ReportLab PDF Library

用户手册

ReportLab Version 3.5.56

版本 3.5.56

Document generated on 2022/12/26 16:48:51

文档生成自 2022/12/26 16:48:51

目录

第 4 章 PDF的特殊功能　　　　　　　　　　　　　　　　　　67

第 5 章 PLATYPUS - 使用脚本进行页面布局和排版。　　　　80

第 6 章 文字段落　　　　　　　　　　　　　　　　　　　　90

第 7 章 表格和表格样式　　　　　　　　　　　　　　　　　105

第 8 章 编写 Flowables　　　　　　　　　　　　　　　　　114

第 1 章 介绍

1.1 关于

This document is an introduction to the ReportLab PDF library. Some previous programming experience is presumed and familiarity with the Python Programming language is recommended. If you are new to Python, we tell you in the next section where to go for orientation.

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。

This manual does not cover 100% of the features, but should explain all the main concepts and help you get started, and point you at other learning resources. After working your way through this, you should be ready to begin writing programs to produce sophisticated reports.

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。

In this chapter, we will cover the groundwork:

在本章, 我们将介绍基础:

- What is ReportLab all about, and why should I use it?
- 什么是ReportLab, 为什么要使用它?
- What is Python?
- 什么是Python?
- How do I get everything set up and running?
- 我该做哪些准备以便开始?

We need your help to make sure this manual is complete and helpful. Please send any feedback to our user mailing list, which is signposted from www.reportlab .com.
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?

This is a software library that lets you directly create documents in Adobe's Portable Document Format (PDF) using the Python programming language. It also creates charts and data graphics in various bitmap and vector formats as well as PDF.

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 is the global standard for electronic documents. It supports high-quality printing yet is totally portable across platforms, thanks to the freely available Acrobat Reader. Any application which previously generated hard copy reports or driving a printer can benefit from making PDF documents instead; these can be archived, emailed, placed on the web, or printed out the old-fashioned way. However, the PDF file format is a complex indexed binary format which is impossible to type directly. The PDF format specification is more than 600 pages long and PDF files must provide precise byte offsets -- a single extra character placed anywhere in a valid PDF document can render it invalid. This makes it harder to generate than HTML.

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。

Most of the world's PDF documents have been produced by Adobe's Acrobat tools, or rivals such as JAWS PDF Creator, which act as 'print drivers'. Anyone wanting to automate PDF production would typically use a product like Quark, Word or Framemaker running in a loop with macros or plugins, connected to Acrobat.

Pipelines of several languages and products can be slow and somewhat unwieldy.

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。

The ReportLab library directly creates PDF based on your graphics commands. There are no intervening steps. Your applications can generate reports extremely fast - sometimes orders of magnitude faster than traditional report-writing tools. This approach is shared by several other libraries - PDFlib for C, iText for Java, iTextSharp for .NET and others. However, The ReportLab library differs in that it can work at much higher levels, with a full featured engine for laying out documents complete with tables and charts.

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。

In addition, because you are writing a program in a powerful general purpose language, there are no restrictions at all on where you get your data from, how you transform it, and the kind of output you can create. And you can reuse code across whole families of reports.

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。

The ReportLab library is expected to be useful in at least the following contexts:

预期ReportLab库至少在以下情况下有用：
- Dynamic PDF generation on the web
- 网络上动态生成PDF.
- High-volume corporate reporting and database publishing
- 大批量公司报告和数据库发布.
- An embeddable print engine for other applications, including a 'report language' so that users can customize their own reports. *This is particularly relevant to cross-platform apps which cannot rely on a consistent printing or previewing API on each operating system.*
- 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。
- A 'build system' for complex documents with charts, tables and text such as management accounts, statistical reports and scientific papers
- 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
- Going from XML to PDF in one step
- 从XML到PDF的一步

1.3 ReportLab's commercial software

1.4 ReportLab的商业软件

The ReportLab library forms the foundation of our commercial solution for PDF generation, Report Markup Language (RML). This is available for evaluation on our web site with full documentation. We believe that RML is the fastest and easiest way to develop rich PDF workflows. You work in a markup language at a similar level to HTML, using your favorite templating system to populate an RML document; then call our rml2pdf API function to generate a PDF. It's what ReportLab staff use to build all of the solutions you can see on reportlab.com. Key differences:

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：

- Fully documented with two manuals, a formal specification (the DTD) and extensive self-documenting tests. (By contrast, we try to make sure the open source documentation isn't wrong, but we don't always keep up with the code)
- 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，但是我们并不总是跟上代码）
- Work in high-level markup rather than constructing graphs of Python objects
- 在高级标记中工作，而不是构造Python对象图
- Requires no Python expertise - your colleagues may thank you after you've left!'
- 不需要Python专业知识-您离开后，您的同事可能会感谢您！"
- Support for vector graphics and inclusion of other PDF documents
- 支持矢量图形并包含其他PDF文档
- Many more useful features expressed with a single tag, which would need a lot of coding in the open source package
- 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
- Commercial support is included
- 包括商业支持

We ask open source developers to consider trying out RML where it is appropriate. You can register on our site and try out a copy before buying. The costs are reasonable and linked to the volume of the project, and the revenue helps us spend more time developing this software.

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。

1.5 什么是 Python?

Python is an *interpreted, interactive, object-oriented* programming language. It is often compared to Tcl, Perl, Scheme or Java.

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 combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC). New built-in modules are easily written in C or C++. Python is also usable as an extension language for applications that need a programmable interface.

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 is as old as Java and has been growing steadily in popularity for years; since our library first came out it has entered the mainstream. Many ReportLab library users are already Python devotees, but if you are not, we feel that the language is an excellent choice for document-generation apps because of its expressiveness and ability to get data from anywhere.

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 is copyrighted but **freely usable and distributable, even for commercial use**.

Python受版权保护，但可自由使用和分发，甚至用于商业用途。

1.6 致谢

Many people have contributed to ReportLab. We would like to thank in particular (in alphabetical order):

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.

Special thanks go to Just van Rossum for his valuable assistance with font technicalities.

特别感谢Just van Rossum在字体技术方面的宝贵帮助。

Moshe Wagner and Hosam Aly deserve a huge thanks for contributing to the RTL patch, which is not yet on the trunk.

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 deserves a big hand for contributing the work on TrueType fonts and we are glad to include these in the toolkit. Finally we thank Michal Kosmulski for the DarkGarden font for and Bitstream Inc. for the Vera fonts.

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。

1.7 安装与设定

To avoid duplication, the installation instructions are kept in the README file in our distribution, which can be viewed online at *https://hg.reportlab.com/hg-public/reportlab/*

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/

This release (3.5.56) of ReportLab requires Python versions 2.7, 3.6+ or higher. If you need to use Python 2.5 or 2.6, please use the latest ReportLab 2.7 package.

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。

1.8 开始

ReportLab is an Open Source project. Although we are a commercial company we provide the core PDF generation sources freely, even for commercial purposes, and we make no income directly from these modules. We also welcome help from the community as much as any other Open Source project. There are many ways in which you can help:

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：

- General feedback on the core API. Does it work for you? Are there any rough edges? Does anything feel clunky and awkward?
  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？
- New objects to put in reports, or useful utilities for the library. We have an open standard for report objects, so if you have written a nice chart or table class, why not contribute it?

- 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?
- Snippets and Case Studies: If you have produced some nice output, register online on *http://www.reportlab.com* and submit a snippet of your output (with or without scripts). If ReportLab solved a problem for you at work, write a little 'case study' and submit it. And if your web site uses our tools to make reports, let us link to it. We will be happy to display your work (and credit it with your name and company) on our site!
- 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）!
- Working on the core code: we have a long list of things to refine or to implement. If you are missing some features or just want to help out, let us know!
- 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!

The first step for anyone wanting to learn more or get involved is to join the mailing list. To Subscribe visit `http://two.pairlist.net/mailman/listinfo/reportlab-users`. From there you can also browse through the group's archives and contributions. The mailing list is the place to report bugs and get support.

- 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。

The code now lives on our website (`http://hg.reportlab.com/hg-public/reportlab/`) in a Mercurial repository, along with an issue tracker and wiki. Everyone should feel free to contribute, but if you are working actively on some improvements or want to draw attention to an issue, please use the mailing list to let us know.

- 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。

1.9 全局配置

There are a number of options which most likely need to be configured globally for a site. The python script module `reportlab/rl_config.py` aggregates the various settings files. You may want inspect the file `reportlab/rl_settings.py` which contains defaults for the currently used variables. There are several overrides for `rl_settings` modules `reportlab.local_rl_settings`, `reportlab_settings` (a script file anywhere on the python path) and finally the file `~/.reportlab_settings` (note no .py). Temporary changes can be made using evironment variables which are the variables from `rl_settings.py` prefixed with RL_ eg RL_verbose=1.

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: set to integer values to control diagnostic output.
- verbose: 设置为整数值以控制诊断输出。
- shapeChecking: set this to zero to turn off a lot of error checking in the graphics modules
- shapeChecking: 将此设置为零可关闭图形模块中的许多错误检查
- defaultEncoding: set this to WinAnsiEncoding or MacRomanEncoding.

- defaultEncoding: 将此设置为WinAnsiEncoding或MacRomanEncoding。
- defaultPageSize: set this to one of the values defined in reportlab/lib/pagesizes.py; as delivered it is set to pagesizes.A4; other values are pagesizes.letter etc.
- 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: set to zero to inhibit the creation of .a85 files on your hard-drive. The default is to create these preprocessed PDF compatible image files for faster loading
- 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: this is a python list of strings representing directories that may be queried for information on Type 1 fonts
- 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: this is a python list of strings representing directories that may be queried for information on TrueType fonts
- 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: this is a python list of strings representing directories that may be queried for information on font code maps.
- 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: set to non-zero to get boundary lines drawn.
- showBoundary: 设置为非零以绘制边界线。
- ZLIB_WARNINGS: set to non-zero to get warnings if the Python compression extension is not found.
- ZLIB_WARNINGS: 如果未找到Python压缩扩展，则设置为非零以获得警告。
- pageCompression: set to non-zero to try and get compressed PDF.
- pageCompression: 设置为非零以尝试获取压缩的PDF。
- allowtableBoundsErrors: set to 0 to force an error on very large Platypus table elements
- allowtableBoundsErrors: 设置为0会在非常大的Platypus表元素上强制执行错误
- emptyTableAction: Controls behaviour for empty tables, can be 'error' (default), 'indicate' or 'ignore'.
- emptyTableAction: 控制空表的行为，可以为“error”（默认），“indicate”或“ignore”。
- trustedHosts: if not `None` a list of glob patterns of trusted hosts; these may be used in places like <img> tags in paragraph texts.
- TrustedHosts: 如果不是`None`，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<img> 标签等地方。
- trustedSchemes: a list of allowed `URL` schemes used with `trustedHosts`
- trustedSchemes: 与`trustedHosts`一起使用的允许的`URL`方案的列表

For the full list of variables see the file `reportlab/rl_settings.py`.

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

其他修改

More complex modifications to the reportlab toolkit environment may be made using one of the modules `rep[ortlab.local_rl_mods` (.py script in reportlab folder), `reportlab_mods` (.py file on the python path) or `~/.reportlab_mods` (note no .py).

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：`rep[ortlab.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）。

1.10 了解有关Python的更多信息

If you are a total beginner to Python, you should check out one or more from the growing number of resources on Python programming. The following are freely available on the web:

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 Documentation.** A list of documentation on the Python.org web site. `http://www.python.org/doc/`
- Python文档 Python.org网站上的文档列表。`http://www.python.org/doc/`
- **Python Tutorial.** The official Python Tutorial , originally written by Guido van Rossum himself. `http://docs.python.org/tutorial/`
- Python教程 官方的Python教程，最初由Guido van Rossum亲自编写。`http://docs.python.org/tutorial/`
- **Learning to Program.** A tutorial on programming by Alan Gauld. Has a heavy emphasis on Python, but also uses other languages. `http://www.freenetpages.co.uk/hp/alan.gauld/`
- 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/`
- **Instant Python**. A 6-page minimal crash course by Magnus Lie Hetland. `http://www.hetland.org/python/instant-python.php`
- 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`
- **Dive Into Python**. A free Python tutorial for experienced programmers. `http://www.diveintopython.net/`
- 深入Python 适用于经验丰富的程序员的免费Python教程。`http://www.diveintopython.net/`

1.11 3.x版本系列的目标

ReportLab 3.0 has been produced to help in the migration to Python 3.x. Python 3.x will be standard in future Ubuntu releases and is gaining popularity, and a good proportion of major Python packages now run on Python 3.

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 compatibility. A single line of code should run on 2.7 and 3.6
- Python 3.x兼容性。 一行代码应在2.7和3.6上运行
- __init__.py restricts to 2.7 or >=3.6
- __init__.py限制为2.7或> = 3.6
- __init__.py allow the import of on optional reportlab.local_rl_mods to allow monkey patching etc.
- __init__.py允许导入可选的reportlab.local_rl_mods，以允许猴子打补丁等。
- rl_config now imports rl_settings, optionally local_rl_settings, reportlab_settings.py & finally ~/.reportlab_settings
- 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 extensions now live inside reportlab; _rl_accel is no longer required. All _rl_accel imports now pass through reportlab.lib.rl_accel
- 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 is gone, alongside the paraparser stuff that caused issues in favour of HTMLParser.
- xmllib以及用于引起HTMLParser问题的paraparser东西一去不复返了。
- some obsolete C extensions (sgmlop and pyHnj) are gone
- 一些过时的C扩展名（sgmlop和pyHnj）消失了
- Improved support for multi-threaded systems to the _rl_accel C extension module.
- _rl_accel C扩展模块对多线程系统的改进支持。

- Removed reportlab/lib/para.py & pycanvas.py. These would better belong in third party packages, which can make use of the monkeypatching feature above.
- 删除了reportlab/lib/para.py和pycanvas.py。
  这些最好属于第三方程序包，可以利用上面的Monkeypatching功能。
- Add ability to output greyscale and 1-bit PIL images without conversion to RGB. (contributed by Matthew Duggan)
- 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）
- highlight annotation (contributed by Ben Echols)
- 高亮注释（由Ben Echols提供）
- full compliance with pip, easy_install, wheels etc
- 完全符合pip，easy_install，wheel等要求

Detailed release notes are available at
`http://www.reportlab.com/software/documentation/relnotes/30/`

有关详细的发行说明，请访问：`http://www.reportlab.com/software/documentation/relnotes/30/`

第 2 章 使用 `pdfgen` 生成图形和文本

2.1 基本概念

The `pdfgen` package is the lowest level interface for generating PDF documents. A `pdfgen` program is essentially a sequence of instructions for "painting" a document onto a sequence of pages. The interface object which provides the painting operations is the `pdfgen canvas`.

`pdfgen`包是生成PDF文档的最低级别接口。 一个`pdfgen`程序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`。

The canvas should be thought of as a sheet of white paper with points on the sheet identified using Cartesian `(X,Y)` coordinates which by default have the `(0,0)` origin point at the lower left corner of the page. Furthermore the first coordinate `x` goes to the right and the second coordinate `y` goes up, by default.

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。

A simple example program that uses a canvas follows.

下面是一个使用画布的简单示例程序。

```python
from reportlab.pdfgen import canvas
def hello(c):
    c.drawString(100,100,"Hello World")
c = canvas.Canvas("hello.pdf")
hello(c)
c.showPage()
c.save()
```

The above code creates a `canvas` object which will generate a PDF file named `hello.pdf` in the current working directory. It then calls the `hello` function passing the `canvas` as an argument. Finally the `showPage` method saves the current page of the canvas and the `save` method stores the file and closes the canvas.

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。

The `showPage` method causes the `canvas` to stop drawing on the current page and any further operations will draw on a subsequent page (if there are any further operations -- if not no new page is created). The `save` method must be called after the construction of the document is complete -- it generates the PDF document, which is the whole purpose of the `canvas` object.

`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

Before describing the drawing operations, we will digress to cover some of the things which can be done to configure a canvas. There are many different settings available. If you are new to Python or can't wait to produce some output, you can skip ahead, but come back later and read this!

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!

First of all, we will look at the constructor arguments for the canvas:

首先，我们来看看画布的构造函数参数:

```python
def __init__(self,filename,
             pagesize=(595.27,841.89),
             bottomup = 1,
```

```
pageCompression=0,
encoding=rl_config.defaultEncoding,
verbosity=0
encrypt=None):
```

The `filename` argument controls the name of the final PDF file. You may also pass in any open binary stream (such as `sys.stdout`, the python process standard output with a binary encoding) and the PDF document will be written to that. Since PDF is a binary format, you should take care when writing other stuff before or after it; you can't deliver PDF documents inline in the middle of an HTML page!

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。

The `pagesize` argument is a tuple of two numbers in points (1/72 of an inch). The canvas defaults to A4 (an international standard page size which differs from the American standard page size of `letter`), but it is better to explicitly specify it. Most common page sizes are found in the library module `reportlab.lib.pagesizes`, so you can use expressions like

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
```

If you have problems printing your document make sure you are using the right page size (usually either A4 or `letter`). Some printers do not work well with pages that are too large or too small.

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`）。

Very often, you will want to calculate things based on the page size. In the example above we extracted the width and height. Later in the program we may use the `width` variable to define a right margin as `width - inch` rather than using a constant. By using variables the margin will still make sense even if the page size changes.

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。

The `bottomup` argument switches coordinate systems. Some graphics systems (like PDF and PostScript) place (0,0) at the bottom left of the page others (like many graphical user interfaces [GUI's]) place the origin at the top left. The `bottomup` argument is deprecated and may be dropped in future

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。
Need to see if it really works for all tasks, and if not then get rid of it

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。

The `pageCompression` option determines whether the stream of PDF operations for each page is compressed. By default page streams are not compressed, because the compression slows the file generation process. If output size is important set `pageCompression=1`, but remember that, compressed documents will be smaller, but slower to generate. Note that images are *always* compressed, and this option will only save space if you have a very large amount of text and vector graphics on each page.

`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。

The `encoding` argument is largely obsolete in version 2.0 and can probably be omitted by 99% of users. Its default value is fine unless you very specifically need to use one of the 25 or so characters which are present

in MacRoman and not in Winansi. A useful reference to these is here: http://www.alanwood.net/demos/charsetdiffs.html. The parameter determines which font encoding is used for the standard Type 1 fonts; this should correspond to the encoding on your system. Note that this is the encoding used *internally by the font*; text you pass to the ReportLab toolkit for rendering should always either be a Python unicode string object or a UTF-8 encoded byte string (see the next chapter)! The font encoding has two values at present: `'WinAnsiEncoding'` or `'MacRomanEncoding'`. The variable `rl_config.defaultEncoding` above points to the former, which is standard on Windows, Mac OS X and many Unices (including Linux). If you are Mac user and don't have OS X, you may want to make a global change: modify the line at the top of *reportlab/pdfbase/pdfdoc.py* to switch it over. Otherwise, you can probably just ignore this argument completely and never pass it. For all TTF and the commonly-used CID fonts, the encoding you pass in here is ignored, since the reportlab library itself knows the right encodings in those cases.

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Unices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
可能就完全无视这个论点，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。

The demo script `reportlab/demos/stdfonts.py` will print out two test documents showing all code points in all fonts, so you can look up characters. Special characters can be inserted into string commands with the usual Python escape sequences; for example \101 = 'A'.

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'。

The `verbosity` argument determines how much log information is printed. By default, it is zero to assist applications which want to capture PDF from standard output. With a value of 1, you will get a confirmation message each time a document is generated. Higher numbers may give more output in future.

参数`verbosity`决定了打印多少日志信息。 默认情况下，它是零，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。

The `encrypt` argument determines if and how the document is encrypted. By default, the document is not encrypted. If `encrypt` is a string object, it is used as the user password for the pdf. If `encrypt` is an instance of `reportlab.lib.pdfencrypt.StandardEncryption`, this object is used to encrypt the pdf. This allows more finegrained control over the encryption settings. Encryption is covered in more detail in Chapter 4.

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 - all the info functions and other non-drawing stuff
要做的事 -- 所有的信息功能和其他非绘图的东西。
Cover all constructor arguments, and setAuthor etc.
覆盖所有构造函数参数，以及setAuthor等。

2.3 绘图操作

Suppose the `hello` function referenced above is implemented as follows (we will not explain each of the operations in detail yet).

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）。

```python
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(0,0,0,1.7*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")
```

Examining this code notice that there are essentially two types of operations performed using a canvas. The first type draws something on the page such as a text string or a rectangle or a line. The second type changes the state of the canvas such as changing the current fill or stroke color or changing the current font type and size.

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。

If we imagine the program as a painter working on the canvas the "draw" operations apply paint to the canvas using the current set of tools (colors, line styles, fonts, etcetera) and the "state change" operations change one of the current tools (changing the fill color from whatever it was to blue, or changing the current font to `Times-Roman` in 15 points, for example).

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`）。

The document generated by the "hello world" program listed above would contain the following graphics.

上面列出的 "hello world "程序生成的文档将包含以下图形。
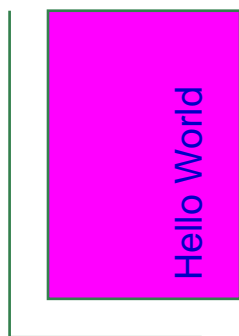
图 2 - 1 : pdfgen 生成 "Hello World"

关于本文档中的演示

This document contains demonstrations of the code discussed like the one shown in the rectangle above. These demos are drawn on a "tiny page" embedded within the real pages of the guide. The tiny pages are 5.5 inches wide and 3 inches tall. The demo displays show the actual output of the demo code. For convenience the size of the output has been reduced slightly.

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

This section briefly lists the tools available to the program for painting information onto a page using the canvas interface. These will be discussed in detail in later sections. They are listed here for easy reference and for summary purposes.

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)
```

The line methods draw straight line segments on the canvas.

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)
```

The shape methods draw common complex shapes on the canvas.

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)
```

The draw string methods draw single lines of text on the canvas.

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)
```

Text objects are used to format text in ways that are not supported directly by the canvas interface. A program creates a text object from the canvas using beginText and then formats text by invoking textobject methods. Finally the textobject is drawn onto the canvas using drawText.

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)
```

Path objects are similar to text objects: they provide dedicated control for performing complex graphical drawing not directly provided by the canvas interface. A program creates a path object using beginPath populates the path with graphics using the methods of the path object and then draws the path on the canvas using drawPath.

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。

It is also possible to use a path as a "clipping region" using the clipPath method -- for example a circular path can be used to clip away the outer parts of a rectangular image leaving only a circular part of the image visible on the page.

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。

If `fill=1` is specified then the `fillMode` argument may be used to set either 0=even-odd or 1=non-zero filling mode. which will alter the way that complex paths are filled. If the default `None` values is used then the canvas `_fillMode` attribute value is used (normally 0 ie even-odd).

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

You need the Python Imaging Library (PIL) to use images with the ReportLab package. Examples of the techniques below can be found by running the script `test_pdfgen_general.py` in our `tests` subdirectory and looking at page 7 of the output.

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。

There are two similar-sounding ways to draw images. The preferred one is the `drawImage` method. This implements a caching system so you can define an image once and draw it many times; it will only be stored once in the PDF file. `drawImage` also exposes one advanced parameter, a transparency mask, and will expose more in future. The older technique, `drawInlineImage` stores bitmaps within the page stream and is thus very inefficient if you use the same image more than once in a document; but can result in PDFs which render faster if the images are very small and not repeated. We'll discuss the oldest one first:

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)
```

The `drawInlineImage` method places an image on the canvas. The `image` parameter may be either a PIL Image object or an image filename. Many common file formats are accepted including GIF and JPEG. It returns the size of the actual image in pixels as a (width, height) tuple.

`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)
```

The arguments and return value work as for `drawInlineImage`. However, we use a caching system; a given image will only be stored the first time it is used, and just referenced on subsequent use. If you supply a filename, it assumes that the same filename means the same image. If you supply a PIL image, it tests if the content has actually changed before re-embedding.

参数和返回值和`drawInlineImage`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。

The `mask` parameter lets you create transparent images. It takes 6 numbers and defines the range of RGB values which will be masked out or treated as transparent. For example with [0,2,40,42,136,139], it will mask out any pixels with a Red value from 0 or 1, Green from 40 or 41 and Blue of 136, 137 or 138 (on a scale of 0-255). It's currently your job to know which color is the 'transparent' or background one.

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 allows for many image features and we will expose more of the over time, probably with extra keyword arguments to drawImage.

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()
```

The showPage method finishes the current page. All additional drawing will be done on another page.

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。

Warning! All state changes (font changes, color settings, geometry transforms, etcetera) are FORGOTTEN when you advance to a new page in pdfgen. Any state settings you wish to preserve must be set up again before the program proceeds with drawing!

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

This section briefly lists the ways to switch the tools used by the program for painting information onto a page using the canvas interface. These too will be discussed in detail in later sections.

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.setStrikeColorCMYK(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 supports three different color models: gray level, additive (red/green/blue or RGB), and subtractive with darkness parameter (cyan/magenta/yellow/darkness or CMYK). The ReportLab packages also provide named colors such as lawngreen. There are two basic color parameters in the graphics state: the Fill color for the interior of graphic figures and the Stroke color for the boundary of graphic figures. The above methods support setting the fill or stroke color using any of the four color specifications.

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)
```

The `setFont` method changes the current text font to a given type and size. The `leading` parameter specifies the distance down to move when advancing from one text line to the next.

`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)
```

Lines drawn in PDF can be presented in a number of graphical styles. Lines can have different widths, they can end in differing cap styles, they can meet in different join styles, and they can be continuous or they can be dotted or dashed. The above methods adjust these various parameters.

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)
```

All PDF drawings fit into a specified page size. Elements drawn outside of the specified page size are not visible. Furthermore all drawn elements are passed through an affine transformation which may adjust their location and/or distort their appearence. The `setPageSize` method adjusts the current page size. The `transform`, `translate`, `scale`, `rotate`, and `skew` methods add additional transformations to the current transformation. It is important to remember that these transformations are *incremental* -- a new transform modifies the current transform (but does not replace it).

