'points'`表示使用绝对值，`'value'`表示使用沿轴的相对值（均由`joinAxisPos`属性表示）。

属性	描述
<code>joinAxis</code>	如果为真，则连接两个轴。 Join both axes if true.
<code>joinAxisMode</code>	用于连接轴的模式 (<code>'bottom'</code> , <code>'top'</code> , <code>'left'</code> , <code>'right'</code> , <code>'value'</code> , <code>'points'</code> , <code>None</code>). Mode used for connecting axis (<code>'bottom'</code> , <code>'top'</code> , <code>'left'</code> , <code>'right'</code> , <code>'value'</code> , <code>'points'</code> , <code>None</code>).
<code>joinAxisPos</code>	与其他轴连接的位置。 Position at which to join with other axis.

表 11-8 - Axes joining 属性

11.5 柱状图

这描述了我们目前的`VerticalBarChart`类，它使用了上面的轴和标签。我们认为这是正确方向的一步，但还远未到最后的阶段。请注意，与我们交谈过的人对这是'垂直'还是'水平'条形图的看法不一。我们选择这个名字是因为'垂直'出现在'条形图'旁边，所以我们认为它的意思是条形图而不是类别轴是垂直的。

与往常一样，我们将从一个示例开始：

```
from reportlab.graphics.shapes import Drawing
from reportlab.graphics.charts.barcharts import VerticalBarChart
from reportlab.lib import colors

drawing = Drawing(400, 200)

data = [(13, 5, 20, 22, 37, 45, 19, 4),
        (14, 6, 21, 23, 38, 46, 20, 5)]

bc = VerticalBarChart()
bc.x = 50
bc.y = 50
bc.height = 125
bc.width = 300
bc.data = data
bc.strokeColor = colors.black

bc.valueAxis.valueMin = 0
bc.valueAxis.valueMax = 50
bc.valueAxis.valueStep = 10

bc.categoryAxis.labels.boxAnchor = 'ne'
```

```
bc.categoryAxis.labels.dx = 8
bc.categoryAxis.labels.dy = -2
bc.categoryAxis.labels.angle = 30
bc.categoryAxis.categoryNames = [
    'Jan-99',
    'Feb-99',
    'Mar-99',
    'Apr-99',
    'May-99',
    'Jun-99',
    'Jul-99',
    'Aug-99',
]

drawing.add(bc)
```

结果:

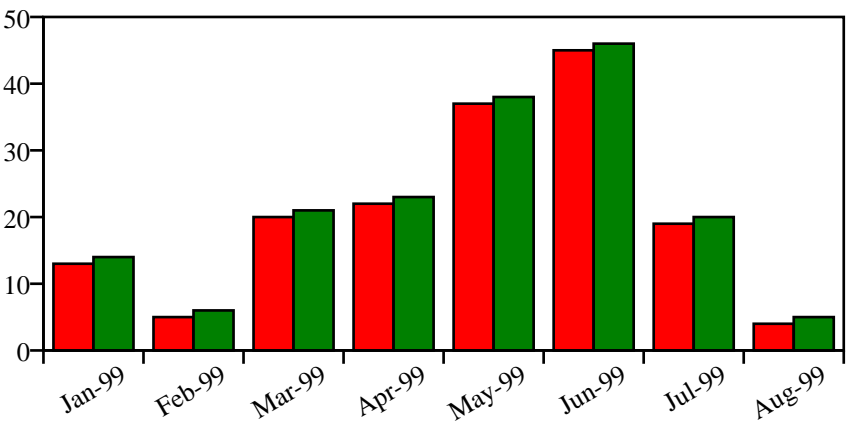


图 11-4 - 具有两个数据系列的简单柱状图

这段代码的大部分内容是关于设置坐标轴和标签的，我们已经介绍过了，下面是VerticalBarChart类的顶层属性。

属性	描述
data	这应该是“数字列表列表”或“数字元组列表”。 如果只有一个系列，则将其写为 data=[(10,20,30,42),] This should be a "list of lists of numbers" or "list of tuples of numbers". If you have just one series, write it as data = [(10,20,30,42),]
x, y, width, height	这些定义了内部的 "绘图矩形"。 我们在上面用黄色边框突出显示了这个矩形。 请注意，您的工作是将图表放置在图纸上，为所有的轴标签和刻度线留出空间。 我们指定这个 "内矩形" 是因为它可以很容易地以一致的方式布置多个图表。 These define the inner 'plot rectangle'. We highlighted this with a yellow border above. Note that it is your job to place the chart on the drawing in a way which leaves room for all the axis labels and tickmarks. We specify this 'inner rectangle' because it makes it very easy to lay out multiple charts in a consistent manner.

<code>strokeColor</code>	默认为 None。这将在绘图矩形周围绘制边框，这可能对调试很有用。轴将覆盖此位置。 Defaults to None. This will draw a border around the plot rectangle, which may be useful in debugging. Axes will overwrite this.
<code>fillColor</code>	默认为 None。这将用纯色填充绘图矩形。 (请注意，我们可以像其他任何实心形状一样实现 <code>\$dashArray\$</code> 等) Defaults to None. This will fill the plot rectangle with a solid color. (Note that we could implement <code>dashArray</code> etc. as for any other solid shape)
<code>useAbsolute</code>	默认为0。如果为1，则下面的三个属性是绝对值，以点为单位 (这意味着你可以制作一个图表，其中的条形图从绘图矩形中伸出来)； 如果为0，则是相对量，表示相关元素的比例宽度。 Defaults to 0. If 1, the three properties below are absolute values in points (which means you can make a chart where the bars stick out from the plot rectangle); if 0, they are relative quantities and indicate the proportional widths of the elements involved.
<code>barWidth</code>	柱状宽度. 默认为 10. As it says. Defaults to 10.
<code>groupSpacing</code>	默认值为5。这是每组条形图之间的间距， 如果只有一个系列，请使用 <code>\$groupSpacing\$</code> 而不是 <code>barSpacing</code> 来分割它们。 <code>\$groupSpacing\$</code> 的一半用于图表的第一个条形图之前，另一半用于结尾。 Defaults to 5. This is the space between each group of bars. If you have only one series, use <code>groupSpacing</code> and not <code>barSpacing</code> to split them up. Half of the <code>groupSpacing</code> is used before the first bar in the chart, and another half at the end.
<code>barSpacing</code>	默认值为0。这是每个组中的条之间的间距。 如果在上面的示例中绿色和红色条形之间要有一点间隙，则可以将其设置为非零。 Defaults to 0. This is the spacing between bars in each group. If you wanted a little gap between green and red bars in the example above, you would make this non-zero.
<code>barLabelFormat</code>	默认为 None。与 <code>\$YValueAxis\$</code> 一样， 如果提供函数或格式字符串，则会在每个显示数值的条旁边绘制标签。 对于正值，它们会自动定位在条的上方， 对于负值，它们会自动位于下方。 Defaults to None. As with the <code>YValueAxis</code> , if you supply a function or format string then labels will be drawn next to each bar showing the numeric value. They are positioned automatically above the bar for positive values and below for negative ones.
<code>barLabels</code>	用于格式化所有条形标签的标签集合。 由于这是一个二维数组，您可以使用此语法明确格式化第二个系列的第三个标签： <code>chart.barLabels[(1,2)].fontSize = 12</code> 。 A collection of labels used to format all bar labels. Since this is a two-dimensional array, you may explicitly format the third label of the second series using this syntax: <code>chart.barLabels[(1,2)].fontSize = 12</code>
<code>valueAxis</code>	值轴，其格式可如前所述。 The value axis, which may be formatted as described previously.
<code>categoryAxis</code>	类别轴，其格式可如前所述。 The category axis, which may be formatted as described previously.

title	还没有实现。这需要像一个标签一样，但也可以让你直接设置文本。它将有一个默认的位置在轴的下方。 Not Implemented Yet. This needs to be like a label, but also lets you set the text directly. It would have a default location below the axis.
-------	--

表 11-9 - VerticalBarChart 属性

从该表中我们可以得出结论，在上面的代码中加入以下几行，应该会使条形图组之间的间距增加一倍(`groupSpacing`属性的默认值为5点)，我们还应该看到同一组的条形组之间有一些微小的空间(`barSpacing`)。

```
bc.groupSpacing = 10
bc.barSpacing = 2.5
```

而且，实际上，这就是在将这些行添加到上面的代码之后所能看到的。
注意各个条的宽度也如何变化。
这是因为随着总图表宽度保持不变，必须从某处“占用”条形之间的空格。

```
from reportlab.graphics.shapes import Drawing
from reportlab.graphics.charts.barcharts import VerticalBarChart
from reportlab.lib import colors

drawing = Drawing(400, 200)

data = [(13, 5, 20, 22, 37, 45, 19, 4),
        (14, 6, 21, 23, 38, 46, 20, 5)]

bc = VerticalBarChart()
bc.x = 50
bc.y = 50
bc.height = 125
bc.width = 300
bc.data = data
bc.strokeColor = colors.black

bc.groupSpacing = 10
bc.barSpacing = 2.5

bc.valueAxis.valueMin = 0
bc.valueAxis.valueMax = 50
bc.valueAxis.valueStep = 10

bc.categoryAxis.labels.boxAnchor = 'ne'
bc.categoryAxis.labels.dx = 8
bc.categoryAxis.labels.dy = -2
bc.categoryAxis.labels.angle = 30
bc.categoryAxis.categoryNames = [
    'Jan-99',
    'Feb-99',
    'Mar-99',
    'Apr-99',
    'May-99',
    'Jun-99',
    'Jul-99',
    'Aug-99',
]

drawing.add(bc)
```

结果:

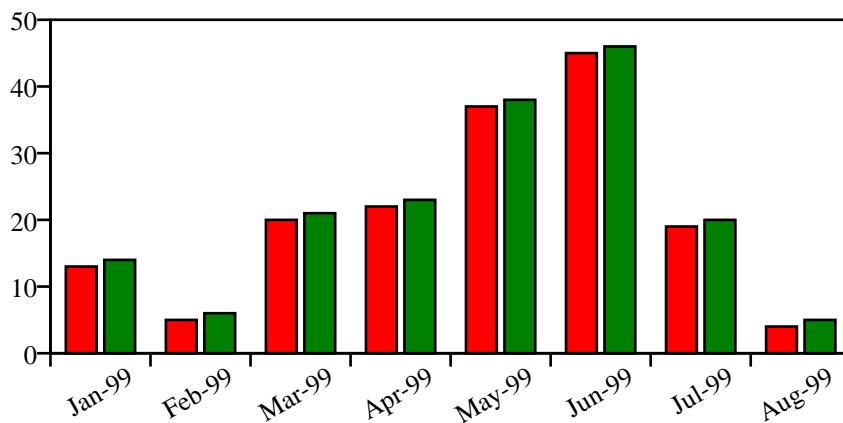


图 11-5 - 像以前一样，但间距已更改

条形图标签将自动显示为条形下限以下的负值，其他条形上限值以上的正值。

垂直条形图也支持堆叠条形图。您可以通过在 `categoryAxis` 上设置 `style` 属性为 'stacked' 来为您的图表启用这种布局。

```
bc.categoryAxis.style = 'stacked'
```

下面是以之前的图表值为例，以叠加的方式排列。

```
from reportlab.graphics.shapes import Drawing
from reportlab.graphics.charts.barcharts import VerticalBarChart
from reportlab.lib import colors

drawing = Drawing(400, 200)

data = [(13, 5, 20, 22, 37, 45, 19, 4),
        (14, 6, 21, 23, 38, 46, 20, 5)]

bc = VerticalBarChart()
bc.x = 50
bc.y = 50
bc.height = 125
bc.width = 300
bc.data = data
bc.strokeColor = colors.black

bc.groupSpacing = 10
bc.barSpacing = 2.5

bc.valueAxis.valueMin = 0
bc.valueAxis.valueMax = 100
bc.valueAxis.valueStep = 20

bc.categoryAxis.labels.boxAnchor = 'ne'
bc.categoryAxis.labels.dx = 8
bc.categoryAxis.labels.dy = -2
bc.categoryAxis.labels.angle = 30
bc.categoryAxis.categoryNames = [
    'Jan-99',
    'Feb-99',
    'Mar-99',
    'Apr-99',
    'May-99',
    'Jun-99',
    'Jul-99',
    'Aug-99',
]
bc.categoryAxis.style = 'stacked'

drawing.add(bc)
```

结果:

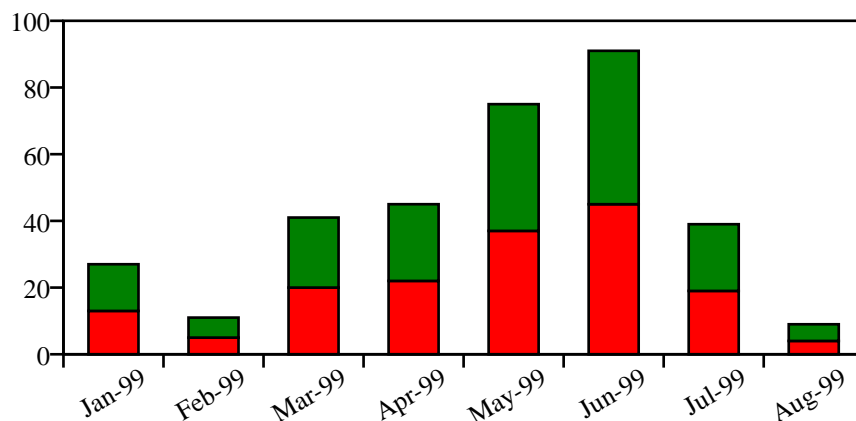


图 11-6 - 柱状叠加图

11.6 折线图

我们认为 "折线图"(Line Charts) 与 "柱状图"(Bar Charts) 本质上是一样的，只是用线代替了条。两者共享同一对类别/数值轴对。这与 "线图"不同，在"线图"(Line Plots) 中，两个轴都是值轴。

以下代码及其输出将作为一个简单的例子。后面会有更多的解释。目前，您也可以研究运行 `reportlab/lib/graphdocpy.py` 工具的输出，并在生成的PDF文档中搜索柱状图(Line Charts)的例子。

```
from reportlab.graphics.shapes import Drawing
from reportlab.graphics.charts.linecharts import HorizontalLineChart
from reportlab.lib import colors

drawing = Drawing(400, 200)

data = [(13, 5, 20, 22, 37, 45, 19, 4), (5, 20, 46, 38, 23, 21, 6, 14)]

lc = HorizontalLineChart()
lc.x = 50
lc.y = 50
lc.height = 125
lc.width = 300
lc.data = data
lc.joinedLines = 1
catNames = 'Jan Feb Mar Apr May Jun Jul Aug'.split(' ')
lc.categoryAxis.categoryNames = catNames
lc.categoryAxis.labels.boxAnchor = 'n'
lc.valueAxis.valueMin = 0
lc.valueAxis.valueMax = 60
lc.valueAxis.valueStep = 15
lc.lines[0].strokeWidth = 2
lc.lines[1].strokeWidth = 1.5
drawing.add(lc)
```

结果:

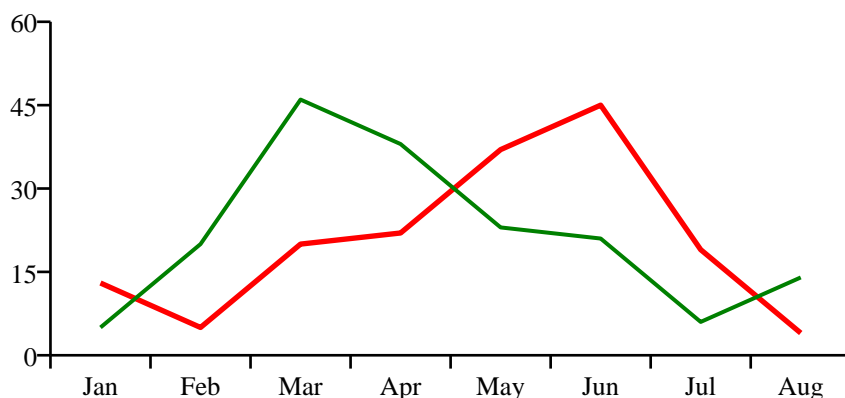


图 11-7 - HorizontalLineChart - 折线图

属性	描述
<code>data</code>	要绘制的数据，（列表）整数清单。 Data to be plotted, list of (lists of) numbers.
<code>x, y, width, height</code>	折线图的边界框。 请注意，x和y不指定中心，而是左下角 Bounding box of the line chart. Note that x and y do NOT specify the centre but the bottom left corner
<code>valueAxis</code>	值轴，其格式可如前所述。 The value axis, which may be formatted as described previously.
<code>categoryAxis</code>	类别轴，其格式可如前所述。 The category axis, which may be formatted as described previously.
<code>strokeColor</code>	默认值为 "None"。这将在绘图矩形周围画一个边框，这在调试时可能很有用。轴将覆盖它。 Defaults to None. This will draw a border around the plot rectangle, which may be useful in debugging. Axes will overwrite this.
<code>fillColor</code>	默认为 None。这将用纯色填充绘图矩形。 Defaults to None. This will fill the plot rectangle with a solid color.
<code>lines.strokeColor</code>	线的颜色 Color of the line.
<code>lines.strokeWidth</code>	线的宽度 Width of the line.
<code>lineLabels</code>	用于格式化所有行标签的标签集合。由于这是一个二维数组，您可以使用以下语法明确格式化第二行的第三个标签： <code>chart.lineLabels[(1,2)].fontSize = 12</code> 。 A collection of labels used to format all line labels. Since this is a two-dimensional array, you may explicitly format the third label of the second line using this syntax: <code>chart.lineLabels[(1,2)].fontSize = 12</code>

<code>lineLabelFormat</code>	默认值为 "None"。与YValueAxis一样，如果你提供一个函数或格式字符串，那么标签将被绘制在每一行的旁边，显示数值。 您也可以将其设置为'values'，以显示在lineLabelArray中明确定义的值。 Defaults to None. As with the YValueAxis, if you supply a function or format string then labels will be drawn next to each line showing the numeric value. You can also set it to 'values' to display the values explicitly defined in lineLabelArray.
<code>lineLabelArray</code>	行标签值的显式数组，如果存在，必须与数据的大小相匹配。 只有当上面的属性\$lineLabelFormat\$被设置为'values'时，这些标签值才会被显示。 Explicit array of line label values, must match size of data if present. These labels values will be displayed only if the property lineLabelFormat above is set to 'values'.

