python-pptx Documentation

Release 0.6.7

Steve Canny

Oct 30, 2017

Contents

Release v0.6.7 (*Installation*) *python-pptx* is a Python library for creating and updating PowerPoint (.pptx) files.

A typical use would be generating a customized PowerPoint presentation from database content, downloadable by clicking a link in a web application. Several developers have used it to automate production of presentation-ready engineering status reports based on information held in their work management system. It could also be used for making bulk updates to a library of presentations or simply to automate the production of a slide or two that would be tedious to get right by hand.

More information is available in the python-pptx documentation.

Browse examples with screenshots to get a quick idea what you can do with python-pptx.

CHAPTER 1

Feature Support

python-pptx has the following capabilities, with many more on the roadmap:

- Round-trip any Open XML presentation (.pptx file) including all its elements
- Add slides
- Populate text placeholders, for example to create a bullet slide
- Add image to slide at arbitrary position and size
- Add textbox to a slide; manipulate text font size and bold
- Add table to a slide
- Add auto shapes (e.g. polygons, flowchart shapes, etc.) to a slide
- Add and manipulate column, bar, line, and pie charts
- Access and change core document properties such as title and subject

Additional capabilities are actively being developed and added on a release cadence of roughly once per month. If you find a feature you need that *python-pptx* doesn't yet have, reach out via the mailing list or issue tracker and we'll see if we can jump the queue for you to pop it in there :)

CHAPTER 2

User Guide

Introduction

python-pptx is a Python library for creating and updating PowerPoint (.pptx) files.

A typical use would be generating a customized PowerPoint presentation from database content, downloadble by clicking a link in a web application. Several developers have used it to automate production of presentation-ready engineering status reports based on information held in their work management system. It could also be used for making bulk updates to a library of presentations or simply to automate the production of a slide or two that would be tedious to get right by hand.

This user guide is tutorial in nature, introducing concepts along with code examples that I hope will allow you to learn what you need while addressing the real-life PowerPoint challenges you're working on.

python-pptx License

Installing

python-pptx is hosted on PyPI, so installing with *pip* is simple:

```
pip install python-pptx
```

It can also be installed using `easy_install`, but that is not recommended:

```
easy_install python-pptx
```

If neither `pip` nor `easy_install` is available, it can be installed manually by downloading the distribution from PyPI, unpacking the tarball, and running `setup.py`:

```
tar xvzf python-pptx-0.1.0a1.tar.gz
cd python-pptx-0.1.0a1
python setup.py install
```

python-pptx depends on the `lxml` package and `Pillow`, the modern version of the Python Imaging Library (`PIL`). The charting features depend on `XlsxWriter`. Both `pip` and `easy_install` will take care of satisfying these dependencies for you, but if you use the `setup.py` installation method you will need to install the dependencies yourself.

Currently *python-pptx* requires Python 2.6, 2.7, 3.3 or 3.4.

Dependencies

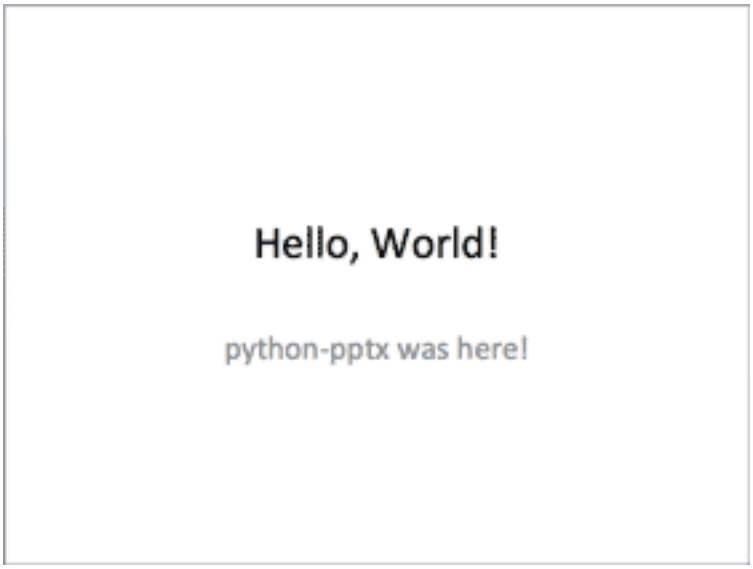
- Python 2.6, 2.7, 3.3, or 3.4
- lxml
- Pillow
- XlsxWriter (to use charting features)

Getting Started

A quick way to get started is by trying out some of the examples below to get a feel for how to use *python-pptx*.

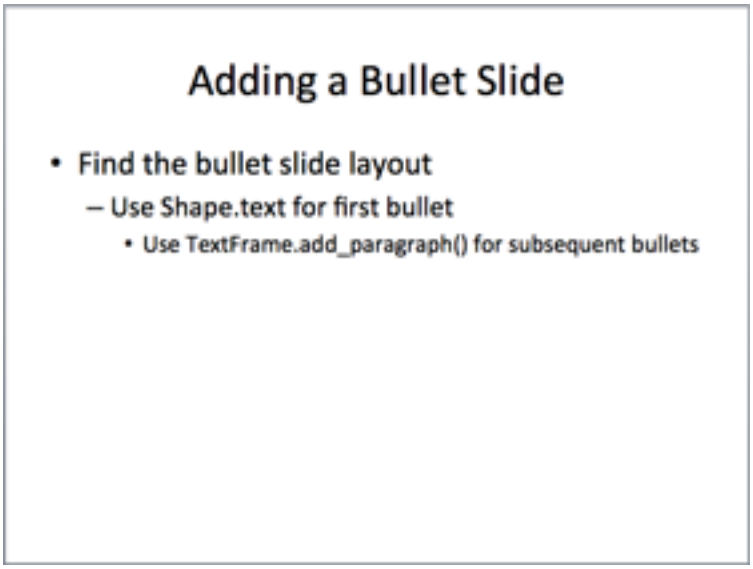
The *API documentation* can help you with the fine details of calling signatures and behaviors.

Hello World! example




```
from pptx import Presentation

prs = Presentation()
title_slide_layout = prs.slide_layouts[0]
slide = prs.slides.add_slide(title_slide_layout)
title = slide.shapes.title
subtitle = slide.placeholders[1]

title.text = "Hello, World!"
subtitle.text = "python-pptx was here!"

prs.save('test.pptx')
```