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()
```

Very often it is important to save the current font, graphics transform, line styles and other graphics state in order to restore them later. The `saveState` method marks the current graphics state for later restoration by a matching `restoreState`. Note that the save and restore method invocation must match -- a restore call restores the state to the most recently saved state which hasn't been restored yet. You cannot save the state on one page and restore it on the next, however -- no state is preserved between pages.

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。

Not all methods of the `canvas` object fit into the "tool" or "toolbox" categories. Below are some of the misfits, included here for completeness.

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)

By default locations on a page are identified by a pair of numbers. For example the pair `(4.5*inch, 1*inch)` identifies the location found on the page by starting at the lower left corner and moving to the right 4.5 inches and up one inch.

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。

For example, the following function draws a number of elements on a `canvas`.

例如，下面的函数在`canvas`上绘制一些元素。

```python
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([inch, 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.setFillColor(red)
c.rect(0,2*inch,0.2*inch,0.3*inch, fill=1)
c.setFillColor(green)
c.circle(4.5*inch, 0.4*inch, 0.2*inch, fill=1)
```

In the default user space the "origin" *(0,0)* point is at the lower left corner. Executing the `coords` function in the default user space (for the "demo minipage") we obtain the following.

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。
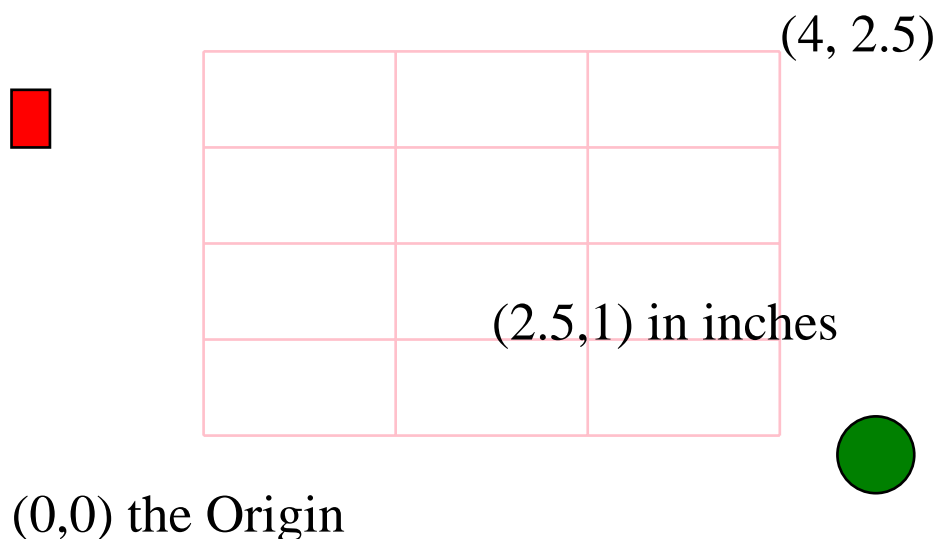
图 2 - 2 : 坐标系统

移动原点： translate方法

Often it is useful to "move the origin" to a new point off the lower left corner. The `canvas.translate(x,y)` method moves the origin for the current page to the point currently identified by *(x,y)*.

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。

For example the following translate function first moves the origin before drawing the same objects as shown above.

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)
```

This produces the following.

这就产生了以下结果:

(4, 2.5)

(2.5,1) in inches

(0,0) the Origin
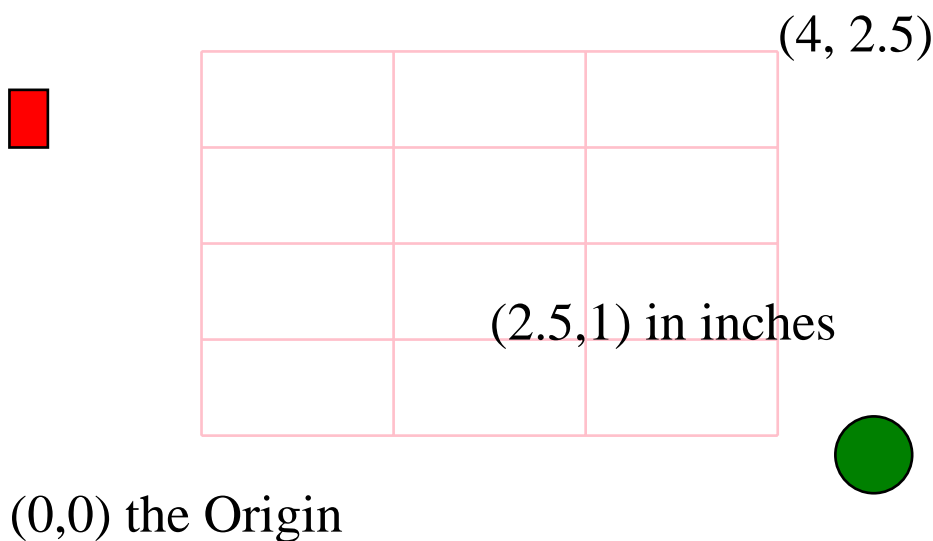
图 2-3:移动原点: `translate`方法

Note: As illustrated in the example it is perfectly possible to draw objects or parts of objects "off the page". In particular a common confusing bug is a translation operation that translates the entire drawing off the visible area of the page. If a program produces a blank page it is possible that all the drawn objects are off the page.

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

Another important operation is scaling. The scaling operation `canvas.scale(dx,dy)` stretches or shrinks the x and y dimensions by the dx, dy factors respectively. Often dx and dy are the same -- for example to reduce a drawing by half in all dimensions use `dx = dy = 0.5`. However for the purposes of illustration we show an example where `dx` and `dy` are different.

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。

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

This produces a "short and fat" reduced version of the previously displayed operations.

这样就会产生一个 "短小精悍"的缩小版的之前显示的操作。
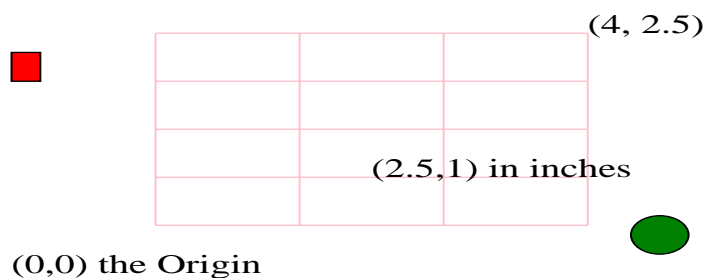
图 2 - 4 : 缩放坐标系统

Note: scaling may also move objects or parts of objects off the page, or may cause objects to "shrink to nothing."

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"。

Scaling and translation can be combined, but the order of the operations are important.

缩放和翻译可以结合起来，但操作的顺序很重要。

```python
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)
```

This example function first saves the current `canvas` state and then does a `scale` followed by a `translate`. Afterward the function restores the state (effectively removing the effects of the scaling and translation) and then does the *same* operations in a different order. Observe the effect below.

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。
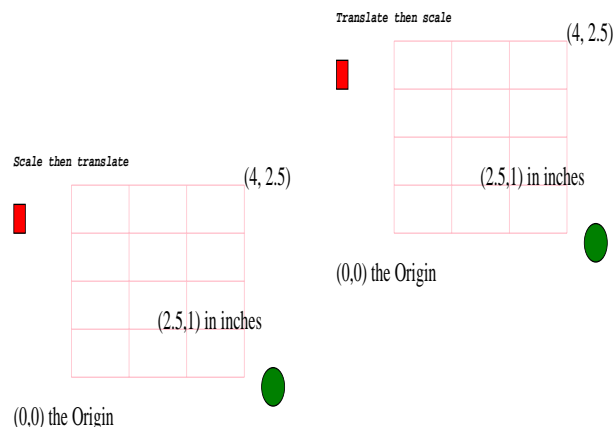
图 2 - 5：缩放和翻译

Note: scaling shrinks or grows everything including line widths so using the canvas.scale method to render a microscopic drawing in scaled microscopic units may produce a blob (because all line widths will get expanded a huge amount). Also rendering an aircraft wing in meters scaled to centimeters may cause the lines to shrink to the point where they disappear. For engineering or scientific purposes such as these scale and translate the units externally before rendering them using the canvas.

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。

The scaletranslate function used an important feature of the canvas object: the ability to save and restore the current parameters of the canvas. By enclosing a sequence of operations in a matching pair of canvas.saveState() an canvas.restoreState() operations all changes of font, color, line style, scaling, translation, or other aspects of the canvas graphics state can be restored to the state at the point of the saveState(). Remember that the save/restore calls must match: a stray save or restore operation may cause unexpected and undesirable behavior. Also, remember that *no* canvas state is preserved across page breaks, and the save/restore mechanism does not work across page breaks.

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/还原机制不会不能跨越分页符工作。

镜像

It is interesting although perhaps not terribly useful to note that scale factors can be negative. For example the following function

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)
```

creates a mirror image of the elements drawn by the `coord` function.

创建`coord`函数绘制的元素的镜像。
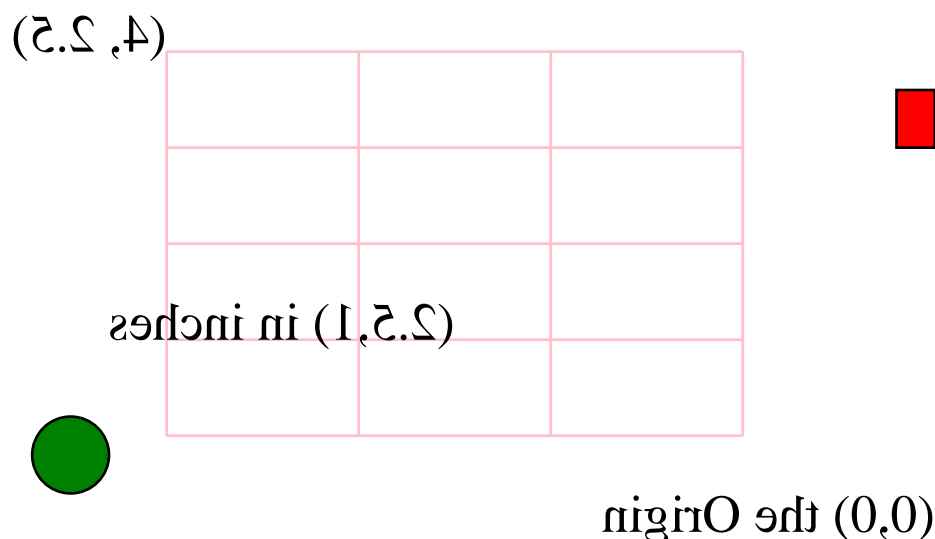
图 2 - 6 : 镜像

Notice that the text strings are painted backwards.

注意，文字串是倒着画的。

2.8 颜色

There are generally two types of colors used in PDF depending on the media where the PDF will be used. The most commonly known screen colors model RGB can be used in PDF, however in professional printing another color model CMYK is mainly used which gives more control over how inks are applied to paper. More on these color models below.

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

The `RGB` or additive color representation follows the way a computer screen adds different levels of the red, green, and blue light to make any color in between, where white is formed by turning all three lights on full (1,1,1).

`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。

There are three ways to specify RGB colors in `pdfgen`: by name (using the `color` module, by red/green/blue (additive, `RGB`) value, or by gray level. The `colors` function below exercises each of the four methods.

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.lawngreen, "lawngreen"],
        [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
```
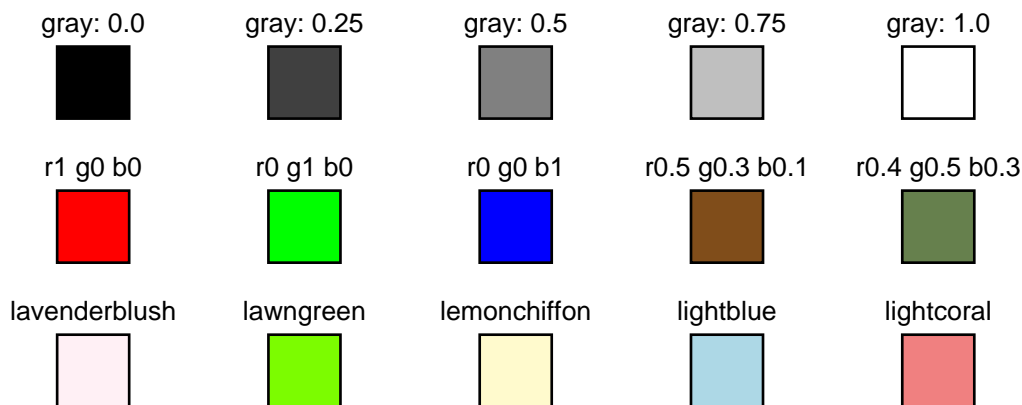

图 2 - 7 : RGB 颜色模块

RGB颜色透明度

Objects may be painted over other objects to good effect in pdfgen. Generally There are two modes of handling objects that overlap in space, the default objects in the top layer will hide any part of other objects that falls underneath it. If you need transparency you got two choices:

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. If your document is intended to be printed in a professional way and you are working in CMYK color space then you can use overPrint. In overPrinting the colors physically mix in the printer and thus a new color is obtained. By default a knockout will be applied and only top object appears. Read the CMYK section if this

is what you intend to use.

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. If your document is intended for screen output and you are using RGB colors then you can set an alpha value, where alpha is the opacity value of the color. The default alpha value is 1 (fully opaque) and you can use any real number value in the range 0-1.

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 transparency (alpha) is similar to overprint but works in RGB color space this example below demonstrates the alpha functionality. Refer to our website http://www.reportlab.com/snippets/ and look for snippets of overPrint and alpha to see the code that generates the graph below.

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。

```python
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)
```
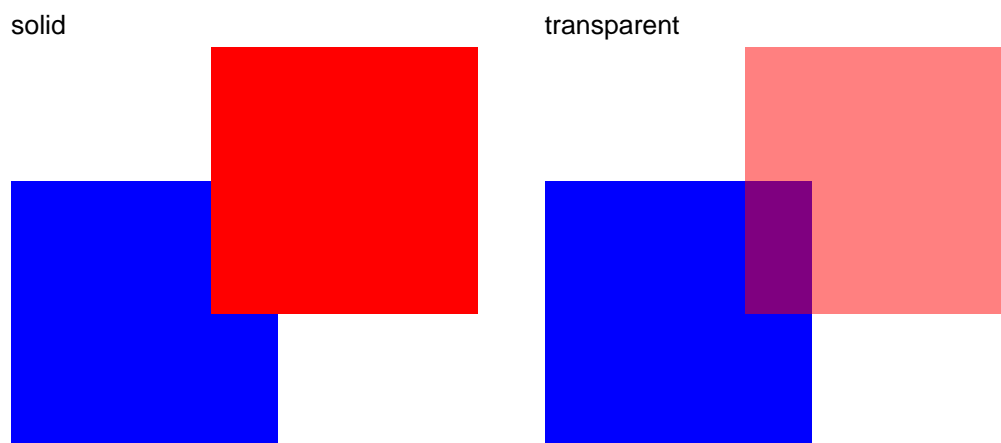
图 2 - 8 : Alpha 例子

CMYK 颜色

The `CMYK` or subtractive method follows the way a printer mixes three pigments (cyan, magenta, and yellow) to form colors. Because mixing chemicals is more difficult than combining light there is a fourth parameter for darkness. For example a chemical combination of the `CMY` pigments generally never makes a perfect black -- instead producing a muddy color -- so, to get black printers don not use the `CMY` pigments but use a direct black ink. Because `CMYK` maps more directly to the way printer hardware works it may be the case that colors specified in `CMYK` will provide better fidelity and better control when printed.

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。

There are two ways of representing CMYK Color: each color can be represented either by a real value between 0 and 1, or integer value between 0 and 100. Depending on your preference you can either use CMYKColor (for real values) or PCMYKColor (for integer values). 0 means 'no ink', so printing on white papers gives you white. 1 (or 100 if you use PCMYKColor) means 'the maximum amount of ink'. e.g. CMYKColor(0,0,0,1) is black, CMYKColor(0,0,0,0) means 'no ink', and CMYKColor(0.5,0,0,0) means 50 percent cyan color.

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。

```python
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.setFillColorCMYK(c,m,y1,k)
        canvas.rect(x+rdx, y+rdy, w, h, fill=1)
        canvas.setFillColor(black)
        canvas.drawCentredString(x+dx/2, y+texty, "c%s m%s y%s k%s"%cmyk)
        x = x+dx
```

c1 m0 y0 k0   c0 m1 y0 k0   c0 m0 y1 k0   c0 m0 y0 k1   c0 m0 y0 k0

图 2 - 9 : CMYK颜色模型

2.9 色彩空间检查

The `enforceColorSpace` argument of the canvas is used to enforce the consistency of the colour model used in a document. It accepts these values: CMYK, RGB, SEP, SEP_BLACK, SEP_CMYK. 'SEP' refers to named color separations such as Pantone spot colors - these can be mixed with CMYK or RGB according to the parameter used. The default is 'MIXED' which allows you to use colors from any color space. An exception is raised if any colors used are not convertible to the specified model, e.g. rgb and cmyk (more information in test_pdfgen_general). This approach doesn't check external images included in document.

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

When two CMYK colored objects overlap in printing, then either the object 'on top' will knock out the color of the the one underneath it, or the colors of the two objects will mix in the overlapped area. This behaviour can be set using the property `overPrint`.

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。

The `overPrint` function will cause ovelapping areas of color to mix. In the example below, the colors of the rectangles on the left should appear mixed where they overlap - If you can't see this effect then you may need to enable the 'overprint preview' option in your PDF viewing software. Some PDF viewers such as `evince` do not support overPrint; however Adobe Acrobat Reader does support it.

`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。
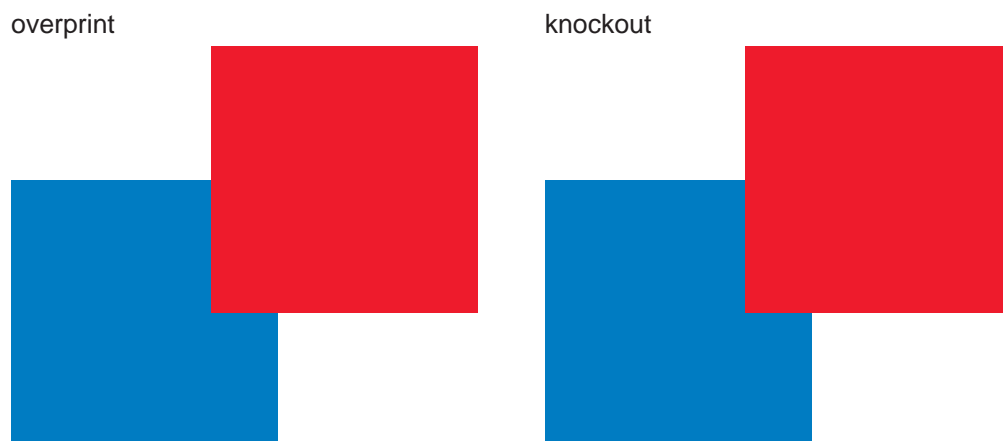
图 2 - 10 : OverPrint示例

其他对象的打印顺序示例

The word "SPUMONI" is painted in white over the colored rectangles, with the apparent effect of "removing" the color inside the body of the word.

"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。

```python
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")
```
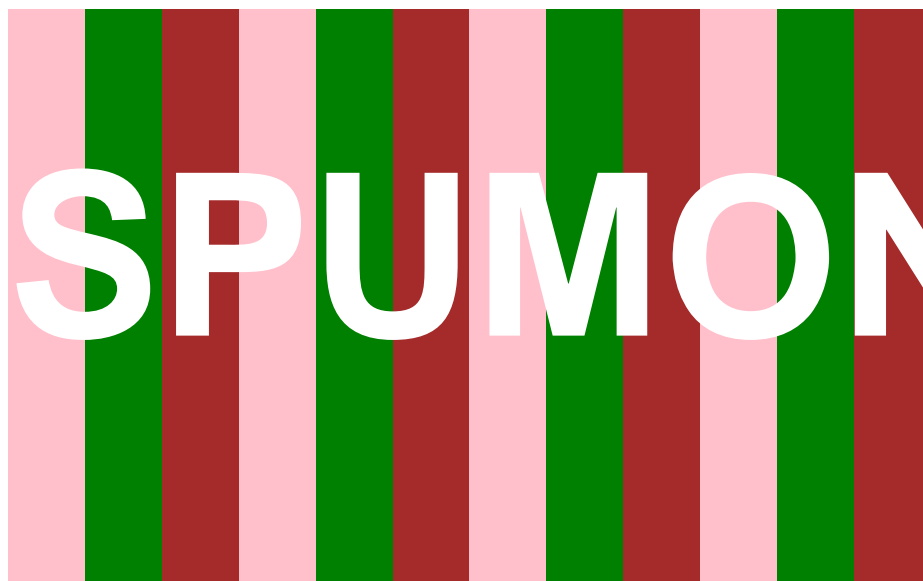
图 2 - 11 : 涂抹颜色

The last letters of the word are not visible because the default `canvas` background is white and painting white letters over a white background leaves no visible effect.

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。

This method of building up complex paintings in layers can be done in very many layers in `pdfgen` -- there are fewer physical limitations than there are when dealing with physical paints.

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。

```python
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.setFillColor(brown)
    canvas.setStrokeColor(black)
    canvas.drawPath(p, fill=1)
    # draw some circles (scoops)
    y = 1.5*inch
    for color in (pink, green, brown):
        canvas.setFillColor(color)
        canvas.circle(xcenter, y, radius, fill=1)
        y = y+radius
```

The `spumoni2` function layers an ice cream cone over the `spumoni` drawing. Note that different parts of the cone and scoops layer over eachother as well.

Spumoni2■■■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。
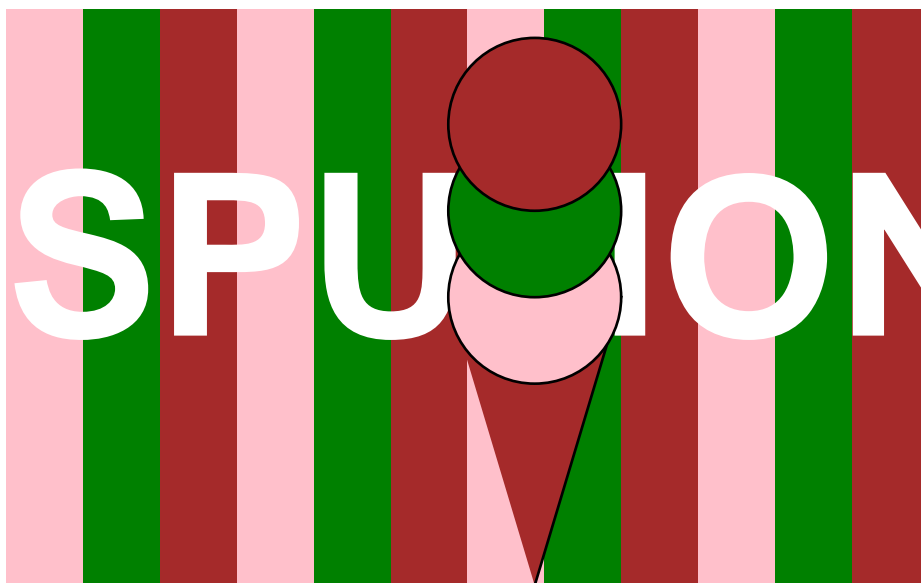
图 2 - 12 : 层层叠加

2.11 标准字体和文本对象

Text may be drawn in many different colors, fonts, and sizes in pdfgen. The textsize function demonstrates how to change the color and font and size of text and how to place text on the page.

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。

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

The textsize function generates the following page.

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

A number of different fonts are always available in pdfgen.

不同字体和大小的文本在pdfgen中总是有许多不同的字体。

```python
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
```

The fonts function lists the fonts that are always available. These don't need to be stored in a PDF document, since they are guaranteed to be present in Acrobat Reader.

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...*
SourceHanSansSC:	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

The Symbol and ZapfDingbats fonts cannot display properly because the required glyphs are not present in those fonts.

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。

For information on how to use arbitrary fonts, see the next chapter.

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

For the dedicated presentation of text in a PDF document, use a text object. The text object interface provides detailed control of text layout parameters not available directly at the `canvas` level. In addition, it results in smaller PDF that will render faster than many separate calls to the `drawString` methods.

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)
```

The text object methods shown above relate to basic text geometry.

上面显示的文本对象方法与基本的文本几何体有关。

A text object maintains a text cursor which moves about the page when text is drawn. For example the `setTextOrigin` places the cursor in a known position and the `textLine` and `textLines` methods move the text cursor down past the lines that have been missing.

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。

```python
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)
```

The `cursormoves` function relies on the automatic movement of the text cursor for placing text after the origin has been set.

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

It is also possible to control the movement of the cursor more explicitly by using the `moveCursor` method (which moves the cursor as an offset from the start of the current *line* NOT the current cursor, and which also has positive y offsets move *down*)in contrast to the normal geometry where positive y usually moves up.

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。

```python
def cursormoves2(canvas):
    from reportlab.lib.units import inch
    textobject = canvas.beginText()
    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.setFillColorRGB(0.4,0,1)
```

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

Here the `textOut` does not move the down a line in contrast to the `textLine` function which does move down.

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

<div align="center">

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

</div>

<div align="center">

图 2 - 16：文本光标如何再次移动

</div>

字符间距

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

The `setCharSpace` method adjusts one of the parameters of text -- the inter-character spacing.

`setCharSpace`方法调整文本的一个参数--字符间的间距。

```python
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)
```

The `charspace` function exercises various spacing settings. It produces the following page.

`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)
```

The setWordSpace method adjusts the space between words.

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.setFillColorCMYK(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)
```

The wordspace function shows what various word space settings look like below.

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)
```

Lines of text can be stretched or shrunken horizontally by the setHorizScale method.

文本行可以通过setHorizScale方法进行水平拉伸或收缩。

```python
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.setFillColorCMYK(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)
```
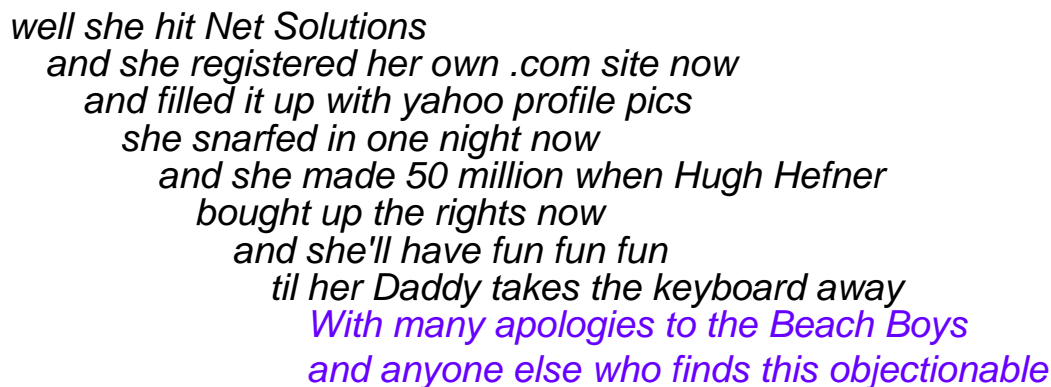
The horizontal scaling parameter *horizScale* is given in percentages (with 100 as the default), so the 80 setting shown below looks skinny.