表 11-10 - HorizontalLineChart 属性

11.7 点线图

下面我们展示了一个更复杂的线型图的例子，也使用了一些实验功能，比如在每个数据点放置线型标记。

```
from reportlab.graphics.shapes import Drawing
from reportlab.graphics.charts.lineplots import LinePlot
from reportlab.graphics.widgets.markers import makeMarker
from reportlab.lib import colors

drawing = Drawing(400, 200)

data = [
    ((1,1), (2,2), (2.5,1), (3,3), (4,5)),
    ((1,2), (2,3), (2.5,2), (3.5,5), (4,6))
]

lp = LinePlot()
lp.x = 50
lp.y = 50
lp.height = 125
lp.width = 300
lp.data = data
lp.joinedLines = 1
lp.lines[0].symbol = makeMarker('FilledCircle')
lp.lines[1].symbol = makeMarker('Circle')
lp.lineLabelFormat = '%2.0f'
lp.strokeColor = colors.black
lp.xValueAxis.valueMin = 0
lp.xValueAxis.valueMax = 5
lp.xValueAxis.valueSteps = [1, 2, 2.5, 3, 4, 5]
lp.xValueAxis.labelTextFormat = '%2.1f'
lp.yValueAxis.valueMin = 0
lp.yValueAxis.valueMax = 7
lp.yValueAxis.valueSteps = [1, 2, 3, 5, 6]

drawing.add(lp)
```

结果:

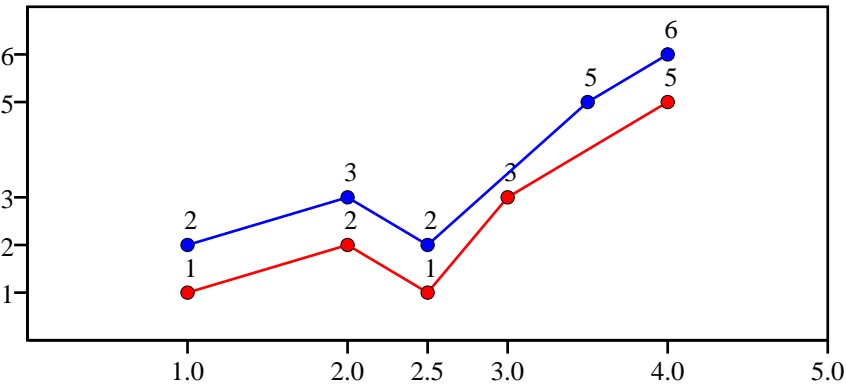


图 11-8 - LinePlot 示例图

属性	描述
data	要绘制的数据，（列表）整数清单。 Data to be plotted, list of (lists of) numbers.
x, y, width, height	折线图的边界框。 请注意，x和y不指定中心，而是左下角 Bounding box of the line chart. Note that x and y do NOT specify the centre but the bottom left corner
xValueAxis	垂直值轴，其格式可以如上所述。 The vertical value axis, which may be formatted as described previously.
yValueAxis	水平值轴，其格式可以如上所述。 The horizontal value axis, which may be formatted as described previously.
strokeColor	默认为None。 这将在绘图矩形周围绘制边框，这可能对调试很有用。 轴将覆盖此位置。 Defaults to None. This will draw a border around the plot rectangle, which may be useful in debugging. Axes will overwrite this.
strokeWidth	默认为None。 绘图矩形周围边框的宽度。 Defaults to None. Width of the border around the plot rectangle.
fillColor	默认为None。 这将在用纯色填充绘图矩形。 Defaults to None. This will fill the plot rectangle with a solid color.
lines.strokeColor	线的颜色。 Color of the line.
lines.strokeWidth	线的宽度 Width of the line.
lines.symbol	每个点使用的标记。 您可以使用函数\$makeMarker()\$创建一个新的标记。 例如，要使用一个圆，函数调用是makeMarker('Circle') Marker used for each point. You can create a new marker using the function makeMarker(). For example to use a circle, the function call would be makeMarker('Circle')

lineLabels	用于格式化所有行标签的标签集合。 由于这是一个二维数组，您可以使用以下语法明确格式化第二行的第三个标签： chart.lineLabels[(1,2)].fontSize = 12。 A collection of labels used to format all line labels. Since this is a two-dimensional array, you may explicitly format the third label of the second line using this syntax: chart.lineLabels[(1,2)].fontSize = 12
lineLabelFormat	默认值为 None。与\$YValueAxis\$一样，如果你提供一个函数或格式字符串，那么标签将被绘制。 Defaults to None. As with the YValueAxis, if you supply a function or format string then labels will be drawn next to each line showing the numeric value. You can also set it to 'values' to display the values explicitly defined in lineLabelArray.
lineLabelArray	行标签值的显式数组，如果存在，必须与数据的大小相匹配。只有当上面的属性lineLabelFormat被设置为'values'时，这些值才会被显示。 Explicit array of line label values, must match size of data if present. These labels values will be displayed only if the property lineLabelFormat above is set to 'values'.

图 11-9 - LinePlot 属性

11.8 饼图

像往常一样，我们将从一个例子开始:

```
from reportlab.graphics.shapes import Drawing
from reportlab.graphics.charts.piecharts import Pie
from reportlab.lib import colors

d = Drawing(200, 100)

pc = Pie()
pc.x = 65
pc.y = 15
pc.width = 70
pc.height = 70
pc.data = [10,20,30,40,50,60]
pc.labels = ['a','b','c','d','e','f']

pc.slices.strokeWidth=0.5
pc.slices[3].popout = 10
pc.slices[3].strokeWidth = 2
pc.slices[3].strokeDashArray = [2,2]
pc.slices[3].labelRadius = 1.75
pc.slices[3].fontColor = colors.red
d.add(pc)
```

结果:

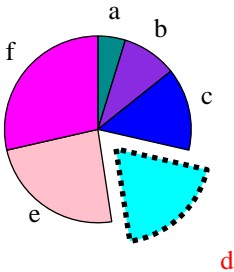


图 11-10 - 凸出的饼图

属性在下面介绍。 pie 具有 “slices” 集合，我们在同一表中记录 wedge 属性。

属性	描述
----	----

<code>data</code>	整数合集
<code>x, y, width, height</code>	饼图的边界框。请注意，x和y不必指定中心，而是指定左下角，并且宽度和高度不必相等。饼图可能是椭圆形的，并且会正确绘制切片。 A list or tuple of numbers Note that x and y do NOT specify the centre but the bottom left corner, and that width and height do not have to be equal; pies may be elliptical and slices will be drawn correctly.
<code>labels</code>	None, 或者字符串合集. 如果您不希望饼图边缘周围有标签，请将其设为“None”。 由于不可能知道切片的大小，因此我们通常不建议在饼中或饼周围放置标签。最好将它们放在图例中。 None, or a list of strings. Make it None if you don't want labels around the edge of the pie. Since it is impossible to know the size of slices, we generally discourage placing labels in or around pies; it is much better to put them in a legend alongside.
<code>startAngle</code>	第一个饼片的起始角度是什么？默认为"90"，即 12 点钟方向。 Where is the start angle of the first pie slice? The default is '90' which is twelve o'clock.
<code>direction</code>	切片的前进方向是什么？默认为 "clockwise"。(顺时针) Which direction do slices progress in? The default is 'clockwise'.
<code>sideLabels</code>	这将创建一个标签在两边各一系列的图表。 This creates a chart with the labels in two columns, one on either side.
<code>sideLabelsOffset</code>	这是饼图宽度的一部分，该宽度定义了饼图和标签列之间的水平距离。 This is a fraction of the width of the pie that defines the horizontal distance between the pie and the columns of labels.
<code>simpleLabels</code>	默认为 1. 设置为0以后启用自定义标签以及在集合切片中使用 label_ 前缀的属性。 Default is 1. Set to 0 to enable the use of customizable labels and of properties prefixed by label_ in the collection slices.
<code>slices</code>	切片的集合。这使您可以自定义每个扇形或单个扇形。 见下文 Collection of slices. This lets you customise each wedge, or individual ones. See below
<code>slices.strokeWidth</code>	扇形边框宽度 Border width for wedge
<code>slices.strokeColor</code>	扇形边框颜色 Border color
<code>slices.strokeDashArray</code>	实线或虚线配置：['solid', 'dashed'] Solid or dashed line configuration
<code>slices.popout</code>	切片应从馅饼的中心伸出多远？默认值为零。 How far out should the slice(s) stick from the centre of the pie? Default is zero.
<code>slices.fontName</code>	标签字体名称 Name of the label font
<code>slices.fontSize</code>	标签字体大小 Size of the label font
<code>slices.fontColor</code>	标签文字的颜色 Color of the label text

<code>slices.labelRadius</code>	这控制文本标签的锚点。它是半径的一小部分； 0.7将文本放置在饼中，1.2将文本放置在饼外。 (请注意，如果我们添加标签，将保留此标签以指定其锚点) This controls the anchor point for a text label. It is a fraction of the radius; 0.7 will place the text inside the pie, 1.2 will place it slightly outside. (note that if we add labels, we will keep this to specify their anchor point)
---------------------------------	---

自定义 Label

通过改变集合slices中以label_为前缀的属性，可以单独定制每个幻灯片标签。例如
`pc.slices[2].label_angle = 10` 改变第三个标签的角度。

在使用这些自定义属性之前，您需要使用以下命令禁用简单标签：`pc.simplesLabels=0`

属性	描述
<code>label_dx</code>	X轴标签偏移 X Offset of the label
<code>label_dy</code>	Y轴标签偏移 Y Offset of the label
<code>label_angle</code>	标签的角度，默认（0）为水平，90为垂直，180为上下颠倒 Angle of the label, default (0) is horizontal, 90 is vertical,180 is upside down
<code>label_boxAnchor</code>	标签的固定点 Anchoring point of the label
<code>label_boxStrokeColor</code>	标签框的边框颜色 Border color for the label box
<code>label_boxStrokeWidth</code>	标签框的边框宽度 Border width for the label box
<code>label_boxFillColor</code>	标签盒的填充颜色 Filling color of the label box
<code>label_strokeColor</code>	标签文字的边框颜色 Border color for the label text
<code>label_strokeWidth</code>	标签文字的边框宽度 Border width for the label text
<code>label_text</code>	标签文字 Text of the label
<code>label_width</code>	标签宽度 Width of the label
<code>label_maxWidth</code>	标签可以增加到的最大宽度 Maximum width the label can grow to
<code>label_height</code>	标签高度 Height of the label
<code>label_textAnchor</code>	标签可以增加到的最大高度 Maximum height the label can grow to
<code>label_visible</code>	如果要绘制标签，则为True True if the label is to be drawn
<code>label_topPadding</code>	盒子的上边距(Padding at top of box)
<code>label_leftPadding</code>	盒子的左边距(Padding at left of box)

label_rightPadding	盒子的右边距(Padding at right of box)
label_bottomPadding	盒子的下边距(Padding at bottom of box)
label_simple_pointer	为简单指针设置为1(Set to 1 for simple pointers)
label_pointer_strokeColor	指示线颜色 Color of indicator line
label_pointer_strokeWidth	指示线宽度 Width of indicator line

表 11-11 - Pie.slices 自定义 label

Side Labels 侧面标签

如果 `sideLabels` 属性被设置为 `true`，那么切片分为两列，两边各一列。
饼和饼的起始角度将被自动设置。右侧栏的锚点设置为 "start"，并设置了左手列的锚点设置为 "end"。饼的边缘与左手列中任何一个的边缘的距离。列由 `sideLabelsOffset` 属性决定，该属性为饼的宽度的一小部分。如果改变 `xradius`，饼会与标签重叠，这样一来...。我们建议将 `xradius` 改为 `None`。下面是一个例子。

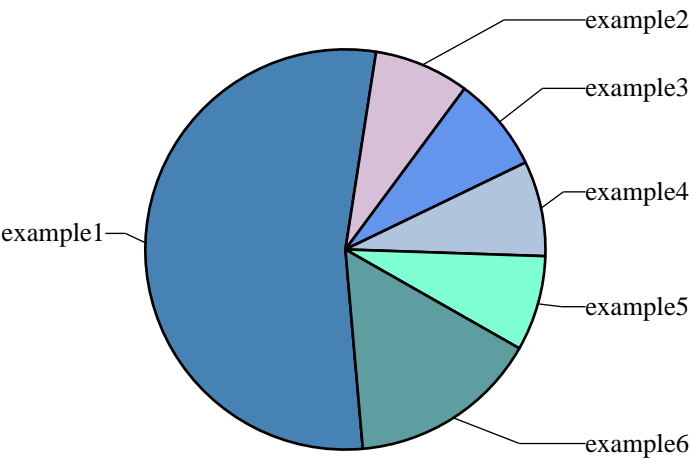


图 11 - 11: 一个 `sideLabels=1` 的饼图示例

如果将 `sideLabels` 设置为 `True`，则某些属性将变得多余，例如 `pointerLabelMode`。同样，`sideLabelsOffset` 仅在将 `sideLabels` 设置为 `true` 时更改饼图。

一些问题

如果切片过多，指针可能会交叉。

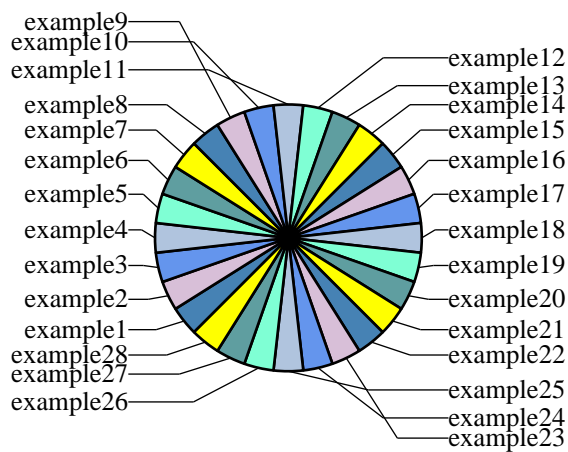


图 11 - 12: 指针交叉的例子

另外，如果标签对应的片子不相邻，尽管有 `checkLabelOverlap`，标签也可以重叠。

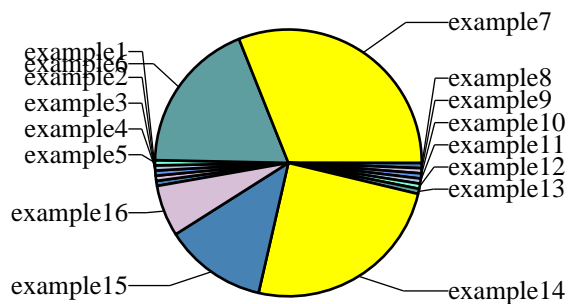


图 11 - 13: 标签重叠的示例

11.9 Legends

可以找到各种初步的图例类，但需要进行清理以与图表模型的其他部分保持一致。图例是指定图表颜色和线条风格的自然场所；我们建议每个图表都创建一个不可见的 `legend` 属性。然后，将执行以下操作以指定颜色：

```
myChart.legend.defaultColors = [red, green, blue]
```

还可以定义一组共享相同图例的图表：

```
myLegend = Legend()
myLegend.defaultColor = [red, green.....] #yuck!
myLegend.columns = 2
# etc.
chart1.legend = myLegend
chart2.legend = myLegend
chart3.legend = myLegend
```


这样行吗？直接指定图表颜色是否可以接受？

存在的问题

有几个问题已经解决了，但现在开始真正公开这些问题还为时过早。不过，这里有一个正在进行的事情清单。

- 颜色规范: -- 现在图表有一个没文档化的属性 `defaultColors`, 该属性提供要循环显示的颜色列表，以便每个数据系列都有自己的颜色。现在，如果你要引入图例，则需要确保它具有相同的颜色列表。最有可能的是，它将被一种方案取代，该方案指定一种图例，该图例包含每个数据系列具有不同值的属性。然后，该图例也可以由多个图表共享，但本身不必可见。
- 额外的图表类型--当目前的设计变得更加稳定时，我们希望增加条形图的变体，以处理百分位条形图以及这里看到的并排变体。

展望 (Outlook)

处理全部图表类型需要一些时间。我们预计将首先完成柱状图和饼图，然后试行更多的普通图。

X-Y Plots

大多数其他图都涉及两个值轴，并以某种形式直接绘制x-y数据。该系列可以绘制为线条，标记符号，两者兼而有之,或自定义图形（例如，开-高-低-闭图形）所有这些都具有缩放和轴/标题格式的概念。在某一点上，一个例程将在数据序列上循环，并在给定的x-y位置上对数据点 "做一些事情"。给定一个基本的折线图，只需覆盖一个方法--比如说，`drawSeries()`，就可以很容易地推导出一个自定义的图表类型。

自定义标记和自定义形状

众所周知的绘图软件包，如 `excel`、`Mathematica`和`Excel`，都提供了一系列的标记类型来添加到图表中。我们可以做得更好--你可以编写任何一种你想要的图表部件，只需告诉图表使用它作为例子。

组合图

结合多种绘图类型真的很容易。你只需在同一个矩形中绘制几个图表（条形图、线形图或其他什么图），根据需要取消显示轴。因此，一个图表可以将一条线与左轴上15年内的苏格兰伤寒病例相关联，右轴上有一组显示通货膨胀率的条形图。如果有谁能提醒我们这个例子的出处，我们会注明出处，并很高兴地展示这个著名的图表作为例子。

互动式编辑

图形包的一个原则是使图形组件的所有 "有趣的" 属性都可以通过设置相应的公共属性的适当值来访问和改变。这使得我们很想建立一个像GUI编辑器一样的工具，来帮助你交互式地完成这些工作。

ReportLab使用Tkinter工具箱构建了这样的工具，该工具箱可加载描述图纸的纯Python代码并记录您的属性编辑操作。然后，此“更改历史记录”用于为该图表的子类创建代码，例如，可以像其他任何图表一样立即保存和使用该代码，或将其用作另一个交互式编辑会话的新起点。

不过这还在进行中，发布的条件还需要进一步细化。

其他

这并不是对所有图表类的详尽介绍。这些类还在不断地改进中。要查看当前发行版中的确切内容，请使用 `graphdocpy.py`工具。默认情况下，它将在 `reportlab/graphics` 上运行，并生成一份完整的报告。(如果你想在其他模块或软件包上运行它，`graphdocpy.py -h` 会打印出一条帮助信息，告诉你如何运行。)

这是“文档小部件” (Documenting Widgets) 部分中提到的工具

11.10 Shapes 多边形

本节介绍形状的概念及其作为图形库生成的所有输出的构建块的重要性。介绍了现有形状的一些属性及其与图表的关系，并简要介绍了针对不同输出格式使用不同渲染器的概念。

可用形状

绘画是由形状组成的。任何东西都可以通过组合相同的原始图形集来构建。模块`shapes.py`提供了一些可以添加到图形中的基本形状和构造。它们是

- Rect - 矩形
- Circle - 圆
- Ellipse - 椭圆
- Wedge (a pie slice) - 扇形
- Polygon - 多边形
- Line - 线
- PolyLine - 折线
- String - 字符串
- Group - 组
- Path (还没有完全实现，但将来会加入) - 路径

下面的图画来自于我们的测试套件，显示了大部分的基本形状（除了组）。那些有绿色填充面的图形也被称为实体图形。（这些是Rect、Circle、Ellipse、Wedge和Polygon）。

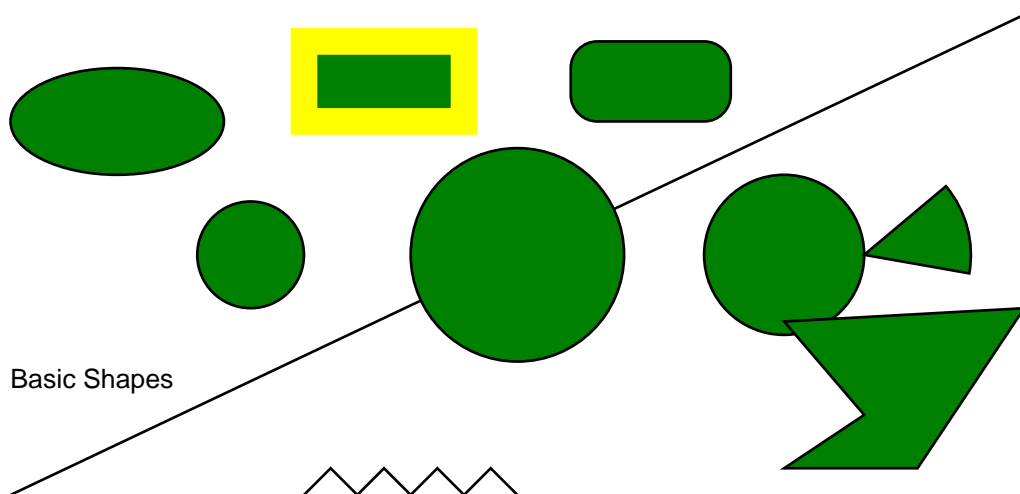


图 11 - 14 : 基本的图形

形状属性

形状有两种属性 --有的用来定义其几何形状，有的用来定义其样式。让我们创建一个红色的矩形，有3点粗的绿色边框。

```
from reportlab.graphics.shapes import Drawing
from reportlab.graphics.shapes import Rect
from reportlab.lib.colors import red, green

d = Drawing(220,120)
r = Rect(5, 5, 200, 100)
r.fillColor = red
r.strokeColor = green
r.strokeWidth = 3
d.add(r)
```

结果:

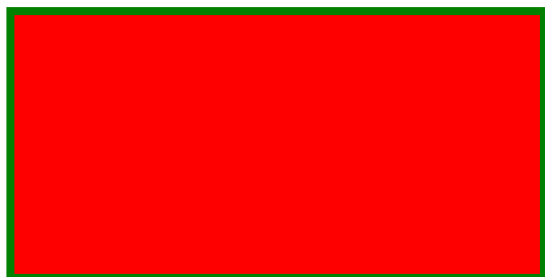


图 11-15 - 红色矩形和绿色边框



注意：在未来的例子中，我们将省略导入语句。

所有的形状都有一些可以设置的属性。在交互式提示下，我们可以使用它们的`dumpProperties()`方法来列出这些属性。下面是你可以用来配置一个`Rect.Rex`的方法。

```
>>> r.dumpProperties()
fillColor = Color(1.00,0.00,0.00)
height = 100
rx = 0
ry = 0
strokeColor = Color(0.00,0.50,0.00)
strokeDashArray = None
strokeLineCap = 0
strokeLineJoin = 0
strokeMiterLimit = 0
strokeWidth = 3
width = 200
x = 5
y = 5
>>>
```