Bullet slide example

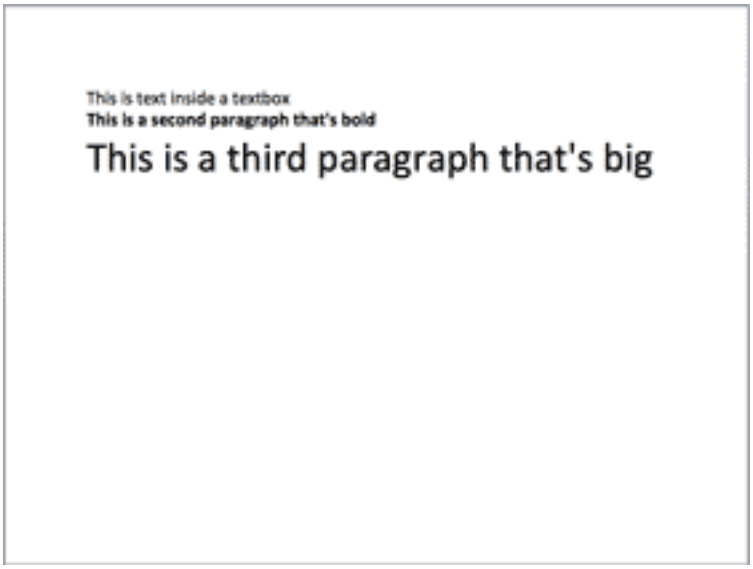



```
from pptx import Presentation

prs = Presentation()
bullet_slide_layout = prs.slide_layouts[1]

slide = prs.slides.add_slide(bullet_slide_layout)
shapes = slide.shapes

title_shape = shapes.title
body_shape = shapes.placeholders[1]

title_shape.text = 'Adding a Bullet Slide'

tf = body_shape.text_frame
tf.text = 'Find the bullet slide layout'

p = tf.add_paragraph()
p.text = 'Use _TextFrame.text for first bullet'
p.level = 1

p = tf.add_paragraph()
p.text = 'Use _TextFrame.add_paragraph() for subsequent bullets'
p.level = 2

prs.save('test.pptx')
```

Not all shapes can contain text, but those that do always have at least one paragraph, even if that paragraph is empty and no text is visible within the shape. `_BaseShape.has_text_frame` can be used to determine whether a shape can contain text. (All shapes subclass `_BaseShape`.) When `_BaseShape.has_text_frame` is `True`, `_BaseShape.text_frame.paragraphs[0]` returns the first paragraph. The text of the first paragraph can be set using `text_frame.paragraphs[0].text`. As a shortcut, the writable properties `_BaseShape.text`

and `_TextFrame.text` are provided to accomplish the same thing. Note that these last two calls delete all the shape's paragraphs except the first one before setting the text it contains.

`add_textbox()` example

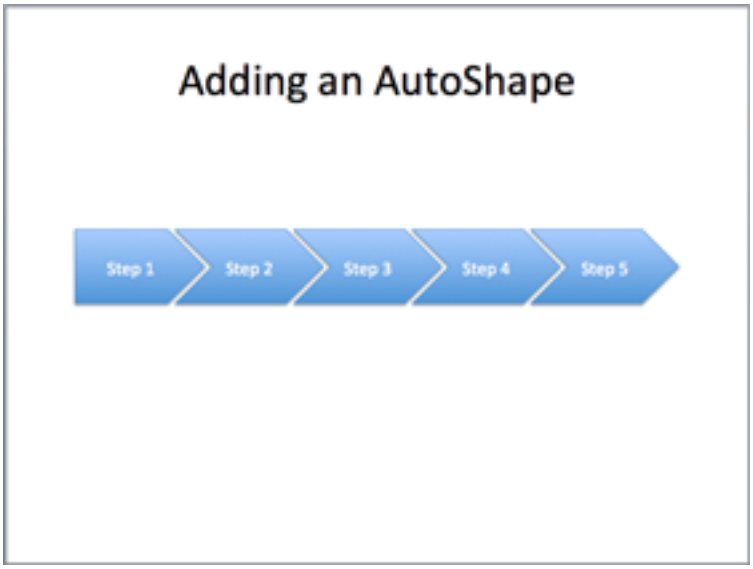



```
from pptx import Presentation
from pptx.util import Inches, Pt

prs = Presentation()
blank_slide_layout = prs.slide_layouts[6]
slide = prs.slides.add_slide(blank_slide_layout)

left = top = width = height = Inches(1)
txBox = slide.shapes.add_textbox(left, top, width, height)
tf = txBox.text_frame

tf.text = "This is text inside a textbox"

p = tf.add_paragraph()
p.text = "This is a second paragraph that's bold"
p.font.bold = True

p = tf.add_paragraph()
p.text = "This is a third paragraph that's big"
p.font.size = Pt(40)

prs.save('test.pptx')
```

`add_picture()` example


```
from pptx import Presentation
from pptx.util import Inches

img_path = 'monty-truth.png'

prs = Presentation()
blank_slide_layout = prs.slide_layouts[6]
slide = prs.slides.add_slide(blank_slide_layout)

left = top = Inches(1)
pic = slide.shapes.add_picture(img_path, left, top)

left = Inches(5)
height = Inches(5.5)
pic = slide.shapes.add_picture(img_path, left, top, height=height)

prs.save('test.pptx')
```

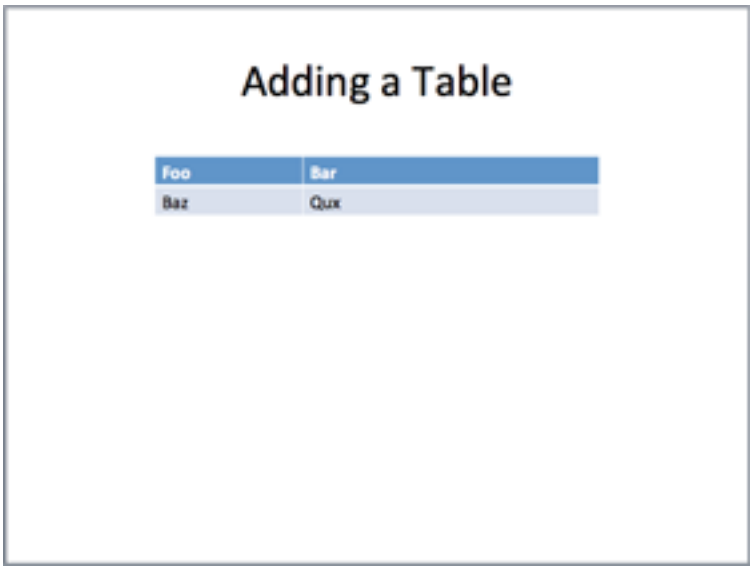
`add_shape()` example




```
from pptx import Presentation
from pptx.enum.shapes import MSO_SHAPE
from pptx.util import Inches

prs = Presentation()
title_only_slide_layout = prs.slide_layouts[5]
slide = prs.slides.add_slide(title_only_slide_layout)
shapes = slide.shapes

shapes.title.text = 'Adding an AutoShape'

left = Inches(0.93)  # 0.93" centers this overall set of shapes
top = Inches(3.0)
width = Inches(1.75)
height = Inches(1.0)

shape = shapes.add_shape(MSO_SHAPE.PENTAGON, left, top, width, height)
shape.text = 'Step 1'

left = left + width - Inches(0.4)
width = Inches(2.0)  # chevrons need more width for visual balance

for n in range(2, 6):
    shape = shapes.add_shape(MSO_SHAPE.CHEVRON, left, top, width, height)
    shape.text = 'Step %d' % n
    left = left + width - Inches(0.4)

prs.save('test.pptx')
```

Constants representing each of the available auto shapes (like MSO_SHAPE.ROUNDED_RECT, MSO_SHAPE.CHEVRON, etc.) are listed on the *autoshape-types* page.