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)
```

The vertical offset between the point at which one line starts and where the next starts is called the leading offset. The setLeading method adjusts the leading offset.

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.setFillColorCMYK(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)
```

As shown below if the leading offset is set too small characters of one line my write over the bottom parts of characters in the previous line.

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)
```

The setTextRenderMode method allows text to be used as a foreground for clipping background drawings, for example.

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

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

The setRise method ^{raises} or $_{lowers}$ text on the line (for creating superscripts or subscripts, for example).

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

These color change operations change the color of the text and are otherwise similar to the color methods for the canvas object.

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

Just as textobjects are designed for the dedicated presentation of text, path objects are designed for the dedicated construction of graphical figures. When path objects are drawn onto a canvas they are drawn as one figure (like a rectangle) and the mode of drawing for the entire figure can be adjusted: the lines of the figure can be drawn (stroked) or not; the interior of the figure can be filled or not; and so forth.

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。

For example the star function uses a path object to draw a star

例如，star函数使用一个路径对象来绘制一个星体

```python
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)
```

The star function has been designed to be useful in illustrating various line style parameters supported by pdfgen.

star函数被设计用来说明pdfgen支持的各种行式参数。

Title Here

Comment here.

图 2 - 21 : 直线样式参数

直线连接设置

The setLineJoin method can adjust whether line segments meet in a point a square or a rounded vertex.

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。

```python
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)
```

The line join setting is only really of interest for thick lines because it cannot be seen clearly for thin lines.

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。

<div align="center">图 2 - 22 : 不同的直线连接样式</div>

直线帽子设置

The line cap setting, adjusted using the `setLineCap` method, determines whether a terminating line ends in a square exactly at the vertex, a square over the vertex or a half circle over the vertex.

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。

```python
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)
```

The line cap setting, like the line join setting, is only clearly visible when the lines are thick.

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。

Default   Round cap   Square cap

no projection   1: ends in half circle 2: projects out half a width
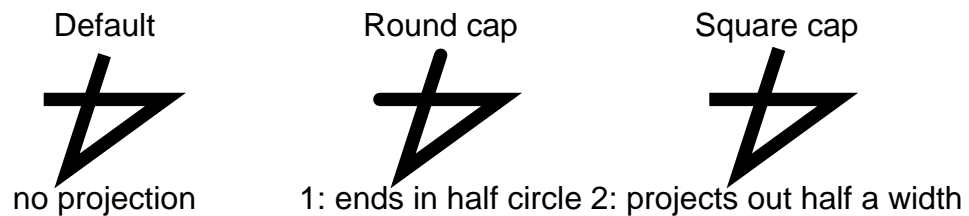
图 2 - 23 : 直线帽子设置

破折号和断线

The setDash method allows lines to be broken into dots or dashes.

用setDash方法可以将线条分成点或破折号。

```python
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)
```

The patterns for the dashes or dots can be in a simple on/off repeating pattern or they can be specified in a complex repeating pattern.

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。

Simple dashes   Dots   Complex Pattern

6 points on, 3 off   One on, two off   [1,1,3,3,1,4,4,1]
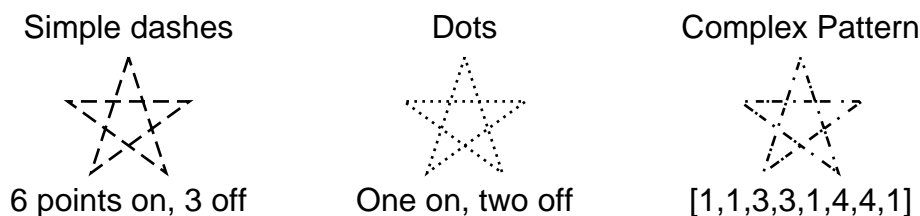
图 2 - 24 : 破折号

用路径对象创建复杂的图形

Combinations of lines, curves, arcs and other figures can be combined into a single figure using path objects. For example the function shown below constructs two path objects using lines and curves. This function will be used later on as part of a pencil icon construction.

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。

```python
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])
```

Note that the interior of the pencil tip is filled as one object even though it is constructed from several lines and curves. The pencil lead is then drawn over it using a new path object.

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。
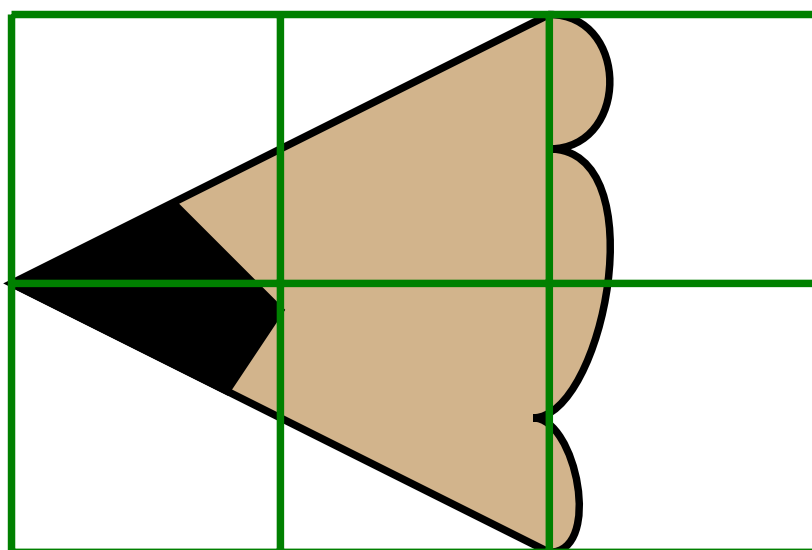
图 2 - 25 : 铅笔头

2.14 矩形、圆形、椭圆形。

The pdfgen module supports a number of generally useful shapes such as rectangles, rounded rectangles, ellipses, and circles. Each of these figures can be used in path objects or can be drawn directly on a canvas. For example the pencil function below draws a pencil icon using rectangles and rounded rectangles with various fill colors and a few other annotations.

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。

```python
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 erasor
    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)
```

Note that this function is used to create the "margin pencil" to the left. Also note that the order in which the elements are drawn are important because, for example, the white rectangles "erase" parts of a black rectangle and the "tip" paints over part of the yellow rectangle.

注意，这个函数是用来创建左边的'边距铅笔'的。还要注意的是，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。
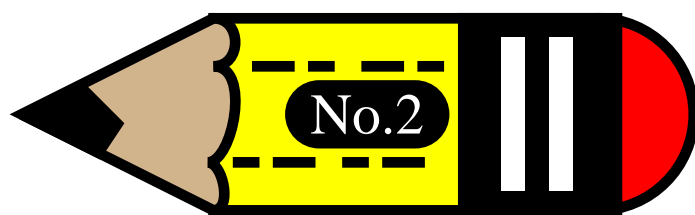
图 2 - 26 : 铅笔

2.15 贝兹尔曲线

Programs that wish to construct figures with curving borders generally use Bezier curves to form the borders.

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。

```python
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)
```

A Bezier curve is specified by four control points (x1,y1),(x2,y2),(x3,y3),(x4,y4). The curve starts at (x1,y1) and ends at (x4,y4) and the line segment from (x1,y1) to (x2,y2) and the line segment from (x3,y3) to (x4,y4) both form tangents to the curve. Furthermore the curve is entirely contained in the convex figure with vertices at the control points.

Bezier曲线由四个控制点(x1,y1)，(x2,y2)，(x3,y3)，(x4,y4)指定。曲线起于(x1,y1)，止于(x4,y4)，从(x1,y1)到(x2,y2)的线段和从(x3,y3)到(x4,y4)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。
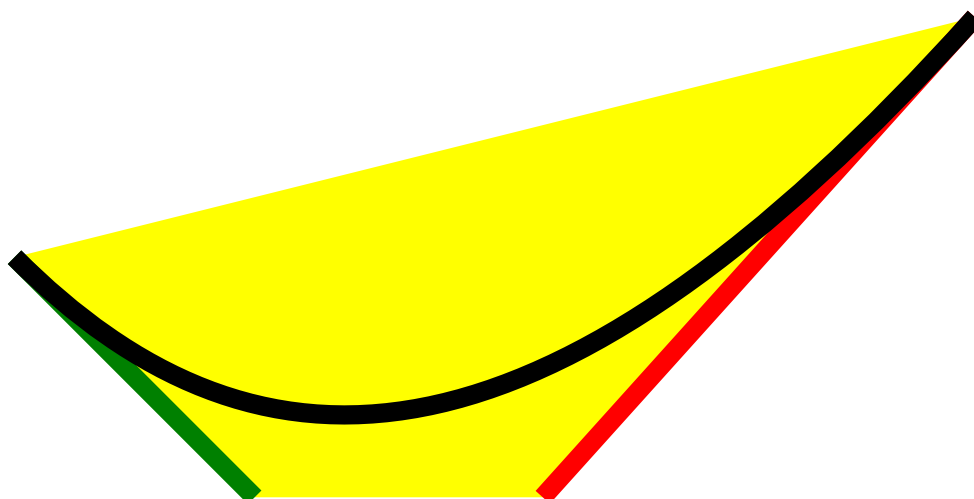
图 2 - 27 : 基本贝塞尔曲线

The drawing above (the output of `testbezier`) shows a bezier curves, the tangent lines defined by the control points and the convex figure with vertices at the control points.

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

It is often useful to join several bezier curves to form a single smooth curve. To construct a larger smooth curve from several bezier curves make sure that the tangent lines to adjacent bezier curves that join at a control point lie on the same line.

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。

```python
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
    dxdy = [(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 dxdy:
            px = xc + xd*xoffset*dx
            py = yc + yd*yoffset*dy
            pointlist.append((px,py))
        yoffset = -xoffset
        for (dy,dx) in dxdy:
            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)
```

The figure created by `testbezier2` describes a smooth complex curve because adjacent tangent lines "line up" as illustrated below.

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。
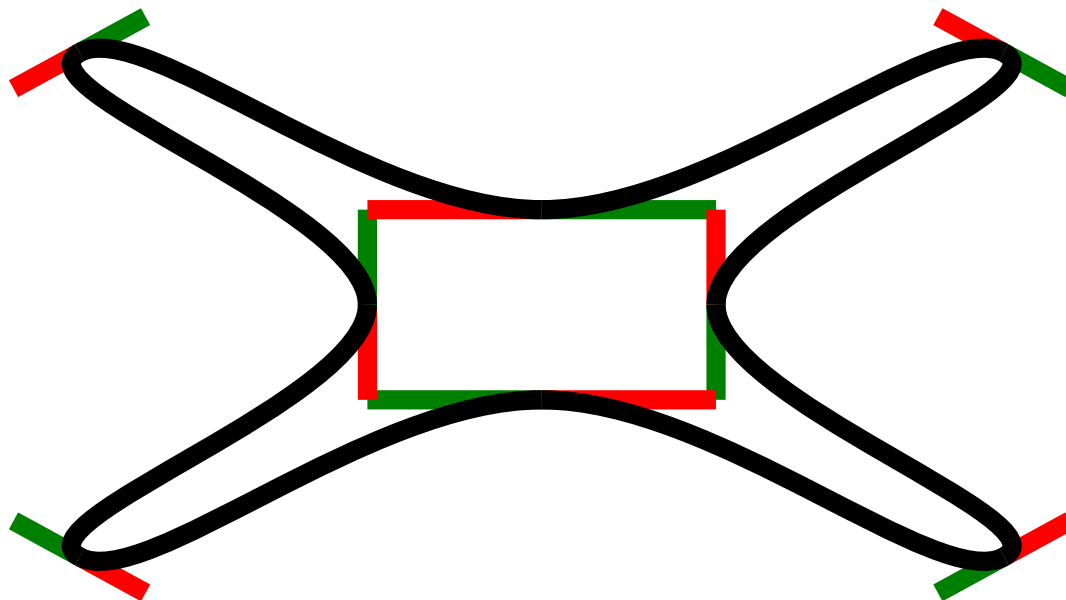
图 2 - 28 : bezier curves

2.16 路径对象方法

Path objects build complex graphical figures by setting the "pen" or "brush" at a start point on the canvas and drawing lines or curves to additional points on the canvas. Most operations apply paint on the canvas starting at the end point of the last operation and leave the brush at a new end point.

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)
```

The `moveTo` method lifts the brush (ending any current sequence of lines or curves if there is one) and replaces the brush at the new *(x,y)* location on the canvas to start a new path sequence.

`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)
```

The `lineTo` method paints straight line segment from the current brush location to the new *(x,y)* location.

`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)
```

The `curveTo` method starts painting a Bezier curve beginning at the current brush location, using *(x1,y1)*,*(x2,y2)*, and *(x3,y3)* as the other three control points, leaving the brush on *(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)
```

The `arc` and `arcTo` methods paint partial ellipses. The `arc` method first "lifts the brush" and starts a new shape sequence. The `arcTo` method joins the start of the partial ellipse to the current shape sequence by line segment before drawing the partial ellipse. The points *(x1,y1)* and *(x2, y2)* define opposite corner points of a rectangle enclosing the ellipse. The `startAng` is an angle (in degrees) specifying where to begin the partial ellipse where the 0 angle is the midpoint of the right border of the enclosing rectangle (when *(x1,y1)* is the lower left corner and *(x2, y2)* is the upper right corner). The `extent` is the angle in degrees to traverse on the ellipse.

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■■■■■(■■)■■■■■■■■■■■■■■■■■0■■■■■■■■■
■■■■■■■■*(x1,y1)*■■■■■*(x2, y2*■■■■■■.extent$是指与椭圆上的横移。

```python
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.setFillColorRGB(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)
```

The `arcs` function above exercises the two partial ellipse methods. It produces the following drawing.

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。
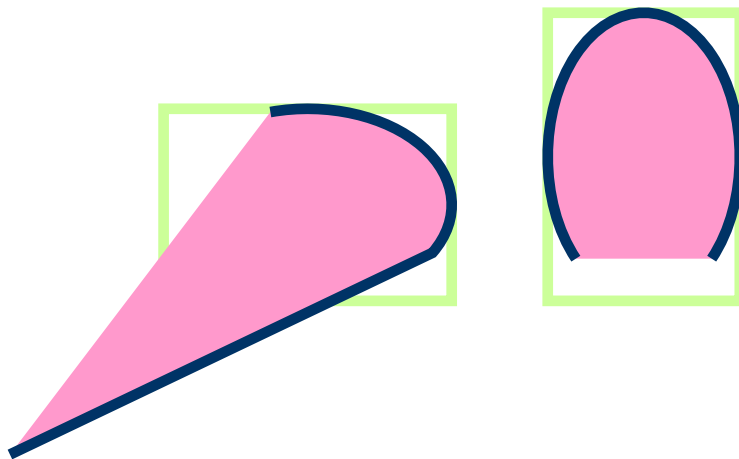
图 2 - 29 : 弧线

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

The `rect` method draws a rectangle with lower left corner at *(x,y)* of the specified *width* and *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)
```

The `ellipse` method draws an ellipse enclosed in the rectange with lower left corner at *(x,y)* of the specified *width* and *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)
```

The `circle` method draws a circle centered at *(x_cen, y_cen)* with radius *r*.

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

```python
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.setFillColorRGB(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)
```

The `variousshapes` function above shows a rectangle, circle and ellipse placed in a frame of reference grid.

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。
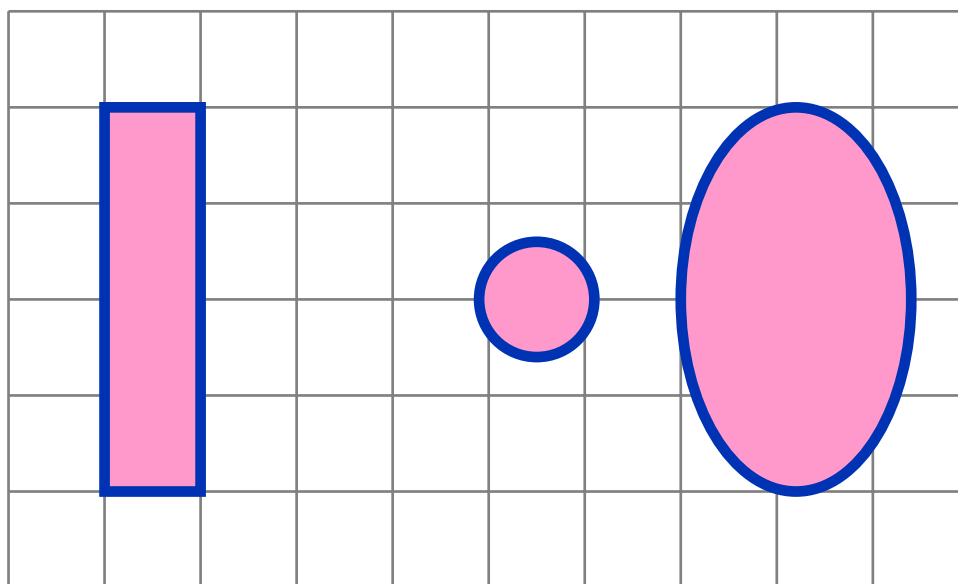
图 2 - 30：路径对象中的矩形、圆形、椭圆形。

```
pathobject.close()
```

The `close` method closes the current graphical figure by painting a line segment from the last point of the figure to the starting point of the figure (the the most recent point where the brush was placed on the paper by `moveTo` or `arc` or other placement operations).

`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。

```python
def closingfigures(canvas):
    from reportlab.lib.units import inch
```

```python
h = inch/3.0; k = inch/2.0
canvas.setStrokeColorRGB(0.2,0.3,0.5)
canvas.setFillColorRGB(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)
```

The `closingfigures` function illustrates the effect of closing or not closing figures including a line segment and a partial ellipse.

`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。
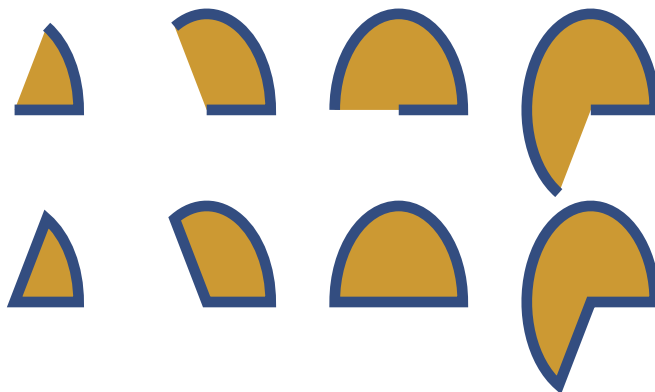
图 2 - 31 : 闭合和不闭合的路径对象数字

Closing or not closing graphical figures effects only the stroked outline of a figure, not the filling of the figure as illustrated above.

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。

For a more extensive example of drawing using a path object examine the `hand` function.

关于使用路径对象绘图的更广泛的例子，请检查hand函数。

```python
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)
```

In debug mode (the default) the hand function shows the tangent line segments to the bezier curves used to compose the figure. Note that where the segments line up the curves join smoothly, but where they do not line up the curves show a "sharp edge".

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"。
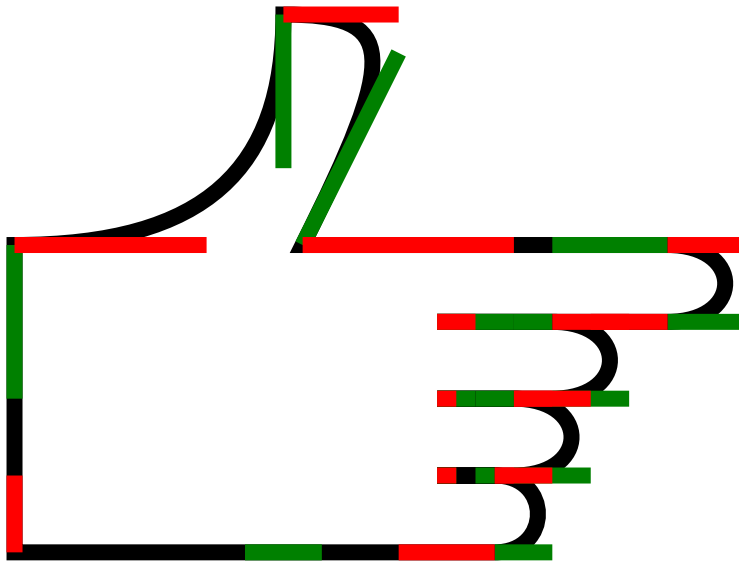
图 2 - 32 : 手形图

Used in non-debug mode the hand function only shows the Bezier curves. With the fill parameter set the figure is filled using the current fill color.

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.setFillColorRGB(0.1, 0.3, 0.9)
    canvas.setStrokeGray(0.5)
    hand(canvas, debug=0, fill=1)
```

Note that the "stroking" of the border draws over the interior fill where they overlap.

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

So far the graphics we have seen were created on a fairly low level. It should be noted, though, that there is another way of creating much more sophisticated graphics using the dedicated high-level *ReportLab Graphics Library*.

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。

It can be used to produce high-quality, platform-independant, reusable graphics for different output formats (vector and bitmap) like PDF, EPS, SVG, JPG and PNG.

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。

A more thorough description of its philosophy and features is now covered in Chapter 11 of this document, *Graphics*, which contains information about the existing components and how to create customized ones.

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。

Chapter 11 also contains details of the ReportLab charting package and its components (labels, axes, legends and different types of charts like bar, line and pie charts) that builds directly on the graphics library.

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、线状图和饼状图），它直接建立在图形库上。

第 3 章 字体和编码

This chapter covers fonts, encodings and Asian language capabilities. If you are purely concerned with generating PDFs for Western European languages, you can just read the "Unicode is the default" section below and skip the rest on a first reading. We expect this section to grow considerably over time. We hope that Open Source will enable us to give better support for more of the world's languages than other tools, and we welcome feedback and help in this area.

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 是默认编码

Starting with reportlab Version 2.0 (May 2006), all text input you provide to our APIs should be in UTF8 or as Python Unicode objects. This applies to arguments to canvas.drawString and related APIs, table cell content, drawing object parameters, and paragraph source text.

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。

We considered making the input encoding configurable or even locale-dependent, but decided that "explicit is better than implicit".

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"。

This simplifies many things we used to do previously regarding greek letters, symbols and so on. To display any character, find out its unicode code point, and make sure the font you are using is able to display it.

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。

If you are adapting a ReportLab 1.x application, or reading data from another source which contains single-byte data (e.g. latin-1 or WinAnsi), you need to do a conversion into Unicode. The Python codecs package now includes converters for all the common encodings, including Asian ones.

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。

If your data is not encoded as UTF8, you will get a UnicodeDecodeError as soon as you feed in a non-ASCII character. For example, this snippet below is attempting to read in and print a series of names, including one with a French accent: *Marc-André Lemburg*. The standard error is quite helpful and tells you what character it doesn't like:

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-Andr/é 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
>>>
```

The simplest fix is just to convert your data to unicode, saying which encoding it comes from, like this:

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

There are still a number of places in the code, including the rl_config defaultEncoding parameter, and arguments passed to various Font constructors, which refer to encodings. These were useful in the past when people needed to use glyphs in the Symbol and ZapfDingbats fonts which are supported by PDF viewing devices. By default the standard fonts (Helvetica, Courier, Times Roman) will offer the glyphs available in Latin-1. However, if our engine detects a character not in the font, it will attempt to switch to Symbol or ZapfDingbats to display these. For example, if you include the Unicode character for a pair of right-facing scissors, \u2702, in a call to *drawString*, you should see them (there is an example in *test_pdfgen_general.py/pdf*). It is not necessary to switch fonts in your code.

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

As discussed in the previous chapter, every copy of Acrobat Reader comes with 14 standard fonts built in. Therefore, the ReportLab PDF Library only needs to refer to these by name. If you want to use other fonts, they must be available to your code and will be embedded in the PDF document.

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。

You can use the mechanism described below to include arbitrary fonts in your documents. We have an open source font named *DarkGardenMK* which we may use for testing and/or documenting purposes (and which you may use as well). It comes bundled with the ReportLab distribution in the directory reportlab/fonts.

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。

Right now font-embedding relies on font description files in the Adobe AFM ('Adobe Font Metrics') and PFB ('Printer Font Binary') format. The former is an ASCII file and contains information about the characters ('glyphs') in the font such as height, width, bounding box info and other 'metrics', while the latter is a binary file that describes the shapes of the font. The reportlab/fonts directory contains the files 'DarkGardenMK.afm' and 'DarkGardenMK.pfb' that are used as an example font.

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。

In the following example locate the folder containing the test font and register it for future use with the pdfmetrics module, after which we can use it like any other standard font.

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。

```python
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')
```

Note that the argument "WinAnsiEncoding" has nothing to do with the input; it's to say which set of characters within the font file will be active and available.

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。
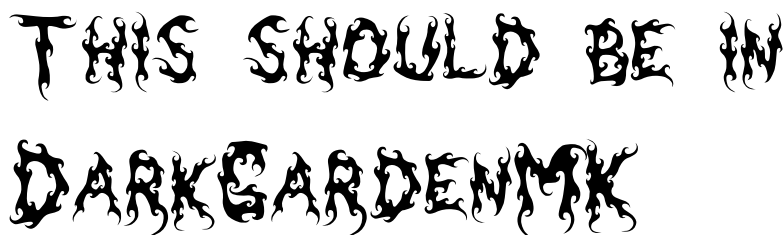
图 3‑1：使用非常不标准的字体

The font's facename comes from the AFM file's FontName field. In the example above we knew the name in advance, but quite often the names of font description files are pretty cryptic and then you might want to retrieve the name from an AFM file automatically. When lacking a more sophisticated method you can use some code as simple as this:

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字是非常神秘的，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。

```python
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
```

In the *DarkGardenMK* example we explicitly specified the place of the font description files to be loaded. In general, you'll prefer to store your fonts in some canonic locations and make the embedding mechanism aware of them. Using the same configuration mechanism we've already seen at the beginning of this section we can indicate a default search path for Type-1 fonts.

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字体指定一个默认的搜索路径。

Unfortunately, there is no reliable standard yet for such locations (not even on the same platform) and, hence, you might have to edit one of the files `reportlab_settings.py` or `~/.reportlab_settings` to modify the value of the `T1SearchPath` identifier to contain additional directories. Our own recommendation is to use the *reportlab/fonts* folder in development; and to have any needed fonts as packaged parts of your application in any kind of controlled server deployment. This insulates you from fonts being installed and uninstalled by other software or system administrator.

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

If you specify an encoding, it is generally assumed that the font designer has provided all the needed glyphs. However, this is not always true. In the case of our example font, the letters of the alphabet are present, but many symbols and accents are missing. The default behaviour is for the font to print a 'notdef' character - typically a blob, dot or space - when passed a character it cannot draw. However, you can ask the library to warn you instead; the code below (executed before loading a font) will cause warnings to be generated for any glyphs not in the font when you register it.

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，字体会打印一个 "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

This section shows you the glyphs available in the common encodings.

本节为您展示常用编码中可用的字形。

The code chart below shows the characters in the `WinAnsiEncoding`. This is the standard encoding on Windows and many Unix systems in America and Western Europe. It is also knows as Code Page 1252, and is practically identical to ISO-Latin-1 (it contains one or two extra characters). This is the default encoding used by the Reportlab PDF Library. It was generated from a standard routine in `reportlab/lib`,

`codecharts.py`, which can be used to display the contents of fonts. The index numbers along the edges are in hex.

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	€	•	,	ƒ	„	…	†	‡	^	‰	Š	‹	Œ	•	Ž	•	•	'	'	"	"	•	–	—	~	™	š	›	œ	•	ž	Ÿ
A0		¡	¢	£	¤	¥	¦	§	¨	©	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

Figure 3-2: WinAnsi Encoding

The code chart below shows the characters in the `MacRomanEncoding`. as it sounds, this is the standard encoding on Macintosh computers in America and Western Europe. As usual with non-unicode encodings, the first 128 code points (top 4 rows in this case) are the ASCII standard and agree with the WinAnsi code chart above; but the bottom 4 rows differ.

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	ö	õ	ú	ù	û	ü
A0	†	°	¢	£	§	•	¶	ß	®	©	™	´	¨	≠	Æ	Ø	∞	±	≤	≥	¥	µ	∂	∑	∏	π	∫	ª	º	Ω	æ	ø
C0	¿	¡	¬	√	ƒ	≈	∆	«	»	…	■	À	Ã	Õ	Œ	œ	–	—	"	"	'	'	÷	◊	ÿ	Ÿ	/	€	‹	›	fi	fl
E0	‡	·	,	„	‰	Â	Ê	Á	Ë	È	Í	Î	Ï	Ì	Ó	Ô	■	Ò	Ú	Û	Ù	ı	^	~	¯	˘	˙	˚	¸	˝	˛	ˇ

Figure 3-3: MacRoman Encoding

These two encodings are available for the standard fonts (Helvetica, Times-Roman and Courier and their variants) and will be available for most commercial fonts including those from Adobe. However, some fonts contain non- text glyphs and the concept does not really apply. For example, ZapfDingbats and Symbol can each be treated as having their own 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，ZapfDingbats和Symbol可以被视为各自拥有自己的编码。

图 3 - 4 : ZapfDingbats和它的唯一编码。

图 3 - 5 : 符号及其唯一的编码

3.5 支持TrueType字体

Marius Gedminas (`mgedmin@delfi.lt`) with the help of Viktorija Zaksiene (`vika@pov.lt`) have contributed support for embedded TrueType fonts. TrueType fonts work in Unicode/UTF8 and are not limited to 256 characters.

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。

We use `reportlab.pdfbase.ttfonts.TTFont` to create a true type font object and register using `reportlab.pdfbase.pdfmetrics.registerFont`. In pdfgen drawing directly to the canvas we can do

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

```python
# 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('VeraIt', 'VeraIt.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

In the above example the True Type font object is created using

在上面的例子中，True Type字体对象是使用

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

so that the ReportLab internal name is given by the first argument and the second argument is a string(or file like object) denoting the font's TTF file. In Marius' original patch the filename was supposed to be exactly correct, but we have modified things so that if the filename is relative then a search for the corresponding file is done in the current directory and then in directories specified by `reportlab.rl_config.TTFSearchpath`!