形状一般有`style`属性和`geometry`属性。`x`, `y`, `width` 和 `height` 是几何属性的一部分，在创建矩形时必须提供，因为没有这些属性就没有什么意义。其他的属性是可选的，并且有合理的默认值。

你可以在随后的行中设置其他属性，或者将它们作为可选参数传递给构造函数。我们也可以用这种方式来创建我们的矩形。

```
>>> r = Rect(5, 5, 200, 100,
            fillColor=red,
            strokeColor=green,
            strokeWidth=3)
```

我们来看看样式属性。`fillColor`是显而易见的。`stroke` 是形状边缘的发布术语；`stroke`有一个颜色、宽度，可能还有一个破折号图案，以及一些（很少使用的）功能，用于当一条线转角时发生的情况。`rx` 和 `ry` 是可选的几何属性，用于定义圆角矩形的角半径。

其他所有的实体形状都具有相同的样式属性。

Lines - 线

我们提供单条直线、多条直线和曲线。线条具有所有的`stroke*`属性，但没有`fillColor`。下面是一些直线和多线的例子以及相应的图形输出。

```
from reportlab.graphics.shapes import Drawing
from reportlab.lib import colors
from reportlab.graphics.shapes import (
    Drawing, Line, PolyLine
)

d = Drawing(400, 200)
d.add(Line(50,50, 300,100,
           strokeColor=colors.blue, strokeWidth=5))
d.add(Line(50,100, 300,50,
           strokeColor=colors.red,
           strokeWidth=10,
```

```
strokeDashArray=[10, 20]))
d.add(PolyLine([120,110, 130,150, 140,110, 150,150, 160,110,
170,150, 180,110, 190,150, 200,110],
strokeWidth=2,
strokeColor=colors.purple))
```

结果:

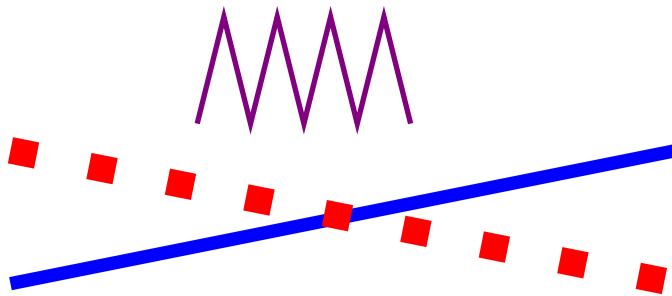


图 11-16 - 线和折线示例

字符串

ReportLab

图形包并不是为花哨的文本布局而设计的，但它可以将字符串放置在所需的位置上，并进行 left/right/center 对齐。让我们指定一个 String 对象，看看它的属性。

```
>>> s = String(10, 50, 'Hello World')
>>> s.dumpProperties()
fillColor = Color(0.00,0.00,0.00)
fontName = Times-Roman
fontSize = 10
text = Hello World
textAnchor = start
x = 10
y = 50
>>>
```

字符串有一个 `textAnchor` 属性，它的值可以是 'start'、'middle'、'end' 之一。如果设置为 'start'，则 `x` 和 `y` 与字符串的开始相关，以此类推。这提供了一个简单的方法来对齐文本。

字符串使用一个通用的字体标准：Acrobat Reader 中存在的 Type 1 Postscript 字体。因此，我们可以在 ReportLab 中使用基本的 14 种字体，并获得准确的指标。我们最近还增加了对额外的 Type 1 字体的支持，渲染器都知道如何渲染 Type 1 字体。

下面是一个使用下面代码片段的更漂亮的例子。请查阅 ReportLab 用户指南，了解像 'DarkGardenMK' 这样的非标准字体是如何被注册的。

```
d = Drawing(400, 200)
for size in range(12, 36, 4):
    d.add(String(10+size*2, 10+size*2, 'Hello World',
                fontName='Times-Roman',
                fontSize=size))

d.add(String(130, 120, 'Hello World',
            fontName='Courier',
            fontSize=36))

d.add(String(150, 160, 'Hello World',
```

```
fontName='DarkGardenMK',
fontSize=36))
```



图 11 - 17 : 花式字体样例

Paths

Postscript paths

是图形学中一个广为人知的概念。它们在reportlab/graphics中还没有实现，但很快就会实现。

Groups

最后，我们有 **Group** 对象。一个组有一个内容列表，就是其他节点。它还可以应用变换 -- 它的内容可以被旋转、缩放或移动。如果你懂数学，你可以直接设置变换。否则它提供了旋转、缩放等方法。在这里，我们做一个旋转和转换的组。

```
>>> g=Group(shape1, shape2, shape3)
>>> g.rotate(30)
>>> g.translate(50, 200)
```

组提供了一个重复使用的工具。你可以做一堆形状来表示某个组件 -- 比如说，一个坐标系 -- 并把它们放在一个叫做 "Axis" (轴) 的组里。然后你可以把这个组放到其他组中，每个组都有不同的平移和旋转，你就会得到一堆轴。它仍然是同一个组，被画在不同的地方。

让我们用一些只稍微多一点的代码来做这件事。

```
d = Drawing(400, 200)

Axis = Group(
    Line(0,0,100,0), # x axis
    Line(0,0,0,50), # y axis
    Line(0,10,10,10), # ticks on y axis
    Line(0,20,10,20),
    Line(0,30,10,30),
    Line(0,40,10,40),
    Line(10,0,10,10), # ticks on x axis
    Line(20,0,20,10),
    Line(30,0,30,10),
    Line(40,0,40,10),
    Line(50,0,50,10),
    Line(60,0,60,10),
    Line(70,0,70,10),
    Line(80,0,80,10),
    Line(90,0,90,10),
    String(20, 35, 'Axes', fill=colors.black)
)
```

```

firstAxisGroup = Group(Axis)
firstAxisGroup.translate(10,10)
d.add(firstAxisGroup)

secondAxisGroup = Group(Axis)
secondAxisGroup.translate(150,10)
secondAxisGroup.rotate(15)

d.add(secondAxisGroup)

thirdAxisGroup = Group(Axis,
                       transform=mmult(translate(300,10),
                                       rotate(30)))
d.add(thirdAxisGroup)

```

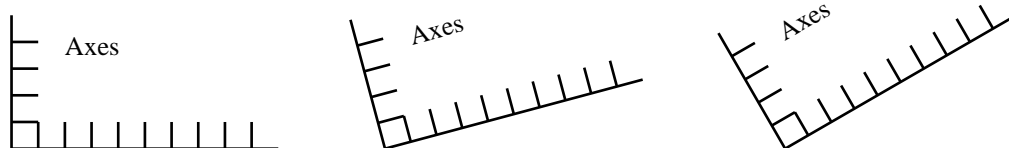


图 11 - 18 : Groups 示例

11.11 小部件

现在，我们描述小部件及其与形状的关系。
通过许多示例，展示了小部件如何使可重用图形组件成为现实。

Shapes vs. Widgets

到目前为止，图纸一直是 "pure data"(纯数据)。除了协助程序员检查和检验图纸外，它们中没有任何代码可以真正做任何事情。事实上，这是整个概念的基石，也是让我们实现可移植性的原因--渲染器只需要实现原始形状。

我们希望构建可重复使用的图形对象，包括一个强大的图表库。要做到这一点，我们需要重用比矩形和圆形更有形的东西。我们应该能够编写对象供其他人重用--箭头、齿轮、文本框、UML图节点，甚至是完全成熟的图表。

小部件标准是建立在shapes模块之上的标准，任何人都可以编写新的小部件，我们可以建立它们的库。**Widget** 支持 **getProperties()** 和 **setProperties()** 方法，所以你可以检查和修改，也可以用统一的方式记录它们。

- 小部件是一个可重复使用的形状
 - 当调用 **draw()** 方法时，它可以在没有参数的情况下进行初始化，它创建了一个原始形状或一个组来表示自己。
 - 它可以有任何你想要的参数，它们可以驱动它的绘制方式。
 - 它有一个 **demo()** 方法，该方法应以 200x100 矩形返回一个精美的绘制示例。这是自动文档编制工具的基础。
- demo()** 方法也应该有一个写得很好的文档字符串，因为它也可以打印！

小部件与图形只是一捆形状的想法背道而驰。他们肯定有自己的代码吗？它们的工作方式是，小部件可以将自身转换为一组原始形状。如果其某些组件本身就是小部件，它们也将被转换。这在渲染过程中自动发生。渲染器将看不到图表小部件，而只会看到矩形，直线和字符串的集合。您还可以显式“flatten out”（扁平化）工程图，从而将所有小部件都转换为基元。

使用一个小部件

让我们想象一个简单的新部件。我们将使用一个小部件来绘制一张脸，然后展示它是如何实现的。

```
>>> from reportlab.lib import colors
>>> from reportlab.graphics import shapes
>>> from reportlab.graphics import widgetbase
>>> from reportlab.graphics import renderPDF
>>> d = shapes.Drawing(200, 100)
>>> f = widgetbase.Face()
>>> f.skinColor = colors.yellow
>>> f.mood = "sad"
>>> d.add(f)
>>> renderPDF.drawToFile(d, 'face.pdf', 'A Face')
```



图 11 - 19 : 小部件示例

让我们看看它有哪些可用的属性，使用我们前面看到的 `setProperties()` 方法。

```
>>> f.dumpProperties()
eyeColor = Color(0.00,0.00,1.00)
mood = sad
size = 80
skinColor = Color(1.00,1.00,0.00)
x = 10
y = 10
>>>
```

上面的代码似乎奇怪的一件事是，当我们制作面孔时，我们没有设置大小或位置。这是必要的折衷方案，以允许使用一个统一的界面来构造小部件并对其进行记录-它们不能在其 `__init__()` 方法中要求参数。取而代之的是，通常将它们设计为适合 200 x 100 的窗口，然后在创建后通过设置诸如 `x`, `y`, `width` 等属性来移动或调整它们的大小。

此外，一个小部件总是提供一个 `demo()` 方法。像这样简单的总是在设置属性之前做一些合理的事情，但更复杂的像图表这样的就没有任何数据可供绘制。文档工具会调用 `demo()`，这样你的花哨的新图表类就可以创建一张图来展示它的能力。

以下是一些简单的小部件，可在模块 `signsandsymbols.py` 中使用。

```
from reportlab.graphics.shapes import Drawing
from reportlab.graphics.widgets import signsandsymbols

d = Drawing(230, 230)

ne = signsandsymbols.NoEntry()
ds = signsandsymbols.DangerSign()
fd = signsandsymbols.FloppyDisk()
ns = signsandsymbols.NoSmoking()
```

```

ne.x, ne.y = 10, 10
ds.x, ds.y = 120, 10
fd.x, fd.y = 10, 120
ns.x, ns.y = 120, 120

```

```

d.add(ne)
d.add(ds)
d.add(fd)
d.add(ns)

```

结果:

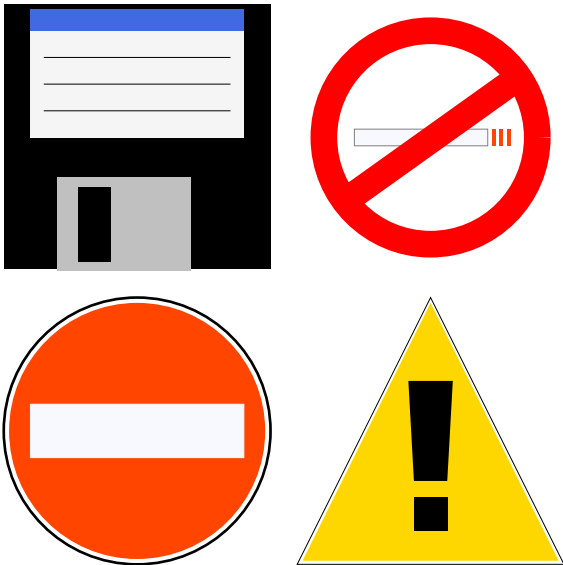


图 11-20 - 一些来自 `signandsymbols.py` 的例子。

而这是生成它们所需要的代码，如上图所示。

复合小部件

让我们想象一下，一个复合小组件可以并排绘制两个面孔。当你有了 `Face widget` 时，就可以很容易地构建这个小部件。

```

>>> tf = widgetbase.TwoFaces()
>>> tf.faceOne.mood
'happy'
>>> tf.faceTwo.mood
'sad'
>>> tf.dumpProperties()
faceOne.eyeColor = Color(0.00,0.00,1.00)
faceOne.mood = happy
faceOne.size = 80
faceOne.skinColor = None
faceOne.x = 10
faceOne.y = 10
faceTwo.eyeColor = Color(0.00,0.00,1.00)
faceTwo.mood = sad
faceTwo.size = 80
faceTwo.skinColor = None
faceTwo.x = 100
faceTwo.y = 10
>>>

```

故意暴露了 `'faceOne'` 和 `'faceTwo'` 这两个属性，这样你就可以直接获取它们。也可以有顶层属性，但本例中没有。

校验小部件

小组件设计者决定验证的策略，但默认情况下，如果设计者提供了检查信息，它们就会像形状一样工作--检查每一个任务。

实现小部件

我们试图让它尽可能容易地实现小部件。下面是一个不做任何类型检查的 **Face** 小组件的代码。

```
class Face(Widget):
    """This draws a face with two eyes, mouth and nose."""

    def __init__(self):
        self.x = 10
        self.y = 10
        self.size = 80
        self.skinColor = None
        self.eyeColor = colors.blue
        self.mood = 'happy'

    def draw(self):
        s = self.size # abbreviate as we will use this a lot
        g = shapes.Group()
        g.transform = [1,0,0,1,self.x, self.y]
        # background
        g.add(shapes.Circle(s * 0.5, s * 0.5, s * 0.5,
                           fillColor=self.skinColor))
        # CODE OMITTED TO MAKE MORE SHAPES
        return g
```

我们在这个文档中省略了所有绘制形状的代码，但你可以从 `widgetbase.py` 中找到它。

默认情况下，任何没有前导下划线的属性都会被 `setProperties` 返回。这是为了鼓励一致的编码惯例而特意制定的政策。

一旦你的 `widget` 工作了，你可能想要添加对验证的支持。这涉及到在类中添加一个名为 `_verifyMap` 的字典，它从属性名映射到“检查函数”。`widgetbase.py` 模块定义了一堆检查函数，比如 `isNumber`, `isListOfShapes` 等等。你也可以简单地使用 `None`，这意味着属性必须存在，但可以有任何类型。而且你也可以写你自己的检查函数。我们想将 `"mood"` 自定义属性限制为 `"happy"`、`"sad"` 或 `"ok"` 等值。所以我们这样做：

```
class Face(Widget):
    """This draws a face with two eyes. It exposes a
    couple of properties to configure itself and hides
    all other details"""
    def checkMood(moodName):
        return (moodName in ('happy','sad','ok'))
    _verifyMap = {
        'x': shapes.isNumber,
        'y': shapes.isNumber,
        'size': shapes.isNumber,
        'skinColor': shapes.isColorOrNone,
        'eyeColor': shapes.isColorOrNone,
        'mood': checkMood
    }
```

这个检查将在每次属性分配时执行；或者，如果 `config.shapeChecking` 是关闭的，每当你调用 `myFace.verify()` 时，这个检查就会被执行。

记录小部件

我们正在开发一个通用工具来记录任何 Python 包或模块；它已经记录到 **ReportLab** 中，并将用于为

ReportLab 包生成一个引用。当它遇到小部件时，它会在手册中添加额外的章节，包括：

- 您的小组件类的文档字符串
- 从您的 `demo()` 方法中提取的代码片段，以便人们可以看到如何使用它
- 由 `demo()` 方法产生的图画。
- 绘图中小组件的属性转储。

这个工具意味着我们可以保证在网站和印刷品上的小部件和图表上都有最新的文档；而且您也可以为自己的小部件做同样的事情！

小工具设计策略

我们无法提出一个一致的架构来设计

widget，所以我们把这个问题留给了作者！如果你不喜欢默认验证策略，或者是 **setProperties/getProperties** 的工作方式，你可以自己覆盖它们。

对于简单的 **widgets**，建议你做我们上面所做的：选择非重叠的属性，在 **__init__** 上初始化每个属性，然后在调用 **draw()** 时构造一切。你可以使用 **__setattr__** 钩子，当某些属性被设置时，就会更新所有的东西。考虑一个饼图。如果你想暴露各个切片，你可以写这样的代码。

```
from reportlab.graphics.charts import piecharts
pc = piecharts.Pie()
pc.defaultColors = [navy, blue, skyblue] #used in rotation
pc.data = [10,30,50,25]
pc.slices[7].strokeWidth = 5
```

最后一行是有问题的，因为我们只创建了四个切片--事实上，我们可能还没有创建它们。**pc.slices[7]** 是否会引起错误？如果定义了第七个楔形图，用来覆盖默认设置，这是不是一个解决方案？我们现在把这个问题直接丢给 **widget** 作者，并建议您在暴露“子对象”之前先做一个简单的工作，因为子对象的存在取决于其他属性的值：-)

我们还讨论了父级小部件可以将属性传递给其子级的规则。

人们似乎普遍希望以一种全局的方式来表示“所有切片均从其父级的 **lineWidth** 获得其 **lineWidth**”，而无需进行大量重复编码。我们没有通用的解决方案，因此请再次将其留给小部件作者。

我们希望人们将尝试下推，下拉和模式匹配方法，并提出一些不错的东西。

同时，我们当然可以编写整体式的图表小部件，这些小部件的工作方式类似于 Visual Basic 和 Delphi。

现在看看下面的示例代码，使用一个早期版本的饼图小组件和它产生的输出。

```
from reportlab.graphics.charts.piecharts import Pie
from reportlab.graphics.shapes import Drawing, String
from reportlab.lib import colors

d = Drawing(400,200)
d.add(String(100,175,"Without labels", textAnchor="middle"))
d.add(String(300,175,"With labels", textAnchor="middle"))

pc = Pie()
pc.x = 25
pc.y = 50
pc.data = [10,20,30,40,50,60]
pc.slices[0].popout = 5
d.add(pc, 'pie1')

pc2 = Pie()
pc2.x = 150
pc2.y = 50
pc2.data = [10,20,30,40,50,60]
pc2.labels = ['a','b','c','d','e','f']
d.add(pc2, 'pie2')

pc3 = Pie()
pc3.x = 275
pc3.y = 50
pc3.data = [10,20,30,40,50,60]
pc3.labels = ['a','b','c','d','e','f']
pc3.slices.labelRadius = 0.65
pc3.slices.fontName = "Helvetica-Bold"
pc3.slices.fontSize = 16
pc3.slices.fontColor = colors.yellow
d.add(pc3, 'pie3')
```

结果:

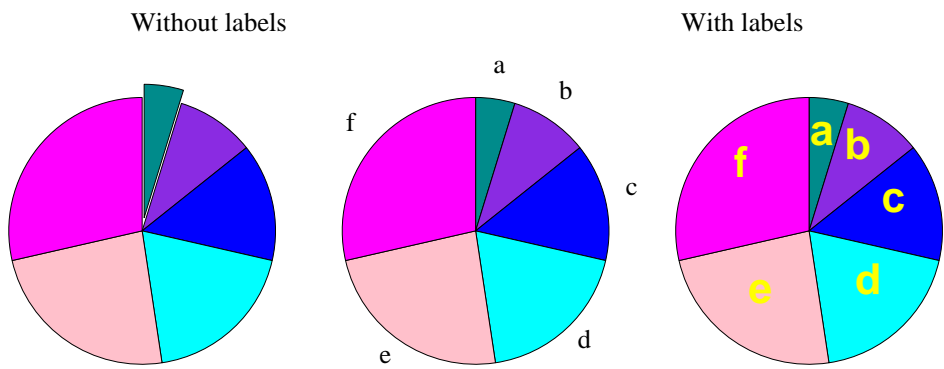


图 11-21 - 一些饼图示例

附录 A ReportLab 示例

在reportlab/demos的子目录中，有许多工作实例，几乎展示了 reportlab 使用的所有方面。

A.1 奥德赛

odyssey.py、dodyssey.py和fodyssey.py这三个脚本都是取文件odyssey.txt并生成PDF文档。包含的 odyssey.txt 很短；更长和更多的测试版本可以在 <ftp://ftp.reportlab.com/odyssey.full.zip> 找到。

```
Windows
cd reportlab\demos\odyssey
python odyssey.py
start odyssey.pdf
```

```
Linux
cd reportlab/demos/odyssey
python odyssey.py
acrord odyssey.pdf
```

odyssey.py 脚本显示了简单的格式化。它的运行速度相当快，但它所做的只是收集文本并将其强行放到画布页面上。它完全不进行段落操作，所以你可以看到 XML < 和 > 标签。