`add_table()` example




```
from pptx import Presentation
from pptx.util import Inches

prs = Presentation()
title_only_slide_layout = prs.slide_layouts[5]
slide = prs.slides.add_slide(title_only_slide_layout)
shapes = slide.shapes

shapes.title.text = 'Adding a Table'

rows = cols = 2
left = top = Inches(2.0)
width = Inches(6.0)
height = Inches(0.8)

table = shapes.add_table(rows, cols, left, top, width, height).table

# set column widths
table.columns[0].width = Inches(2.0)
table.columns[1].width = Inches(4.0)

# write column headings
table.cell(0, 0).text = 'Foo'
table.cell(0, 1).text = 'Bar'

# write body cells
table.cell(1, 0).text = 'Baz'
```

```
table.cell(1, 1).text = 'Qux'

prs.save('test.pptx')
```

Extract all text from slides in presentation

```
from pptx import Presentation

prs = Presentation(path_to_presentation)

# text_runs will be populated with a list of strings,
# one for each text run in presentation
text_runs = []

for slide in prs.slides:
    for shape in slide.shapes:
        if not shape.has_text_frame:
            continue
        for paragraph in shape.text_frame.paragraphs:
            for run in paragraph.runs:
                text_runs.append(run.text)
```

Working with Presentations

python-pptx allows you to create new presentations as well as make changes to existing ones. Actually, it only lets you make changes to existing presentations; it's just that if you start with a presentation that doesn't have any slides, it feels at first like you're creating one from scratch.

However, a lot of how a presentation looks is determined by the parts that are left when you delete all the slides, specifically the theme, the slide master, and the slide layouts that derive from the master. Let's walk through it a step at a time using examples, starting with the two things you can do with a presentation, open it and save it.

Opening a presentation

The simplest way to get started is to open a new presentation without specifying a file to open:

```
from pptx import Presentation

prs = Presentation()
prs.save('test.pptx')
```

This creates a new presentation from the built-in default template and saves it unchanged to a file named 'test.pptx'. A couple things to note:

- The so-called "default template" is actually just a PowerPoint file that doesn't have any slides in it, stored with the installed *python-pptx* package. It's the same as what you would get if you created a new presentation from a fresh PowerPoint install, a 4x3 aspect ratio presentation based on the "White" template. Well, except it won't contain any slides. PowerPoint always adds a blank first slide by default.
- You don't need to do anything to it before you save it. If you want to see exactly what that template contains, just look in the 'test.pptx' file this creates.

- We've called it a *template*, but in fact it's just a regular PowerPoint file with all the slides removed. Actual PowerPoint template files (.potx files) are something a bit different. More on those later maybe, but you won't need them to work with *python-pptx*.

REALLY opening a presentation

Okay, so if you want any control at all to speak of over the final presentation, or if you want to change an existing presentation, you need to open one with a filename:

```
prs = Presentation('existing-prs-file.pptx')
prs.save('new-file-name.pptx')
```

Things to note:

- You can open any PowerPoint 2007 or later file this way (.ppt files from PowerPoint 2003 and earlier won't work). While you might not be able to manipulate all the contents yet, whatever is already in there will load and save just fine. The feature set is still being built out, so you can't add or change things like Notes Pages yet, but if the presentation has them *python-pptx* is polite enough to leave them alone and smart enough to save them without actually understanding what they are.
- If you use the same filename to open and save the file, *python-pptx* will obediently overwrite the original file without a peep. You'll want to make sure that's what you intend.

Opening a 'file-like' presentation

python-pptx can open a presentation from a so-called *file-like* object. It can also save to a file-like object. This can be handy when you want to get the source or target presentation over a network connection or from a database and don't want to (or aren't allowed to) fuss with interacting with the file system. In practice this means you can pass an open file or StringIO/BytesIO stream object to open or save a presentation like so:

```
f = open('foobar.pptx')
prs = Presentation(f)
f.close()

# or

with open('foobar.pptx') as f:
    source_stream = StringIO(f.read())
prs = Presentation(source_stream)
source_stream.close()
...
target_stream = StringIO()
prs.save(target_stream)
```

Okay, so you've got a presentation open and are pretty sure you can save it somewhere later. Next step is to get a slide in there ...

Working with Slides

Every slide in a presentation is based on a slide layout. Not surprising then that you have to specify which slide layout to use when you create a new slide. Let's take a minute to understand a few things about slide layouts that we'll need so the slide we add looks the way we want it to.

Slide layout basics

A slide layout is like a template for a slide. Whatever is on the slide layout "shows through" on a slide created with it and formatting choices made on the slide layout are inherited by the slide. This is an important feature for getting a professional-looking presentation deck, where all the slides are formatted consistently. Each slide layout is based on the slide master in a similar way, so you can make presentation-wide formatting decisions on the slide master and layout-specific decisions on the slide layouts. There can actually be multiple slide masters, but I'll pretend for now there's only one. Usually there is.

The presentation themes that come with PowerPoint have about nine slide layouts, with names like *Title*, *Title and Content*, *Title Only*, and *Blank*. Each has zero or more placeholders (mostly not zero), preformatted areas into which you can place a title, multi-level bullets, an image, etc. More on those later.

The slide layouts in a standard PowerPoint theme always occur in the same sequence. This allows content from one deck to be pasted into another and be connected with the right new slide layout:

- Title (presentation title slide)
- Title and Content
- Section Header (sometimes called Segue)
- Two Content (side by side bullet textboxes)
- Comparison (same but additional title for each side by side content box)
- Title Only
- Blank
- Content with Caption
- Picture with Caption

In *python-pptx*, these are `prs.slide_layouts[0]` through `prs.slide_layouts[8]`. However, there's no rule they have to appear in this order, it's just a convention followed by the themes provided with PowerPoint. If the deck you're using as your template has different slide layouts or has them in a different order, you'll have to work out the slide layout indices for yourself. It's pretty easy. Just open it up in Slide Master view in PowerPoint and count down from the top, starting at zero.

Now we can get to creating a new slide.

Adding a slide

Let's use the Title and Content slide layout; a lot of slides do:

```
SLD_LAYOUT_TITLE_AND_CONTENT = 1

prs = Presentation()
slide_layout = prs.slide_layouts[SLD_LAYOUT_TITLE_AND_CONTENT]
slide = prs.slides.add_slide(slide_layout)
```

A few things to note:

- Using a "constant" value like `SLD_LAYOUT_TITLE_AND_CONTENT` is up to you. If you're creating many slides it can be handy to have constants defined so a reader can more easily make sense of what you're doing. There isn't a set of these built into the package because they can't be assured to be right for the starting deck you're using.

- `prs.slide_layouts` is the collection of slide layouts contained in the presentation and has list semantics, at least for item access which is about all you can do with that collection at the moment. Using `prs` for the Presentation instance is purely conventional, but I like it and use it consistently.
- `prs.slides` is the collection of slides in the presentation, also has list semantics for item access, and len() works on it. Note that the method to add the slide is on the slide collection, not the presentation. The `add_slide()` method appends the new slide to the end of the collection. At the time of writing it's the only way to add a slide, but sooner or later I expect someone will want to insert one in the middle, and when they post a feature request for that I expect I'll add an `insert_slide(idx, ...)` method.

