Welcome to PythonPoint

...a library for creating presentation slides.

PythonPoint lets you create attractive and consistent presentation slides on any platform. It is a demo app built on top of the PDFgen PDF library and the PLATYPUS Page Layout library. Essentially, it converts slides in an XML format to PDF.

It can be used right now to create slide shows, but will undoubtedly change and evolve. Read on for a tutorial...

Part 1 – Feature Overview

XML Notation

You create slides in a text editor with a basic XML syntax looking like this:

Pythonpoint then converts these into slides. Just enter "pythonpoint.py myfile.xml" to create a PDF document (usually called "myfile.pdf", but you specify that in the XML).

Page Layout Model

The Page Layout model comes from PLATYPUS (Page Layout and Typography Using Scripts), a key component of the toolkit. This covers concepts such as:

- Reusable 'Drawable Objects'
- Frames into which objects flow (like this one, around which we have drawn a border)
- Style Sheets for text, table cells, line styles etc.
- Wrapping, page breaking an document management logic

Everything is open and extensible. I hope a library of reusable objects such as charts and diagrams will grow up.

ReportLab

Reuse and Consistency – Sections

You can create a 'section' spanning some or all tags in the presentation and place graphics on this. The blue border and title come from the section. Here's how we did the border:

Thus you can re-brand an entire presentation for a new audience in seconds.

Style Sheets

Paragraph styles are defined externally. You may specify a filename from which to load a stylesheet with the stylesheet tag.

Thus you can have different sizes and formats by switching stylesheets, or colour and black-and-white options.

When they are added, tables will be driven by line and cell styles in a similar way.

Special Effects

Acrobat Reader supports tags to define page transition effects. If you are reading this on screen, you should have seen a selection of these:

- Split
- Blinds
- Box
- Wipe
- Dissolve
- Glitter

Each has a range of options to fine-tune.

When they are added, tables will be driven by line and cell styles in a similar way.

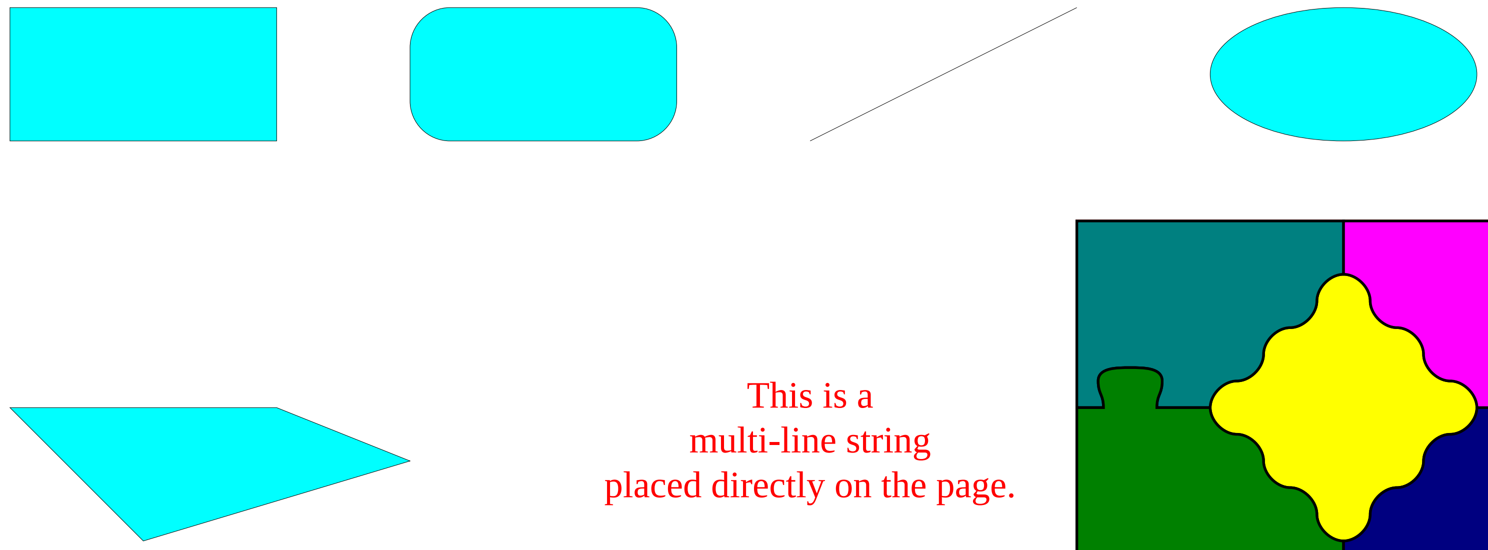
Outlines and Hyperlinks

By default, we generate an outline view in the left pane to help you navigate. Hyperlinks within documents are also possible.