脚本 fodyssey.py 和 dodyssey.py 处理段落格式，所以你可以看到颜色变化等。这两个脚本都使用了文档模板类，dodyssey.py 脚本显示了做双列布局的能力，并使用了多个页面模板。

A.2 标准字体和颜色

在reportlab/demos/stdfonts中，脚本 stdfonts.py 可以用来说明 ReportLab 的标准字体。使用以下方法运行该脚本

```
cd reportlab\demos\stdfonts
python stdfonts.py
```

生成两个PDF文档，StandardFonts_MacRoman.pdf 和 StandardFonts_WinAnsi.pdf，其中显示了两种最常见的内置字体编码。

在reportlab/demos/colors中的 colortest.py 脚本展示了 reportlab 设置和使用颜色的不同方式。

试着运行该脚本并查看输出文档，colortest.pdf。这显示了不同的颜色空间和大量在reportlab.lib.colors模块中命名的颜色选择。

Dinu Gherman 贡献了这个有用的脚本，它使用 reportlab 从 Python 脚本中生成漂亮的彩色PDF文档，包括类、方法和函数的书签。要得到一个漂亮的主脚本版本，可以尝试一下

```
cd reportlab/demos/py2pdf
python py2pdf.py py2pdf.py
acrord py2pdf.pdf
```

即我们使用py2pdf在文档中生成一个漂亮的py2pdf.py版本，根名相同，扩展名为.pdf。

py2pdf.py脚本有很多选项，这些选项超出了这个简单介绍的范围，请参考脚本开头的注释。

A.3 Gadflypaper

reportlab/demos/gadflypaper中的Python脚本gfe.py使用了内联式的文档编制方式。这个脚本几乎完全由Aaron

Watters制作，产生了一个描述Aaron的gadfly内存数据库的Python文档。要生成该文档，请使用

```
cd reportlab\gadflypaper
python gfe.py
start gfe.pdf
```

PDF文档中的所有内容都是由脚本制作的，这就是为什么这是一种内联式文档制作方式。所以，为了生成一个标题，后面跟着一些文本，脚本使用了函数**header**和**p**，这两个函数接收一些文本并附加到一个全局故事列表中。

```
header("Conclusion")

p("""The revamped query engine design in Gadfly 2 supports
.....
and integration. """)
```

A.4 Pythonpoint

Andy Robinson改进了**pythonpoint.py**脚本(在reportlab\demos\pythonpoint中)，直到它成为一个真正有用的脚本。它接收一个包含XML标记的输入文件，并使用一个**xmllib**样式分析器将标记映射到PDF幻灯片中。当在它自己的目录下运行时，**pythonpoint.py** 将文件**pythonpoint.xml** 作为默认输入，并生成 **pythonpoint.pdf**，这是 Pythonpoint 的文档。您也可以通过一篇较早的文章看到它的运行情况。

```
cd reportlab\demos\pythonpoint
python pythonpoint.py monterey.xml
start monterey.pdf
```

pythonpoint 不仅是自文档，而且还演示了 **reportlab** 和 **PDF**。它使用了 **reportlab** 的许多功能（文档模板、表格等）。**PDF** 的奇特功能，如淡入和书签也被展示得很好。**XML** 文档的使用可以与 **gadflypaper** 演示中的**inline**风格形成对比；内容与格式完全分离。

附录 B 译者备注

后面的信息为译者在翻译本手册时，学习，查询资料做的备注，或者说总结，没有对应的英文版...

B.1 Win Ansi 编码 和 Mac Roman 编码

参考: [ANSI, ISO-8859-1和MacRoman字符集之间的差异](#)

以下为原文和译文:

Of the three main 8-bit character sets, only ISO-8859-1 is produced by a standards organization. The three sets are identical for the 95 characters from 32 to 126, the ASCII character set. The ANSI character set, also known as Windows-1252, has become a Microsoft proprietary character set; it is a superset of ISO-8859-1 with the addition of 27 characters in locations that ISO designates for control codes. Apple's proprietary MacRoman character set contains a similar variety of characters from 128 to 255, but with very few of them assigned the same numbers, and also assigns characters to the control-code positions.

The characters that appear in the first column of the following tables are generated from Unicode numeric character references, and so they should appear correctly in any Web browser that supports Unicode and that has suitable fonts available, regardless of the operating system.

- ANSI characters not present in ISO-8859-1
- ANSI characters not present in MacRoman
- ISO-8859-1 characters not present in ANSI
- ISO-8859-1 characters not present in MacRoman
- MacRoman characters not present in ANSI
- MacRoman characters not present in ISO-8859-1

译文:

在三种主要的8位字符集中，只有**ISO-8859-1**是由标准组织制作的。

这三个字符集对于32到126这95个字符，即**ASCII**字符集是相同的。**ANSI**字符集，也就是**Windows-1252**，已经成为微软的专有字符集；它是**ISO-8859-1**的超集，在ISO指定控制代码的位置增加了27个字符。苹果公司专有的**MacRoman**字符集包含从128到255个类似的各种字符，但其中很少有相同的数字，而且还将字符分配到控制码的位置。

下表第一栏中显示的字符是从**Unicode**数字字符引用生成的，因此，无论使用什么操作系统，它们都应该在支持**Unicode**并具有合适的可用字体的任何**Web**浏览器中正确显示。

- **ISO-8859-1** 中不存在 **ANSI** 字符
- **MacRoman** 中不存在 **ANSI** 字符
- **ANSI** 中不存在 **ISO-8859-1** 字符
- **MacRoman** 中不存在 **ISO-8859-1** 字符
- **ANSI** 中不存在 **MacRoman** 字符
- **ISO-8859-1** 中不存在 **MacRoman** 字符

其他参考:

Win Ansi 编码总体来说就是,

微软公司根据**windows**系统的本地化和国际化地区，默认采取的某种编码.参考:[ANSI是什么编码？](#)

B.2 生成pdf文件，推荐开源字体：思源黑体

True Type 字体介绍: <https://blog.csdn.net/gaojinshan/article/details/80319856>

思源黑体、思源宋体TTF介绍: <https://www.v2ex.com/t/399030>

GitHub 资源:

1. <https://github.com/be5invis/source-han-sans-ttf/releases>
2. <https://github.com/junmer/source-han-serif-ttf>
3. <https://github.com/Pal3love/Source-Han-TrueType>

附录 C 官方示例: Line

参考: <https://www.reportlab.com/chartgallery/line/>

C.1 Line with markers (serious)

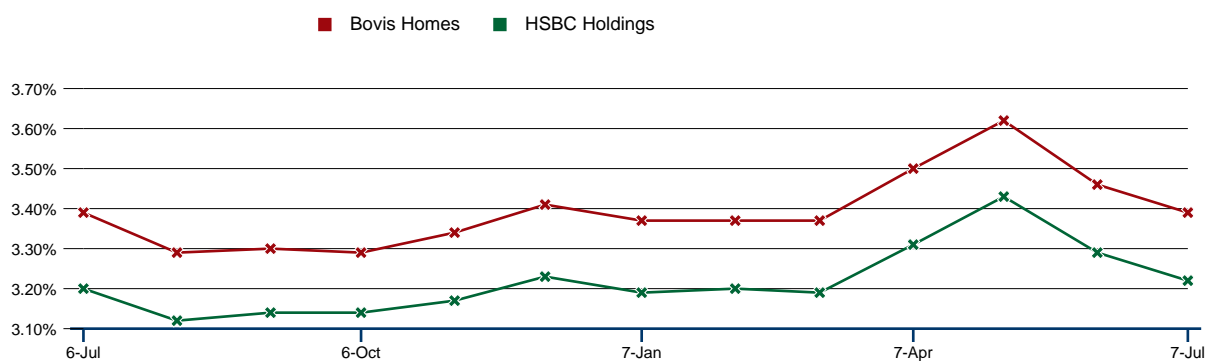


图 C-1 - 折线图(serious)

C.2 Line with markers (silly)

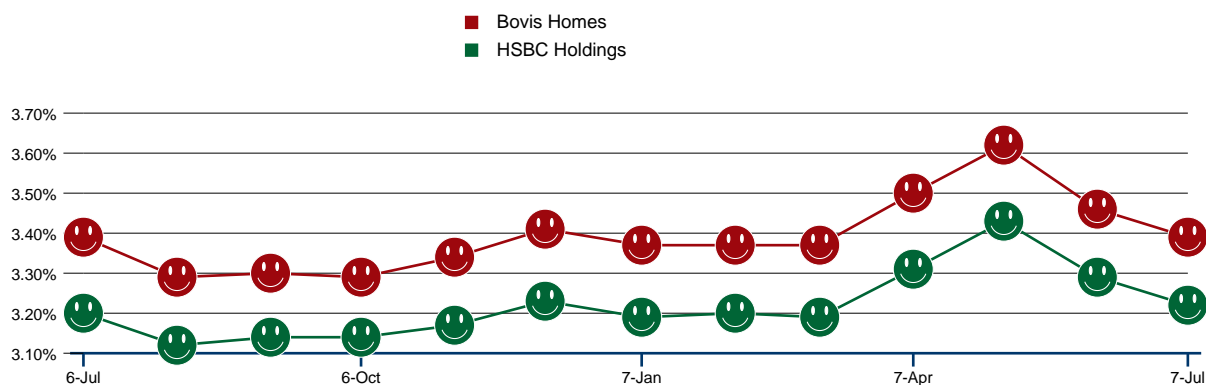


图 C-2 - 折线图(silly)