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`!

Before using the TT Fonts in Platypus we should add a mapping from the family name to the individual font names that describe the behaviour under the `<b>` and `<i>` attributes.

在使用Platypus中的TT字体之前，我们应该在`<b>`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='VeraIt',
boldItalic='VeraBI')
```

If we only have a Vera regular font, no bold or italic then we must map all to the same internal fontname. *<b>* and *<i>* tags may now be used safely, but have no effect. After registering and mapping the Vera font as above we can use paragraph text like

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。*<b>*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!

Figure 3-7: Using TTF fonts in paragraphs

3.6 亚洲字体支持

The Reportlab PDF Library aims to expose full support for Asian fonts. PDF is the first really portable solution for Asian text handling. There are two main approaches for this: Adobe's Asian Language Packs, or TrueType fonts.

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

This approach offers the best performance since nothing needs embedding in the PDF file; as with the standard fonts, everything is on the reader.

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 makes available add-ons for each main language. In Adobe Reader 6.0 and 7.0, you will be prompted to download and install these as soon as you try to open a document using them. In earlier versions, you would see an error message on opening an Asian document and had to know what to do.

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。

Japanese, Traditional Chinese (Taiwan/Hong Kong), Simplified Chinese (mainland China) and Korean are all supported and our software knows about the following fonts:

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'

Since many users will not have the font packs installed, we have included a rather grainy *bitmap* of some Japanese characters. We will discuss below what is needed to generate them.

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	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥	旭	葦	芦	鯵
20	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或	粟	袷	安	庵	按	暗	案	闇
40	鞍	杏	以	伊	位	依	偉	囲	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異
60	移	維	緯	胃	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
80	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭					

Prior to Version 2.0, you had to specify one of many native encodings when registering a CID Font. In version 2.0 you should a new UnicodeCIDFont class.

在2.0版本之前，当你注册一个CID
Font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。

```python
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)
```

The old coding style with explicit encodings should still work, but is now only relevant if you need to construct vertical text. We aim to add more readable options for horizontal and vertical text to the UnicodeCIDFont constructor in future. The following four test scripts generate samples in the corresponding languages:

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
```

In previous versions of the ReportLab PDF Library, we had to make use of Adobe's CMap files (located near Acrobat Reader if the Asian Language packs were installed). Now that we only have one encoding to deal with, the character width data is embedded in the package, and CMap files are not needed for generation. The CMap search path in $rl\_config.py$ is now deprecated and has no effect if you restrict yourself to UnicodeCIDFont.

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

This is the easy way to do it. No special handling at all is needed to work with Asian TrueType fonts. Windows users who have installed, for example, Japanese as an option in Control Panel, will have a font "msmincho.ttf" which can be used. However, be aware that it takes time to parse the fonts, and that quite large subsets may need to be embedded in your PDFs. We can also now parse files ending in .ttc, which are a slight variation of .ttf.

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

We expect to be developing this area of the package for some time .accept2dyear Here is an outline of the main priorities. We welcome help!

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

- Ensure that we have accurate character metrics for all encodings in horizontal and vertical writing.

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。

- Add options to *UnicodeCIDFont* to allow vertical and proportional variants where the font permits it.
- 为*UnicodeCIDFont*添加选项，在字体允许的情况下，允许垂直和比例变体。
- Improve the word wrapping code in paragraphs and allow vertical writing.
- 改进段落中的包字代码，允许竖写。

3.7 RenderPM 测试

This may also be the best place to mention the test function of `reportlab/graphics/renderPM.py`, which can be considered the cannonical place for tests which exercise renderPM (the "PixMap Renderer", as opposed to renderPDF, renderPS or renderSVG).

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。

If you run this from the command line, you should see lots of output like the following.

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
```

This runs a number of tests progressing from a "Hello World" test, through various tests of Lines; text strings in a number of sizes, fonts, colour and alignments; the basic shapes; translated and rotated groups; scaled coordinates; rotated strings; nested groups; anchoring and non-standard fonts.

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。

It creates a subdirectory called `pmout`, writes the image files into it, and writes an `index.html` page which makes it easy to refer to all the results.

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。

The font-related tests which you may wish to look at are test #11 ('Text strings in a non-standard font') and test #12 ('Test Various Fonts').

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 provides a number of features to make electronic document viewing more efficient and comfortable, and our library exposes a number of these.

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

The Form feature lets you create a block of graphics and text once near the start of a PDF file, and then simply refer to it on subsequent pages. If you are dealing with a run of 5000 repetitive business forms - for example, one-page invoices or payslips - you only need to store the backdrop once and simply draw the changing text on each page. Used correctly, forms can dramatically cut file size and production time, and apparently even speed things up on the printer.

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。

Forms do not need to refer to a whole page; anything which might be repeated often should be placed in a form.

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。

The example below shows the basic sequence used. A real program would probably define the forms up front and refer to them from another location.

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。

```python
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 supports internal hyperlinks. There is a very wide range of link types, destination types and events which can be triggered by a click. At the moment we just support the basic ability to jump from one part of a document to another, and to control the zoom level of the window after the jump. The *bookmarkPage* method defines a destination that is the endpoint of a jump.

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。

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

By default the `bookmarkPage` method defines the page itself as the destination. After jumping to an endpoint defined by bookmarkPage, the PDF browser will display the whole page, scaling it to fit the screen:

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)
```

The bookmarkPage method can be instructed to display the page in a number of different ways by providing a fit parameter.

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

Note : fit settings are case-sensitive so fit="FIT" is invalid

注意：`fit`的设置是区分大小写的，所以`fit="FIT"`是无效的。

Sometimes you want the destination of a jump to be some part of a page. The FitR fit allows you to identify a particular rectangle, scaling the area to fit the entire page.

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

To set the display to a particular x and y coordinate of the page and to control the zoom directly use fit="XYZ".

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)
```

This destination is at the leftmost of the page with the top of the screen at position 200. Because zoom was not set the zoom remains at whatever the user had it set to.

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)
```

This time zoom is set to expand the page 2X its normal size.

这次的缩放设置为将页面扩大2倍其正常大小。

Note : Both XYZ and FitR fit types require that their positional parameters (top, bottom, left, right) be specified in terms of the default user space. They ignore any geometric transform in effect in the canvas graphic state.

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。

Note: Two previous bookmark methods are supported but deprecated now that bookmarkPage is so general. These are bookmarkHorizontalAbsolute and bookmarkHorizontal.

注意：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)
```

The linkAbsolute method defines a starting point for a jump. When the user is browsing the generated document using a dynamic viewer (such as Acrobat Reader) when the mouse is clicked when the pointer is within the rectangle specified by Rect the viewer will jump to the endpoint associated with destinationname. As in the case with bookmarkHorizontalAbsolute the rectangle Rect must be specified in terms of the default user space. The contents parameter specifies a chunk of text which displays in the viewer if the user left-clicks on the region.

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。

The rectangle Rect must be specified in terms of a tuple *(x1,y1,x2, y2)* identifying the lower left and upper right points of the rectangle in default user space.

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。

For example the code

示例代码

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

defines a location as the whole of the current page with the identifier Meaning_of_life. To create a rectangular link to it while drawing a possibly different page, we would use this code:

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))
```

By default during interactive viewing a rectangle appears around the link. Use the keyword argument Border='[0 0 0]' to suppress the visible rectangle around the during viewing link. For example

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]')
```

The thickness, color and dashArray arguments may be used alternately to specify a border if no Border argument is specified. If Border is specified it must be either a string representation of a PDF array or a PDFArray (see the pdfdoc module). The color argument (which should be a Color instance) is equivalent to a keyword argument C which should resolve to a PDF color definition (Normally a three entry PDF array).

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)。

The `canvas.linkRect` method is similar in intent to the `linkAbsolute` method, but has an extra argument `relative=1` so is intended to obey the local user space transformation.

`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 has a navigation page which can hold a document outline; it should normally be visible when you open this guide. We provide some simple methods to add outline entries. Typically, a program to make a document (such as this user guide) will call the method `canvas.addOutlineEntry(`*self, title, key, level=0, closed=None*`)` as it reaches each heading in the document.

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 is the caption which will be displayed in the left pane. The *key* must be a string which is unique within the document and which names a bookmark, as with the hyperlinks. The *level* is zero - the uppermost level - unless otherwise specified, and it is an error to go down more than one level at a time (for example to follow a level 0 heading by a level 2 heading). Finally, the *closed* argument specifies whether the node in the outline pane is closed or opened by default.

*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。

The snippet below is taken from the document template that formats this user guide. A central processor looks at each paragraph in turn, and makes a new outline entry when a new chapter occurs, taking the chapter heading text as the caption text. The key is obtained from the chapter number (not shown here), so Chapter 2 has the key 'ch2'. The bookmark to which the outline entry points aims at the whole page, but it could as easily have been an individual paragraph.

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')
```

The `setPageTransition` method specifies how one page will be replaced with the next. By setting the page transition effect to "dissolve" for example the current page will appear to melt away when it is replaced by the next page during interactive viewing. These effects are useful in spicing up slide presentations, among other places. Please see the reference manual for more detail on how to use this method.

`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)
```

These methods have no automatically seen visible effect on the document. They add internal annotations to the document. These annotations can be viewed using the "Document Info" menu item of the browser and they also can be used as a simple standard way of providing basic information about the document to archiving software which need not parse the entire file. To find the annotations view the `*.pdf` output file using a standard text editor (such as `notepad` on MS/Windows or `vi` or `emacs` on unix) and look for the string `/Author` in the file contents.

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。

```python
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")
```

If you want the subject, title, and author to automatically display in the document when viewed and printed you must paint them onto the document like any other text.

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。

setAuthor, setTitle, setSubject have no visible effect

But if you are viewing this document   dynamically

please look at File/Document Info

图 4 - 1 : 设置文档内部注释

4.6 加密

关于加密PDF文件

Adobe's PDF standard allows you to do three related things to a PDF file when you encrypt it:

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。

- Apply password protection to it, so a user must supply a valid password before being able to read it,
- 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。
- Encrypt the contents of the file to make it useless until it is decrypted, and
- 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。
- Control whether the user can print, copy and paste or modify the document while viewing it.
- 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。

The PDF security handler allows two different passwords to be specified for a document:

PDF安全处理程序允许为一个文档指定两个不同的密码。

- The 'owner' password (aka the 'security password' or 'master password')
- 所有者 "密码"（也就是 "安全密码 "或 "主密码"）。
- The 'user' password (aka the 'open password')
- 用户 "密码"（也就是 "打开密码"）。

When a user supplies either one of these passwords, the PDF file will be opened, decrypted and displayed on screen.

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。

If the owner password is supplied, then the file is opened with full control - you can do anything to it, including changing the security settings and passwords, or re-encrypting it with a new password.

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。

If the user password was the one that was supplied, you open it up in a more restricted mode. The restrictions were put in place when the file was encrypted, and will either allow or deny the user permission to do the following:

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。

- Modifying the document's contents
- Copying text and graphics from the document
- Adding or modifying text annotations and interactive form fields
- Printing the document
- 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

Note that all password protected PDF files are encrypted, but not all encrypted PDFs are password protected. If a document's user password is an empty string, there will be no prompt for the password when the file is opened. If you only secure a document with the owner password, there will also not be a prompt for the password when you open the file. If the owner and user passwords are set to the same string when encrypting the PDF file, the document will always open with the user access privileges. This means that it is possible to create a file which, for example, is impossible for anyone to print out, even the person who created it.

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，任何人都不可能打印出来，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。
-	是	打开文件时需要用户密码。适用于所有人。
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。

When a PDF file is encrypted, encryption is applied to all the strings and streams in the file. This prevents people who don't have the password from simply removing the password from the PDF file to gain access to it - it renders the file useless unless you actually have the password.

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's standard encryption methods use the MD5 message digest algorithm (as described in RFC 1321, The MD5 Message-Digest Algorithm) and an encryption algorithm known as RC4. RC4 is a symmetric stream cipher - the same algorithm is used both for encryption and decryption, and the algorithm does not change the length of the data.

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

Documents can be encrypted by passing an argument to the canvas object.

文档可以通过向画布对象传递一个参数来加密。

If the argument is a string object, it is used as the User password to the PDF.

如果参数是一个字符串对象，它被用作PDF的用户密码。

The argument can also be an instance of the class `reportlab.lib.pdfencrypt.StandardEncryption`, which allows more finegrained control over encryption settings.

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。

The `StandardEncryption` constructor takes the following arguments:

`StandardEncryption`构造函数接受以下参数。

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

The `userPassword` and `ownerPassword` parameters set the relevant password on the encrypted PDF.

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

The boolean flags `canPrint`, `canModify`, `canCopy`, `canAnnotate` determine wether a user can perform the corresponding actions on the PDF when only a user password has been supplied.

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。

If the user supplies the owner password while opening the PDF, all actions can be performed regardless of the flags.

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

To create a document named hello.pdf with a user password of 'rptlab' on which printing is not allowed, use the following code:

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。

```python
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

The PDF standard allows for various kinds of interactive elements, the ReportLab toolkit currently supports only a fraction of the possibilities and should be considered a work in progress.At present we allow choices with *checkbox*, *radio*, *choice* & *listbox* widgets; text values can be entered with a *textfield* widget. All the widgets are created by calling methods on the *canvas.acroform* property.

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 & listbox widget进行选择；文本值可以使用textfield widget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

This shows the basic mechanism of creating an interactive element on the current page.

这显示了在当前页面上创建交互式元素的基本机制。

```python
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)
```

NB note that the *acroform* canvas property is created automatically on demand and that there is only one form allowd in a document.

NB请注意，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

The *canvas.acroform.checkbox* method creates a *checkbox* widget on the current page. The value of the checkbox is either **YES** or **OFF**. The arguments are

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	None
x	在页面上的水平位置(绝对坐标)	0
y	在页面上的垂直位置(绝对坐标)	0
size	轮廓尺寸：size x size	20
checked	如果为真，则该复选框被初始选中	False
buttonStyle	如果为真，则该复选框最初被选中，复选框样式（见下文）。	'check'
shape	小组件的轮廓（见下文）。	'square'
fillColor	用来填充小组件的颜色	None
textColor	符号的颜色	None
borderWidth	边框宽度	1
borderColor	小组件的边框颜色	None
borderStyle	边框样式名称	'solid'
tooltip	悬停在小组件上时要显示的文本。	None
annotationFlags	空白分隔的注解标志字符串	'print'
fieldFlags	空白分隔的字段标志（见下文）。	'required'
forceBorder	当真的时候会强行画边框	False
relative	如果为真，服从当前画布的变换	False
dashLen	如果 borderStyle=='dashed'，要使用的折线。	3

Radio Usage

单选框使用方法

The *canvas.acroform.radio* method creates a *radio* widget on the current page. The value of the radio is the value of the radio group's selected value or **OFF** if none are selected. The arguments are

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)	None
value	单选框组的名称	None
x	在页面上的水平位置(绝对坐标)	0
y	在页面上的垂直位置(绝对坐标)	0
size	轮廓尺寸，size x size	20
selected	if True this radio is the selected one in its group	False
buttonStyle	如果为 'true'，则该单选机在其组中被选中。	'check'
shape	小组件的轮廓（见下文）。	'square'
fillColor	用来填充小组件的颜色	None
textColor	符号的颜色	None
borderWidth	边框宽度	1
borderColor	边框颜色	None
borderStyle	边框样式名称	'solid'
tooltip	悬停在小组件上时要显示的文本。	None
annotationFlags	空白分隔的注解标志字符串	'print'
fieldFlags	空白分隔的字段标志（见下文）。	'noToggleToOff required radio'

canvas.acroform.radio 参数列表		
参数名	描述	默认值
forceBorder	当真的时候会强行画边框	False
relative	if true obey the current canvas transform	False
dashLen	如果borderStyle=='dashed'，要使用的折线。	3

列表框用法

The *canvas.acroform.listbox* method creates a *listbox* widget on the current page. The listbox contains a list of options one or more of which (depending on fieldFlags) may be selected.

canvas.acroform.listbox方法在当前页面上创建了一个listbox小组件。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	None
options	可用选项的列表或元组	[]
value	单个或选定选项的字符串列表。	[]
x	在页面上的水平位置(绝对坐标)	0
y	在页面上的垂直位置(绝对坐标)	0
width	小组件宽度	120
height	小组件高度	36
fontName	要使用的第1种字体的名称。	'Helvetica'
fontSize	要使用的字体大小	12
fillColor	用来填充小组件的颜色	None
textColor	符号的颜色	None
borderWidth	边框宽度	1
borderColor	边框颜色	None
borderStyle	边框样式	'solid'
tooltip	悬停在小组件上时要显示的文本。	None
annotationFlags	空白分隔的注解标志字符串	'print'
fieldFlags	空白分隔的字段标志（见下文）。	''
forceBorder	当真的时候会强行画边框	False
relative	如果为真，服从当前画布的变换	False
dashLen	如果borderStyle=='dashed'，要使用的折线。	3

下拉菜单的使用

The *canvas.acroform.choice* method creates a *dropdown* widget on the current page. The dropdown contains a list of options one or more of which (depending on fieldFlags) may be selected. If you add *edit* to the *fieldFlags* then the result may be edited.

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	None
options	可用选项的列表或元组	[]
value	单个或选定选项的字符串列表。	[]
x	在页面上的水平位置(绝对坐标)	0

canvas.acroform.choice 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)	0
width	小组件宽度	120
height	小组件高度	36
fontName	要使用的第1种字体的名称。	'Helvetica'
fontSize	要使用的字体大小	12
fillColor	用来填充小组件的颜色	None
textColor	符号的颜色	None
borderWidth	边框宽度	1
borderColor	边框颜色	None
borderStyle	边框样式	'solid'
tooltip	悬停在小组件上时要显示的文本。	None
annotationFlags	空白分隔的注解标志字符串	'print'
fieldFlags	空白分隔的字段标志（见下文）。	'combo'
forceBorder	当真的时候会强行画边框	False
relative	如果为真，服从当前画布的变换	False
dashLen	如果borderStyle=='dashed'，要使用的破折号。	3
maxlen	无或小组件值的最大长度	None

文本字段用法

The *canvas.acroform.textfield* method creates a *textfield* entry widget on the current page. The textfield may be edited to change tha value of the widget

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	None
value	文本字段的值	''
maxlen	无或小组件值的最大长度	100
x	在页面上的水平位置(绝对坐标)	0
y	在页面上的垂直位置(绝对坐标)	0
width	小组件宽度	120
height	小组件高度	36
fontName	要使用的第1种字体的名称。	'Helvetica'
fontSize	要使用的字体大小	12
fillColor	用来填充小组件的颜色	None
textColor	符号或文本的颜色	None
borderWidth	边框宽度	1
borderColor	边框颜色	None
borderStyle	边框样式	'solid'
tooltip	悬停在小组件上时要显示的文本。	None
annotationFlags	空白分隔的注解标志字符串	'print'
fieldFlags	空白分隔的字段标志（见下文）。	''
forceBorder	当真的时候会强行画边框	False
relative	如果为真，服从当前画布的变换	False

canvas.acroform.textfield 参数列表		
参数	描述	默认值
dashLen	如果borderStyle=='dashed'，要使用破折号。	3

按钮样式

The button style argument indicates what style of symbol should appear in the button when it is selected. There are several choices

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
```

note that the document renderer can make some of these symbols wrong for their intended application. Acrobat reader prefers to use its own rendering on top of what the specification says should be shown (especially when the forms hihlighting features are used

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

The shape argument describes how the outline of the checkbox or radio widget should appear you can use

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
```

The renderer may make its own decisions about how the widget should look; so Acrobat Reader prefers circular outlines for radios.

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

The borderStyle argument changes the 3D appearance of the widget on the page alternatives are

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

The fieldFlags arguments can be an integer or a string containing blank separate tokens the values are shown in the table below. For more information consult the PDF specification.

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	1<<0
required	小部件是必需的	1<<1
noExport	不要导出小组件的值	1<<2

字段标志 Tokens 和 values		
Token	描述	值
noToggleToOff	单选框组必选选择一个	1<<14
radio	单选法	1<<15
pushButton	当按钮为公共按钮时	1<<16
radiosInUnison	单选框拥有一样的值时一起切换	1<<25
multiline	用于多行文本小组件	1<<12
password	密码文本域	1<<13
fileSelect	文件选择小组件	1<<20
doNotSpellCheck	不拼写检查	1<<22
doNotScroll	文本框不滚动	1<<23
comb	根据最大长度值制作 comb 样式的文字	1<<24
richText	如果使用富文本	1<<25
combo	针对下拉框	1<<17
edit	如果选择是可编辑的	1<<18
sort	是否要对数值进行排序	1<<19
multiSelect	如果选择允许多选	1<<21
commitOnSelChange	reportlab 没有使用	1<<26

annotationFlags 参数

PDF widgets are annotations and have annotation properties these are shown in the table below

PDF小组件是注释，并具有注释属性，这些属性显示在下面的表格中。

注释标志 Tokens 和 values		
Token	描述	值
invisible	小组件不显示	1<<0
hidden	小组件隐藏	1<<1
print	小组件打印	1<<2
nozoom	注释不会随渲染页面的大小而缩放。	1<<3
norotate	小组件不会随着页面旋转。	1<<4
noview	不要渲染小组件	1<<5
readonly	小组件只读，不交互	1<<6
locked	小组件无法更改	1<<7
togglenoview	在某些事件发生后，可以展示该小部件。	1<<8
lockedcontents	小部件的内容是固定的	1<<9

第 5 章 PLATYPUS - 使用脚本进行页面布局和排版。

5.1 设计目标

Platypus stands for "Page Layout and Typography Using Scripts". It is a high level page layout library which lets you programmatically create complex documents with a minimum of effort.

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。

The design of Platypus seeks to separate "high level" layout decisions from the document content as much as possible. Thus, for example, paragraphs are constructed using paragraph styles and pages are constructed using page templates with the intention that hundreds of documents with thousands of pages can be reformatted to different style specifications with the modifications of a few lines in a single shared file which contains the paragraph styles and page layout specifications.

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。

The overall design of Platypus can be thought of has having several layers, top down, these are

Platypus的整体设计可以认为有几个层次，自上而下，这些是

`DocTemplates` the outermost container for the document;

`DocTemplates`作为最外层容器。

`PageTemplates` specifications for layouts of pages of various kinds;

`PageTemplates`各种页面布局的规格。

`Frames` specifications of regions in pages that can contain flowing text or graphics.

`Frames`页面中可包含流动文本或图形的区域规格。

`Flowables` text or graphic elements that should be "flowed into the document (i.e. things like images, paragraphs and tables, but not things like page footers or fixed page graphics).

`Flowables`对应 "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` the lowest level which ultimately receives the painting of the document from the other layers.

`pdfgen.Canvas`为最终从其他图层接收文档绘画的最低层。

DocTemplate

| two column | chapter page | title page |
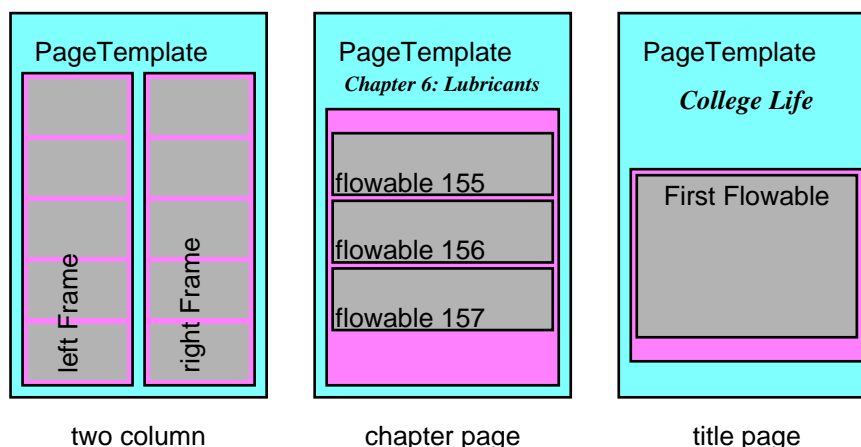
图 5 - 1 : DocTemplate 结构说明

The illustration above graphically illustrates the concepts of `DocTemplates`, `PageTemplates` and `Flowables`. It is deceptive, however, because each of the `PageTemplates` actually may specify the format for any number of pages (not just one as might be inferred from the diagram).

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）。

`DocTemplates` contain one or more `PageTemplates` each of which contain one or more `Frames`. `Flowables` are things which can be *flowed* into a `Frame` e.g. a `Paragraph` or a `Table`.

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

To use platypus you create a document from a `DocTemplate` class and pass a list of `Flowables` to its `build` method. The document `build` method knows how to process the list of flowables into something reasonable.

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。

Internally the `DocTemplate` class implements page layout and formatting using various events. Each of the events has a corresponding handler method called `handle_XXX` where XXX is the event name. A typical event is `frameBegin` which occurs when the machinery begins to use a frame for the first time.

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`，它发生在程序开始第一次使用一个框架的时候。

A Platypus story consists of a sequence of basic elements called `Flowables` and these elements drive the data driven Platypus formatting engine. To modify the behavior of the engine a special kind of flowable, `ActionFlowables`, tell the layout engine to, for example, skip to the next column or change to another `PageTemplate`.

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

Consider the following code sequence which provides a very simple "hello world" example for Platypus.

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。