As far as we know, this is the first PDF library to expose these features.

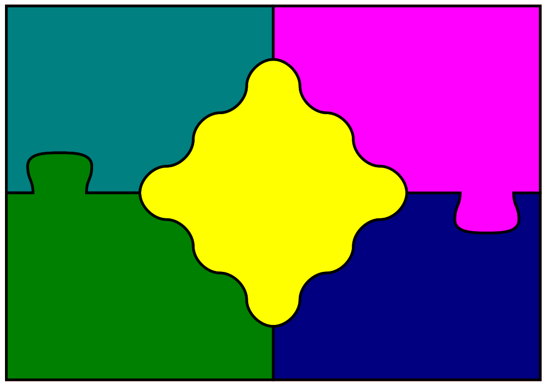
Basic Shapes

Here are some of the basic shapes available for decorating pages...



This is a
multi-line string
placed directly on the page.

It can be left-aligned,
centred,
or right-aligned.



A Custom Shape

Tables

The Table tag lets you paste in bulk data and format it attractively:

Division	Jan	Feb	Mar	Q1 Total
North	100	115	120	335
South	215	145	180	540
East	75	90	135	300
West	100	120	115	335

Features Coming Soon

This is the first version that runs. A lot can now be added fairly easily:

- Preprocessor to let you enter paragraphs and bullets as one block of text, with less tag typing!
- PIDDLE drawings
- PINGO drawings – 'Object Graphics' tags with grouping and coordinate transformations
- Speaker notes and a mode to print them
- Tools to archive slides in a database and build presentations to order

...what else can YOU think of?

Part 2 – Reference

This section covers all command line options and tags currently in use.

Command Line Options

Usage (NT, or executable Unix script):
pythonpoint.py [/notes] myslides.xml
or (Win9x or non-executable script)
python pythonpoint.py [/notes] myslides.xml

Notes:

- The resulting PDF file has the same name as the input file.
- The Speaker Notes mode prints a shrunken canvas with room for notes around the edge. To create notes, make an extra frame off the page. See the source of Pythonpoint.xml slide 001 for an example.

Tag "presentation"

This is the outermost tag in an XML file and is always required.

Attributes:

filename (required)

Can Contain:

section, stylesheet, slides

To Do:

Support for page sizes, opening modes

Tag "stylesheet"

This defines an external style sheet full of paragraph styles. For now, these are Python modules conforming to a common interface, and examples are given. If not declared, a default style sheet is used. You are strongly advised to define your own style sheet, as the built-in one will change a few more times.

Attributes:

path, module, function

Contained By:

Presentation, Section

Can Contain:

nothing

Example

Tag "section"

A Section spans across a number of slides and can apply an overall background to them. Place graphics directly within the section tag, either before or after slides. This lets you re-brand a presentation very quickly. Documents may contain multiple sections; nesting of sections is not (yet) permitted.

Attributes:

name (required, but not used yet)

Contained By:

Presentation

Can Contain:

all graphic shapes; slides

Tag "slide"

Defines a single slide. The presentation effects are defined in the PDF reference; best to just try the combinations.

Attributes (with defaults):

id (required)
title (required)
effectname: one of Split, Blinds, Box, Wipe, Dissolve, Glitter
effectdirection: '0','90','180' or '270'
effectdimension: 'H' or 'V' (Horiz./Vert.)
effectmotion: 'I' for inwards or 'O' for outwards
effectduration: time in seconds

Contained By:

Presentation

Can Contain:

all graphic shapes; frames

Tag "frame"

Defines a frame on the page which can hold content. You may have as many frames as you like, to allow multi-column text or flow around pictures.

Attributes:

x, y, width, height (all required): in points
leftmargin, rightmargin, topmargin, bottommargin (optional, default to zero) – define the 'inner rectangle' within which content flows
border (defaults to 'false'): whether to show a frame border – useful when designing pages.

Contained By:

Slide

Can Contain:

all 'flowable objects' – paragraphs, images

Tag family – "Flowable Objects"

Flowable Objects currently include Paragraphs, Preformatted text (used for code printing, where the line breaks and spaces matter) and inline Images. More will be added in future. They negotiate with their containing frame about height and width; paragraphs do what you would expect, while images are centred.

Contained By:

Slide

Can Contain:

The three instances so far contain nothing.