C.3 char with background color

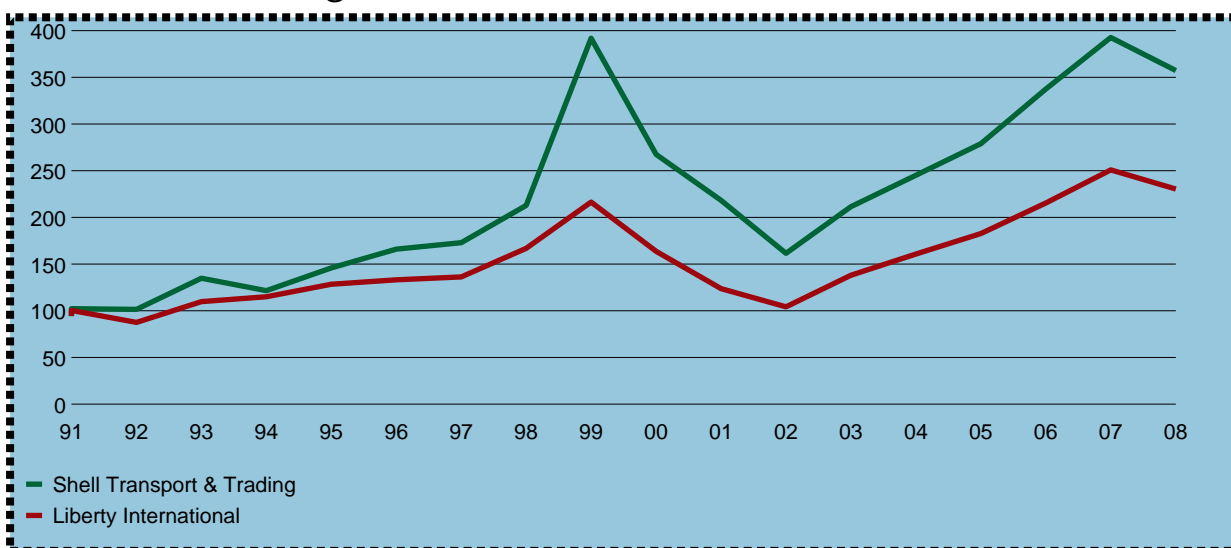


图 C-3 - 折线+背景图示例

C.4 dashed lines and number formats

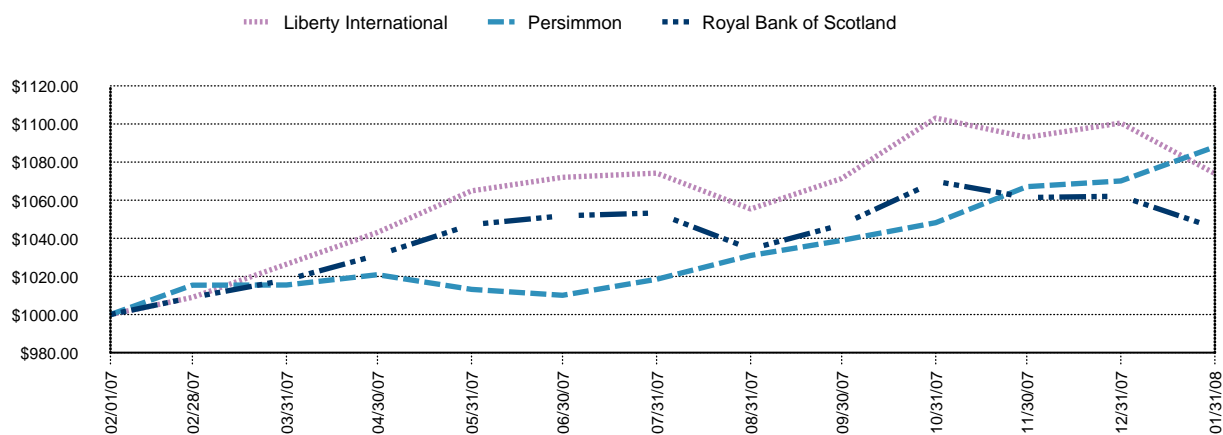


图 C-4 - 虚线+数字格式

C.5 time serious plot

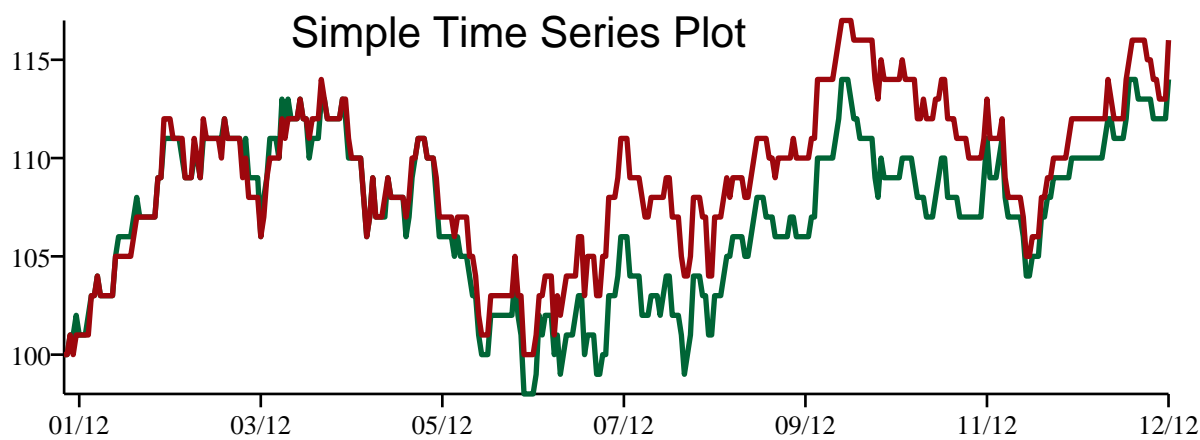


图 C-5 - 时间走势图

附录 D 官方示例: Pie

参考: <https://www.reportlab.com/chartgallery/pie/>

D.1 Basic pie

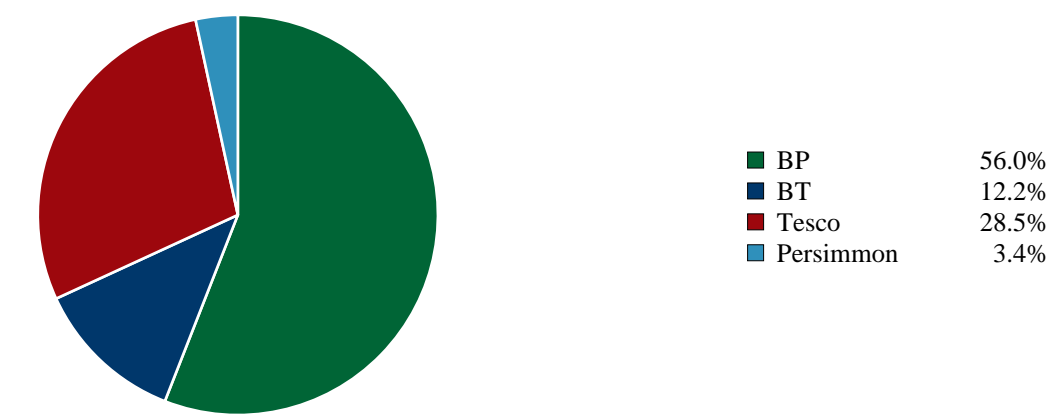


图 D-1 - 基本饼图

D.2 Pie with multi-column legend

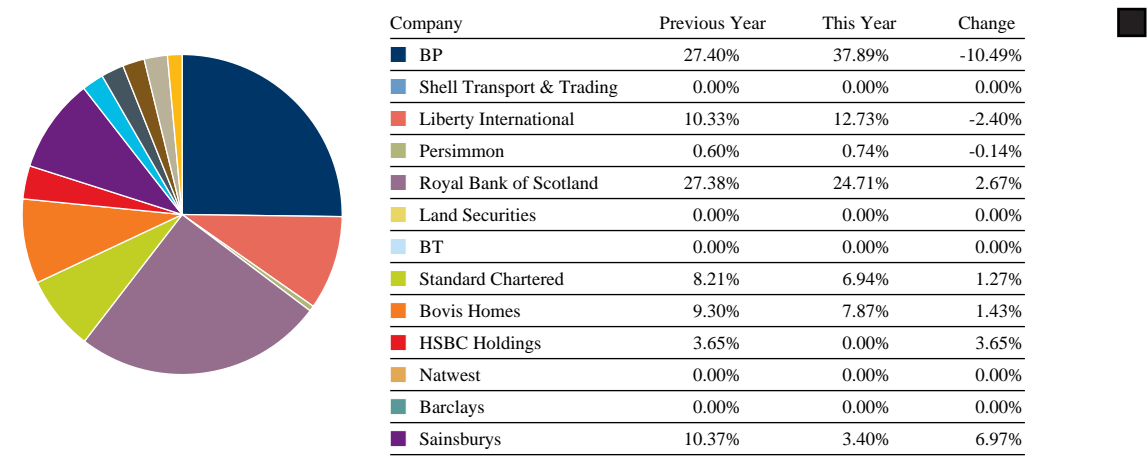


图 D-2 - 饼图+多列

D.3 Exploding pie

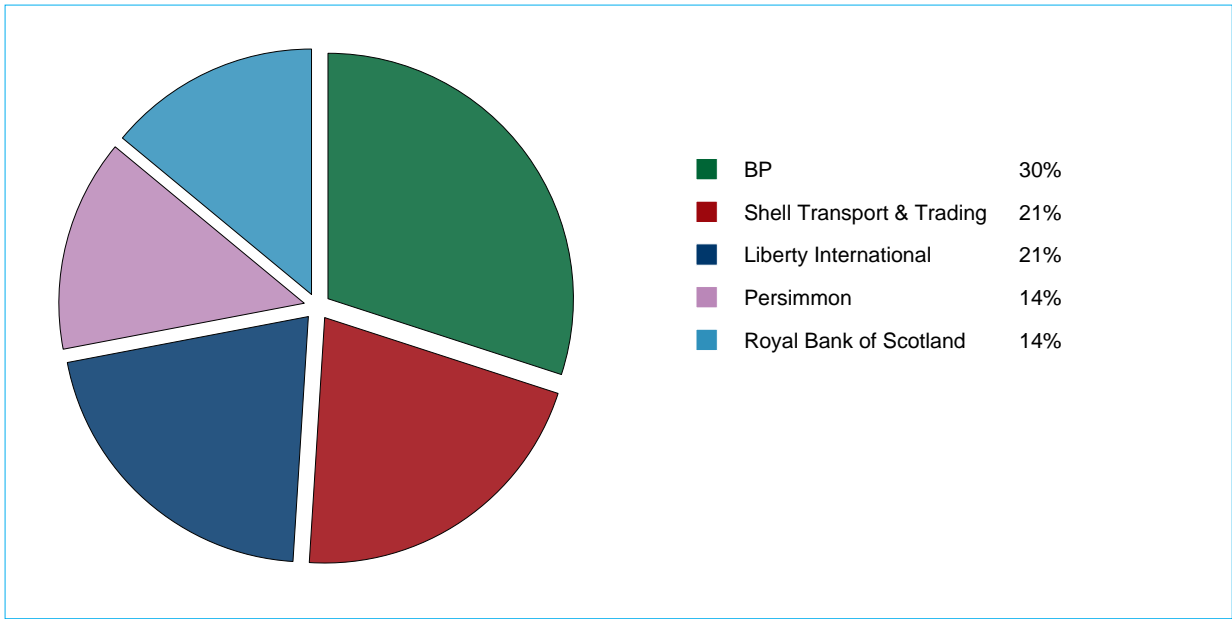


图 D-3 - Exploding 饼图

D.4 Pie with nested legend

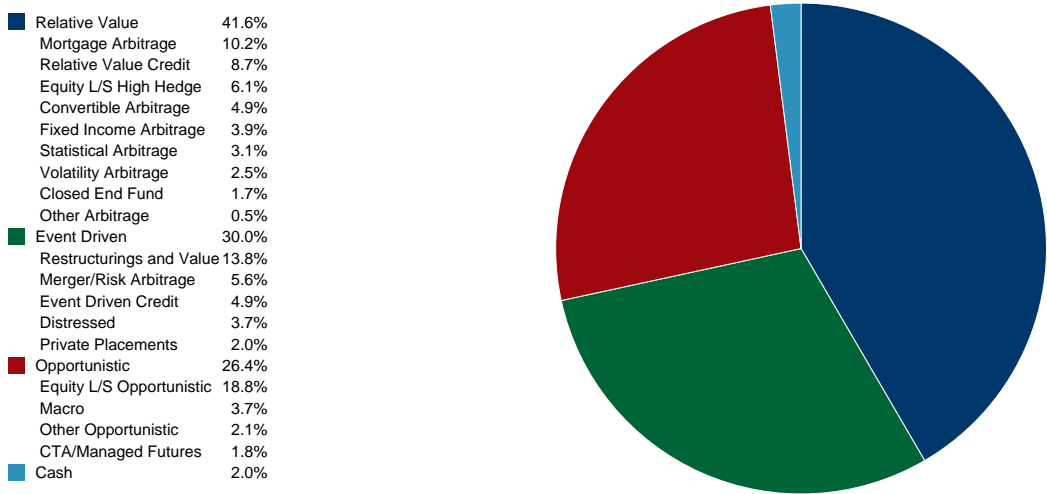


图 D-4 - 嵌套Legend 饼图

D.5 Pie with a pie

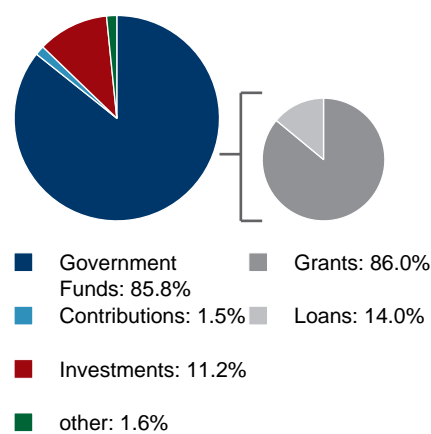


图 D-5 - 嵌套 饼图

D.6 Legend with text wrapping

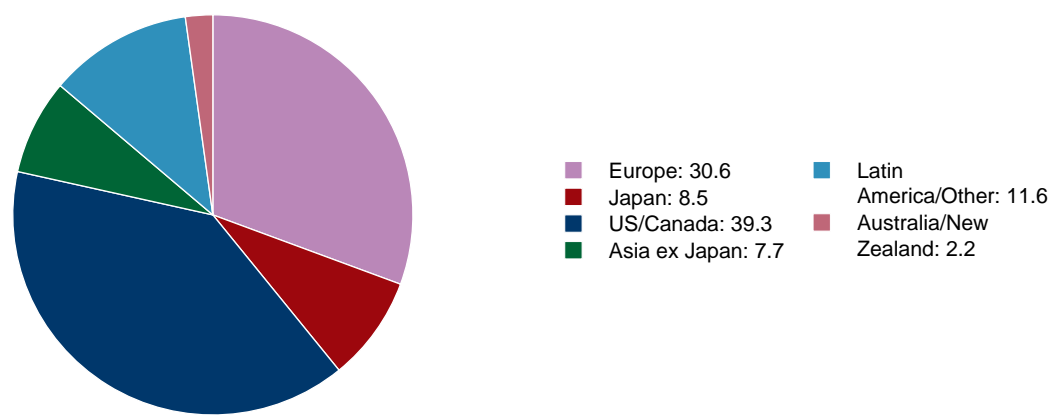


图 D-6 - 图例附带文本

附录 E 官方示例: Scatter

参考: <https://www.reportlab.com/chartgallery/scatter/>

E.1 Scatter plot with legend

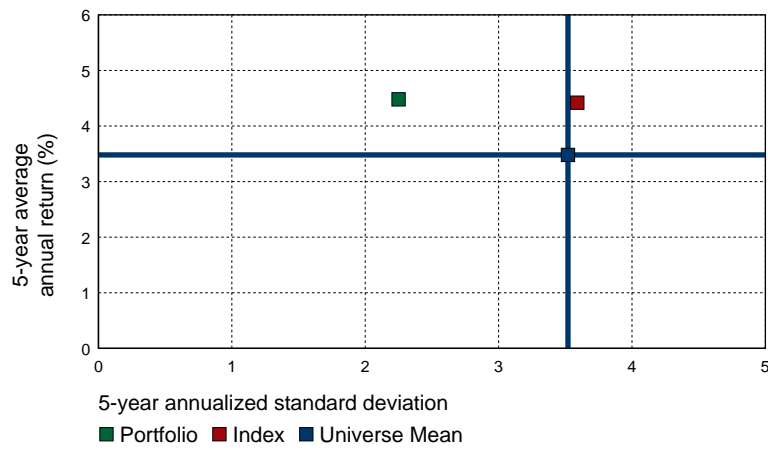


图 E-1 - 散点图

附录 F 官方示例: bar

参考: <https://www.reportlab.com/chartgallery/bar/>

F.1 Four category eight month

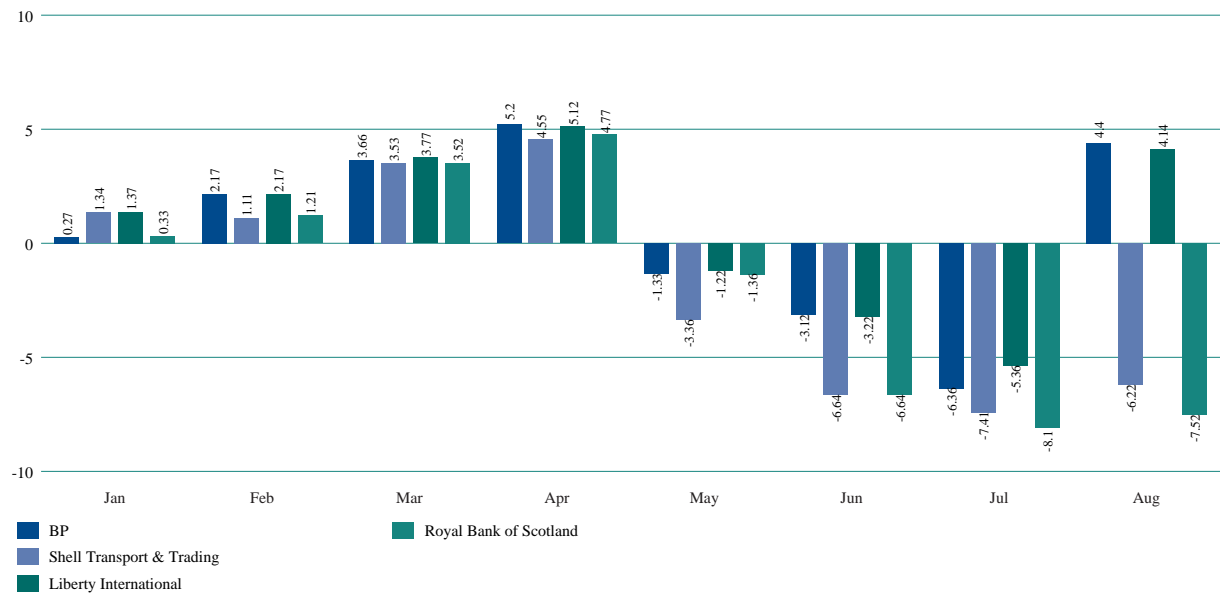


图 F-1 - 4个类别8个月对比图

F.2 Comparison chart

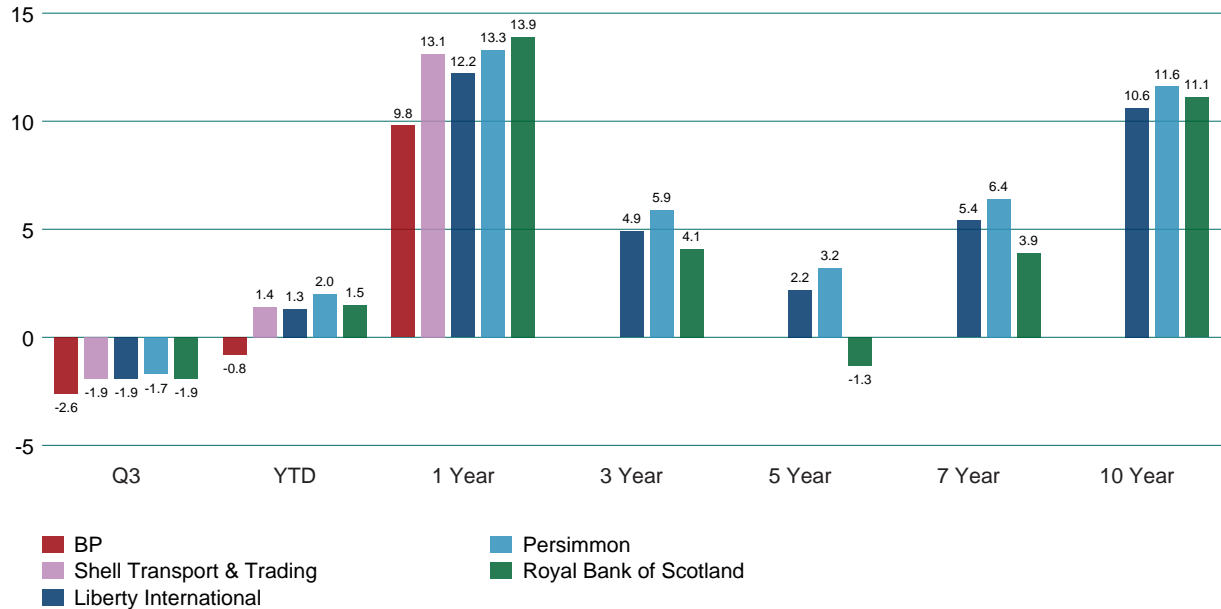


图 F-2 - 对比图

F.3 Multi-line x-axis labels

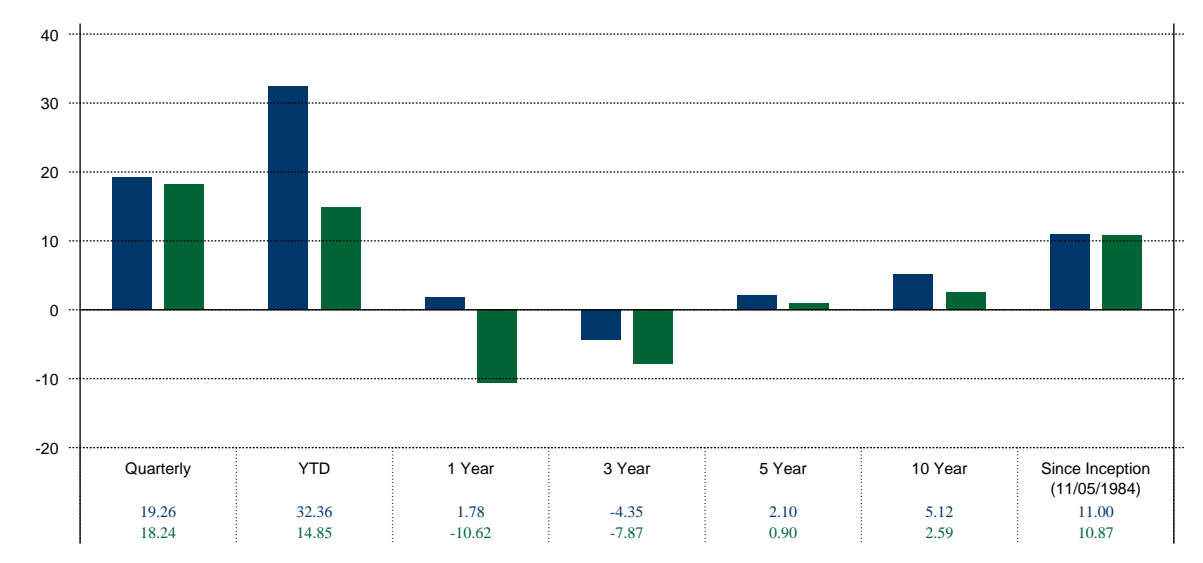


图 F-3 - 多行x轴标签

F.4 BarChart with table

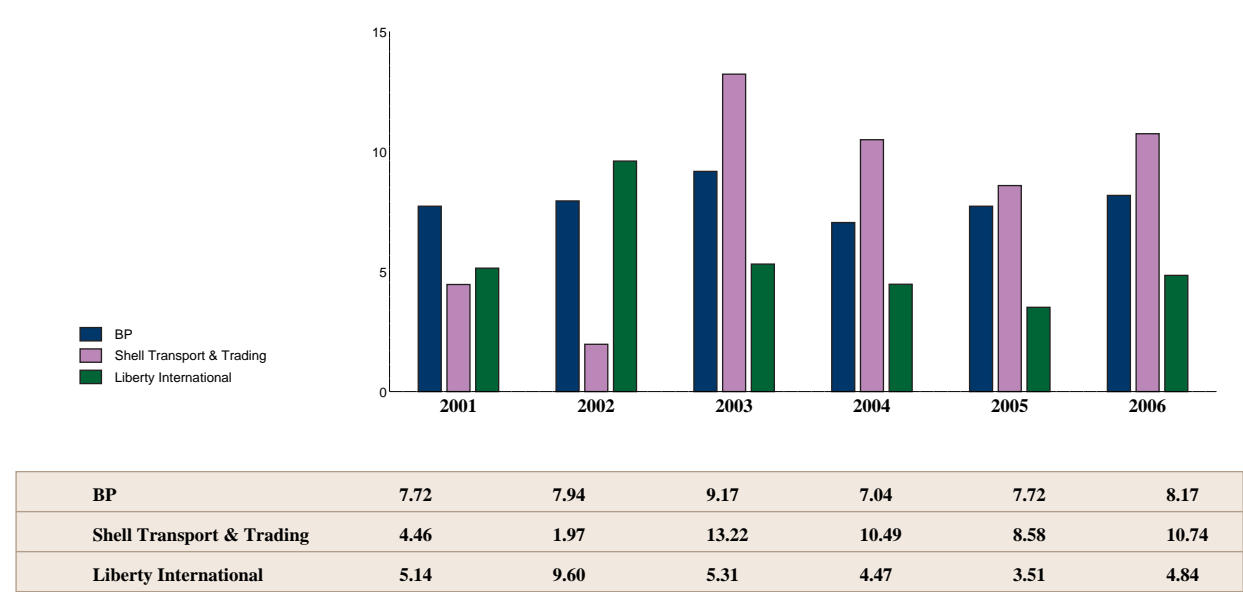


图 F-4 - 柱状图和表格

F.5 Vertical labels

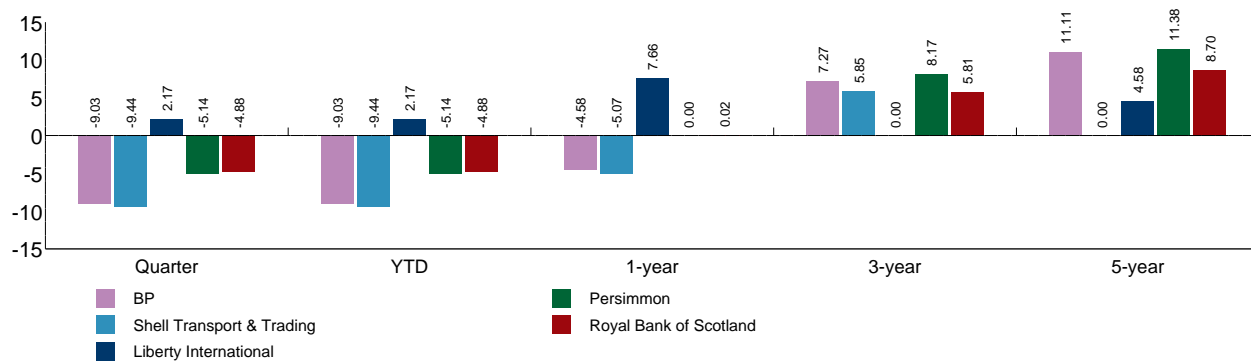
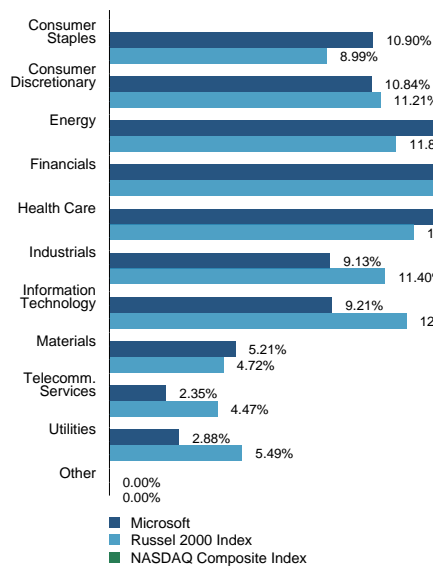