```python
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()
```

First we import some constructors, some paragraph styles and other conveniences from other modules.

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。

```python
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()
```

We define the fixed features of the first page of the document with the function above.

我们用上面的函数定义文档首页的固定特征。

```python
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()
```

Since we want pages after the first to look different from the first we define an alternate layout for the fixed features of the other pages. Note that the two functions above use the pdfgen level canvas operations to paint the annotations for the pages.

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canvas(画布)操作来为页面绘制注释。

```python
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)
```

Finally, we create a story and build the document. Note that we are using a "canned" document template here which comes pre-built with page templates. We are also using a pre-built paragraph style. We are only using two types of flowables here -- Spacers and Paragraphs. The first Spacer ensures that the Paragraphs skip past the title string.

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"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。

To see the output of this example program run the module docs/userguide/examples.py (from the ReportLab docs distribution) as a "top level script". The script interpretation python examples.py will generate the Platypus output phello.pdf.

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/examples.py（来自ReportLab docs发行版）。 脚本解释 python examples.py 将生成 Platypus输出 phello.pdf。

5.3 Flowables

Flowables are things which can be drawn and which have wrap, draw and perhaps split methods. Flowable is an abstract base class for things to be drawn and an instance knows its size and draws in its own coordinate system (this requires the base API to provide an absolute coordinate system when the Flowable.draw method is called). To get an instance use f=Flowable().

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()。

It should be noted that the Flowable class is an *abstract* class and is normally only used as a base class.

需要注意的是，Flowable类是一个抽象类，通常只作为基类使用。

To illustrate the general way in which Flowables are used we show how a derived class Paragraph is used and drawn on a canvas. Paragraphs are so important they will get a whole chapter to themselves.

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。

```python
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 用户方法

```
Flowable.draw()
```

This will be called to ask the flowable to actually render itself. The Flowable class does not implement draw. The calling code should ensure that the flowable has an attribute canv which is the pdfgen.Canvas which should be drawn to an that the Canvas is in an appropriate state (as regards translations rotations, etc). Normally this method will only be called internally by the drawOn method. Derived classes must implement this method.

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)
```

This is the method which controlling programs use to render the flowable to a particular canvas. It handles the translation to the canvas coordinate (*x*,*y*) and ensuring that the flowable has a canv attribute so that the draw method (which is not implemented in the base class) can render in an absolute coordinate frame.

这是控制程序用来将 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)

This will be called by the enclosing frame before objects are asked their size, drawn or whatever. It returns the size actually used.

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)

This will be called by more sophisticated frames when wrap fails. Stupid flowables should return [] meaning that they are unable to split. Clever flowables should split themselves and return a list of flowables. It is up to the client code to ensure that repeated attempts to split are avoided. If the space is sufficient the split method should return [self]. Otherwise the flowable should rearrange itself and return a list [f0, ...] of flowables which will be considered in order. The implemented split method should avoid changing `self` as this will allow sophisticated layout mechanisms to do multiple passes over a list of flowables.

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

Two methods, which by default return zero, provide guidance on vertical spacing of flowables:

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):

These methods return how much space should follow or precede the flowable. The space doesn't belong to the flowable itself i.e. the flowable's `draw` method shouldn't consider it when rendering. Controlling programs will use the values returned in determining how much space is required by a particular flowable in context.

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。

All flowables have an hAlign property: ('LEFT', 'RIGHT', 'CENTER' or 'CENTRE'). For paragraphs, which fill the full width of the frame, this has no effect. For tables, images or other objects which are less than the width of the frame, this determines their horizontal placement.

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，这决定了它们的水平位置。

The chapters which follow will cover the most important specific types of flowables: Paragraphs and Tables.

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 are active containers which are themselves contained in PageTemplates. Frames have a location and size and maintain a concept of remaining drawable space. The command

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)
```

creates a Frame instance with lower left hand corner at coordinate (x1,y1) (relative to the canvas at use time) and with dimensions width x height. The Padding arguments are positive quantities used to reduce the space available for drawing. The id argument is an identifier for use at runtime e.g. 'LeftColumn' or 'RightColumn' etc. If the showBoundary argument is non-zero then the boundary of the frame will get drawn at run time (this is useful sometimes).

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 用户方法

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

consumes Flowables from the front of drawlist until the frame is full. If it cannot fit one object, raises an exception.

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)
```

Asks the flowable to split using up the available space and return the list of flowables.

要求flowable使用可用空间进行分割，并返回flowable的列表。

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

draws the frame boundary as a rectangle (primarily for debugging).

将框架边界画成一个矩形（主要用于调试）。

使用 Frames

Frames can be used directly with canvases and flowables to create documents. The Frame.addFromList method handles the wrap & drawOn calls for you. You don't need all of the Platypus machinery to get something useful into PDF.

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机器来获得有用的东西到PDF中。

```python
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

The `BaseDocTemplate` class implements the basic machinery for document formatting. An instance of the class contains a list of one or more `PageTemplates` that can be used to describe the layout of information on a single page. The `build` method can be used to process a list of `Flowables` to produce a **PDF** document.

`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` 类。

```
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)
```

creates a document template suitable for creating a basic document. It comes with quite a lot of internal machinery, but no default page templates. The required `filename` can be a string, the name of a file to receive the created **PDF** document; alternatively it can be an object which has a `write` method such as a `BytesIO` or `file` or `socket`.

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，但没有默认的页面模板。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`。

The allowed arguments should be self explanatory, but `showBoundary` controls whether or not `Frame` boundaries are drawn which can be useful for debugging purposes. The `allowSplitting` argument determines whether the builtin methods should try to *split* individual `Flowables` across `Frames`. The `_pageBreakQuick` argument determines whether an attempt to do a page break should try to end all the frames on the page or not, before ending the page. The encrypt argument determines wether or not and how the document is encrypted. By default, the document is not encrypted. If `encrypt` is a string object, it is used as the user password for the pdf. If `encrypt` is an instance of `reportlab.lib.pdfencrypt.StandardEncryption`, this object is used to encrypt the pdf. This allows more finegrained control over the encryption settings.

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` 用户方法

These are of direct interest to client programmers in that they are normally expected to be used.

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)
```

This method is used to add one or a list of `PageTemplates` to an existing documents.

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)

This is the main method which is of interest to the application programmer. Assuming that the document instance is correctly set up the `build` method takes the *story* in the shape of the list of flowables (the `flowables` argument) and loops through the list forcing the flowables one at a time through the formatting machinery. Effectively this causes the `BaseDocTemplate` instance to issue calls to the instance `handle_XXX` methods to process the various events.

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` 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` 用户虚拟方法

These have no semantics at all in the base class. They are intended as pure virtual hooks into the layout machinery. Creators of immediately derived classes can override these without worrying about affecting the properties of the layout engine.

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)

This is called after initialisation of the base class; a derived class could overide the method to add default `PageTemplates`.

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)

This is called after page processing, and immediately after the afterDrawPage method of the current page template. A derived class could use this to do things which are dependent on information in the page such as the first and last word on the page of a dictionary.

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)

This is called before any processing is done on the document, but after the processing machinery is ready. It can therefore be used to do things to the instance's `pdfgen.canvas` and the like.

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)

This is called at the beginning of page processing, and immediately before the beforeDrawPage method of the current page template. It could be used to reset page specific information holders.

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)

This is called to filter flowables at the start of the main handle_flowable method. Upon return if flowables[0] has been set to None it is discarded and the main method returns immediately.

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)

Called after a flowable has been rendered. An interested class could use this hook to gather information about what information is present on a particular page or frame.

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` 事件处理方法

These methods constitute the greater part of the layout engine. Programmers shouldn't have to call or override these methods directly unless they are trying to modify the layout engine. Of course, the experienced programmer who wants to intervene at a particular event, XXX, which does not correspond to one of the virtual methods can always override and call the base method from the drived class version. We make this easy by providing a base class synonym for each of the handler methods with the same name prefixed by an underscore '_'.

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。

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

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

Here we list the methods only as an indication of the events that are being handled. Interested programmers can take a look at the source.

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。

```python
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)
```

Using document templates can be very easy; `SimpleDoctemplate` is a class derived from `BaseDocTemplate` which provides its own `PageTemplate` and `Frame` setup.

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。

```python
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

The `PageTemplate` class is a container class with fairly minimal semantics. Each instance contains a list of `Frames` and has methods which should be called at the start and end of each page.

`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)

is used to initialize an instance, the `frames` argument should be a list of `Frames` whilst the optional `onPage` and `onPageEnd` arguments are callables which should have signature `def XXX(canvas,document)` where `canvas` and `document` are the canvas and document being drawn. These routines are intended to be used to paint non-flowing (i.e. standard) parts of pages. These attribute functions are exactly parallel to the pure virtual methods `PageTemplate.beforPage` and `PageTemplate.afterPage` which have signature `beforPage(self,canvas,document)`. The methods allow class derivation to be used to define standard behaviour, whilst the attributes allow instance changes. The `id` argument is used at run time to perform `PageTemplate` switching so `id='FirstPage'` or `id='TwoColumns'` are typical.

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.beforPage` 和 `PageTemplate.afterPage`完全平行，这两个方法的签名是 `beforPage(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 文字段落

The `reportlab.platypus.Paragraph` class is one of the most useful of the Platypus `Flowables`; it can format fairly arbitrary text and provides for inline font style and colour changes using an XML style markup. The overall shape of the formatted text can be justified, right or left ragged or centered. The XML markup can even be used to insert greek characters or to do subscripts.

`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。

The following text creates an instance of the `Paragraph` class:

下面的文字创建了一个`Paragraph`类的实例。

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

The `text` argument contains the text of the paragraph; excess white space is removed from the text at the ends and internally after linefeeds. This allows easy use of indented triple quoted text in **Python** scripts. The `bulletText` argument provides the text of a default bullet for the paragraph. The font and other properties for the paragraph text and bullet are set using the style argument.

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。

The `style` argument should be an instance of class `ParagraphStyle` obtained typically using

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

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

this container class provides for the setting of multiple default paragraph attributes in a structured way. The styles are arranged in a dictionary style object called a `stylesheet` which allows for the styles to be accessed as `stylesheet['BodyText']`. A sample style sheet is provided.

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']
```

The options which can be set for a `Paragraph` can be seen from the `ParagraphStyle` defaults. The values with leading underscore ('_') are derived from the defaults in module `reportlab.rl_config` which are derived from module `reportlab.rl_settings`.

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。

class ParagraphStyle

类 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,
}
```

6.1 使用文字段落样式

The `Paragraph` and `ParagraphStyle` classes together handle most common formatting needs. The following examples draw paragraphs in various styles, and add a bounding box so that you can see exactly what space is taken up.

`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 = SourceHanSansSC
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 0
fontName = SourceHanSansSC
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 6-1: 默认 `ParagraphStyle`

The two attributes `spaceBefore` and `spaceAfter` do what they say, except at the top or bottom of a frame. At the top of a frame, `spaceBefore` is ignored, and at the bottom, `spaceAfter` is ignored. This means that you could specify that a 'Heading2' style had two inches of space before when it occurs in mid-page, but will not get acres of whitespace at the top of a page. These two attributes should be thought of as 'requests' to the Frame and are not part of the space occupied by the Paragraph itself.

`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'Heading2'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对Frame的 "请求"，而不是段落本身所占空间的一部分。

The `fontSize` and `fontName` tags are obvious, but it is important to set the `leading`. This is the spacing between adjacent lines of text; a good rule of thumb is to make this 20% larger than the point size. To get double-spaced text, use a high `leading`. If you set `autoLeading`(default `"off"`) to `"min"`(use observed leading even if smaller than specified) or `"max"`(use the larger of observed and specified) then an attempt is made to determine the leading on a line by line basis. This may be useful if the lines contain different font sizes etc.

`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。

The figure below shows space before and after and an increased leading:

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 = SourceHanSansSC
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 0
fontName = SourceHanSansSC
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

The attribute `borderPadding` adjusts the padding between the paragraph and the border of its background. This can either be a single value or a tuple containing 2 to 4 values. These values are applied the same way as in Cascading Style Sheets (CSS). If a single value is given, that value is applied to all four sides. If more than one value is given, they are applied in clockwise order to the sides starting at the top. If two or three values are given, the missing values are taken from the opposite side(s). Note that in the following example the yellow box is drawn by the paragraph itself.

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 = SourceHanSansSC
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 0
fontName = SourceHanSansSC
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`

The `leftIndent` and `rightIndent` attributes do exactly what you would expect; `firstLineIndent` is added to the `leftIndent` of the first line. If you want a straight left edge, remember to set `firstLineIndent` equal to 0.

`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 = SourceHanSansSC
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 24
fontName = SourceHanSansSC
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

Setting `firstLineIndent` equal to a negative number, `leftIndent` much higher, and using a different font (we'll show you how later!) can give you a definition list:.

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 = SourceHanSansSC
bulletFontSize = 10
bulletIndent = 0
embeddedHyphenation = 0
endDots = None
firstLineIndent = 0
fontName = SourceHanSansSC
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: 定义列表

There are four possible values of `alignment`, defined as constants in the module *reportlab.lib.enums*. These are TA_LEFT, TA_CENTER or TA_CENTRE, TA_RIGHT and TA_JUSTIFY, with values of 0, 1, 2 and 4 respectively. These do exactly what you would expect.

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。

Set `wordWrap` to `'CJK'` to get Asian language linewrapping. For normal western text you can change the way the line breaking algorithm handles *widows* and *orphans* with the `allowWidows` and `allowOrphans` values. Both should normally be set to 0, but for historical reasons we have allowed *widows*. The default color of the text can be set with `textColor` and the paragraph background colour can be set with `backColor`. The paragraph's border properties may be changed using `borderWidth`, `borderPadding`, `borderColor` and `borderRadius`.

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。

The `textTransform` attribute can be **None**, *'uppercase'* or *'lowercase'* to get the obvious result and *'capitalize'* to get initial letter capitalization.

`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。

Attribute `endDots` can be **None**, a string, or an object with attributes text and optional fontName, fontSize, textColor, backColor and dy(y offset) to specify trailing matter on the last line of left/right justified paragraphs.

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。

The `splitLongWords` attribute can be set to a false value to avoid splitting very long words.

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

Attribute `bulletAnchor` can be *'start'*, *'middle'*, *'end'* or *'numeric'* to control where the bullet is anchored.

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

The `justifyBreaks` attribute controls whether lines deliberately broken with a `<br/>` tag should be justified

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

Attribute `spaceShrinkage` is a fractional number specifiying by how much the space of a paragraph line may be shrunk in order to make it fit; typically it is something like 0.05

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。

The `underlineWidth`, `underlineOffset`, `underlineGap` & `underlineColor` attributes control the underline behaviour when the `<u>` or a linking tag is used. Those tags can have override values of these attributes. The attribute value for width & offset is a `fraction * Letter` where letter can be one of P, L, f or F representing fontSize proportions. P uses the fontsize at the tag, F is the maximum fontSize in the tag, f is the initial fontsize inside the tag. L means the global (paragrpah style) font size. `strikeWidth`, `strikeOffset`, `strikeGap` & `strikeColor` attributes do the same for strikethrough lines.

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。

Attribute `linkUnderline` controls whether link tags are automatically underlined.

属性 `linkUnderline` 控制链接标签是否自动下划线。

If the `pyphen` python module is installed attribute `hyphenationLang` controls which language will be used to hyphenate words without explicit embedded hyphens.

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。

If `embeddedHyphenation` is set then attempts will be made to split words with embedded hyphens.

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。

Attribute `uriWasteReduce` controls how we attempt to split long uri's. It is the fraction of a line that we regard as too much waste. The default in module `reportlab.rl_settings` is *0.5* which means that we will try and split a word that looks like a uri if we would waste at least half of the line.

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。

Currently the hyphenation and uri splitting are turned off by default. You need to modify the default settings by using the file ~/.rl_settings or adding a module reportlab_settings.py to the python path. Suitable values are

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 markup can be used to modify or specify the overall paragraph style, and also to specify intra-paragraph markup.

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

The paragraph text may optionally be surrounded by <para attributes....> </para> tags. The attributes if any of the opening <para> tag affect the style that is used with the Paragraph text and/or bulletText.

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
alignment	align, alignment
allowOrphans	allowOrphans, alloworphans
allowWidows	allowWidows, allowwidows
autoLeading	autoLeading, autoleading
backColor	backColor, backcolor, bg, bgcolor
borderColor	borderColor, bordercolor
borderRadius	borderRadius, borderradius
borderWidth	borderWidth, borderwidth
borderpadding	borderpadding
bulletAnchor	banchor, bulletAnchor, bulletanchor
bulletColor	bcolor, bulletColor, bulletcolor
bulletFontName	bfont, bulletFontName, bulletfontname
bulletFontSize	bfontsize, bulletFontSize, bulletfontsize
bulletIndent	bindent, bulletIndent, bulletindent
bulletOffsetY	boffsety, bulletOffsetY, bulletoffsety
embeddedHyphenation	embeddedHyphenation, embeddedhyphenation
endDots	endDots, enddots
firstLineIndent	findent, firstLineIndent, firstlineindent
fontName	face, font, fontName, fontname
fontSize	fontSize, fontsize, size
hyphenationLang	hyphenationLang, hyphenationLanguage, hyphenationlang
hyphenationMinWordLength	hyphenationMinWordLength, hyphenationminwordlength
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 6-2 - 样式属性的同义词

Some useful synonyms have been provided for our Python attribute names, including lowercase versions, and the equivalent properties from the HTML standard where they exist. These additions make it much easier to build XML-printing applications, since much intra-paragraph markup may not need translating. The table below shows the allowed attributes and synonyms in the outermost paragraph tag.

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[Within each paragraph, we use a basic set of XML tags to provide markup. The most basic of these are bold (...), italic (<i>...</i>) and underline (<u>...</u>). Other tags which are allowed are strong (...), and strike through (<strike>...</strike>). The <link> and <a> tags may be used to refer to URIs, documents or bookmarks in the current document. The a variant of the <a> tag can be used to mark a position in a document. A break (<br/>) tag is also allowed.]]>

<! [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 (<br/>)标签。]]>。

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。 <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

这是一个锚标签的链接，即<a name="MYANCHOR"/>这里。这是另一个指向同一锚标签的<link href="#MYANCHOR" color="blue" fontName="SourceHanSansSC">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

The **link** tag can be used as a reference, but not as an anchor. The a and link hyperlink tags have additional attributes *fontName*, *fontSize*, *color* & *backColor* attributes. The hyperlink reference can have a scheme of **http:**(*external webpage*), **pdf:**(*different pdf document*) or **document:**(*same pdf document*); a missing scheme is treated as **document** as is the case when the reference starts with # (in which case the anchor should omit it). Any other scheme is treated as some kind of URI.

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 backColor属性。超链接引用可以有http:（外部网页）、pdf:（不同的pdf文档）或document:（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。

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？

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？

图 6-8: 强调, 删除线, 和换行标签

 标签

The tag can be used to change the font name, size and text color for any substring within the paragraph. Legal attributes are size, face, name (which is the same as face), color, and fg (which is the same as color). The name is the font family name, without any 'bold' or 'italic' suffixes. Colors may be HTML color names or a hex string encoded in a variety of ways; see *reportlab.lib.colors* for the formats allowed.

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。

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？

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？

图 6-9: font 标签

上标和下标

Superscripts and subscripts are supported with the <![CDATA[<super>/<sup> and <sub> tags, which work exactly as you might expect. Additionally these three tags have attributes rise and size to optionally set the rise/descent and font size for the superscript/subscript text. In addition, most greek letters can be accessed by using the <greek></greek> tag, or with mathML entity names.]]>

上标和下标是由<![CDATA[<super>/<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></greek>标签，或者使用mathML实体名来访问。]]>。

```
Equation (&alpha;):
<greek>e</greek> <super rise=9
size=6><greek>ip</greek></super>
= -1
```

Equation (α): $\varepsilon^{\iota\pi} = -1$

Figure 6-10: Greek letters and superscripts

等式（α）。<greek>e</greek> <super
rise=9
size=6><greek>ip</greek></super>=-1。

等式（α）。$\varepsilon^{\iota\pi} = -1$。

图 6-11: 希腊字母和上标

内联图片

We can embed images in a paragraph with the <img/> tag which has attributes `src`, `width`, `height` whose meanings are obvious. The `valign` attribute may be set to a css like value from "baseline", "sub", "super", "top", "text-top", "middle", "bottom", "text-bottom"; the value may also be a numeric percentage or an absolute value.