Doing other things with slides

Right now, adding a slide is the only operation on the slide collection. On the backlog at the time of writing is deleting a slide and moving a slide to a different position in the list. Copying a slide from one presentation to another turns out to be pretty hard to get right in the general case, so that probably won't come until more of the backlog is burned down.

Up next ...

Ok, now that we have a new slide, let's talk about how to put something on it ...

Understanding Shapes

Pretty much anything on a slide is a shape; the only thing I can think of that can appear on a slide that's not a shape is a slide background. There are between six and ten different types of shape, depending how you count. I'll explain some of the general shape concepts you'll need to make sense of how to work with them and then we'll jump right into working with the specific types.

Technically there are six and only six different types of shapes that can be placed on a slide:

auto shape This is a regular shape, like a rectangle, an ellipse, or a block arrow. They come in a large variety of preset shapes, in the neighborhood of 180 different ones. An auto shape can have a fill and an outline, and can contain text. Some auto shapes have adjustments, the little yellow diamonds you can drag to adjust how round the corners of a rounded rectangle are for example. A text box is also an autoshape, a rectangular one, just by default without a fill and without an outline.

picture A raster image, like a photograph or clip art is referred to as a *picture* in PowerPoint. It's its own kind of shape with different behaviors than an autoshape. Note that an auto shape can have a picture fill, in which an image "shows through" as the background of the shape instead of a fill color or gradient. That's a different thing. But cool.

graphic frame This is the technical name for the container that holds a table, a chart, a smart art diagram, or media clip. You can't add one of these by itself, it just shows up in the file when you add a graphical object. You probably won't need to know anything more about these.

group shape In PowerPoint, a set of shapes can be *grouped*, allowing them to be selected, moved, resized, and even filled as a unit. When you group a set of shapes a group shape gets created to contain those member shapes. You can't actually see these except by their bounding box when the group is selected, and *python-pptx* doesn't support these yet (other than to preserve ones that are already there) so I won't say more about them for now.

line/connector Lines are different from auto shapes because, well, they're linear. Some lines can be connected to other shapes and stay connected when the other shape is moved. These aren't supported yet either so I don't know much more about them. I'd better get to these soon though, they seem like they'd be very handy.

content part I actually have only the vaguest notion of what these are. It has something to do with embedding "foreign" XML like SVG in with the presentation. I'm pretty sure PowerPoint itself doesn't do anything with these. My strategy is to ignore them. Working good so far.

As for real-life shapes, there are these nine types:

- shape shapes – auto shapes with fill and an outline
- text boxes – auto shapes with no fill and no outline
- placeholders – auto shapes that can appear on a slide layout or master and be inherited on slides that use that layout, allowing content to be added that takes on the formatting of the placeholder
- line/connector – as described above
- picture – as described above
- table – that row and column thing
- chart – pie chart, line chart, etc. *python-pptx* doesn't support creating these yet.
- smart art – not supported yet, although preserved if present
- media clip – not supported yet, although preserved if present

Accessing the shapes on a slide

Each slide has a *shape tree* that holds its shapes. It's called a tree because it's hierarchical in the general case; a node in the shape tree can be a group shape which itself can contain shapes and has the same semantics as the shape tree. For most purposes the shape tree has list semantics. You gain access to it like so:

```
shapes = slide.shapes
```

We'll see a lot more of the shape tree in the next few sections.

Up next ...

Okay. That should be enough noodle work to get started. Let's move on to working with AutoShapes.

Working with AutoShapes

Auto shapes are regular shape shapes. Squares, circles, triangles, stars, that sort of thing. There are 182 different auto shapes to choose from. 120 of these have adjustment "handles" you can use to change the shape, sometimes dramatically.

Many shape types share a common set of properties. We'll introduce many of them here because several of those shapes are just a specialized form of AutoShape.

Adding an auto shape

The following code adds a rounded rectangle shape, one inch square, and positioned one inch from the top-left corner of the slide:

```
from pptx.enum.shapes import MSO_SHAPE

shapes = slide.shapes
left = top = width = height = Inches(1.0)
shape = shapes.add_shape(
    MSO_SHAPE.ROUNDED_RECTANGLE, left, top, width, height
)
```

See the *MSO_AUTO_SHAPE_TYPE* enumeration page for a list of all 182 auto shape types.

Understanding English Metric Units

In the prior example we set the position and dimension values to the expression `Inches(1.0)`. What's that about?

Internally, PowerPoint stores length values in *English Metric Units* (EMU). This term might be worth a quick Googling, but the short story is EMU is an integer unit of length, 914400 to the inch. Most lengths in Office documents are stored in EMU. 914400 has the great virtue that it is evenly divisible by a great many common factors, allowing exact conversion between inches and centimeters, for example. Being an integer, it can be represented exactly across serializations and across platforms.

As you might imagine, working directly in EMU is inconvenient. To make it easier, python-pptx provides a collection of value types to allow easy specification and conversion into convenient units:

```
>>> from pptx.util import Inches, Pt
>>> length = Inches(1)
>>> length
914400
>>> length.inches
1.0
>>> length.cm
2.54
>>> length.pt
72.0
>>> length = Pt(72)
>>> length
914400
```

More details are available in the *API documentation for pptx.util*

Shape position and dimensions

All shapes have a position on their slide and have a size. In general, position and size are specified when the shape is created. Position and size can also be read from existing shapes and changed:

```
>>> from pptx.enum.shapes import MSO_SHAPE
>>> left = top = width = height = Inches(1.0)
>>> shape = shapes.add_shape(
>>>     MSO_SHAPE.ROUNDED_RECTANGLE, left, top, width, height
>>> )
>>> shape.left, shape.top, shape.width, shape.height
(914400, 914400, 914400, 914400)
>>> shape.left.inches
1.0
>>> shape.left = Inches(2.0)
>>> shape.left.inches
2.0
```

Fill

AutoShapes have an outline around their outside edge. What appears within that outline is called the shape's *fill*.

The most common type of fill is a solid color. A shape may also be filled with a gradient, a picture, a pattern (like cross-hatching for example), or may have no fill (transparent).

When a color is used, it may be specified as a specific RGB value or a color from the theme palette.