图 F-5 - 垂直标签

F.6 Dual Bar charts on one canvas

Equity Composition



Fixed Income Composition

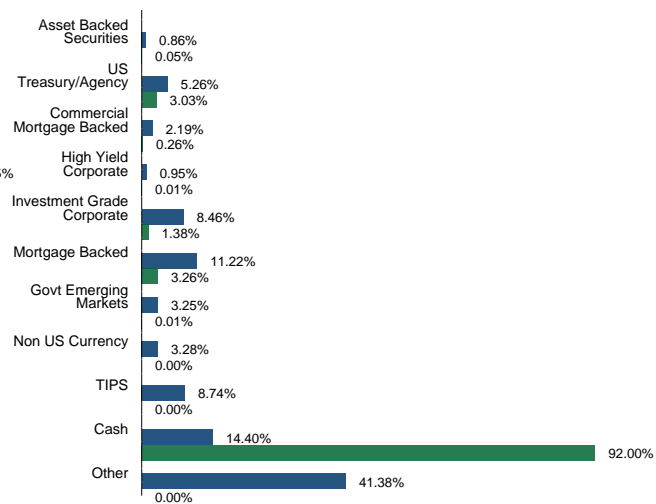


图 F-6 - 一块画布上的双条形图

F.7 Vertical bar chart with mixed stacked & parallel bars

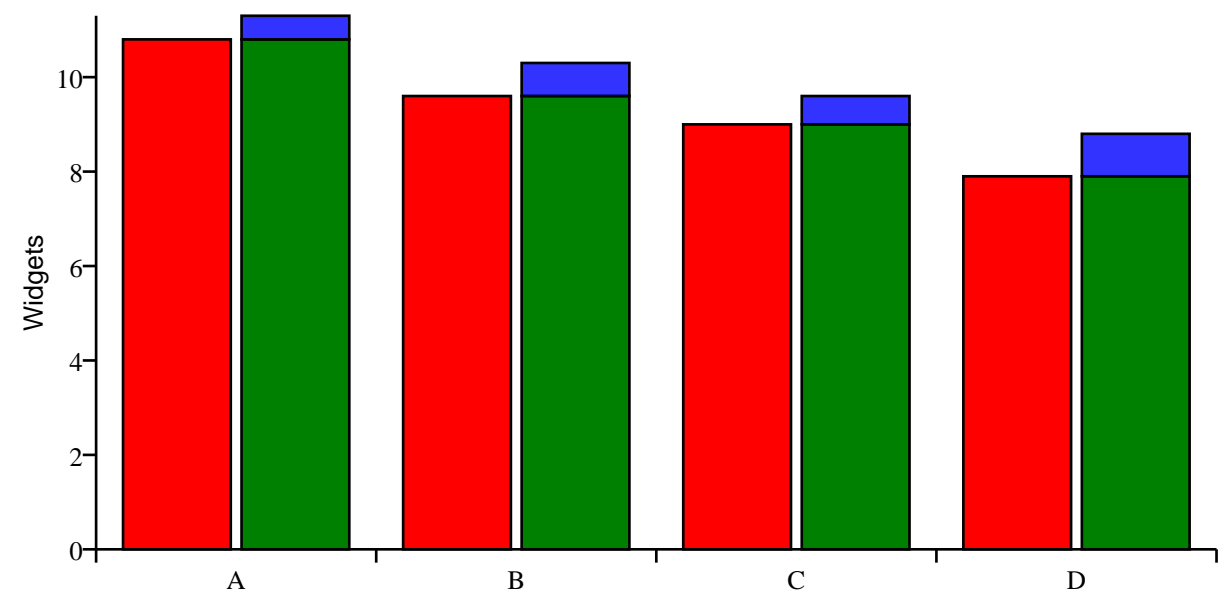


图 F-7 - 混合堆叠平行条形图

F.8 Vertical 3D bar chart with mixed stacked & parallel bars

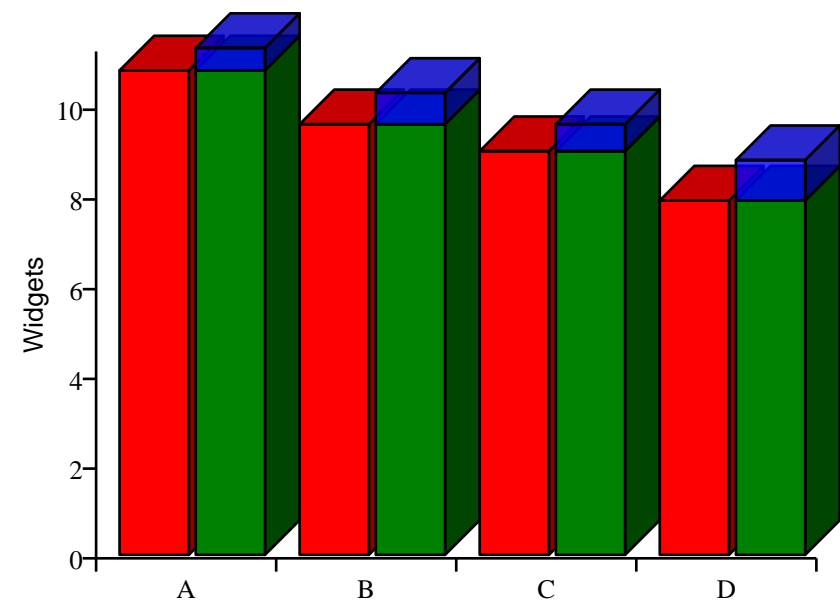


图 F-8 - 垂直3D柱状图

F.9 Horizontal bar with red axis negative labels

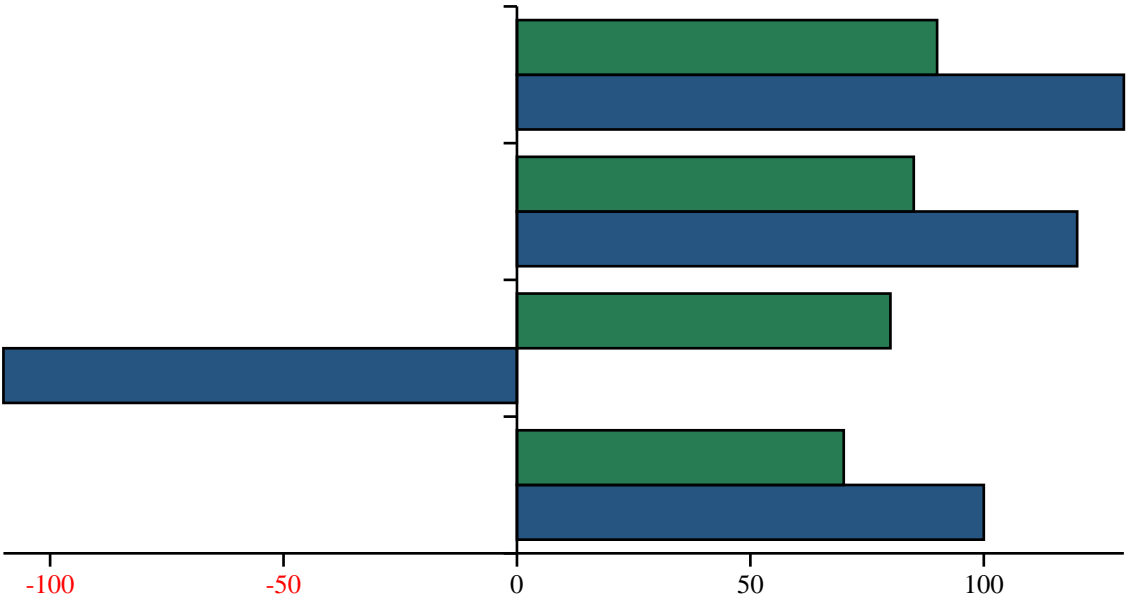


图 F-9 - 带红色轴负标签的单标签

F.10 Vertical bar with line labels

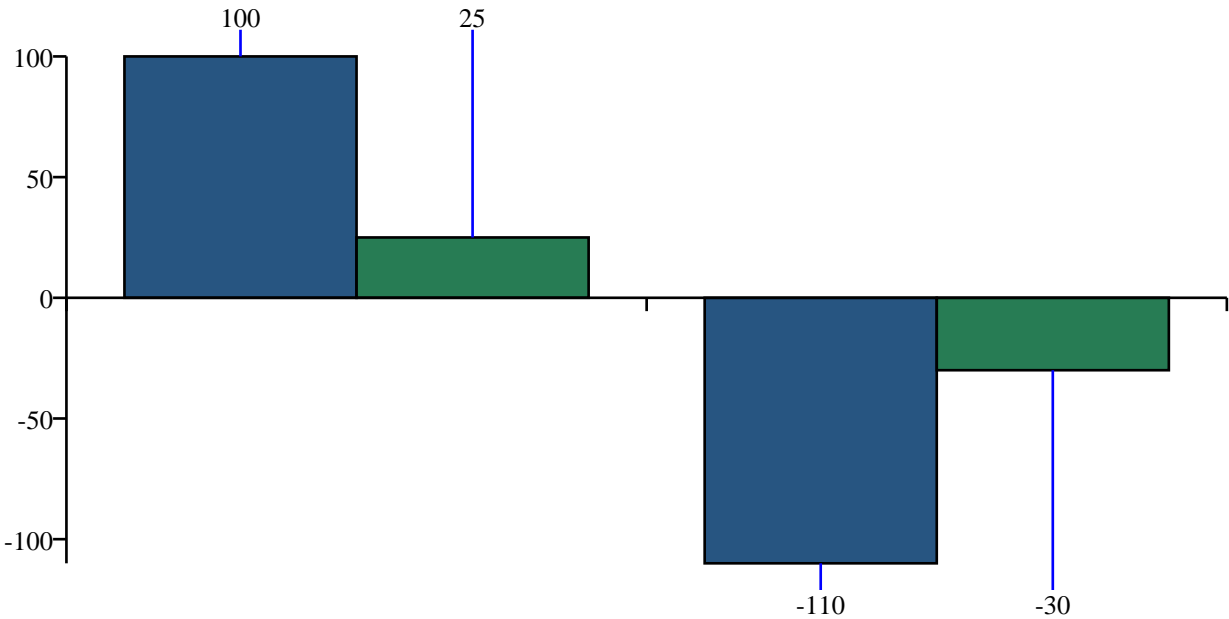


图 F-10 - 带线标签的竖线柱状图

F.11 A Vertical Bar Chart With Line Indicated Bar Labels

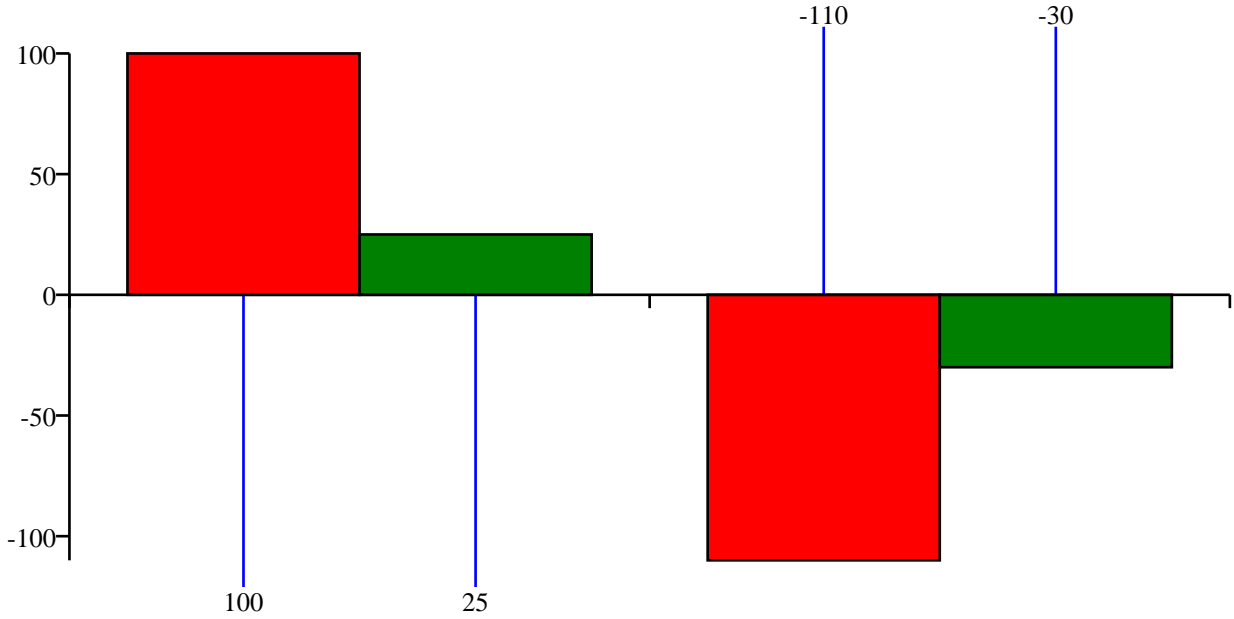


图 F-11 - 带线标签的竖线柱状图2

附录 G 官方示例: quickcharts

参考: <https://www.reportlab.com/chartgallery/quickcharts/>

ReportLab Quick Charts

Unlike the other chart types shown in this gallery, Quick Charts do not have any predetermined structure.

Rather, they can accept similar data structures and depending on the particular Quick Chart type, create a chart.

You will lose some of the precision and control of some of the specialist chart classes seen elsewhere in the gallery, but Quick Charts will let you experiment more quickly with a range of different ways to visualise your data.

ReportLab 快速图形

与此库中显示的其他图表类型不同，快速图表没有任何预定的结构。

相反，他们可以接受类似的数据结构，并根据特定的“快速图表”类型创建图表。

您将失去在图库中其他地方看到的某些专业图表类别的精度和控制力，但是“快速图表”将使您可以使用各种不同的方式更快地进行实验以可视化数据。

G.1 面积图

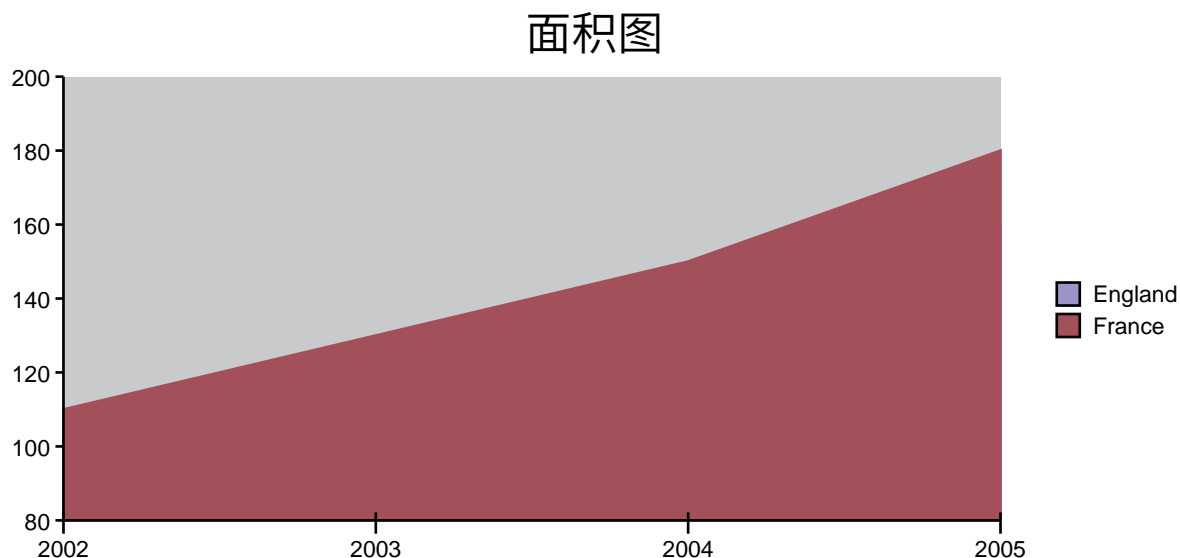


图 G-1 - 面积图)

G.2 3D条形图

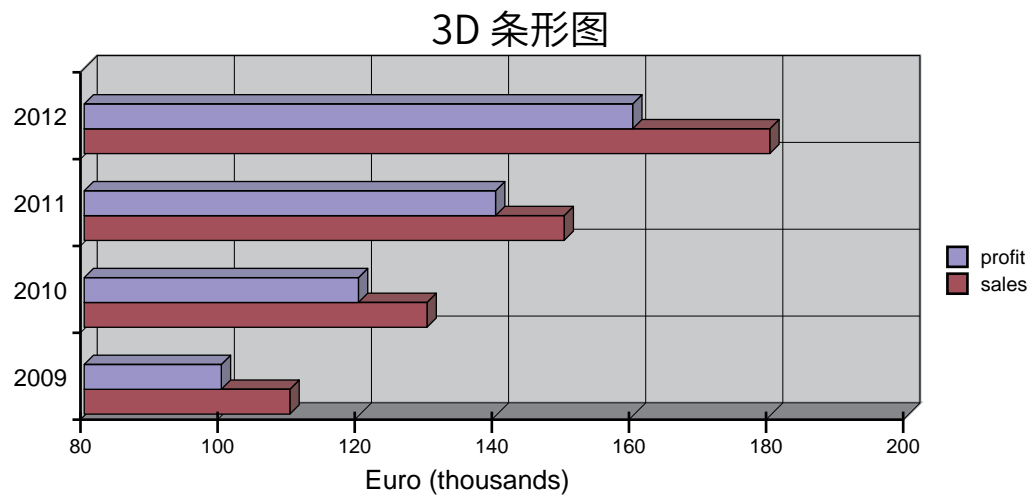


图 G-2 - 3D条形图)

G.3 背景柱状图

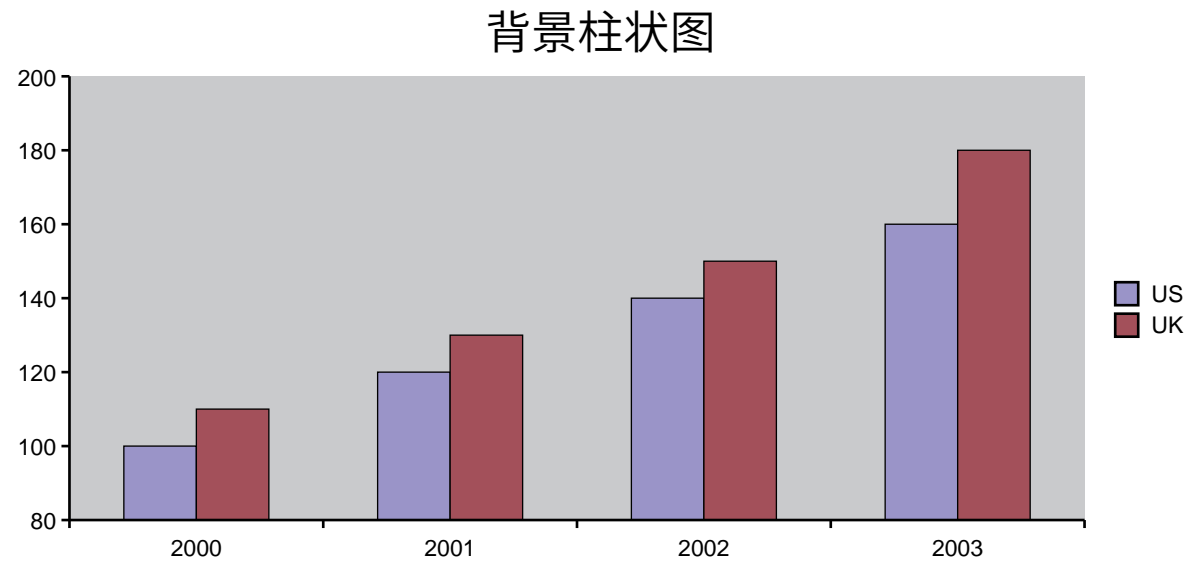


图 G-3 - 背景柱状图)

G.4 柱状图

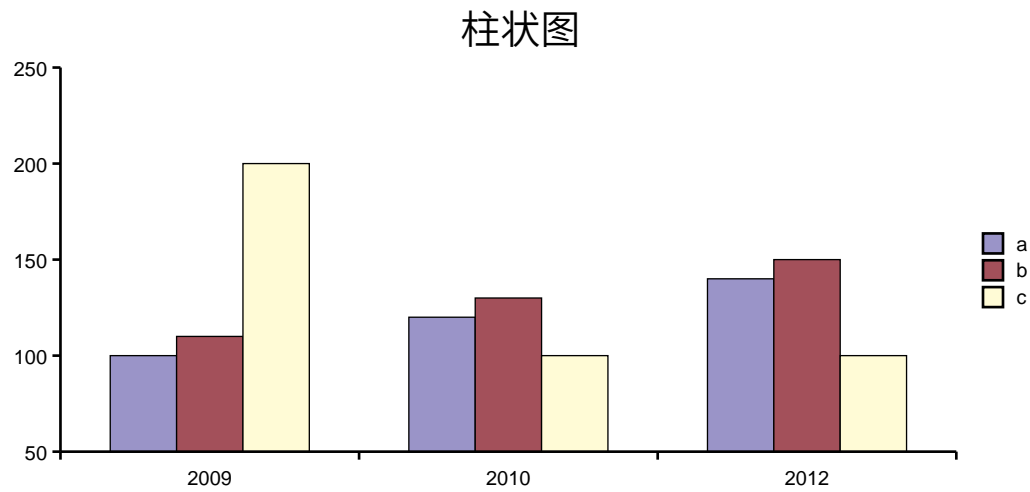


图 G-4 - 柱状图)

G.5 3D背景柱状图

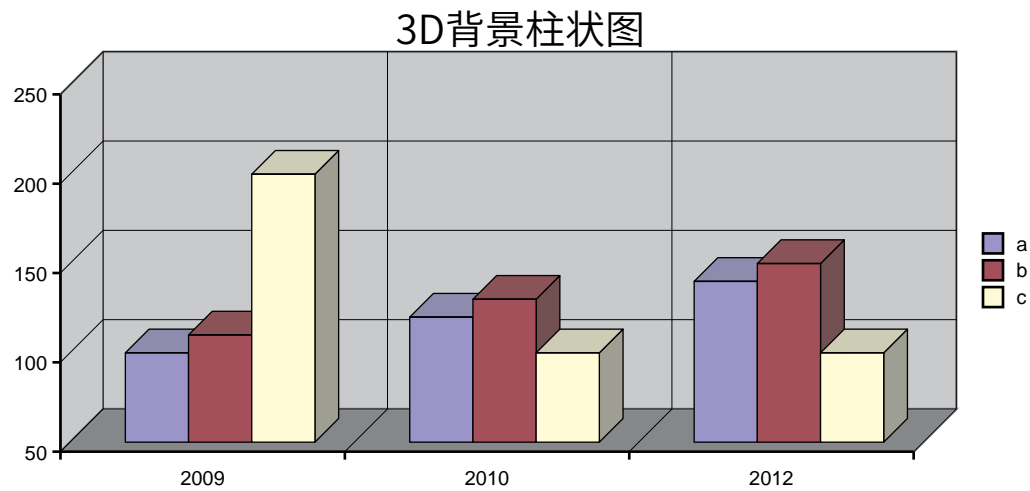


图 G-5 - 3D背景柱状图)