我们可以用<img/>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。`valign`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; <img src="images/testimg.gif"
valign="top"/> is aligned
<b>top</b>.<br/><br/>This &lt;img/&gt;
<img src="images/testimg.gif"
valign="bottom"/> is aligned
<b>bottom</b>.<br/><br/> This &lt;img/&gt;
<img src="images/testimg.gif"
valign="middle"/> is aligned
<b>middle</b>.<br/><br/>This &lt;img/&gt;
<img src="images/testimg.gif" valign="-4"/>
is aligned <b>-4</b>.<br/><br/>This
&lt;img/&gt; <img src="images/testimg.gif"
valign="+4"/> is aligned
<b>+4</b>.<br/><br/>This &lt;img/&gt; <img
src="images/testimg.gif" width="10"/> has
width<b>10</b>.<br/><br/></para>
```

This <img/> text is aligned top.

This <img/> text is aligned bottom.

This <img/> text is aligned middle.

This <img/> text is aligned -4.

This <img/> text is aligned +4.

This <img/> text has width10.

图 6-12: 内联图片

The `src` attribute can refer to a remote location eg `src="https://www.reportlab.com/images/logo.gif"`. By default we set `rl_config.trustedShemes` to `['https','http', 'file', 'data', 'ftp']` and `rl_config.trustedHosts=None` the latter meaning no-restriction. You can modify these variables using one of the override files eg `reportlab_settings.py` or `~/.reportlab_settings`. Or as comma separated strings in the environment variables `RL_trustedSchemes` & `RL_trustedHosts`. Note that the `trustedHosts` values may contain **glob** wild cars so *.reportlab.com* will match the obvious domains.
NB use of *trustedHosts* and or *trustedSchemes* may not control behaviour & actions when `URI` patterns are detected by the viewer application.

`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.trustedShemes` 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

These tags can be used to carry out explicit underlineing or strikethroughs. These tags have attributes `width`, `offset`, `color`, `gap` & `kind`. The `kind` attribute controls how many lines will be drawn (default `kind=1`) and when `kind>1` the `gap` attribute controls the disatnce between lines.

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

If hyphenation is in operation the `<nobr>` tag suppresses it so `<nobr>averylongwordthatwontbebroken</nobr>` won't be broken.

如果使用连字符，`<nobr>`标签会抑制它，所以`<nobr>`■■■■■■■■■■■■■`</nobr>`不会被打断。

文字段落和列表的编号

The `<seq>` tag provides comprehensive support for numbering lists, chapter headings and so on. It acts as an interface to the `Sequencer` class in *reportlab.lib.sequencer*. These are used to number headings and figures throughout this document. You may create as many separate 'counters' as you wish, accessed with the `id` attribute; these will be incremented by one each time they are accessed. The `seqreset` tag resets a counter. If you want it to resume from a number other than 1, use the syntax <seqreset id="mycounter" base="42">. Let's have a go:

`<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 6-13: 基本序列

You can save specifying an ID by designating a counter ID as the *default* using the <seqdefault id="Counter"> tag; it will then be used whenever a counter ID is not specified. This saves some typing, especially when doing multi-level lists; you just change counter ID when stepping in or out a level.

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.

Figure 6-14: The default sequence

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

Continued... 11,12, 13, 14, 15, 16, 17.

图 6-15: 默认序列

Finally, one can access multi-level sequences using a variation of Python string formatting and the `template` attribute in a <seq> tags. This is used to do the captions in all of the figures, as well as the level two headings. The substring `%(counter)s` extracts the current value of a counter without incrementing it; appending a plus sign as in `%(counter)s` increments the counter. The figure captions use a pattern like the one below:

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 6-16: 多级模板

We cheated a little - the real document used 'Figure', but the text above uses 'FigureNo' - otherwise we would have messed up our numbering!

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

In addition to the three indent properties, some other parameters are needed to correctly handle bulleted and numbered lists. We discuss this here because you have now seen how to handle numbering. A paragraph may have an optional *bulletText* argument passed to its constructor; alternatively, bullet text may be placed in a tag at its head. This text will be drawn on the first line of the paragraph, with its x origin determined by the `bulletIndent` attribute of the style, and in the font given in the `bulletFontName` attribute. The "bullet" may be a single character such as (doh!) a bullet, or a fragment of text such as a number in some numbering sequence, or even a short title as used in a definition list. Fonts may offer various bullet characters but we suggest first trying the Unicode bullet (•), which may be written as `•`, `•` or (in utf8) `\xe2\x80\xa2`):

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（嘟！）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 6-3 <bullet> 属性和同义词

The <bullet> tag is only allowed once in a given paragraph and its use overrides the implied bullet style and *bulletText* specified in the *Paragraph* creation.

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 = SourceHanSansSC
fontSize = 10
hyphenationLang =
justifyBreaks = 0
justifyLastLine = 0
leading = 12
leftIndent = 54
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 spam spamspam spam spam spam spam spam spam spam spam spam
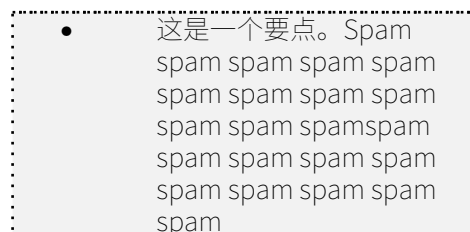
图 6-17: 项目点的基本使用

Exactly the same technique is used for numbers, except that a sequence tag is used. It is also possible to put a multi-character string in the bullet; with a deep indent and bold bullet font, you can make a compact definition list.

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

The `Table` and `LongTable` classes derive from the `Flowable` class and are intended as a simple textual gridding mechanisms. The `longTable` class uses a greedy algorithm when calculating column widths and is intended for long tables where speed counts. `Table` cells can hold anything which can be converted to a **Python** `string` or `Flowables` (or lists of `Flowables`).

`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。

Our present tables are a trade-off between efficient drawing and specification and functionality. We assume the reader has some familiarity with HTML tables. In brief, they have the following characteristics:

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。

- They can contain anything convertible to a string; flowable objects such as other tables; or entire sub-stories
- They can work out the row heights to fit the data if you don't supply the row height. (They can also work out the widths, but generally it is better for a designer to set the width manually, and it draws faster).
- They can split across pages if needed (see the canSplit attribute). You can specify that a number of rows at the top and bottom should be repeated after the split (e.g. show the headers again on page 2,3,4...)
- They have a simple and powerful notation for specifying shading and gridlines which works well with financial or database tables, where you don't know the number of rows up front. You can easily say 'make the last row bold and put a line above it'
- The style and data are separated, so you can declare a handful of table styles and use them for a family of reports. Styes can also 'inherit', as with paragraphs.
- 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。

There is however one main limitation compared to an HTML table. They define a simple rectangular grid. There is no simple row or column spanning; if you need to span cells, you must nest tables inside table cells instead or use a more complex scheme in which the lead cell of a span contains the actual contents.

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。

`Tables` are created by passing the constructor an optional sequence of column widths, an optional sequence of row heights, and the data in row order. Drawing of the table can be controlled by using a `TableStyle` instance. This allows control of the color and weight of the lines (if any), and the font, alignment and padding of the text. A primitive automatic row height and or column width calculation mechanism is provided for.

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` 用户方法

These are the main methods which are of interest to the client programmer.

这些是客户端程序员感兴趣的主要方法。

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

The `data` argument is a sequence of sequences of cell values each of which should be convertible to a string value using the `str` function or should be a Flowable instance (such as a `Paragraph`) or a list (or tuple) of such instances. If a cell value is a `Flowable` or list of `Flowables` these must either have a determined width or the containing column must have a fixed width. The first row of cell values is in `data[0]` i.e. the values are in row order. The i, j$^{th.}$ cell value is in `data[i][j]`. Newline characters `'\n'` in cell values are treated as line split characters and are used at *draw* time to format the cell into lines.

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, j$^{th.}$ 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。

The other arguments are fairly obvious, the `colWidths` argument is a sequence of numbers or possibly `None`, representing the widths of the columns. The number of elements in `colWidths` determines the number of columns in the table. A value of `None` means that the corresponding column width should be calculated automatically.

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。

The `rowHeights` argument is a sequence of numbers or possibly `None`, representing the heights of the rows. The number of elements in `rowHeights` determines the number of rows in the table. A value of `None` means that the corresponding row height should be calculated automatically.

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。

The `style` argument can be an initial style for the table.

参数 `style` 可以是表的初始样式。

The `splitByRow` argument is only needed for tables both too tall and too wide to fit in the current context. In this case you must decide whether to 'tile' down and across, or across and then down. This parameter is a Boolean indicating that the `Table` should split itself by row before attempting to split itself by column when too little space is available in the current drawing area and the caller wants the `Table` to split. Splitting a `Table` by column is currently not implemented, so setting `splitByRow` to `False` will result in a `NotImplementedError`.

`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`NotImplementedError`。

The `repeatRows` argument specifies the number or a tuple of leading rows that should be repeated when the `Table` is asked to split itself. If it is a tuple it should specify which of the leading rows should be repeated; this allows for cases where the first appearance of the table hsa more leading rows than later split parts. The `repeatCols` argument is currently ignored as a `Table` cannot be split by column.

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。

The spaceBefore & spaceAfter arguments may be used to put extra space before or after the table when renedered in a platypus story.

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。

The rowSplitRange argument may be used to control the splitting of the table to a subset of its rows; that can be to prevent splitting too close to the beginning or end of the table.

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)

This method applies a particular instance of class TableStyle (discussed below) to the Table instance. This is the only way to get tables to appear in a nicely formatted way.

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。

Successive uses of the setStyle method apply the styles in an additive fashion. That is, later applications override earlier ones where they overlap.

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

This class is created by passing it a sequence of *commands*, each command is a tuple identified by its first element which is a string; the remaining elements of the command tuple represent the start and stop cell coordinates of the command and possibly thickness and colors, etc.

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 用户方法

TableStyle(commandSequence)

The creation method initializes the TableStyle with the argument command sequence as an example:

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)

This method allows you to add commands to an existing TableStyle, i.e. you can build up TableStyles in multiple statements.

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()
```

This method returns the sequence of commands of the instance.

此方法返回实例的命令序列。

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

7.4 `TableStyle` **Commands**

The commands passed to `TableStyles` come in three main groups which affect the table background, draw lines, or set cell styles.

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。

The first element of each command is its identifier, the second and third arguments determine the cell coordinates of the box of cells which are affected with negative coordinates counting backwards from the limit values as in **Python** indexing. The coordinates are given as (column, row) which follows the spreadsheet 'A1' model, but not the more natural (for mathematicians) 'RC' ordering. The top left cell is (0, 0) the bottom right is (-1, -1). Depending on the command various extra (???) occur at indices beginning at 3 on.

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*

The cell formatting commands all begin with an identifier, followed by the start and stop cell definitions and the perhaps other arguments. the cell formatting commands are:

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。

```
FONT            - 字体名称, 可选字体大小和字符间距(leading).
FONTNAME (or FACE)    - 字体名称.
FONTSIZE (or 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 (or 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。
```

This sets the background cell color in the relevant cells. The following example shows the `BACKGROUND`, and `TEXTCOLOR` commands in action:

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。

```
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)]))
```

produces

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

To see the effects of the alignment styles we need some widths and a grid, but it should be easy to see where the styles come from.

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。

```
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),
                       ]))
```

produces

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

TableStyle 行的命令集

Line commands begin with the identifier, the start and stop cell coordinates and always follow this with the thickness (in points) and color of the desired lines. Colors can be names, or they can be specified as a (R, G, B) tuple, where R, G and B are floats and (0, 0, 0) is black. The line command names are: GRID, BOX, OUTLINE, INNERGRID, LINEBELOW, LINEABOVE, LINEBEFORE and LINEAFTER. BOX and OUTLINE are equivalent, and GRID is the equivalent of applying both BOX and INNERGRID.

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。

We can see some line commands in action with the following example.

我们可以通过下面的例子看到一些行命令的作用。

```
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),
                    ])
```

produces

Line commands cause problems for tables when they split; the following example shows a table being split in various positions

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

```
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),
                    ])
```

produces

When unsplit and split at the first or second row.

当在第一行或第二行进行拆分和分割时。

复杂的单元格值

As mentioned above we can have complicated cell values including `Paragraphs`, `Images` and other `Flowables` or lists of the same. To see this in operation consider the following code and the table it produces. Note that the `Image` has a white background which will obscure any background you choose for the cell. To get better results you should use a transparent background.

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。

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

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

t._argW[3]=1.5*inch
```

produces

A	B	C	A pa**r**a*graph* [1]	D
00	01	02	The **ReportLab Left** **Logo** Image	04

			The **ReportLab Left** **Logo** Image	
10	11	12		14
20	21	22	23	24
30	31	32	33	34

TableStyle 跨单元格命令

Our `Table` classes support the concept of spanning, but it isn't specified in the same wayas html. The style specification

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)
```

indicates that the cells in columns `sc` - `ec` and rows `sr` - `er` should be combined into a super cell with contents determined by the cell `(sc, sr)`. The other cells should be present, but should contain empty strings or you may get unexpected results.

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。

```
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)),
            ])
```

produces

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

notice that we don't need to be conservative with our `GRID` command. The spanned cells are not drawn through.

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

To control `Table` splitting the `NOSPLIT` command may be used The style specification

要控制 `Table` 的拆分，可以使用 `NOSPLIT` 命令。

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

demands that the cells in columns `sc` - `ec` and rows `sr` - `er` may not be split.

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

In any style command the first row index may be set to one of the special strings `'splitlast'` or `'splitfirst'` to indicate that the style should be used only for the last row of a split table, or the first row of a continuation. This allows splitting tables with nicer effects around the split.

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 编写 `Flowables`

The following flowables let you conditionally evaluate and execute expressions and statements at wrap time:

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

Assigns a variable of name `var` to the expression `expr`. E.g.:

为表达式`expr`指定一个名称为`var`的变量。例如：

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

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

Executes the statement `stmt`. E.g.:

执行语句`stmt`。例如：

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

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

Creates a paragraph with the value of expr as text. If format is specified it should use %(__expr__)s for string interpolation of the expression expr (if any). It may also use %(name)s interpolations for other variables in the namespace. E.g.:

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(self, cond, format=None)`

Raises an `AssertionError` containing the `format` string if `cond` evaluates as `False`.

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

If `cond` evaluates as `True`, this flowable is replaced by the `thenBlock` elsethe `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(self, cond, whileBlock)`

Runs the `whileBlock` while `cond` evaluates to `True`. E.g.:

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 其他可用的`Flowables`

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

Creates a preformatted paragraph which does no wrapping, line splitting or other manipulations. No XML style tags are taken account of in the text. If dedent is non zero `dedent` common leading spaces will be removed from the front of each line.

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

You can use the property `maxLineLength` to define a maximum line length. If a line length exceeds this maximum value, the line will be automatically splitted.

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。

The line will be split on any single character defined in `splitChars`. If no value is provided for this property, the line will be split on any of the following standard characters: space, colon, full stop, semi-colon, coma, hyphen, forward slash, back slash, left parenthesis, left square bracket and left curly brace

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。

Characters can be automatically inserted at the beginning of each line that has been created. You can set the property `newLineChars` to the characters you want to use.

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.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='> ')
```

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

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.

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.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 <b>Paragraph</b> class.
    As for <b>Preformatted</b>, if dedent is non zero <font color="red" size="+1">dedent</font>
        common leading spaces will be removed from the
    front of each line.
    You can have &amp;amp; style entities as well for &amp; &lt; &gt; and &quot;.

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

produces

```
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 & < > and ".
```

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

Create a flowable which will contain the image defined by the data in file filename which can be filepath, file like object or an instance of a reportlab.graphics.shapes.Drawing. The default **PDF** image type *jpeg* is supported and if the **PIL** extension to **Python** is installed the other image types can also be handled. If width and or height are specified then they determine the dimension of the displayed image in *points*. If either dimension is not specified (or specified as None) then the corresponding pixel dimension of the image is assumed to be in *points* and used.

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("lj8100.jpg")
```

将显示为

whereas

然而

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

produces

产出

9.4 `Spacer(width, height)`

This does exactly as would be expected; it adds a certain amount of space into the story. At present this only works for vertical space.

这与预期的一样，它为故事增加了一定的空间。目前，这只对垂直空间有效。

9.5 `PageBreak()`

This `Flowable` represents a page break. It works by effectively consuming all vertical space given to it. This is sufficient for a single `Frame` document, but would only be a frame break for multiple frames so the `BaseDocTemplate` mechanism detects `pageBreaks` internally and handles them specially.

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(height)`

This `Flowable` attempts to force a `Frame` break if insufficient vertical space remains in the current `Frame`. It is thus probably wrongly named and should probably be renamed as `CondFrameBreak`.

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(flowables)`

This compound `Flowable` takes a list of `Flowables` and attempts to keep them in the same `Frame`. If the total height of the `Flowables` in the list `flowables` exceeds the current frame's available space then all the space is used and a frame break is forced.

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()`

A table of contents can be generated by using the `TableOfContents` flowable. The following steps are needed to add a table of contents to your document:

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。

Create an instance of `TableOfContents`. Override the level styles (optional) and add the object to the story:

创建一个 `TableOfContents` 的实例。覆盖关卡样式（可选）并将对象添加到故事中。

```
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)
```

Entries to the table of contents can be done either manually by calling the `addEntry` method on the `TableOfContents` object or automatically by sending a `'TOCEntry'` notification in the `afterFlowable` method of the `DocTemplate` you are using. The data to be passed to `notify` is a list of three or four items countaining a level number, the entry text, the page number and an optional destination key which the entry should point to. This list will usually be created in a document template's method like afterFlowable(), making notification calls using the notify() method with appropriate data like this:

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))
        # ...
```

This way, whenever a paragraph of style `'Heading1'` or `'Heading2'` is added to the story, it will appear in the table of contents. `Heading2` entries will be clickable because a bookmarked key has been supplied.

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。

Finally you need to use the `multiBuild` method of the DocTemplate because tables of contents need several passes to be generated:

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)
```

Below is a simple but working example of a document with a table of contents:

下面是一个简单但可行的带目录的文档例子。

```python
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()

An index can be generated by using the SimpleIndex flowable. The following steps are needed to add an index to your document:

可以通过使用 SimpleIndex flowable生成一个索引。下面的步骤是为您的文档添加索引所需要的。

Use the index tag in paragraphs to index terms:

在段落中使用索引标签来索引术语。

```python
story = []

...

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

Create an instance of `SimpleIndex` and add it to the story where you want it to appear:

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)
```

The parameters which you can pass into the SimpleIndex constructor are explained in the reportlab reference. Now, build the document by using the canvas maker returned by SimpleIndex.getCanvasMaker():

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())
```

To build an index with multiple levels, pass a comma-separated list of items to the item attribute of an index tag:

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 will respresent the top-most level and termc the most specific term. termd and termb will appear in the same level inside terma.

`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。

If you need to index a term containing a comma, you will need to escape it by doubling it. To avoid the ambiguity of three consecutive commas (an escaped comma followed by a list separator or a list separator followed by an escaped comma?) introduce a space in the right position. Spaces at the beginning or end of terms will be removed.

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(,,), ,, ,...   " />
```

This indexes the terms "comma (,)", "," and "...".

此处索引了 "逗号（，）"、","，"和"......"等术语。

9.10 `ListFlowable(),ListItem()`

Use these classes to make ordered and unordered lists. Lists can be nested.

使用这些类来制作有序和无序的列表。 列表可以被嵌套。

`ListFlowable()` will create an ordered list, which can contain any flowable. The class has a number of parameters to change font, colour, size, style and position of list numbers, or of bullets in unordered lists. The type of numbering can also be set to use lower or upper case letters ('A,B,C' etc.) or Roman numerals (capitals or lowercase) using the bulletType property. To change the list to an unordered type, set bulletType='bullet'.

`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'.

Items within a `ListFlowable()` list can be changed from their default appearance by wrapping them in a `ListItem()` class and setting its properties.

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。

The following will create an ordered list, and set the third item to an unordered sublist.

下面将创建一个有序列表，并将第三项设置为无序子列表。

```python
from reportlab.platypus import ListFlowable, ListItem
from reportlab.lib.styles import getSampleStyleSheet
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'
)
```

produces

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

To cope with nesting the `start` parameter can be set to a list of possible starts; for ul acceptable starts are any unicode character or specific names known to flowables.py eg `bulletchar`, `circle`, `square`, `disc`, `diamond`, `diamondwx`, `rarrowhead`, `sparkle`, `squarelrs` or `blackstar`. For ol the `start` can be any character from `'1iaAI'` to indicate different number styles.

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()`

Use the `BalancedColumns` class to make a flowable that splits its content flowables into two or more roughly equal sized columns. Effectively n frames are synthesized to take the content and the flowable tries to balance the content between them. The created frames will be split when the total height is too large and the split will maintain the balance.

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。

```python
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 编写你自己的`Flowable`对象

Flowables are intended to be an open standard for creating reusable report content, and you can easily create your own objects. We hope that over time we will build up a library of contributions, giving reportlab users a rich selection of charts, graphics and other "report widgets" they can use in their own reports. This section shows you how to create your own flowables.

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`。
we should put the Figure class in the standard library, as it is a very useful base.
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`

Recall the `hand` function from the `pdfgen` section of this user guide which generated a drawing of a hand as a closed figure composed from Bezier curves.

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。
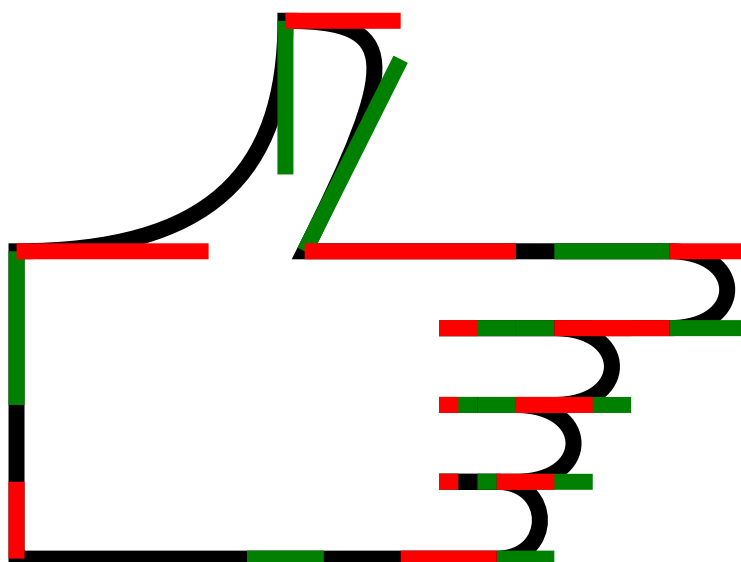
图 10 - 1: 一只手

To embed this or any other drawing in a Platypus flowable we must define a subclass of `Flowable` with at least a `wrap` method and a `draw` method.

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.setFillColor(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)
```

The wrap method must provide the size of the drawing -- it is used by the Platypus mainloop to decide
whether this element fits in the space remaining on the current frame. The draw method performs the
drawing of the object after the Platypus mainloop has translated the (0,0) origin to an appropriate location
in an appropriate frame.

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。

Below are some example uses of the HandAnnotation flowable.

以下是 HandAnnotation flowable 的一些使用示例。
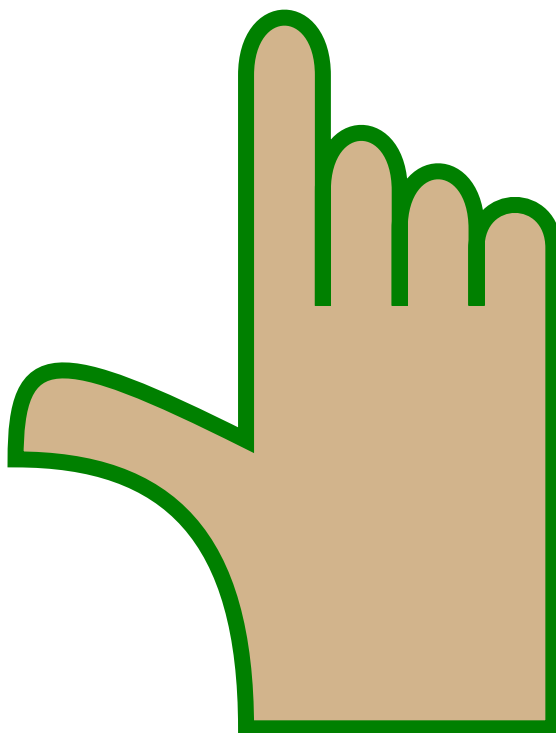
The default.

默认情况下
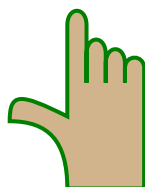
Just one inch high.

■■■■■■

One inch high and shifted to the left with blue and cyan.

高1英寸，向左偏蓝和青色。

10.2 修改内置的 `Flowable`

To modify an existing flowable, you should create a derived class and override the methods you need to change to get the desired behaviour

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。

As an example to create a rotated image you need to override the wrap and draw methods of the existing Image class

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

```python
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('images/replogo.gif')
I.hAlign = 'CENTER'
```

produces

第 11 章 绘图

11.1 简介

ReportLab Graphics is one of the sub-packages to the ReportLab library. It started off as a stand-alone set of programs, but is now a fully integrated part of the ReportLab toolkit that allows you to use its powerful charting and graphics features to improve your PDF forms and reports.

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

In this section we will present some of the more fundamental principles of the graphics library, which will show-up later in various places.

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

A *Drawing* is a platform-independent description of a collection of shapes. It is not directly associated with PDF, Postscript or any other output format. Fortunately, most vector graphics systems have followed the Postscript model and it is possible to describe shapes unambiguously.

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。

A drawing contains a number of primitive *Shapes*. Normal shapes are those widely known as rectangles, circles, lines, etc. One special (logic) shape is a *Group*, which can hold other shapes and apply a transformation to them. Groups represent composites of shapes and allow to treat the composite as if it were a single shape. Just about anything can be built up from a small number of basic shapes.

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Group
(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。

The package provides several *Renderers* which know how to draw a drawing into different formats. These include PDF (renderPDF), Postscript (renderPS), and bitmap output (renderPM). The bitmap renderer uses Raph Levien's *libart* rasterizer and Fredrik Lundh's *Python Imaging Library* (PIL). The SVG renderer makes use of Python's standard library XML modules, so you don't need to install the XML-SIG's additional package named PyXML. If you have the right extensions installed, you can generate drawings in bitmap form for the web as well as vector form for PDF documents, and get "identical output".

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"。

The PDF renderer has special "privileges" - a Drawing object is also a *Flowable* and, hence, can be placed directly in the story of any Platypus document, or drawn directly on a *Canvas* with one line of code. In addition, the PDF renderer has a utility function to make a one-page PDF document quickly.

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。

The SVG renderer is special as it is still pretty experimental. The SVG code it generates is not really optimised in any way and maps only the features available in ReportLab Graphics (RLG) to SVG. This means there is no support for SVG animation, interactivity, scripting or more sophisticated clipping, masking or graduation shapes. So, be careful, and please report any bugs you find!

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

The Y-direction in our X-Y coordinate system points from the bottom *up*. This is consistent with PDF, Postscript and mathematical notation. It also appears to be more natural for people, especially when working with charts. Note that in other graphics models (such as SVG) the Y-coordinate points *down*. For the SVG renderer this is actually no problem as it will take your drawings and flip things as needed, so your SVG output looks just as expected.

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您的图纸进行翻转，因此您的 SVG 输出看起来和预期的一样。

The X-coordinate points, as usual, from left to right. So far there doesn't seem to be any model advocating the opposite direction - at least not yet (with interesting exceptions, as it seems, for Arabs looking at time series charts...).

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

Let's create a simple drawing containing the string "Hello World" and some special characters, displayed on top of a coloured rectangle. After creating it we will save the drawing to a standalone PDF file.

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。

```python
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')
```

This will produce a PDF file containing the following graphic:

这将产生一个包含以下图形的PDF文件。

Figure 11-1: 'Hello World'

Each renderer is allowed to do whatever is appropriate for its format, and may have whatever API is needed. If it refers to a file format, it usually has a `drawToFile` function, and that's all you need to know about the renderer. Let's save the same drawing in Encapsulated Postscript format:

每个渲染器都可以做任何适合其格式的事情，并且可以有任何需要的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')
```

This will produce an EPS file with the identical drawing, which may be imported into publishing tools such as Quark Express. If we wanted to generate the same drawing as a bitmap file for a website, say, all we need to do is write code like this:

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')
```

Many other bitmap formats, like GIF, JPG, TIFF, BMP and PPN are genuinely available, making it unlikely you'll need to add external postprocessing steps to convert to the final format you need.

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。

To produce an SVG file containing the identical drawing, which may be imported into graphical editing tools such as Illustrator all we need to do is write code like this:

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 is very dynamic and lets us execute statements at run time that can easily be the source for unexpected behaviour. One subtle 'error' is when assigning to an attribute that the framework doesn't know about because the used attribute's name contains a typo. Python lets you get away with it (adding a new attribute to an object, say), but the graphics framework will not detect this 'typo' without taking special counter-measures.

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。

There are two verification techniques to avoid this situation. The default is for every object to check every assignment at run time, such that you can only assign to 'legal' attributes. This is what happens by default. As this imposes a small performance penalty, this behaviour can be turned off when you need it to be.

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
```

These statements would be caught by the compiler in a statically typed language, but Python lets you get away with it. The first error could leave you staring at the picture trying to figure out why the colors were wrong. The second error would probably become clear only later, when some back-end tries to draw the rectangle. The third, though less likely, results in an invalid object that would not know how to draw itself.

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
>>>
```

This imposes a performance penalty, so this behaviour can be turned off when you need it to be. To do this, you should use the following lines of code before you first import reportlab.graphics.shapes:

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
>>>
```

Once you turn off shapeChecking, the classes are actually built without the verification hook; code should get faster, then. Currently the penalty seems to be about 25% on batches of charts, so it is hardly worth disabling. However, if we move the renderers to C in future (which is eminently possible), the remaining 75% would shrink to almost nothing and the saving from verification would be significant.

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，如果我们将来把渲染器转移到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。

Each object, including the drawing itself, has a verify() method. This either succeeds, or raises an exception. If you turn off automatic verification, then you should explicitly call verify() in testing when developing the code, or perhaps once in a batch process.

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

A cornerstone of the reportlab/graphics which we will cover below is that you can automatically document widgets. This means getting hold of all of their editable properties, including those of their subcomponents.