Tag "para" – Paragraphs

Paragraphs are used for wrapping text. They are very simple – they have a style attribute, and the stylesheet defines most attributes externally. Currently we use a hack to handle 'bullets', which may be in a different font (such as 'ZapfDingbats, specified in style) and offset to the left. These are used for bullets, numbering and definition lists This will vanish as soon as one can switch fonts in mid-paragraph (due mid April).

Attributes:

style (defaults to 'Normal') – reference to stylesheet
bullettext – text for the optional 'bullet' section. To be deprecated.

Contained By:

Frame

Can Contain:

Their text

Tag "prefmt"

This is used for printing code, or other text which contains line breaks.

Attributes:

style (defaults to 'Normal') – reference to stylesheet

Contained By:

Frame

Can Contain:

The text to be displayed

Tag "image" – flowing images

This is used for an image to be displayed inline within the frame. It will be drawn at a scale of 1 pixel to 1 point, and centred in the frame.

Attributes:

filename (required)

Contained By:

Frame

Can Contain:

Nothing

To do

Rename it 'flowing image'? Control over alignment and size if needed. Image caching.

Tag "table" – tables

This lets you draw tables with a wide variety of formatting options.

Attributes:

widths (optional) in points (auto-sizes if not given)
heights (optional) in points (auto-sizes if not given)
style (optional) – name of a ReportLab tablestyle defined in the stylesheet.
colDelim (optional) – the column delimiter string for bulk data; defaults to a comma.
rowDelim (optional) – the line delimiter string for bulk data; defaults to a carriage return.

Contained By:

Frame

Can Contain:

Bulk data, with the row and column delimiters specified

Tag family – "Drawable Objects"

These are objects which know how to draw themselves directly on the page (or section template). Use them for designing the backdrop, and for custom graphics.

Contained By:

Slide, Section

Can Contain:

Varies.

To Do:

Will include the full PINGO object model – a subset of SVG – allowing any vector graphics at all.

Tag "fixedimage"

This is an image draw directly at a fixed position – for example, the logo at top left of the page.

Attributes:

filename: required
x, y: required
width, height: optional, stretches image to fit box if present.

Contained By:

Slide, Section

Can Contain:

Nothing

Tag "rectangle"

Attributes:

x, y, width, height: required
fill, stroke: either 'None', or an (r,g,b) tuple.
linewidth: defaults to 0 (hairline)

Contained By:

Slide, Section

Can Contain:

Nothing

Tag "roundrect"

This is exactly like Rectangle, but with an extra 'radius' attribute defining the corner radius in points – default is 6 points.

Tag "ellipse"

Draws an ellipse, defined by its bounding box. Note that it can create circles if height and width are equal.

Attributes:

x1, y1, x2, y2: required
fill, stroke: either 'None', or an (r,g,b) tuple.
linewidth: defaults to 0 (hairline)

Contained By:

Slide, Section

Can Contain:

Nothing

Tag "polygon"

Draws a polygon from a list of points you provide.

Attributes:

points: list such as "(0,0),(50,0),(25,25)"
fill, stroke: either 'None', or an (r,g,b) tuple.
linewidth: defaults to 0 (hairline)

Contained By:

Slide, Section

Can Contain:

Nothing

Tag "line"

Draws a line.

Attributes:

x1, y1, x2, y2
stroke: either 'None', or an (r,g,b) tuple.
width: defaults to 0 (hairline)

Contained By:

Slide, Section

Can Contain:

Nothing

Tag "string"

This places strings directly on the page. They may have embedded newlines (use a '\n' in the XML), in which case multi-line strings are printed. Left, right and centre alignment are allowed.

Attributes:

x, y: required
color: RGB colour tuple such as '(0,1,0)'
font: default is 'Times-Roman'
size: default 12
align: default 'left', allows also 'right' or 'center'

Contained By:

Slide, Section

Can Contain:

The text of the string

Tag "customshape"

This looks in a specified Python module for a 'drawable object' you write, and initialises it with arguments you provide before drawing. This must provide a 'self.drawOn(canvas)' method.

Attributes:

path: where to look; searches Python path if None
module: module name
class: class name to create
initargs: tuple of arguments with which to initialize the class.
align: default 'left', allows also 'right' or 'center'

Contained By:

Slide, Section

Can Contain:

Nothing

Part 3 – To Do

- Lots of testing
- Text preprocessor to let you input text, styles and images in something easier to type
- Support for Pingo (http://pingo.sourceforge.net/) drawings using the Scalable Vector Graphics imaging model
- Proper caching of flowing images
- Basic Tables and Charts
- Use new XML parsers as wel as xmllib
- Slide indexing and database search tools
- Speaker Notes mode

Naturally, help is extremely welcome :-)