G.6 嵌套饼图

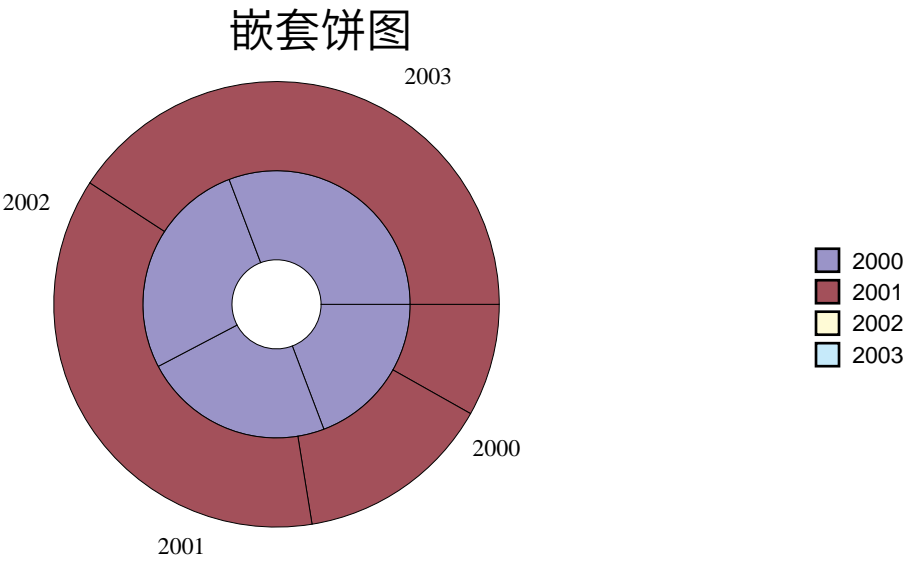


图 G-6 - 嵌套饼图)

G.7 分裂饼图

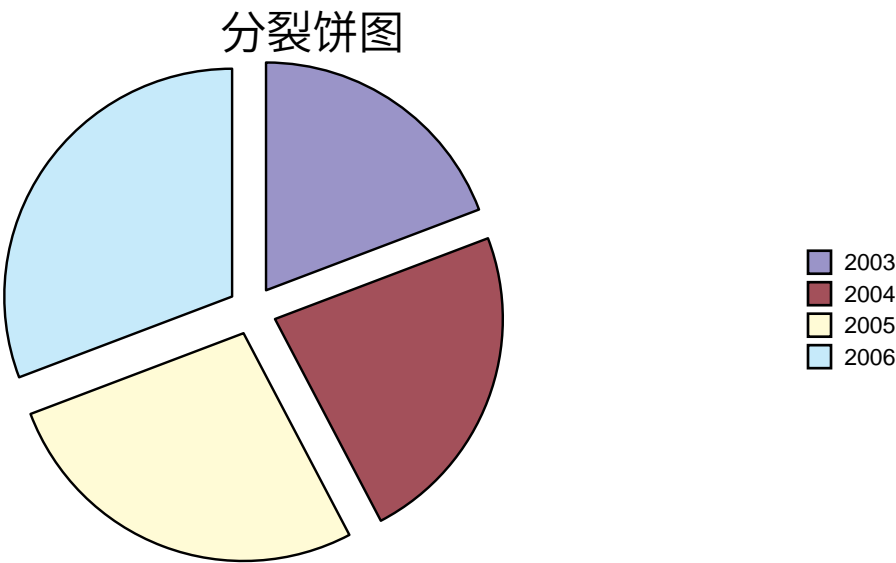


图 G-7 - 分裂饼图)

G.8 3D分裂饼图

3D 分裂饼图

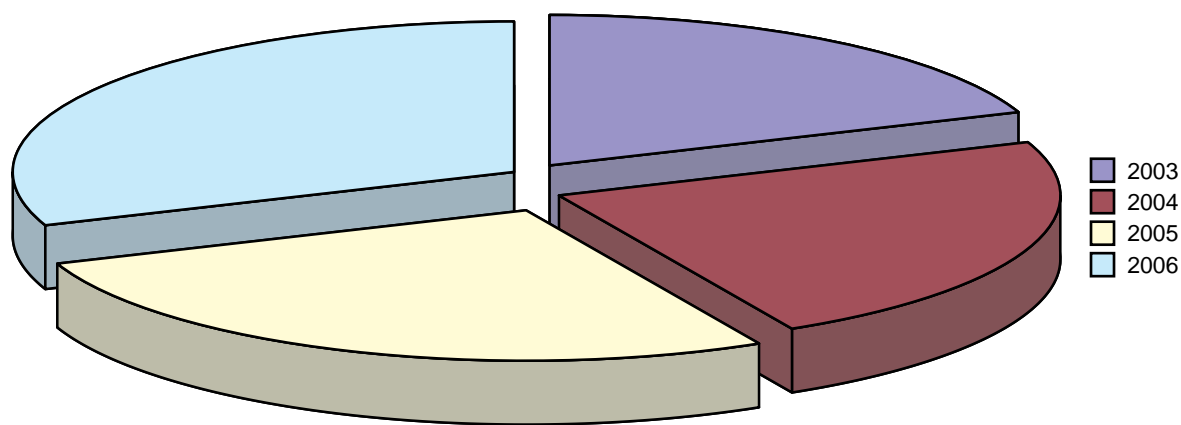


图 G-8 - 3D分裂饼图)

G.9 填充雷达图

填充雷达图

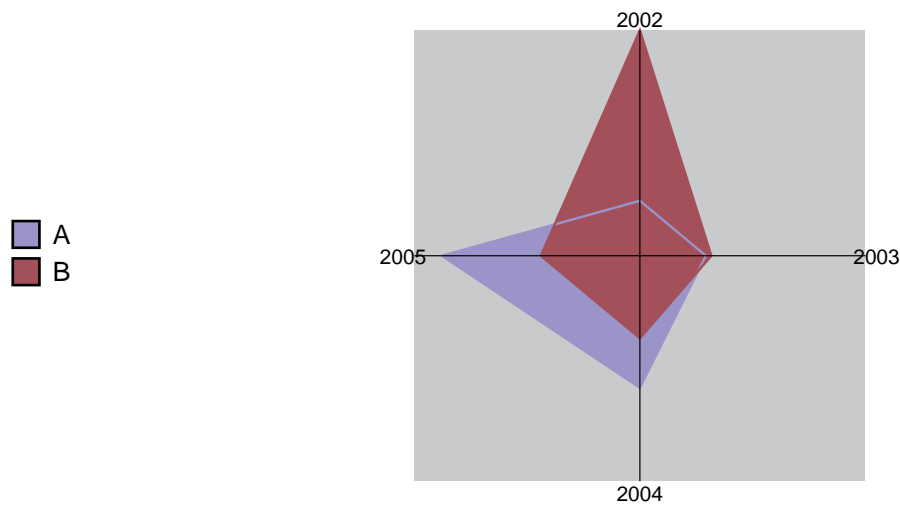


图 G-9 - 填充雷达图)

G.10 3D折线图

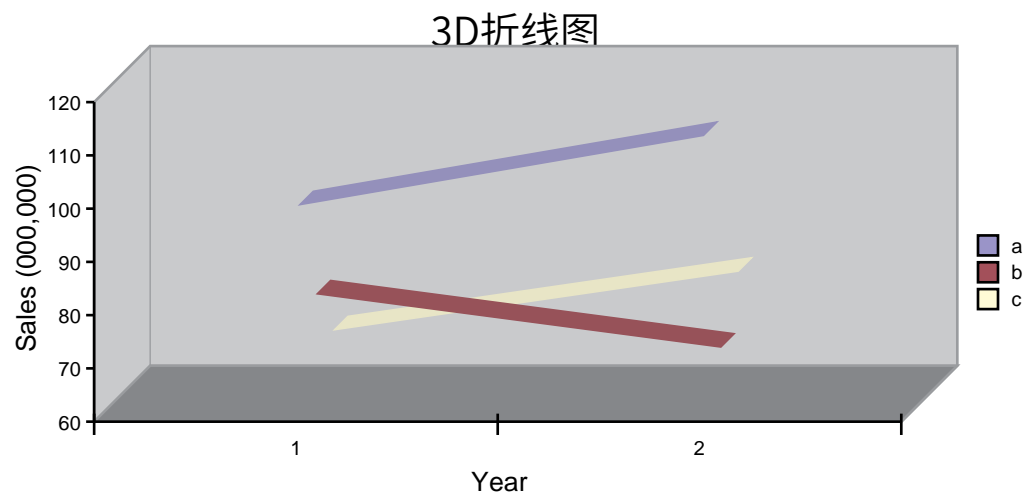


图 G-10 - 3D折线图)

G.11 折线图

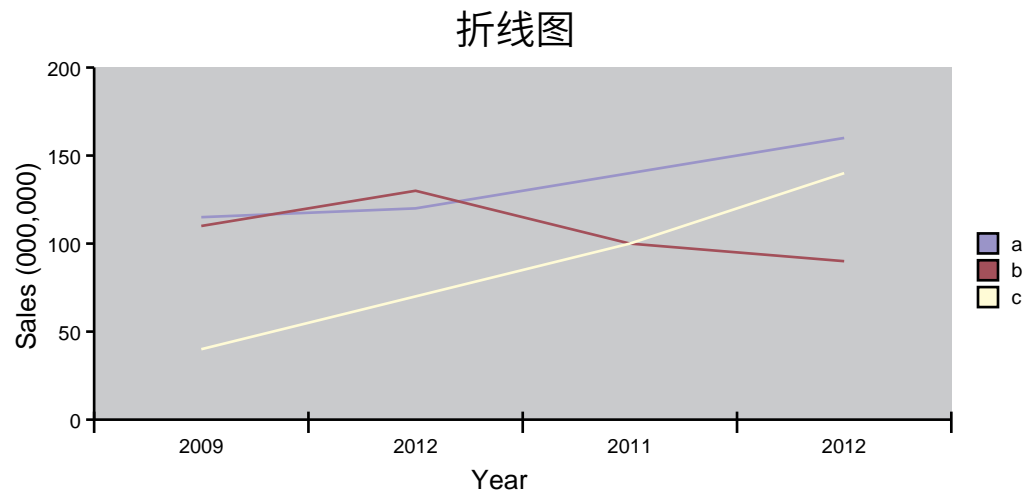


图 G-11 - 折线图)

G.12 标记折线图

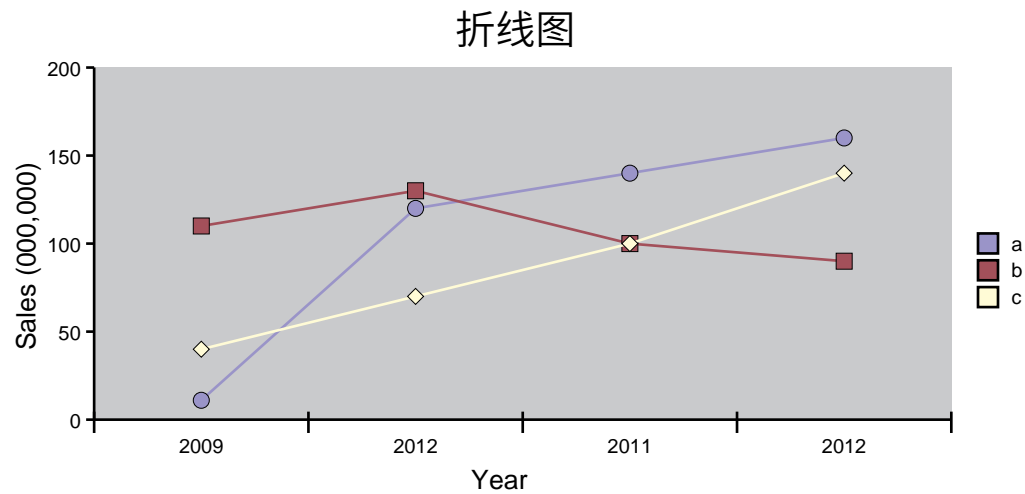


图 G-12 - 标记折线图)

G.13 直线图

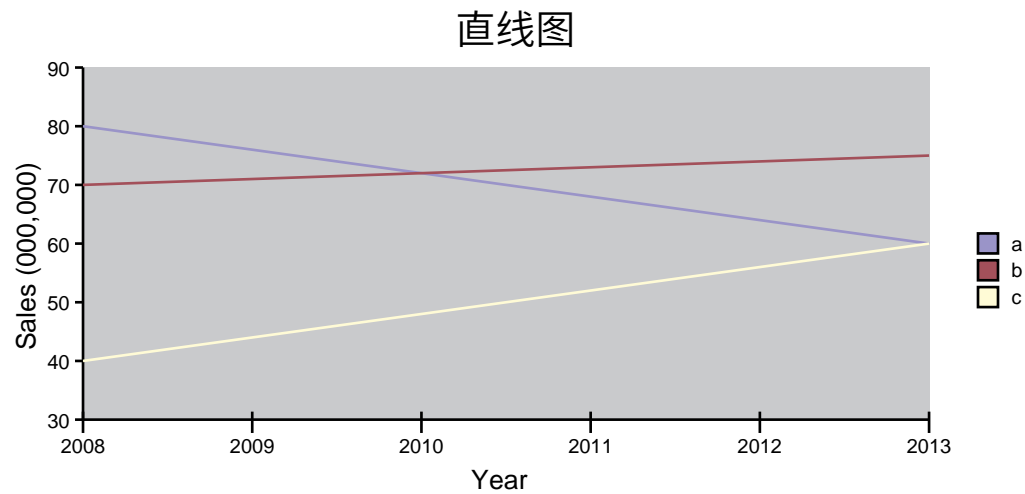


图 G-13 - 直线图)

G.14 3D直线图

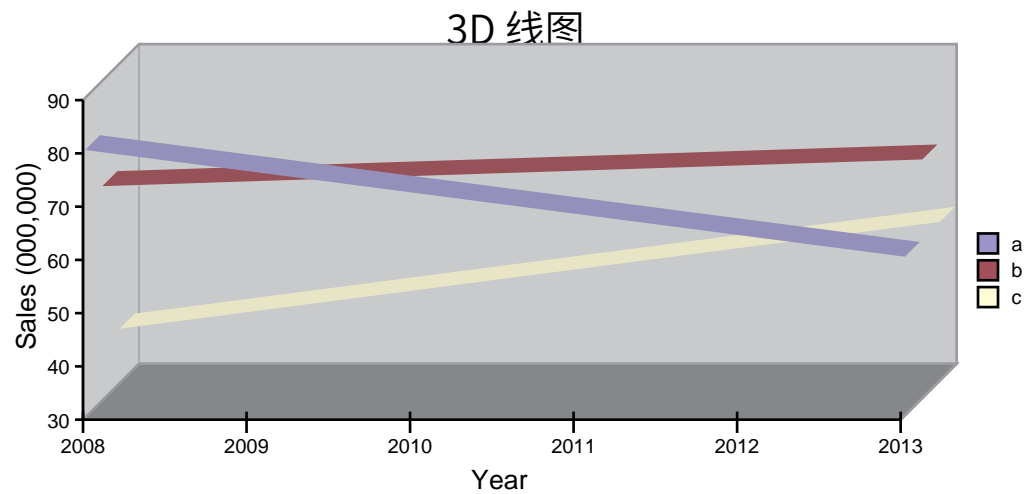


图 G-14 - 3D直线图)

G.15 直线标记图

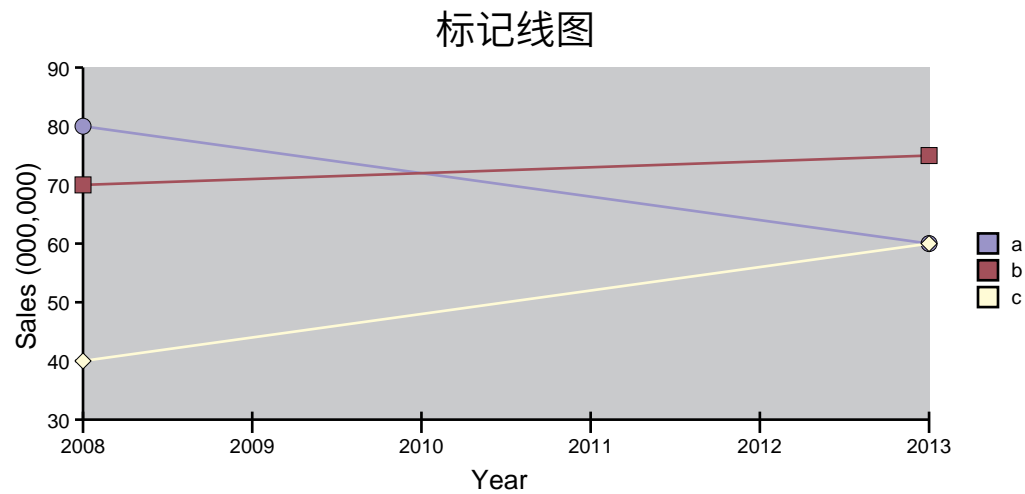


图 G-15 - 直线标记图)

G.16 矩形面积图

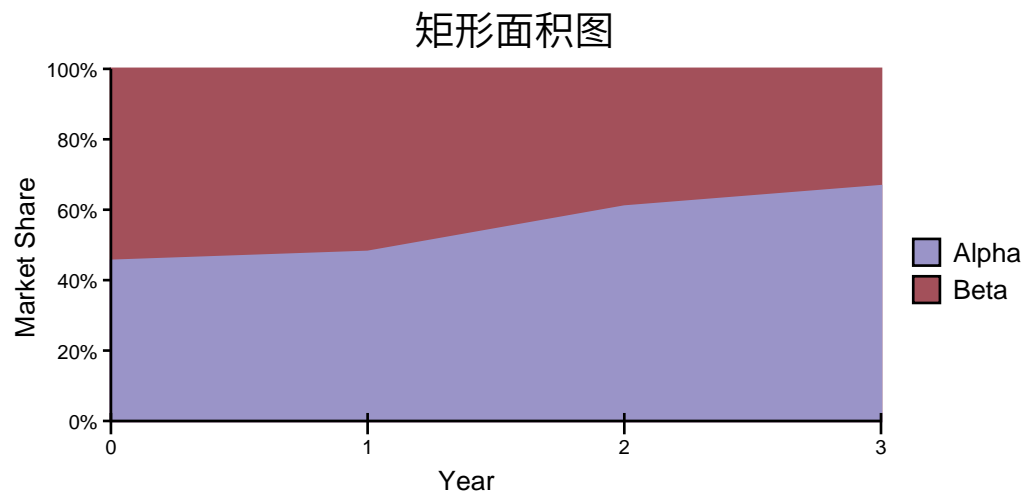


图 G-16 - 矩形面积图)

G.17 矩形条形图

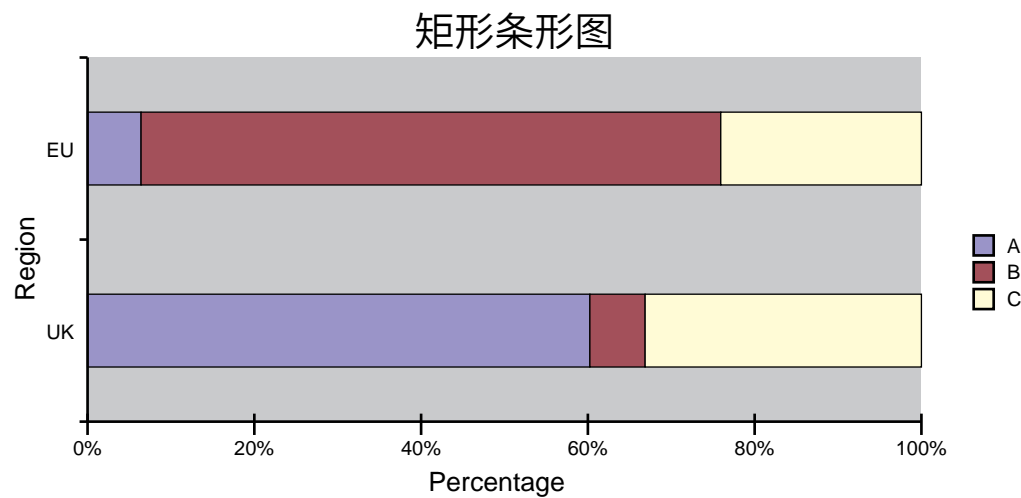


图 G-17 - 矩形条形图)