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。

Another goal is to be able to create GUIs and config files for drawings. A generic GUI can be built to show all editable properties of a drawing, and let you modify them and see the results. The Visual Basic or Delphi development environment are good examples of this kind of thing. In a batch charting application, a file could list all the properties of all the components in a chart, and be merged with a database query to make a batch of charts.

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。

To support these applications we have two interfaces, `getProperties` and `setProperties`, as well as a convenience method `dumpProperties`. The first returns a dictionary of the editable properties of an object; the second sets them en masse. If an object has publicly exposed 'children' then one can recursively set and get their properties too. This will make much more sense when we look at *Widgets* later on, but we need to put the support into the base of the framework.

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
>>>
```

Note: `pprint` *is the standard Python library module that allows you to 'pretty print' output over multiple lines rather than having one very long line.*

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。

These three methods don't seem to do much here, but as we will see they make our widgets framework much more powerful when dealing with non-primitive objects.

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

You can add objects to the `Drawing` and `Group` objects. These normally go into a list of contents. However, you may also give objects a name when adding them. This allows you to refer to and possibly change any element of a drawing after constructing it.

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'
>>>
```

Note that you can use the same shape instance in several contexts in a drawing; if you choose to use the same `Circle` object in many locations (e.g. a scatter plot) and use different names to access it, it will still be a shared object and the changes will be global.

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。

This provides one paradigm for creating and modifying interactive drawings.

这为创建和修改交互式图纸提供了一种范式。

11.3 图表

The motivation for much of this is to create a flexible chart package. This section presents a treatment of the ideas behind our charting model, what the design goals are and what components of the chart package already exist.

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

Here are some of the design goals:

以下是一些设计目标。

Make simple top-level use really simple

让简单的顶层使用变得真正简单

It should be possible to create a simple chart with minimum lines of code, yet have it 'do the right things' with sensible automatic settings. The pie chart snippets above do this. If a real chart has many subcomponents, you still should not need to interact with them unless you want to customize what they do.

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。

Allow precise positioning

让位精确定位

允许精确定位

An absolute requirement in publishing and graphic design is to control the placing and style of every element. We will try to have properties that specify things in fixed sizes and proportions of the drawing, rather than having automatic resizing. Thus, the 'inner plot rectangle' will not magically change when you make the font size of the y labels bigger, even if this means your labels can spill out of the left edge of the chart rectangle. It is your job to preview the chart and choose sizes and spaces which will work.

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。

Some things do need to be automatic. For example, if you want to fit N bars into a 200 point space and don't know N in advance, we specify bar separation as a percentage of the width of a bar rather than a point size, and let the chart work it out. This is still deterministic and controllable.

有些事情确实需要自动完成。例如，如果你想在一个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。

Control child elements individually or as a group

单独或分组控制子元素

We use smart collection classes that let you customize a group of things, or just one of them. For example you can do this in our experimental pie chart:

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。

```
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, '')
```

pc.slices[3] actually lazily creates a little object which holds information about the slice in question; this will be used to format a fourth slice at draw-time if there is one.

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。

Only expose things you should change

只揭露你应该改变的事情

It would be wrong from a statistical viewpoint to let you directly adjust the angle of one of the pie slices in the above example, since that is determined by the data. So not everything will be exposed through the public properties. There may be 'back doors' to let you violate this when you really need to, or methods to provide advanced functionality, but in general properties will be orthogonal.

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。

Composition and component based

组成和成分

Charts are built out of reusable child widgets. A Legend is an easy-to-grasp example. If you need a specialized type of legend (e.g. circular colour swatches), you should subclass the standard Legend widget. Then you could either do something like...

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...
```

...or create/modify your own chart or drawing class which creates one of these by default. This is also very relevant for time series charts, where there can be many styles of x axis.

...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。

Top level chart classes will create a number of such components, and then either call methods or set private properties to tell them their height and position - all the stuff which should be done for you and which you cannot customise. We are working on modelling what the components should be and will publish their APIs here as a consensus emerges.

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--所有这些东西都应该为你做，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。

Multiples

倍数

A corollary of the component approach is that you can create diagrams with multiple charts, or custom data graphics. Our favourite example of what we are aiming for is the weather report in our gallery contributed by a user; we'd like to make it easy to create such drawings, hook the building blocks up to their legends, and feed that data in a consistent way.

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。

(If you want to see the image, it is available on our website here)

(如果你想看图片，可以在我们的网站上找到。这儿)

概述

A chart or plot is an object which is placed on a drawing; it is not itself a drawing. You can thus control where it goes, put several on the same drawing, or add annotations.

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。

Charts have two axes; axes may be Value or Category axes. Axes in turn have a Labels property which lets you configure all text labels or each one individually. Most of the configuration details which vary from chart to chart relate to axis properties, or axis labels.

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。

Objects expose properties through the interfaces discussed in the previous section; these are all optional and are there to let the end user configure the appearance. Things which must be set for a chart to work, and essential communication between a chart and its components, are handled through methods.

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。

You can subclass any chart component and use your replacement instead of the original provided you implement the essential methods and properties.

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。

11.4 `Labels` 属性

A label is a string of text attached to some chart element. They are used on axes, for titles or alongside axes, or attached to individual data points. Labels may contain newline characters, but only one font.

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。

The text and 'origin' of a label are typically set by its parent object. They are accessed by methods rather than properties. Thus, the X axis decides the 'reference point' for each tick mark label and the numeric or date text for each label. However, the end user can set properties of the label (or collection of labels) directly to affect its position relative to this origin and all of its formatting.

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。

```python
from reportlab.graphics import shapes
from reportlab.graphics.charts.textlabels import Label

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
Multi-Line
Label')

d.add(lab)
```


图 11-2: `Label` 示例

In the drawing above, the label is defined relative to the green blob. The text box should have its north-east corner ten points down from the origin, and be rotated by 45 degrees about that corner.

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。

At present labels have the following properties, which we believe are sufficient for all charts we have seen to date:

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
dx	标签的 X 位移。
dy	标签的 Y 位移。
angle	标签的旋转角度（逆时针）。
boxAnchor	标签的框锚，'n'、'e'、'w'、's'、'ne'、'nw'、'se'、'sw'中的一种。
textAnchor	标签文字的固定位置，"start"、"middle"、"end"中的一个。
boxFillColor	标签框中使用的填充颜色。
boxStrokeColor	标签框中使用的笔触颜色。
boxStrokeWidth	标签框的线宽。
fontName	标签的字体名称。
fontSize	标签的字体大小。
leading	标签文字行的前导值。(leading)
x	参考点的X坐标。
y	参考点的Y坐标。
width	标签的宽度。
height	标签的高度。

表 11 - 4 - Label 属性

To see many more examples of `Label` objects with different combinations of properties, please have a look into the ReportLab test suite in the folder `tests`, run the script `test_charts_textlabels.py` and look at the PDF document it generates!

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。

11.5 Axes

We identify two basic kinds of axes - *Value* and *Category* ones. Both come in horizontal and vertical flavors. Both can be subclassed to make very specific kinds of axis. For example, if you have complex rules for which dates to display in a time series application, or want irregular scaling, you override the axis and make a new one.

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。

Axes are responsible for determining the mapping from data to image coordinates; transforming points on request from the chart; drawing themselves and their tick marks, grid lines and axis labels.

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。

This drawing shows two axes, one of each kind, which have been created directly without reference to any chart:

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。
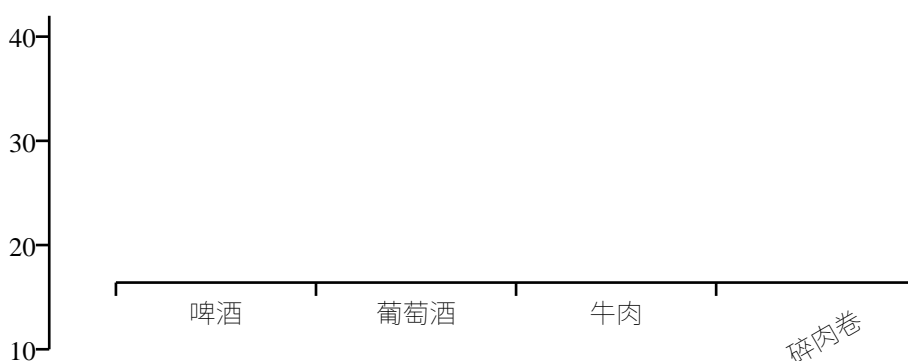
图 11 - 3：两个孤立的轴

Here is the code that created them:

下面是创建它们的代码。

```python
from reportlab.graphics import shapes
from reportlab.graphics.charts.axes import XCategoryAxis,YValueAxis

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 = ['██', '███', '██', '███']
xAxis.labels.boxAnchor = 'n'
xAxis.labels[3].dy = -15
xAxis.labels[3].angle = 30
# xAxis.labels[3].fontName = 'Times-Bold'
xAxis.labels.fontName = 'SourceHanSansSC'

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

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

Remember that, usually, you won't have to create axes directly; when using a standard chart, it comes with ready-made axes. The methods are what the chart uses to configure it and take care of the geometry. However, we will talk through them in detail below. The orthogonally dual axes to those we describe have essentially the same properties, except for those refering to ticks.

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

A Category Axis doesn't really have a scale; it just divides itself into equal-sized buckets. It is simpler than a value axis. The chart (or programmer) sets its location with the method setPosition(x, y, length). The next stage is to show it the data so that it can configure itself. This is easy for a category axis - it just counts the number of data points in one of the data series. The reversed attribute (if 1) indicates that the categories should be reversed. When the drawing is drawn, the axis can provide some help to the chart with

its `scale()` method, which tells the chart where a given category begins and ends on the page. We have not yet seen any need to let people override the widths or positions of categories.

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。

An XCategoryAxis has the following editable properties:

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。
`strokeColor`	轴的颜色
`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"。
`strokeWidth`	轴的宽度，以点为单位（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）
`tickDown`	刻度线应突出到轴下方多远？
`categoryNames`	$None$ 或字符串列表。 该长度应与每个数据系列的长度相同。
`labels`	刻度线标签的集合。默认情况下，每个文本标签的 "$north$"（北）（即顶部中心）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。
`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。

<center>表 11-5- <code>XCategoryAxis</code> 类的属性</center>

`YValueAxis` 类

The left axis in the diagram is a YValueAxis. A Value Axis differs from a Category Axis in that each point along its length corresponds to a y value in chart space. It is the job of the axis to configure itself, and to convert Y values from chart space to points on demand to assist the parent chart in plotting.

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)` and `configure(data)` work exactly as for a category axis. If you have not fully specified the maximum, minimum and tick interval, then `configure()` results in the axis choosing suitable values. Once configured, the value axis can convert y data values to drawing space with the `scale()` method. Thus:

`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
>>>
```

By default, the highest data point is aligned with the top of the axis, the lowest with the bottom of the axis, and the axis choose 'nice round numbers' for its tickmark points. You may override these settings with the properties below.

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。
strokeColor	轴的颜色
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"。
strokeWidth	轴的宽度，以点为单位（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）
tickRight	刻度线应突出到轴的右侧多远？
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。
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。
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"。
valueSteps	放置刻度的数字列表。
minimumTickSpacing	当$valueStep$设置为$None$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 $step$。
labelTextFormat	这确定了标签中的内容。与接受固定字符串的类别轴不同， $ValueAxis$ 上的标签应为数字。 您可以提供 "$%0.2f$" 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。
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。

表 11-6 - `YValueAxis` 类属性

The `valueSteps` property lets you explicitly specify the tick mark locations, so you don't have to follow regular intervals. Hence, you can plot month ends and month end dates with a couple of helper functions, and without needing special time series chart classes. The following code show how to create a simple `XValueAxis` with special tick intervals. Make sure to set the `valueSteps` attribute before calling the configure method!

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，你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!

```python
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)
```
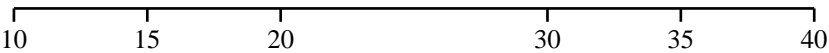

图 11 - 4 : 带非等距刻度线的轴

In addition to these properties, all axes classes have three properties describing how to join two of them to each other. Again, this is interesting only if you define your own charts or want to modify the appearance of an existing chart using such axes. These properties are listed here only very briefly for now, but you can find a host of sample functions in the module reportlab/graphics/axes.py which you can examine...

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......

One axis is joined to another, by calling the method joinToAxis(otherAxis, mode, pos) on the first axis, with mode and pos being the properties described by joinAxisMode and joinAxisPos, respectively. 'points' means to use an absolute value, and 'value' to use a relative value (both indicated by the the joinAxisPos property) along the axis.

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
joinAxis	如果为真，则连接两个轴。
joinAxisMode	用于连接轴的模式 ('bottom', 'top', 'left', 'right', 'value', 'points', None).
joinAxisPos	与其他轴连接的位置。

表 11 - 7 - Axes joining 属性列表

11.6 柱状图

This describes our current `VerticalBarChart` class, which uses the axes and labels above. We think it is step in the right direction but is is far from final. Note that people we speak to are divided about 50/50 on whether to call this a 'Vertical' or 'Horizontal' bar chart. We chose this name because 'Vertical' appears next to 'Bar', so we take it to mean that the bars rather than the category axis are vertical.

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。

As usual, we will start with an example:

与往常一样，我们将从一个示例开始：
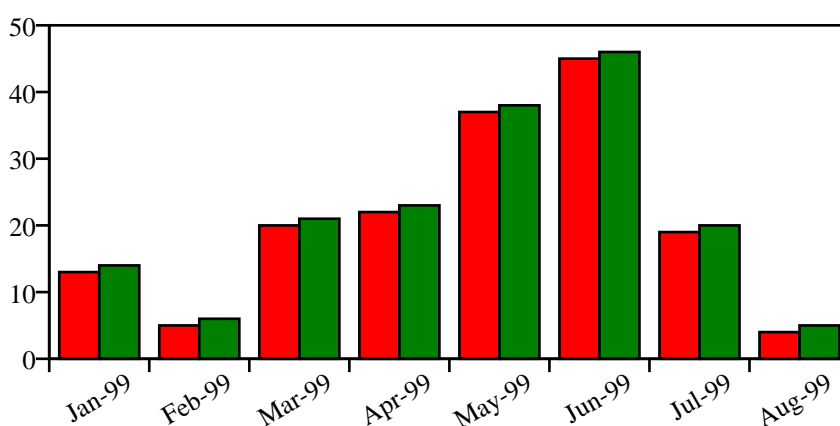
图 11 - 5：具有两个数据系列的简单柱状图

```python
# code to produce the above chart

from reportlab.graphics.shapes import Drawing
from reportlab.graphics.charts.barcharts import VerticalBarChart

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)
```

Most of this code is concerned with setting up the axes and labels, which we have already covered. Here are the top-level properties of the `VerticalBarChart` class:

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),]
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。
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。
fillColor	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 $dashArray$ 等）
useAbsolute	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。
barWidth	柱状宽度. 默认为 10.
groupSpacing	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$groupSpacing$而不是barSpacing来分割它们。 $groupSpacing$的一半用于图表的第一个条形图之前，另一半用于结尾。
barSpacing	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。
barLabelFormat	默认为 None。 与$YValueAxis$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。
barLabels	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： chart.barLabels[(1,2)].fontSize = 12。
valueAxis	值轴，其格式可如前所述。
categoryAxis	类别轴，其格式可如前所述。
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。

表 11 - 8 - 柱状图(`VerticalBarChart`) 属性列表

From this table we deduce that adding the following lines to our code above should double the spacing between bar groups (the `groupSpacing` attribute has a default value of five points) and we should also see some tiny space between bars of the same group (`barSpacing`).

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
```

And, in fact, this is exactly what we can see after adding these lines to the code above. Notice how the width of the individual bars has changed as well. This is because the space added between the bars has to be 'taken' from somewhere as the total chart width stays unchanged.

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。
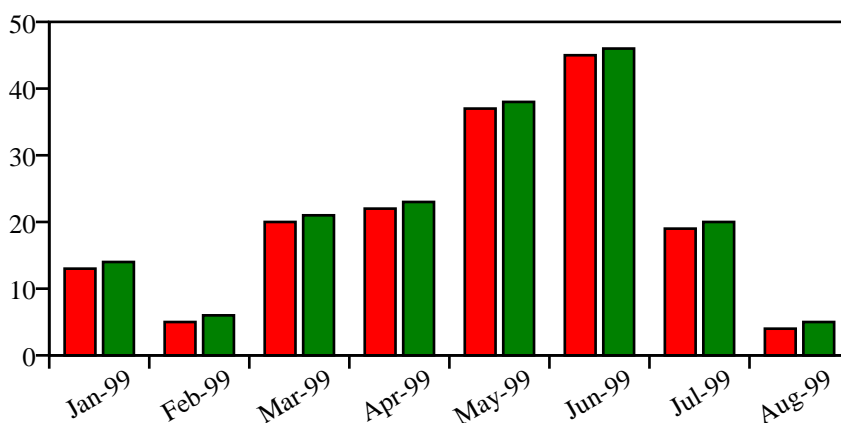注意各个条的宽度也如何变化。
这是因为随着总图表宽度保持不变，必须从某处"占用"条形之间的空格。

图 11-6：像以前一样，但间距已更改

Bars labels are automatically displayed for negative values *below* the lower end of the bar for positive values *above* the upper end of the other ones.

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。

Stacked bars are also supported for vertical bar graphs. You enable this layout for your chart by setting the `style` attribute to `'stacked'` on the `categoryAxis`.

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'
```

Here is an example of the previous chart values arranged in the stacked style.

下面是以之前的图表值为例，以叠加的方式排列。
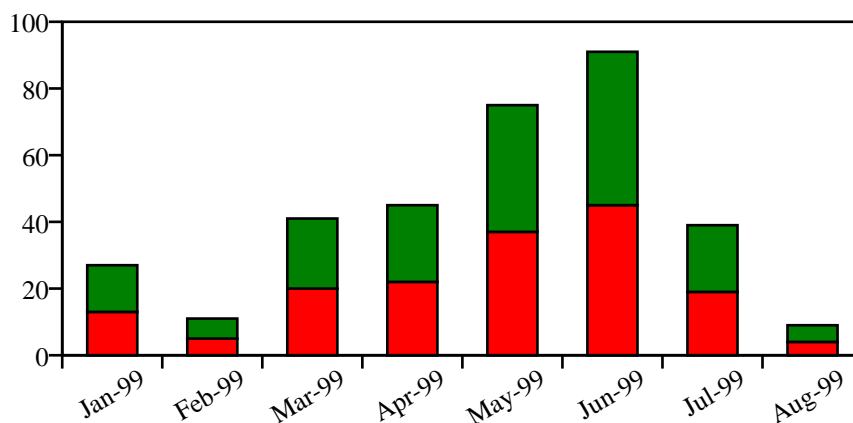
图 11 - 7 : 柱状叠加图

11.7 折线图

We consider "Line Charts" to be essentially the same as "Bar Charts", but with lines instead of bars. Both share the same pair of Category/Value axes pairs. This is in contrast to "Line Plots", where both axes are *Value* axes.

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。

The following code and its output shall serve as a simple example. More explanation will follow. For the time being you can also study the output of running the tool reportlab/lib/graphdocpy.py withough any arguments and search the generated PDF document for examples of Line Charts.

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。

```python
from reportlab.graphics.charts.linecharts import HorizontalLineChart

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)
```
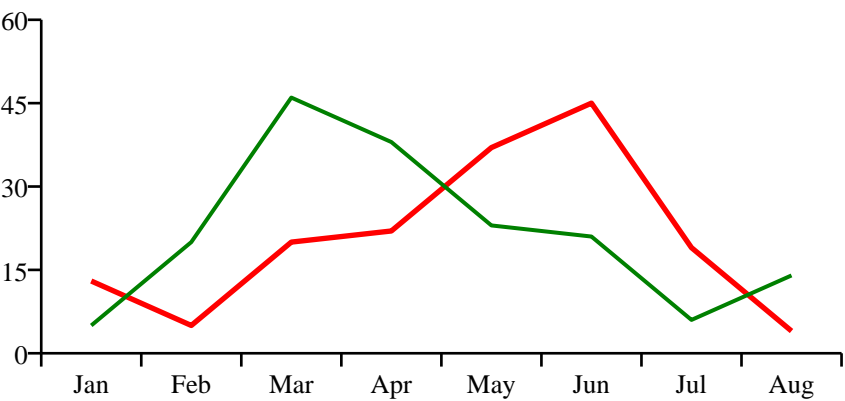
图 11-8: 折线图示例 `HorizontalLineChart`

属性	描述
`data`	要绘制的数据，（列表）整数清单。
`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
`valueAxis`	值轴，其格式可如前所述。
`categoryAxis`	类别轴，其格式可如前所述。
`strokeColor`	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。
`fillColor`	默认为 None。 这将用纯色填充绘图矩形。
`lines.strokeColor`	线的颜色
`lines.strokeWidth`	线的宽度
`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。
`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。
`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， 这些标签值才会被显示。

表 11-9- 折线图属性列表 `HorizontalLineChart`

11.8 点线图

Below we show a more complex example of a Line Plot that also uses some experimental features like line markers placed at each data point.

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.charts.lineplots import LinePlot
from reportlab.graphics.widgets.markers import makeMarker
```

```
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)
```
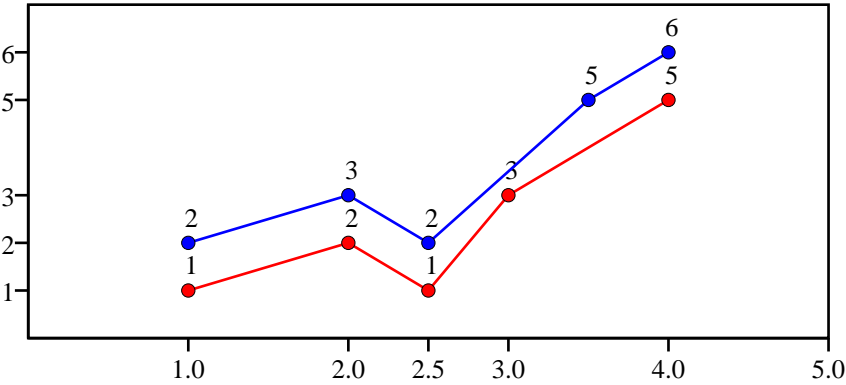

图 11-9：点线图示例 LinePlot

属性	描述
data	要绘制的数据，（列表）整数清单。
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
xValueAxis	垂直值轴，其格式可以如上所述。
yValueAxis	水平值轴，其格式可以如上所述。
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。
strokeWidth	默认为None。绘图矩形周围边框的宽度。
fillColor	默认为None。这将用纯色填充绘图矩形。
lines.strokeColor	线的颜色。
lines.strokeWidth	线的宽度

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')
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。
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
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lineLabelForm

表 11-10 - 点线图属性 LinePlot

11.9 饼图

As usual, we will start with an example:

像往常一样，我们将从一个例子开始:

```
from reportlab.graphics.charts.piecharts import Pie
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)
```
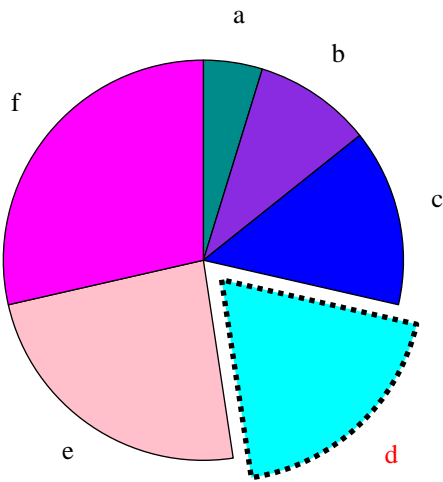

Figure 11-10: A bare bones pie chart
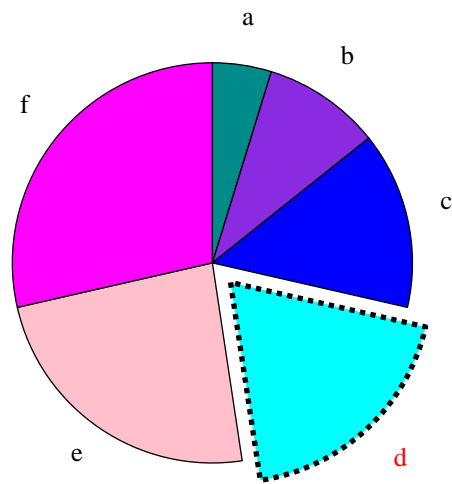
图 11 - 11 : 凸出的饼图

Properties are covered below. The pie has a 'slices' collection and we document wedge properties in the same table.

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
`data`	整数合集
`x, y, width, height`	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。
`labels`	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。
`startAngle`	第一个饼片的起始角度是什么？默认为"90"，即 12 点钟方向。
`direction`	切片的前进方向是什么？默认为 "clockwise"。(顺时针)
`sideLabels`	这将创建一个标签在两边各一列的图表。
`sideLabelsOffset`	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。
`simpleLabels`	默认为 1. 设置为0以启用自定义标签以及在集合切片中使用 label_ 前缀的属性。
`slices`	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
`slices.strokeWidth`	扇形边框宽度
`slices.strokeColor`	扇形边框颜色
`slices.strokeDashArray`	实线或虚线配置：['solid', 'dashed']
`slices.popout`	切片应从馅饼的中心伸出多远？ 默认值为零。
`slices.fontName`	标签字体名称
`slices.fontSize`	标签字体大小
`slices.fontColor`	标签文字的颜色

slices.labelRadius	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）

<div align="center">表 11-11 - 饼图属性 Pie</div>

自定义标签

Each slide label can be customized individually by changing the properties prefixed by `label_` in the collection `slices`. For example `pc.slices[2].label_angle = 10` changes the angle of the third 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。

Before being able to use these customization properties, you need to disable simple labels with: `pc.simplesLabels = 0`

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
label_dx	X轴标签偏移
label_dy	Y轴标签偏移
label_angle	标签的角度，默认（0）为水平，90为垂直，180为上下颠倒
label_boxAnchor	标签的固定点
label_boxStrokeColor	标签框的边框颜色
label_boxStrokeWidth	标签框的边框宽度
label_boxFillColor	标签盒的填充颜色
label_strokeColor	标签文字的边框颜色
label_strokeWidth	标签文字的边框宽度
label_text	标签文字
label_width	标签宽度
label_maxWidth	标签可以增加到的最大宽度
label_height	标签高度
label_textAnchor	标签可以增加到的最大高度
label_visible	如果要绘制标签，则为True
label_topPadding	盒子的上边距(Padding at top of box)
label_leftPadding	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
label_pointer_strokeWidth	指示线宽度

<div align="center">表 11 - 12 - 扇形标签自定义属性 Pie.slices custom label</div>

侧面标签

If the sideLabels attribute is set to true, then the labels of the slices are placed in two columns, one on either side of the pie and the start angle of the pie will be set automatically. The anchor of the right hand column is set to 'start' and the anchor of the left hand column is set to 'end'. The distance from the edge of the pie from the edge of either column is decided by the sideLabelsOffset attribute, which is a fraction of the width of the pie. If xradius is changed, the pie can overlap the labels, and so we advise leaving xradius as None. There is an example below.