Because there are so many options, the API for fill is a bit complex. This code sets the fill of a shape to red:

```
>>> fill = shape.fill
>>> fill.solid()
>>> fill.fore_color.rgb = RGBColor(255, 0, 0)
```

This sets it to the theme color that appears as 'Accent 1 - 25% Darker' in the toolbar palette:

```
>>> from pptx.enum.dml import MSO_THEME_COLOR
>>> fill = shape.fill
>>> fill.solid()
>>> fill.fore_color.theme_color = MSO_THEME_COLOR.ACCENT_1
>>> fill.fore_color.brightness = -0.25
```

This sets the shape fill to transparent, or 'No Fill' as it's called in the PowerPoint UI:

```
>>> shape.fill.background()
```

As you can see, the first step is to specify the desired fill type by calling the corresponding method on fill. Doing so actually changes the properties available on the fill object. For example, referencing `.fore_color` on a fill object after calling its `.background()` method will raise an exception:

```
>>> fill = shape.fill
>>> fill.solid()
>>> fill.fore_color
<pptx.dml.color.ColorFormat object at 0x10ce20910>
>>> fill.background()
>>> fill.fore_color
Traceback (most recent call last):
  ...
TypeError: a transparent (background) fill has no foreground color
```

Line

The outline of an AutoShape can also be formatted, including setting its color, width, dash (solid, dashed, dotted, etc.), line style (single, double, thick-thin, etc.), end cap, join type, and others. At the time of writing, color and width can be set using python-pptx:

```
>>> line = shape.line
>>> line.color.rgb = RGBColor(255, 0, 0)
>>> line.color.brightness = 0.5  # 50% lighter
>>> line.width = Pt(2.5)
```

Theme colors can be used on lines too:

```
>>> line.color.theme_color = MSO_THEME_COLOR.ACCENT_6
```

`Shape.line` has the attribute `.color`. This is essentially a shortcut for:

```
>>> line.fill.solid()
>>> line.fill.fore_color
```

This makes sense for line formatting because a shape outline is most frequently set to a solid color. Accessing the fill directly is required, for example, to set the line to transparent:

```
>>> line.fill.background()
```

Line width

The shape outline also has a read/write width property:

```
>>> line.width
9525
>>> line.width.pt
0.75
>>> line.width = Pt(2.0)
>>> line.width.pt
2.0
```

Adjusting an autoshape

Many auto shapes have adjustments. In PowerPoint, these show up as little yellow diamonds you can drag to change the look of the shape. They're a little fiddly to work with via a program, but if you have the patience to get them right, you can achieve some remarkable effects with great precision.

Shape Adjustment Concepts

There are a few concepts it's worthwhile to grasp before trying to do serious work with adjustments.

First, adjustments are particular to a specific auto shape type. Each auto shape has between zero and eight adjustments. What each of them does is arbitrary and depends on the shape design.

Conceptually, adjustments are guides, in many ways like the light blue ones you can align to in the PowerPoint UI and other drawing apps. These don't show, but they operate in a similar way, each defining an x or y value that part of the shape will align to, changing the proportions of the shape.

Adjustment values are large integers, each based on a nominal value of 100,000. The effective value of an adjustment is proportional to the width or height of the shape. So a value of 50,000 for an x-coordinate adjustment corresponds to half the width of the shape; a value of 75,000 for a y-coordinate adjustment corresponds to 3/4 of the shape height.

Adjustment values can be negative, generally indicating the coordinate is to the left or above the top left corner (origin) of the shape. Values can also be subject to limits, meaning their effective value cannot be outside a prescribed range. In practice this corresponds to a point not being able to extend beyond the left side of the shape, for example.

Spending some time fooling around with shape adjustments in PowerPoint is time well spent to build an intuitive sense of how they behave. You also might want to have `opc-diag` installed so you can look at the XML values that are generated by different adjustments as a head start on developing your adjustment code.

The following code formats a callout shape using its adjustments:

```
callout_sp = shapes.add_shape(
    MSO_SHAPE.LINE_CALLOUT_2_ACCENT_BAR, left, top, width, height
)

# get the callout line coming out of the right place
adjs = callout_sp.adjustments
adjs[0] = 0.5   # vert pos of junction in margin line, 0 is top
adjs[1] = 0.0   # horz pos of margin ln wrt shape width, 0 is left side
adjs[2] = 0.5   # vert pos of elbow wrt margin line, 0 is top
adjs[3] = -0.1  # horz pos of elbow wrt shape width, 0 is margin line
adjs[4] = 3.0   # vert pos of line end wrt shape height, 0 is top
a5 = adjs[3] - (adjs[4] - adjs[0]) * height/width
adjs[5] = a5    # horz pos of elbow wrt shape width, 0 is margin line

# rotate 45 degrees counter-clockwise
callout_sp.rotation = -45.0
```

Understanding placeholders

Intuitively, a placeholder is a pre-formatted container into which content can be placed. By providing pre-set formatting to its content, it places many of the formatting choices in the hands of the template designer while allowing the end-user to concentrate on the actual content. This speeds the presentation development process while encouraging visual consistency in slides created from the same template.

While their typical end-user behaviors are relatively simple, the structures that support their operation are more complex. This page is for those who want to better understand the architecture of the placeholder subsystem and perhaps be less prone to confusion at its sometimes puzzling behavior. If you don't care why they work and just want to know how to work with them, you may want to skip forward to the following page *Working with placeholders*.

A placeholder is a shape

Placeholders are an orthogonal category of shape, which is to say multiple shape types can be placeholders. In particular, the auto shape (*p:sp* element), picture (*p:pic* element), and graphic frame (*p:graphicFrame*) shape types can be a placeholder. The group shape (*p:grpSp*), connector (*p:cxnSp*), and content part (*p:contentPart*) shapes cannot be a placeholder. A graphic frame placeholder can contain a table, a chart, or SmartArt.

Placeholder types

There are 18 types of placeholder.

Title, Center Title, Subtitle, Body These placeholders typically appear on a conventional "word chart" containing text only, often organized as a title and a series of bullet points. All of these placeholders can accept text only.

Content This multi-purpose placeholder is the most commonly used for the body of a slide. When unpopulated, it displays 6 buttons to allow insertion of a table, a chart, SmartArt, a picture, clip art, or a media clip.

Picture, Clip Art These both allow insertion of an image. The insert button on a clip art placeholder brings up the clip art gallery rather than an image file chooser, but otherwise these behave the same.

Chart, Table, Smart Art These three allow the respective type of rich graphical content to be inserted.

Media Clip Allows a video or sound recording to be inserted.

Date, Footer, Slide Number These three appear on most slide masters and slide layouts, but do not behave as most users would expect. These also commonly appear on the Notes Master and Handout Master.

Header Only valid on the Notes Master and Handout Master.

Vertical Body, Vertical Object, Vertical Title Used with vertically oriented languages such as Japanese.

Unpopulated vs. populated

A placeholder on a slide can be empty or filled. This is most evident with a picture placeholder. When unpopulated, a placeholder displays customizable prompt text. A rich content placeholder will also display one or more content insertion buttons when empty.

A text-only placeholder enters "populated" mode when the first character of text is entered and returns to "unpopulated" mode when the last character of text is removed. A rich-content placeholder enters populated mode when content such as a picture is inserted and returns to unpopulated mode when that content is deleted. In order to delete a populated placeholder, the shape must be deleted *twice*. The first delete removes the content and restores the placeholder to unpopulated mode. An additional delete will remove the placeholder itself. A deleted placeholder can be restored by reapplying the layout.

Placholders inherit

A placeholder appearing on a slide is only part of the overall placeholder mechanism. Placeholder behavior requires three different categories of placeholder shape; those that exist on a slide master, those on a slide layout, and those that ultimately appear on a slide in a presentation.

These three categories of placeholder participate in a property inheritance hierarchy, either as an inheritor, an inheritee, or both. Placeholder shapes on masters are inheritees only. Conversely placeholder shapes on slides are inheritors only. Placeholders on slide layouts are both, a possible inheritor from a slide master placeholder and an inheritee to placeholders on slides linked to that layout.