G.18 矩形柱状图

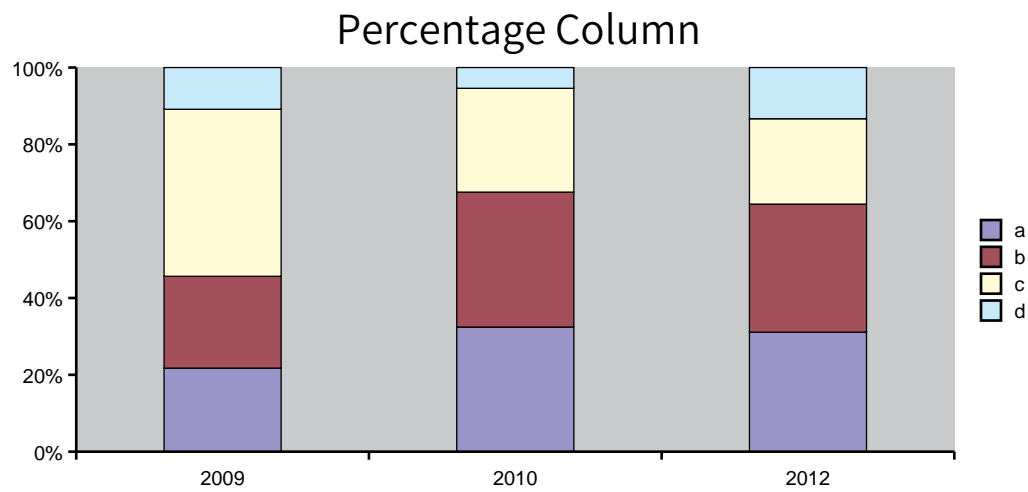


图 G-18 - 矩形柱状图)

G.19 3D饼图

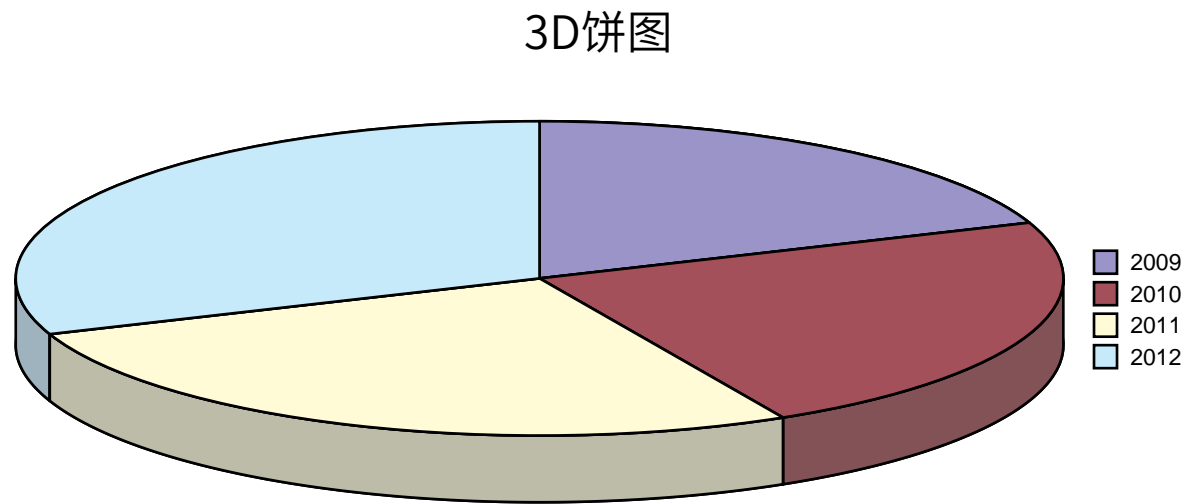


图 G-19 - 3D饼图)

G.20 饼图

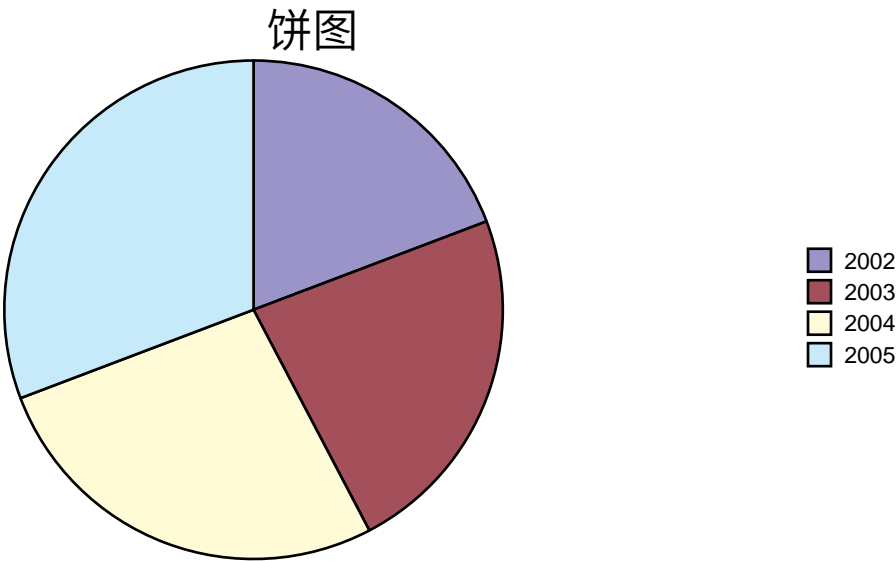


图 G-20 - 饼图)

G.21 雷达图

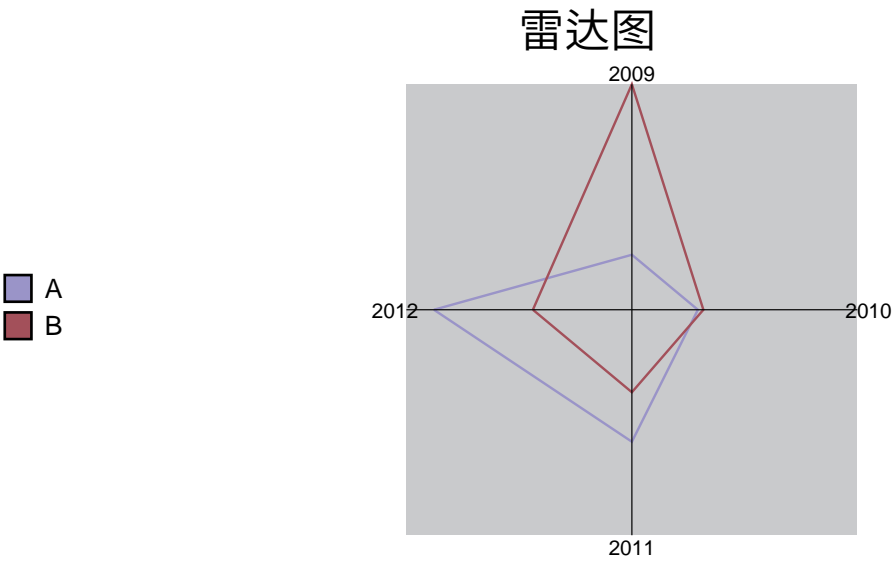


图 G-21 - 雷达图)

G.22 标记雷达图

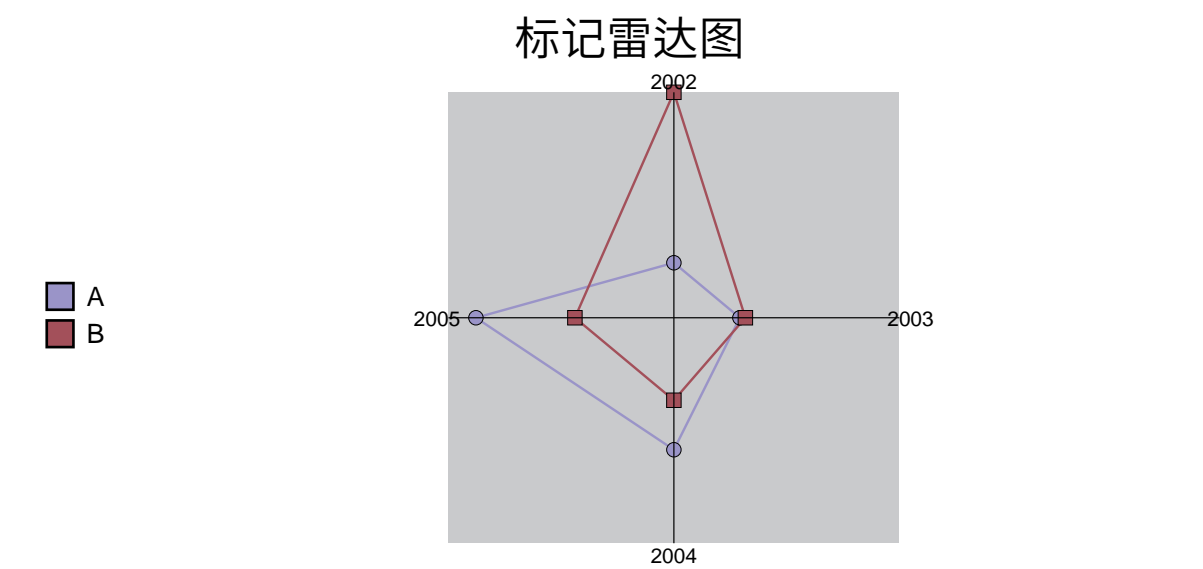


图 G-22 - 标记雷达图)

G.23 堆叠面积图

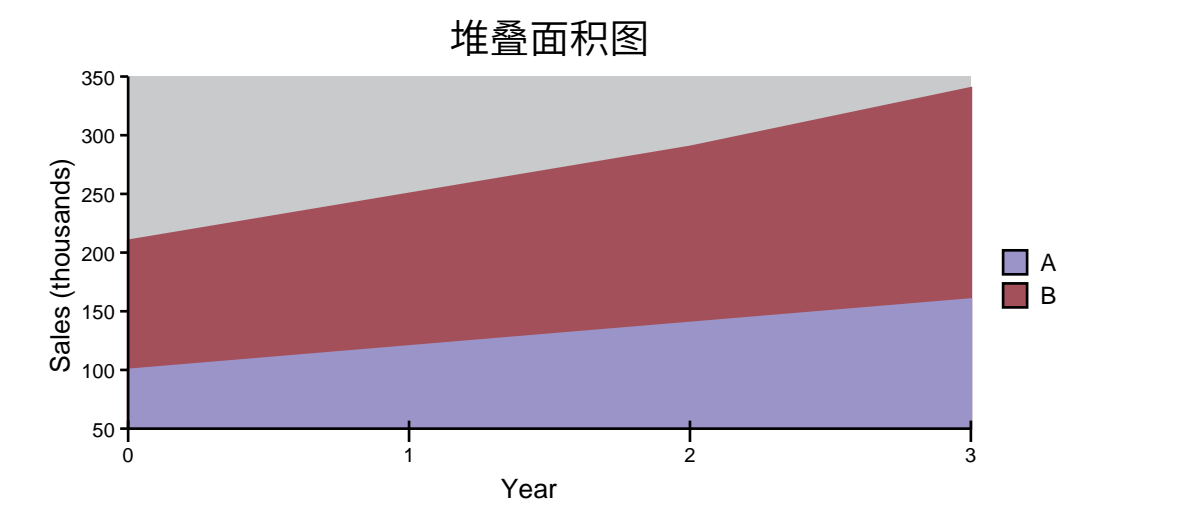


图 G-23 - 堆叠面积图)

G.24 堆叠条形图

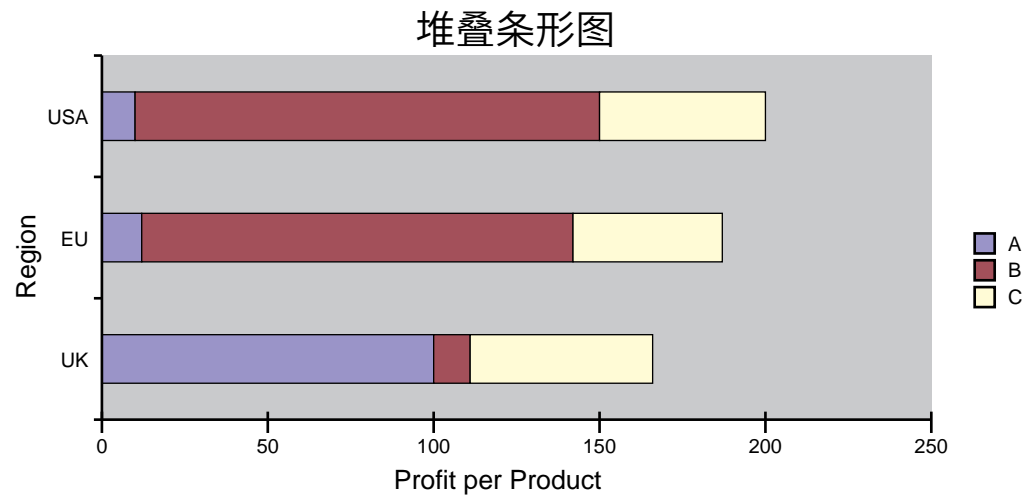


图 G-24 - 堆叠条形图)

G.25 堆叠柱状图

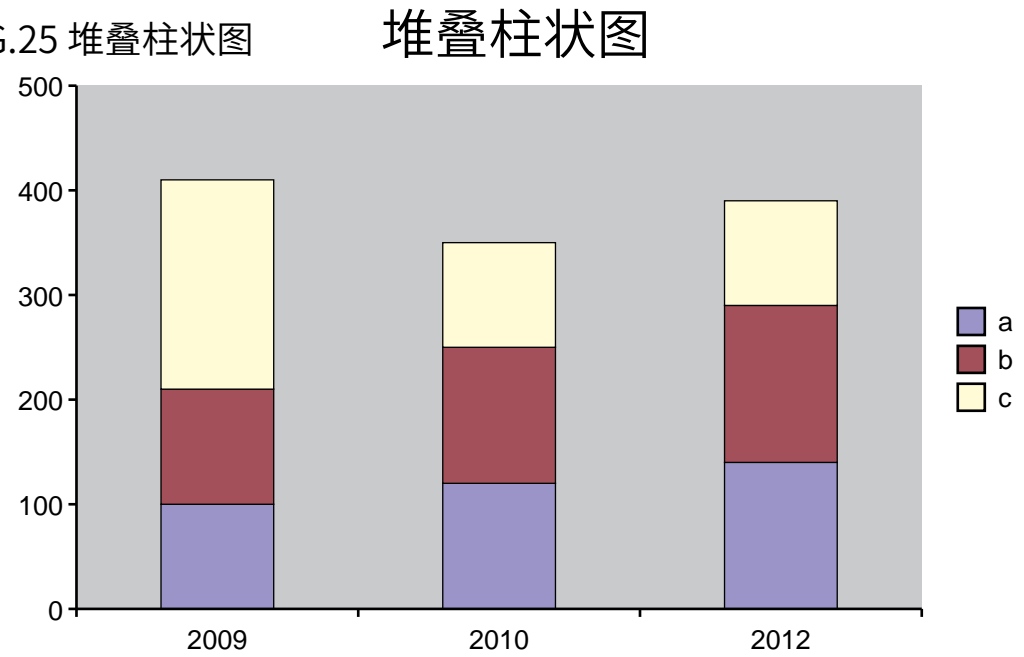


图 G-25 - 堆叠柱状图)

附录 H 官方示例: Area

参考: <https://www.reportlab.com/chartgallery/area/>

H.1 面积折线图

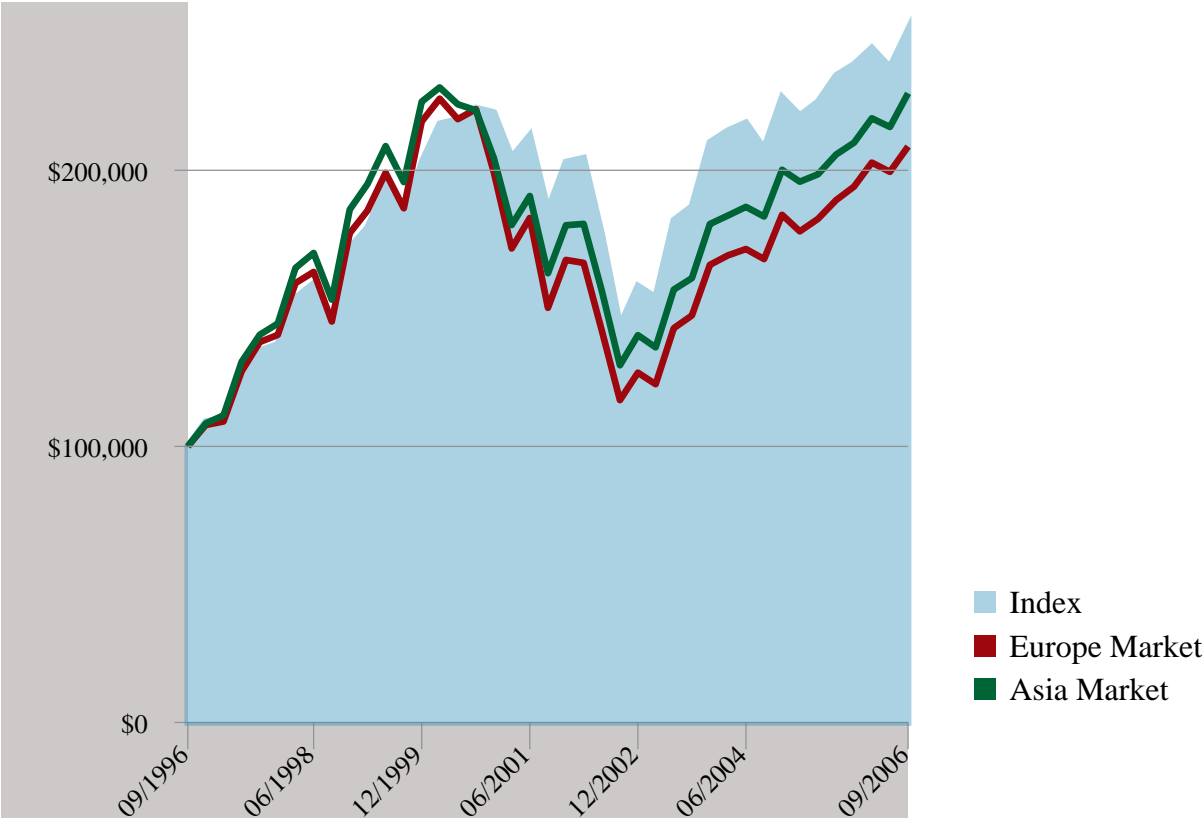


图 H-1 - 面积折线图

H.2 面积动态标签图

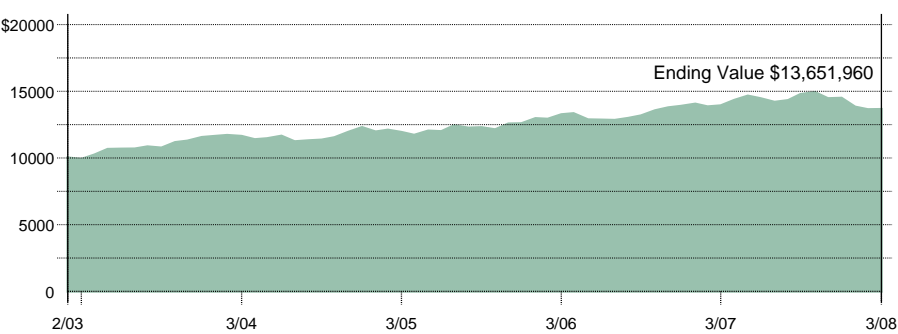


图 H-2 - 面积动态标签图