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。
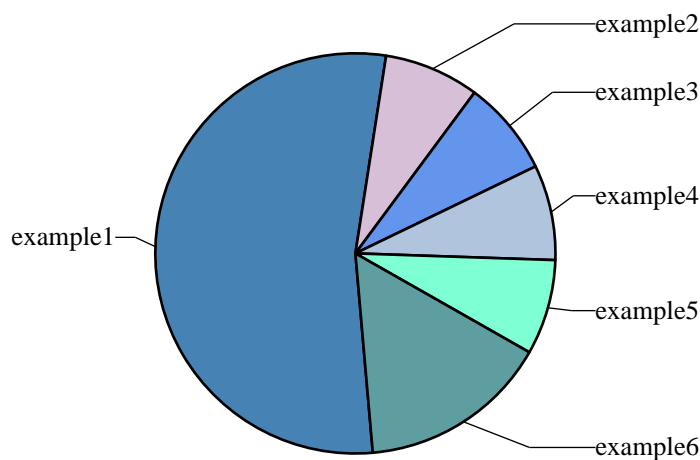
图 11‑12：一个 sideLabels=1 的饼图示例

If you have sideLabels set to True, then some of the attributes become redundant, such as pointerLabelMode. Also sideLabelsOffset only changes the piechart if sideLabels is set to true.

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

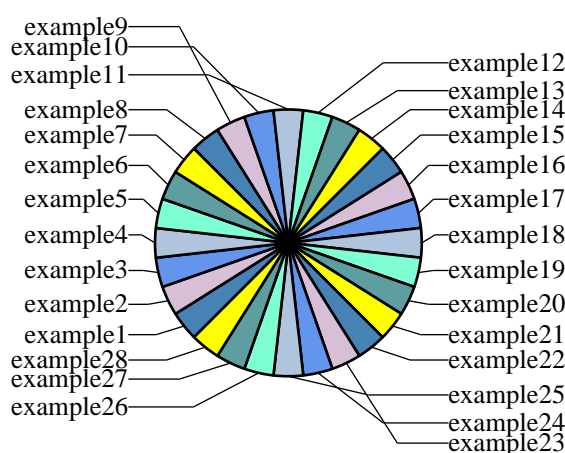The pointers can cross if there are too many slices.

如果切片过多，指针可能会交叉。

图 11 - 13 : 指针交叉的例子

Also the labels can overlap despite checkLabelOverlap if they correspond to slices that are not adjacent.

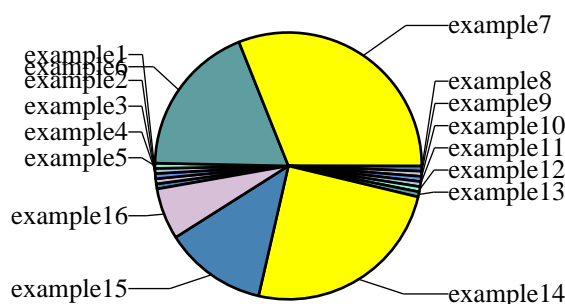另外，如果标签对应的片子不相邻，尽管有 `checkLabelOverlap`，标签也可以重叠。

图 11 - 14 : 标签重叠的示例

11.10 Legends

Various preliminary legend classes can be found but need a cleanup to be consistent with the rest of the charting model. Legends are the natural place to specify the colors and line styles of charts; we propose that each chart is created with a `legend` attribute which is invisible. One would then do the following to specify colors:

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]
```

One could also define a group of charts sharing the same legend:

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
```

Does this work? Is it an acceptable complication over specifying chart colors directly?
这样行吗？ 直接指定图表颜色是否可以接受？

存在的问题

There are several issues that are *almost* solved, but for which is is a bit too early to start making them really public. Nevertheless, here is a list of things that are under way:

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。

- Color specification - right now the chart has an undocumented property `defaultColors`, which provides a list of colors to cycle through, such that each data series gets its own color. Right now, if you introduce a legend, you need to make sure it shares the same list of colors. Most likely, this will be replaced with a scheme to specify a kind of legend containing attributes with different values for each data series. This legend can then also be shared by several charts, but need not be visible itself.
- Additional chart types - when the current design will have become more stable, we expect to add variants of bar charts to deal with percentile bars as well as the side-by-side variant seen here.
- 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`)

It will take some time to deal with the full range of chart types. We expect to finalize bars and pies first and to produce trial implementations of more general plots, thereafter.

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

aaa

Most other plots involve two value axes and directly plotting x-y data in some form. The series can be plotted as lines, marker symbols, both, or custom graphics such as open-high-low-close graphics. All share the concepts of scaling and axis/title formatting. At a certain point, a routine will loop over the data series and 'do something' with the data points at given x-y locations. Given a basic line plot, it should be very easy to derive a custom chart type just by overriding a single method - say, drawSeries().

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

Well known plotting packages such as excel, Mathematica and Excel offer ranges of marker types to add to charts. We can do better - you can write any kind of chart widget you want and just tell the chart to use it as an example.

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

Combining multiple plot types is really easy. You can just draw several charts (bar, line or whatever) in the same rectangle, suppressing axes as needed. So a chart could correlate a line with Scottish typhoid cases over a 15 year period on the left axis with a set of bars showing inflation rates on the right axis. If anyone can remind us where this example came from we'll attribute it, and happily show the well-known graph as an example.

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

One principle of the Graphics package is to make all 'interesting' properties of its graphic components accessible and changeable by setting apropriate values of corresponding public attributes. This makes it very tempting to build a tool like a GUI editor that that helps you with doing that interactively.

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 has built such a tool using the Tkinter toolkit that loads pure Python code describing a drawing and records your property editing operations. This "change history" is then used to create code for a subclass of that chart, say, that can be saved and used instantly just like any other chart or as a new starting point for another interactive editing session.

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。

This is still work in progress, though, and the conditions for releasing this need to be further elaborated.

不过这还在进行中，发布的条件还需要进一步细化。

其他

This has not been an exhaustive look at all the chart classes. Those classes are constantly being worked on. To see exactly what is in the current distribution, use the graphdocpy.py utility. By default, it will run on reportlab/graphics, and produce a full report. (If you want to run it on other modules or packages, graphdocpy.py -h prints a help message that will tell you how.)

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。)

This is the tool that was mentioned in the section on 'Documenting Widgets'.

这是"文档小部件"（Documenting Widgets）部分中提到的工具

11.11 多边形

This section describes the concept of shapes and their importance as building blocks for all output generated by the graphics library. Some properties of existing shapes and their relationship to diagrams are presented and the notion of having different renderers for different output formats is briefly introduced.

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。

Available Shapes

可用形状

Drawings are made up of Shapes. Absolutely anything can be built up by combining the same set of primitive shapes. The module `shapes.py` supplies a number of primitive shapes and constructs which can be added to a drawing. They are:

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 - ■■
- Circle - ■
- Ellipse - ■■
- Wedge (a pie slice) - ■■
- Polygon - ■■■
- Line - ■
- PolyLine - ■■
- String - ■■■
- Group - ■
- `Path` (还没有完全实现，但将来会加入) - 路径

The following drawing, taken from our test suite, shows most of the basic shapes (except for groups). Those with a filled green surface are also called *solid shapes* (these are `Rect`, `Circle`, `Ellipse`, `Wedge` and `Polygon`).

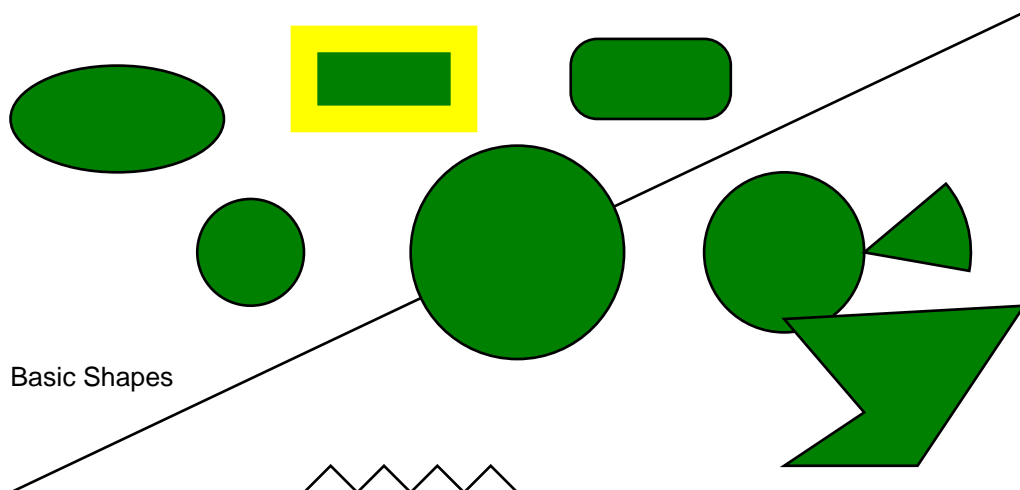下面的图画来自于我们的测试套件，显示了大部分的基本形状（除了组）。那些有绿色填充面的图形也被称为实体图形。(这些是`Rect`、`Circle`、`Ellipse`、`Wedge`和`Polygon`)。

图 11 - 15 : 基本的图形

形状属性

Shapes have two kinds of properties - some to define their geometry and some to define their style. Let's create a red rectangle with 3-point thick green borders:

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 Rect
>>> from reportlab.lib.colors import red, green
>>> r = Rect(5, 5, 200, 100)
>>> r.fillColor = red
>>> r.strokeColor = green
>>> r.strokeWidth = 3
>>>
```
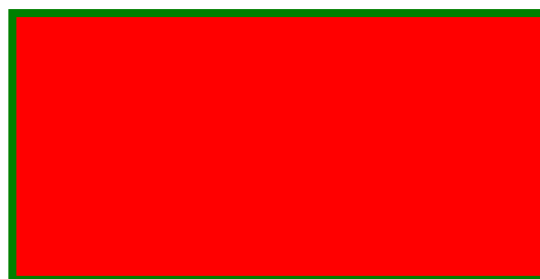
图 11 - 16 : 红色矩形和绿色边框

Note: In future examples we will omit the import statements.

注意：在未来的例子中，我们将省略导入语句。

All shapes have a number of properties which can be set. At an interactive prompt, we can use their *dumpProperties()* method to list these. Here's what you can use to configure a Rect:

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
>>>
```

Shapes generally have *style properties* and *geometry properties*. x, y, width and height are part of the geometry and must be provided when creating the rectangle, since it does not make much sense without those properties. The others are optional and come with sensible defaults.

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。

You may set other properties on subsequent lines, or by passing them as optional arguments to the constructor. We could also have created our rectangle this way:

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)
```

Let's run through the style properties. `fillColor` is obvious. `stroke` is publishing terminology for the edge of a shape; the stroke has a color, width, possibly a dash pattern, and some (rarely used) features for what happens when a line turns a corner. `rx` and `ry` are optional geometric properties and are used to define the corner radius for a rounded rectangle.

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。

All the other solid shapes share the same style properties.

其他所有的实体形状都具有相同的样式属性。

线

We provide single straight lines, PolyLines and curves. Lines have all the `stroke*` properties, but no `fillColor`. Here are a few Line and PolyLine examples and the corresponding graphics output:

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。

```
Line(50,50, 300,100,
     strokeColor=colors.blue, strokeWidth=5)
Line(50,100, 300,50,
     strokeColor=colors.red,
     strokeWidth=10,
     strokeDashArray=[10, 20])
PolyLine([120,110, 130,150, 140,110, 150,150, 160,110,
          170,150, 180,110, 190,150, 200,110],
         strokeWidth=2,
         strokeColor=colors.purple)
```
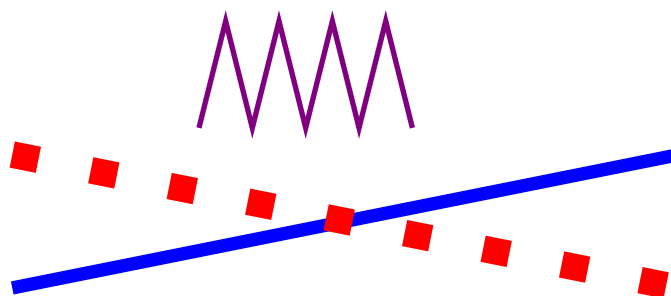
图 11 - 17：线型和多线型示例

字符串

The ReportLab Graphics package is not designed for fancy text layout, but it can place strings at desired locations and with left/right/center alignment. Let's specify a `String` object and look at its properties:

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
>>>
```

Strings have a textAnchor property, which may have one of the values 'start', 'middle', 'end'. If this is set to 'start', x and y relate to the start of the string, and so on. This provides an easy way to align text.

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。

Strings use a common font standard: the Type 1 Postscript fonts present in Acrobat Reader. We can thus use the basic 14 fonts in ReportLab and get accurate metrics for them. We have recently also added support for extra Type 1 fonts and the renderers all know how to render Type 1 fonts.

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。

Here is a more fancy example using the code snippet below. Please consult the ReportLab User Guide to see how non-standard like 'DarkGardenMK' fonts are being registered!

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。

```python
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 11‑18：花式字体样例

Paths

Postscript paths are a widely understood concept in graphics. They are not implemented in `reportlab/graphics` as yet, but they will be, soon.

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

Finally, we have Group objects. A group has a list of contents, which are other nodes. It can also apply a transformation - its contents can be rotated, scaled or shifted. If you know the math, you can set the transform directly. Otherwise it provides methods to rotate, scale and so on. Here we make a group which is rotated and translated:

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)
```

Groups provide a tool for reuse. You can make a bunch of shapes to represent some component - say, a coordinate system - and put them in one group called "Axis". You can then put that group into other groups, each with a different translation and rotation, and you get a bunch of axis. It is still the same group, being drawn in different places.

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。

Let's do this with some only slightly more code:

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)
```
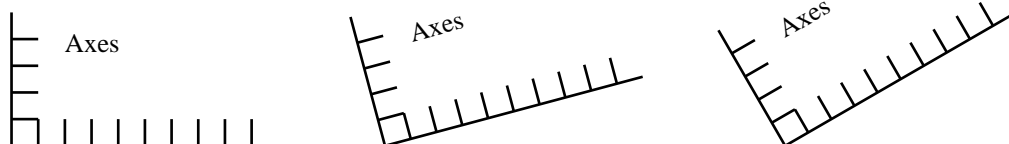

图 11 - 19：Groups 示例

11.12 小部件

We now describe widgets and how they relate to shapes. Using many examples it is shown how widgets make reusable graphics components.

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

Up until now, Drawings have been 'pure data'. There is no code in them to actually do anything, except assist the programmer in checking and inspecting the drawing. In fact, that's the cornerstone of the whole concept and is what lets us achieve portability - a renderer only needs to implement the primitive shapes.

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。

We want to build reusable graphic objects, including a powerful chart library. To do this we need to reuse more tangible things than rectangles and circles. We should be able to write objects for other to reuse - arrows, gears, text boxes, UML diagram nodes, even fully fledged charts.

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。

The Widget standard is a standard built on top of the shapes module. Anyone can write new widgets, and we can build up libraries of them. Widgets support the getProperties() and setProperties() methods, so you can inspect and modify as well as document them in a uniform way.

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。

- A widget is a reusable shape
- it can be initialized with no arguments when its draw() method is called it creates a primitive Shape or a Group to represent itself
- It can have any parameters you want, and they can drive the way it is drawn
- it has a demo() method which should return an attractively drawn example of itself in a 200x100 rectangle. This is the cornerstone of the automatic documentation tools. The demo() method should also have a well written docstring, since that is printed too!
- 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！

Widgets run contrary to the idea that a drawing is just a bundle of shapes; surely they have their own code? The way they work is that a widget can convert itself to a group of primitive shapes. If some of its components are themselves widgets, they will get converted too. This happens automatically during rendering; the renderer will not see your chart widget, but just a collection of rectangles, lines and strings. You can also explicitly 'flatten out' a drawing, causing all widgets to be converted to primitives.

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

Let's imagine a simple new widget. We will use a widget to draw a face, then show how it was implemented.

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')
```
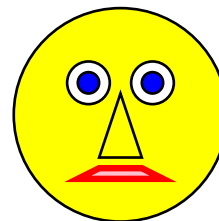
图 11 - 20 : 小部件示例

Let's see what properties it has available, using the setProperties() method we have seen earlier:

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
>>>
```

One thing which seems strange about the above code is that we did not set the size or position when we made the face. This is a necessary trade-off to allow a uniform interface for constructing widgets and documenting them - they cannot require arguments in their __init__() method. Instead, they are generally designed to fit in a 200 x 100 window, and you move or resize them by setting properties such as x, y, width and so on after creation.

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。

In addition, a widget always provides a demo() method. Simple ones like this always do something sensible before setting properties, but more complex ones like a chart would not have any data to plot. The documentation tool calls demo() so that your fancy new chart class can create a drawing showing what it can do.

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。

Here are a handful of simple widgets available in the module *signsandsymbols.py*:
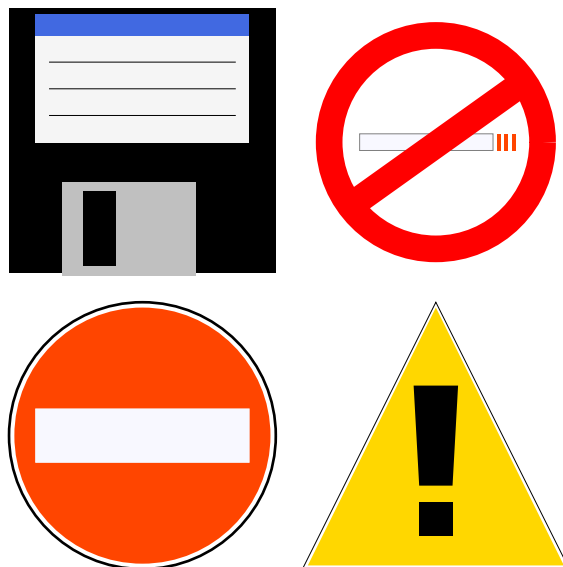
以下是一些简单的小部件，可在模块 signsandsymbols.py 中使用。

图 11‑21：一些来自 `signandsymbols.py` 的例子。

And this is the code needed to generate them as seen in the drawing above:

而这是生成它们所需要的代码，如上图所示。

```python
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)
```

复合小部件

Let's imagine a compound widget which draws two faces side by side. This is easy to build when you have the Face widget.

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
>>>
```

The attributes 'faceOne' and 'faceTwo' are deliberately exposed so you can get at them directly. There could also be top-level attributes, but there aren't in this case.

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

The widget designer decides the policy on verification, but by default they work like shapes - checking every assignment - if the designer has provided the checking information.

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

We tried to make it as easy to implement widgets as possible. Here's the code for a Face widget which does not do any type checking:

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
```

We left out all the code to draw the shapes in this document, but you can find it in the distribution in widgetbase.py.

我们在这个文档中省略了所有绘制形状的代码，但你可以在 widgetbase.py 中找到它。

By default, any attribute without a leading underscore is returned by setProperties. This is a deliberate policy to encourage consistent coding conventions.

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。

Once your widget works, you probably want to add support for verification. This involves adding a dictionary to the class called _verifyMap, which map from attribute names to 'checking functions'. The widgetbase.py module defines a bunch of checking functions with names like isNumber, isListOfShapes and so on. You can also simply use None, which means that the attribute must be present but can have any type. And you can and should write your own checking functions. We want to restrict the "mood" custom attribute to the values "happy", "sad" or "ok". So we do this:

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。而且你可以也应该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：

```python
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
        }
```

This checking will be performed on every attribute assignment; or, if config.shapeChecking is off, whenever you call myFace.verify().

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

We are working on a generic tool to document any Python package or module; this is already checked into ReportLab and will be used to generate a reference for the ReportLab package. When it encounters widgets, it adds extra sections to the manual including:

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：

- the doc string for your widget class
- the code snippet from your *demo()* method, so people can see how to use it
- the drawing produced by the *demo()* method
- the property dump for the widget in the drawing.
- 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。

This tool will mean that we can have guaranteed up-to-date documentation on our widgets and charts, both on the web site and in print; and that you can do the same for your own widgets, too!

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

We could not come up with a consistent architecture for designing widgets, so we are leaving that problem to the authors! If you do not like the default verification strategy, or the way setProperties/getProperties works, you can override them yourself.

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！如果你不喜欢默认的验证策略，或者是
setProperties/getProperties 的工作方式，你可以自己覆盖它们。

For simple widgets it is recommended that you do what we did above: select non-overlapping properties, initialize every property on __init__ and construct everything when draw() is called. You can instead have __setattr__ hooks and have things updated when certain attributes are set. Consider a pie chart. If

you want to expose the individual slices, you might write code like this:

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
。

```python
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
```

The last line is problematic as we have only created four slices - in fact we might not have created them yet. Does pc.slices[7] raise an error? Is it a prescription for what should happen if a seventh wedge is defined, used to override the default settings? We dump this problem squarely on the widget author for now, and recommend that you get a simple one working before exposing 'child objects' whose existence depends on other properties' values :-)

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 :-)

We also discussed rules by which parent widgets could pass properties to their children. There seems to be a general desire for a global way to say that 'all slices get their lineWidth from the lineWidth of their parent' without a lot of repetitive coding. We do not have a universal solution, so again leave that to widget authors. We hope people will experiment with push-down, pull-down and pattern-matching approaches and come up with something nice. In the meantime, we certainly can write monolithic chart widgets which work like the ones in, say, Visual Basic and Delphi.

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。

For now have a look at the following sample code using an early version of a pie chart widget and the output it generates:

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。

```python
from reportlab.lib.colors import *
from reportlab.graphics import shapes,renderPDF
from reportlab.graphics.charts.piecharts import Pie

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')
```
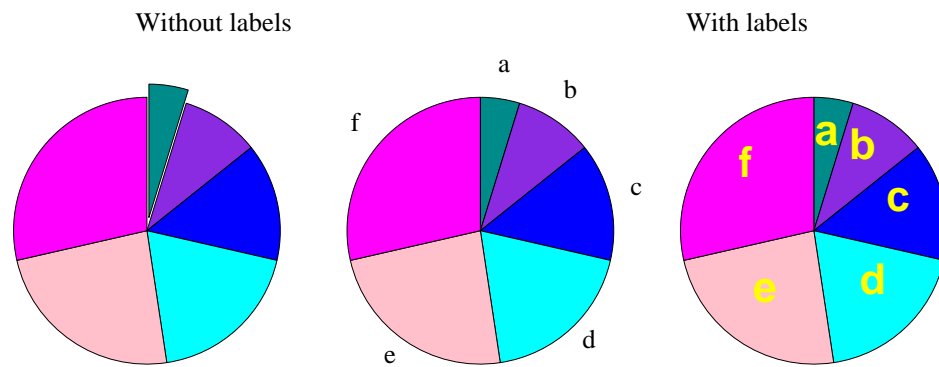
图 11 - 22 : 一些饼图示例

附录 A ReportLab 示例

In the subdirectories of `reportlab/demos` there are a number of working examples showing almost all aspects of reportlab in use.

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

The three scripts odyssey.py, dodyssey.py and fodyssey.py all take the file odyssey.txt and produce PDF documents. The included odyssey.txt is short; a longer and more testing version can be found at ftp://ftp.reportlab.com/odyssey.full.zip.

`odyssey.py`、`dodyssey.py`和`fodyssey.py`这三个脚本都是取文件`oddyssey.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
```

Simple formatting is shown by the odyssey.py script. It runs quite fast, but all it does is gather the text and force it onto the canvas pages. It does no paragraph manipulation at all so you get to see the XML < & > tags.

`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。

The scripts fodyssey.py and dodyssey.py handle paragraph formatting so you get to see colour changes etc. Both scripts use the document template class and the dodyssey.py script shows the ability to do dual column layout and uses multiple page templates.

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

In `reportlab/demos/stdfonts` the script stdfonts.py can be used to illustrate ReportLab's standard fonts. Run the script using

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
```

to produce two PDF documents, StandardFonts_MacRoman.pdf & StandardFonts_WinAnsi.pdf which show the two most common built in font encodings.

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。

The colortest.py script in `reportlab/demos/colors` demonstrates the different ways in which reportlab can set up and use colors.

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。

Try running the script and viewing the output document, colortest.pdf. This shows different color spaces and a large selection of the colors which are named in the `reportlab.lib.colors` module.

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.cols`模块中命名的颜色选择。

A.3 Py2pdf

Dinu Gherman contributed this useful script which uses reportlab to produce nicely colorized PDF documents from Python scripts including bookmarks for classes, methods and functions. To get a nice version of the main script try

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
```

i.e. we used py2pdf to produce a nice version of py2pdf.py in the document with the same rootname and a .pdf extension.

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`。

The py2pdf.py script has many options which are beyond the scope of this simple introduction; consult the comments at the start of the script.

`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.4 Gadflypaper

The Python script, gfe.py, in `reportlab/demos/gadflypaper` uses an inline style of document preparation. The script almost entirely produced by Aaron Watters produces a document describing Aaron's `gadfly` in memory database for Python. To generate the document use

`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
```

everything in the PDF document was produced by the script which is why this is an inline style of document production. So, to produce a header followed by some text the script uses functions `header` and `p` which take some text and append to a global story list.

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.5 Pythonpoint

Andy Robinson has refined the pythonpoint.py script (in `reportlab\demos\pythonpoint`) until it is a really useful script. It takes an input file containing an XML markup and uses an xmllib style parser to map the tags into PDF slides. When run in its own directory pythonpoint.py takes as a default input the file pythonpoint.xml and produces pythonpoint.pdf which is documentation for Pythonpoint! You can also see it in action with an older paper

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
```

Not only is pythonpoint self documenting, but it also demonstrates reportlab and PDF. It uses many features of reportlab (document templates, tables etc). Exotic features of PDF such as fadeins and bookmarks are also shown to good effect. The use of an XML document can be contrasted with the *inline* style of the gadflypaper demo; the content is completely separate from the formatting

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: <u>ANSI，ISO-8859-1和MacRoman字符集之间的差异</u>

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:<u>ANSI是什么编码?</u>