A layout inherits from its master differently than a slide inherits from its layout. A layout placeholder inherits from the master placeholder sharing the same type. A slide placeholder inherits from the layout placeholder having the same *idx* value.

In general, all formatting properties are inherited from the "parent" placeholder. This includes position and size as well as fill, line, and font. Any directly applied formatting overrides the corresponding inherited value. Directly applied formatting can be removed be reapplying the layout.

Glossary

placeholder shape A shape on a slide that inherits from a layout placeholder.

layout placeholder a shorthand name for the placeholder shape on the slide layout from which a particular placeholder on a slide inherits shape properties

master placeholder the placeholder shape on the slide master which a layout placeholder inherits from, if any.

Working with placeholders

Placeholders can make adding content a lot easier. If you've ever added a new textbox to a slide from scratch and noticed how many adjustments it took to get it the way you wanted you understand why. The placeholder is in the right position with the right font size, paragraph alignment, bullet style, etc., etc. Basically you can just click and type in some text and you've got a slide.

A placeholder can be also be used to place a rich-content object on a slide. A picture, table, or chart can each be inserted into a placeholder and so take on the position and size of the placeholder, as well as certain of its formatting attributes.

Access a placeholder

Every placeholder is also a shape, and so can be accessed using the *shapes* property of a slide. However, when looking for a particular placeholder, the *placeholders* property can make things easier.

The most reliable way to access a known placeholder is by its `idx` value. The `idx` value of a placeholder is the integer key of the slide layout placeholder it inherits properties from. As such, it remains stable throughout the life of the slide and will be the same for any slide created using that layout.

It's usually easy enough to take a look at the placeholders on a slide and pick out the one you want:

```
>>> prs = Presentation()
>>> slide = prs.slides.add_slide(prs.slide_layouts[8])
>>> for shape in slide.placeholders:
...     print('%d %s' % (shape.placeholder_format.idx, shape.name))
...
0  Title 1
1  Picture Placeholder 2
2  Text Placeholder 3
```

... then, having the known index in hand, to access it directly:

```
>>> slide.placeholders[1]
<pptx.parts.slide.PicturePlaceholder object at 0x10d094590>
>>> slide.placeholders[2].name
'Text Placeholder 3'
```

Note: Item access on the placeholders collection is like that of a dictionary rather than a list. While the key used above is an integer, the lookup is on *idx* values, not position in a sequence. If the provided value does not match the *idx* value of one of the placeholders, `KeyError` will be raised. *idx* values are not necessarily contiguous.

In general, the `idx` value of a placeholder from a built-in slide layout (one provided with PowerPoint) will be between 0 and 5. The title placeholder will always have `idx` 0 if present and any other placeholders will follow in sequence, top to bottom and left to right. A placeholder added to a slide layout by a user in PowerPoint will receive an `idx` value starting at 10.

Identify and Characterize a placeholder

A placeholder behaves differently that other shapes in some ways. In particular, the value of its *shape_type* attribute is unconditionally `MSO_SHAPE_TYPE.PLACEHOLDER` regardless of what type of placeholder it is or what type of content it contains:

```
>>> prs = Presentation()
>>> slide = prs.slides.add_slide(prs.slide_layouts[8])
>>> for shape in slide.shapes:
...     print('%s' % shape.shape_type)
...
PLACEHOLDER (14)
PLACEHOLDER (14)
PLACEHOLDER (14)
```

To find out more, it's necessary to inspect the contents of the placeholder's *placeholder_format* attribute. All shapes have this attribute, but accessing it on a non-placeholder shape raises `ValueError`. The *is_placeholder* attribute can be used to determine whether a shape is a placeholder:

```
>>> for shape in slide.shapes:
...     if shape.is_placeholder:
...         phf = shape.placeholder_format
...         print('%d, %s' % (phf.idx, phf.type))
...
0, TITLE (1)
1, PICTURE (18)
2, BODY (2)
```

Another way a placeholder acts differently is that it inherits its position and size from its layout placeholder. This inheritance is overridden if the position and size of a placeholder are changed.

Insert content into a placeholder

Certain placeholder types have specialized methods for inserting content. In the current release, the *picture*, *table*, and *chart* placeholders have content insertion methods. Text can be inserted into *title* and *body* placeholders in the same way text is inserted into an auto shape.

PicturePlaceholder.insert_picture()

The picture placeholder has an *insert_picture()* method:

```
>>> prs = Presentation()
>>> slide = prs.slides.add_slide(prs.slide_layouts[8])
>>> placeholder = slide.placeholders[1]  # idx key, not position
>>> placeholder.name
'Picture Placeholder 2'
>>> placeholder.placeholder_format.type
PICTURE (18)
>>> picture = placeholder.insert_picture('my-image.png')
```

Note: A reference to a picture placeholder becomes invalid after its *insert_picture()* method is called. This is because the process of inserting a picture replaces the original *p:sp* XML element with a new *p:pic* element containing the picture. Any attempt to use the original placeholder reference after the call will raise `AttributeError`. The new placeholder is the return value of the `insert_picture()` call and may also be obtained from the placeholders collection using the same *idx* key.

A picture inserted in this way is stretched proportionately and cropped to fill the entire placeholder. Best results are achieved when the aspect ratio of the source image and placeholder are the same. If the picture is taller in aspect than the placeholder, its top and bottom are cropped evenly to fit. If it is wider, its left and right sides are cropped evenly. Cropping can be adjusted using the crop properties on the placeholder, such as *crop_bottom*.

TablePlaceholder.insert_table()

The table placeholder has an *insert_table()* method. The built-in template has no layout containing a table placeholder, so this example assumes a starting presentation named `having-table-placeholder.pptx` having a table placeholder with idx 10 on its second slide layout:

```
>>> prs = Presentation('having-table-placeholder.pptx')
>>> slide = prs.slides.add_slide(prs.slide_layouts[1])
>>> placeholder = slide.placeholders[10]  # idx key, not position
>>> placeholder.name
'Table Placeholder 1'
>>> placeholder.placeholder_format.type
TABLE (12)
>>> graphic_frame = placeholder.insert_table(rows=2, cols=2)
>>> table = graphic_frame.table
>>> len(table.rows), len(table.columns)
(2, 2)
```

A table inserted in this way has the position and width of the original placeholder. Its height is proportional to the number of rows.

Like all rich-content insertion methods, a reference to a table placeholder becomes invalid after its *insert_table()* method is called. This is because the process of inserting rich content replaces the original *p:sp* XML element with a new element, a *p:graphicFrame* in this case, containing the rich-content object. Any attempt to use the original placeholder reference after the call will raise `AttributeError`. The new placeholder is the return value of the `insert_table()` call and may also be obtained from the placeholders collection using the original *idx* key, 10 in this case.

Note: The return value of the *insert_table()* method is a *PlaceholderGraphicFrame* object, which has all the properties and methods of a *GraphicFrame* object along with those specific to placeholders. The inserted table is contained in the graphic frame and can be obtained using its *table* property.

ChartPlaceholder.insert_chart()

The chart placeholder has an *insert_chart()* method. The presentation template built into *python-pptx* has no layout containing a chart placeholder, so this example assumes a starting presentation named `having-chart-placeholder.pptx` having a chart placeholder with idx 10 on its second slide layout:

```
>>> from pptx.chart.data import ChartData
>>> from pptx.enum.chart import XL_CHART_TYPE

>>> prs = Presentation('having-chart-placeholder.pptx')
>>> slide = prs.slides.add_slide(prs.slide_layouts[1])

>>> placeholder = slide.placeholders[10]  # idx key, not position
>>> placeholder.name
'Chart Placeholder 9'
>>> placeholder.placeholder_format.type
CHART (12)

>>> chart_data = ChartData()
>>> chart_data.categories = ['Yes', 'No']
>>> chart_data.add_series('Series 1', (42, 24))

>>> graphic_frame = placeholder.insert_chart(XL_CHART_TYPE.PIE, chart_data)
>>> chart = graphic_frame.chart
>>> chart.chart_type
PIE (5)
```

A chart inserted in this way has the position and size of the original placeholder.

Note the return value from *insert_chart()* is a *PlaceholderGraphicFrame* object, not the chart itself. A *PlaceholderGraphicFrame* object has all the properties and methods of a *GraphicFrame* object along with those specific to placeholders. The inserted chart is contained in the graphic frame and can be obtained using its *chart* property.

Like all rich-content insertion methods, a reference to a chart placeholder becomes invalid after its *insert_chart()* method is called. This is because the process of inserting rich content replaces the original *p:sp* XML element with a new element, a *p:graphicFrame* in this case, containing the rich-content object. Any attempt to use the original placeholder reference after the call will raise `AttributeError`. The new placeholder is the return value of the `insert_chart()` call and may also be obtained from the placeholders collection using the original *idx* key, 10 in this case.

Setting the slide title

Almost all slide layouts have a title placeholder, which any slide based on the layout inherits when the layout is applied. Accessing a slide's title is a common operation and there's a dedicated attribute on the shape tree for it:

```
title_placeholder = slide.shapes.title
title_placeholder.text = 'Air-speed Velocity of Unladen Swallows'
```

Working with text

Auto shapes and table cells can contain text. Other shapes can't. Text is always manipulated the same way, regardless of its container.

Text exists in a hierarchy of three levels:

- *Shape.text_frame*
- *TextFrame.paragraphs*
- *_Paragraph.runs*

All the text in a shape is contained in its *text frame*. A text frame has vertical alignment, margins, wrapping and auto-fit behavior, a rotation angle, some possible 3D visual features, and can be set to format its text into multiple columns. It also contains a sequence of paragraphs, which always contains at least one paragraph, even when empty.

A paragraph has line spacing, space before, space after, available bullet formatting, tabs, outline/indentation level, and horizontal alignment. A paragraph can be empty, but if it contains any text, that text is contained in one or more runs.

A run exists to provide character level formatting, including font typeface, size, and color, an optional hyperlink target URL, bold, italic, and underline styles, strikethrough, kerning, and a few capitalization styles like all caps.

Let's run through these one by one. Only features available in the current release are shown.

Accessing the text frame

As mentioned, not all shapes have a text frame. So if you're not sure and you don't want to catch the possible exception, you'll want to check before attempting to access it:

```
for shape in slide.shapes:
    if not shape.has_text_frame:
        continue
    text_frame = shape.text_frame
    # do things with the text frame
    ...
```

Accessing paragraphs

A text frame always contains at least one paragraph. This causes the process of getting multiple paragraphs into a shape to be a little clunkier than one might like. Say for example you want a shape with three paragraphs:

```
paragraph_strs = [
    'Egg, bacon, sausage and spam.',
    'Spam, bacon, sausage and spam.',
    'Spam, egg, spam, spam, bacon and spam.'
]

text_frame = shape.text_frame
text_frame.clear()  # remove any existing paragraphs, leaving one empty one

p = text_frame.paragraphs[0]
p.text = paragraph_strs[0]

for para_str in paragraph_strs[1:]:
    p = text_frame.add_paragraph()
    p.text = para_str
```

Adding text

Only runs can actually contain text. Assigning a string to the `.text` attribute on a shape, text frame, or paragraph is a shortcut method for placing text in a run contained by those objects. The following two snippets produce the same result:

```
shape.text = 'foobar'

# is equivalent to ...

text_frame = shape.text_frame
text_frame.clear()
p = text_frame.paragraphs[0]
run = p.add_run()
run.text = 'foobar'
```

Applying text frame-level formatting

The following produces a shape with a single paragraph, a slightly wider bottom than top margin (these default to 0.05”), no left margin, text aligned top, and word wrapping turned off. In addition, the auto-size behavior is set to adjust the width and height of the shape to fit its text. Note that vertical alignment is set on the text frame. Horizontal alignment is set on each paragraph:

```
from pptx.util import Inches
from pptx.enum.text import MSO_ANCHOR, MSO_AUTO_SIZE

text_frame = shape.text_frame
text_frame.text = 'Spam, eggs, and spam'
text_frame.margin_bottom = Inches(0.08)
text_frame.margin_left = 0
```

```
text_frame.vertical_anchor = MSO_ANCHOR.TOP
text_frame.word_wrap = False
text_frame.auto_size = MSO_AUTO_SIZE.SHAPE_TO_FIT_TEXT
```

The possible values for `TextFrame.auto_size` and `TextFrame.vertical_anchor` are specified by the enumeration *MSO_AUTO_SIZE* and *MSO_VERTICAL_ANCHOR* respectively.

Applying paragraph formatting

The following produces a shape containing three left-aligned paragraphs, the second and third indented (like sub-bullets) under the first:

```
from pptx.enum.text import PP_ALIGN

paragraph_strs = [
    'Egg, bacon, sausage and spam.',
    'Spam, bacon, sausage and spam.',
    'Spam, egg, spam, spam, bacon and spam.'
]

text_frame = shape.text_frame
text_frame.clear()

p = text_frame.paragraphs[0]
p.text = paragraph_strs[0]
p.alignment = PP_ALIGN.LEFT

for para_str in paragraph_strs[1:]:
    p = text_frame.add_paragraph()
    p.text = para_str
    p.alignment = PP_ALIGN.LEFT
    p.level = 1
```

Applying character formatting

Character level formatting is applied at the run level, using the `.font` attribute. The following formats a sentence in 18pt Calibri Bold and applies the theme color Accent 1.

```
from pptx.dml.color import RGBColor
from pptx.enum.dml import MSO_THEME_COLOR
from pptx.util import Pt

text_frame = shape.text_frame
text_frame.clear()  # not necessary for newly-created shape

p = text_frame.paragraphs[0]
run = p.add_run()
run.text = 'Spam, eggs, and spam'

font = run.font
font.name = 'Calibri'
font.size = Pt(18)
font.bold = True
font.italic = None  # cause value to be inherited from theme
font.color.theme_color = MSO_THEME_COLOR.ACCENT_1
```

If you prefer, you can set the font color to an absolute RGB value. Note that this will not change color when the theme is changed:

```
font.color.rgb = RGBColor(0xFF, 0x7F, 0x50)
```

A run can also be made into a hyperlink by providing a target URL:

```
run.hyperlink.address = 'https://github.com/scanny/python-pptx'
```

Working with Notes Slides

A slide can have notes associated with it. These are perhaps most commonly encountered in the notes pane, below the slide in PowerPoint "Normal" view where it may say "Click to add notes".

The notes added here appear each time that slide is present in the main pane. They also appear in *Presenter View* and in *Notes Page* view, both available from the menu.

Notes can contain rich text, commonly bullets, bold, varying font sizes and colors, etc. The Notes Page view has somewhat more powerful tools for editing the notes text than the note pane in Normal view.

In the API and the underlying XML, the object that contains the text is known as a *Notes Slide*. This is because internally, a notes slide is actually a specialized instance of a slide. It contains shapes, many of which are placeholders, and allows inserting of new shapes such as pictures (a logo perhaps) auto shapes, tables, and charts. Consequently, working with a notes slide is very much like working with a regular slide.

Each slide can have zero or one notes slide. A notes slide is created the first time it is used, generally perhaps by adding notes text to a slide. Once created, it stays, even if all the text is deleted.

The Notes Master

A new notes slide is created using the *Notes Master* as a template. A presentation has no notes master when newly created in PowerPoint. One is created according to a PowerPoint-internal preset default the first time it is needed, which is generally when the first notes slide is created. It's possible one can also be created by entering the Notes Master view and almost certainly is created by editing the master found there (haven't tried it though). A presentation can have at most one notes master.

The notes master governs the look and feel of notes pages, which can be viewed on-screen but are really designed for printing out. So if you want your notes page print-outs to look different from the default, you can make a lot of customizations by editing the notes master. You access the notes master editor using View > Master > Notes Master on the menu (on my version at least). Notes slides created using *python-pptx* will have the look and feel of the notes master in the presentation file you opened to create the presentation.

On creation, certain placeholders (slide image, notes, slide number) are copied from the notes master onto the new notes slide (if they have not been removed from the master). These "cloned" placeholders inherit position, size, and formatting from their corresponding notes master placeholder. If the position, size, or formatting of a notes slide placeholder is changed, the changed property is no long inherited (unchanged properties, however, continue to be inherited).

Notes Slide basics

Enough talk, let's show some code. Let's say you have a slide you're working with and you want to see if it has a notes slide yet:

```
>>> slide.has_notes_slide
False
```

Ok, not yet. Good. Let's add some notes:

```
>>> notes_slide = slide.notes_slide
>>> text_frame = notes_slide.notes_text_frame
>>> text_frame.text = 'foobar'
```

Alright, simple enough. Let's look at what happened here:

- `slide.notes_slide` gave us the notes slide. In this case, it first created that notes slide based on the notes master. If there was no notes master, it created that too. So a lot of things can happen behind the scenes with this call the first time you call it, but if we called it again it would just give us back the reference to the same notes slide, which it caches, once retrieved.
- `notes_slide.notes_text_frame` gave us the *TextFrame* object that contains the actual notes. The reason it's not just `notes_slide.text_frame` is that there are potentially more than one. What this is doing behind the scenes is finding the placeholder shape that contains the notes (as opposed to the slide image, header, slide number, etc.) and giving us *that* particular text frame.
- A text frame in a notes slide works the same as one in a regular slide. More precisely, a text frame on a shape in a notes slide works the same as in any other shape. We used the `.text` property to quickly pop some text in there.

Using the text frame, you can add an arbitrary amount of text, formatted however you want.

Notes Slide Placeholders

What we haven't explicitly seen so far is the shapes on a slide master. It's easy to get started with that:

```
>>> notes_placeholder = notes_slide.notes_placeholder
```

This notes placeholder is just like a body placeholder we saw a couple sections back. You can change its position, size, and many other attributes, as well as get at its text via its text frame.

You can also access the other placeholders:

```
>>> for placeholder in notes_slide.placeholders:
...   print placeholder.placeholder_format.type
...
SLIDE_IMAGE (101)
BODY (2)
SLIDE_NUMBER (13)
```

and also the shapes (a superset of the placeholders):

```
>>> for shape in notes_slide.shapes:
...   print shape
...
<pptx.shapes.placeholder.NotesSlidePlaceholder object at 0x11091e890>
<pptx.shapes.placeholder.NotesSlidePlaceholder object at 0x11091e750>
<pptx.shapes.placeholder.NotesSlidePlaceholder object at 0x11091e990>
```

In the common case, the notes slide contains only placeholders. However, if you added an image, for example, to the notes slide, that would show up as well. Note that if you added that image to the notes master, perhaps a logo, it would appear on the notes slide "visually", but would not appear as a shape in the notes slide shape collection. Rather, it is visually "inherited" from the notes master.

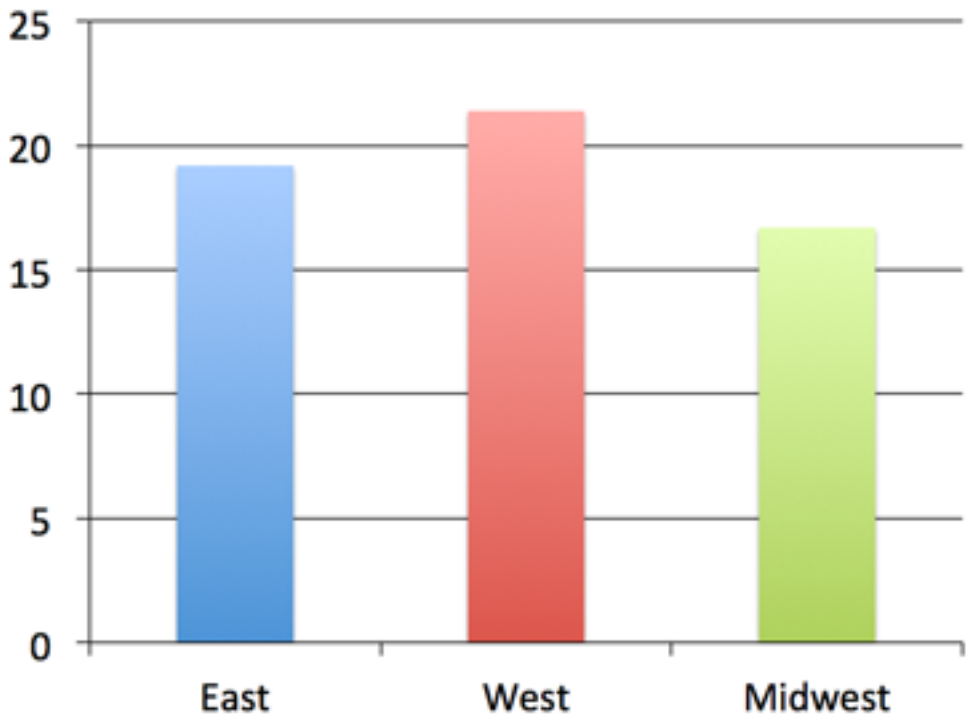
Working with charts

python-pptx supports adding charts and modifying existing ones. So far, 2D bar and column, line, and pie charts are supported.

Adding a chart

The following code adds a single-series column chart in a new presentation:

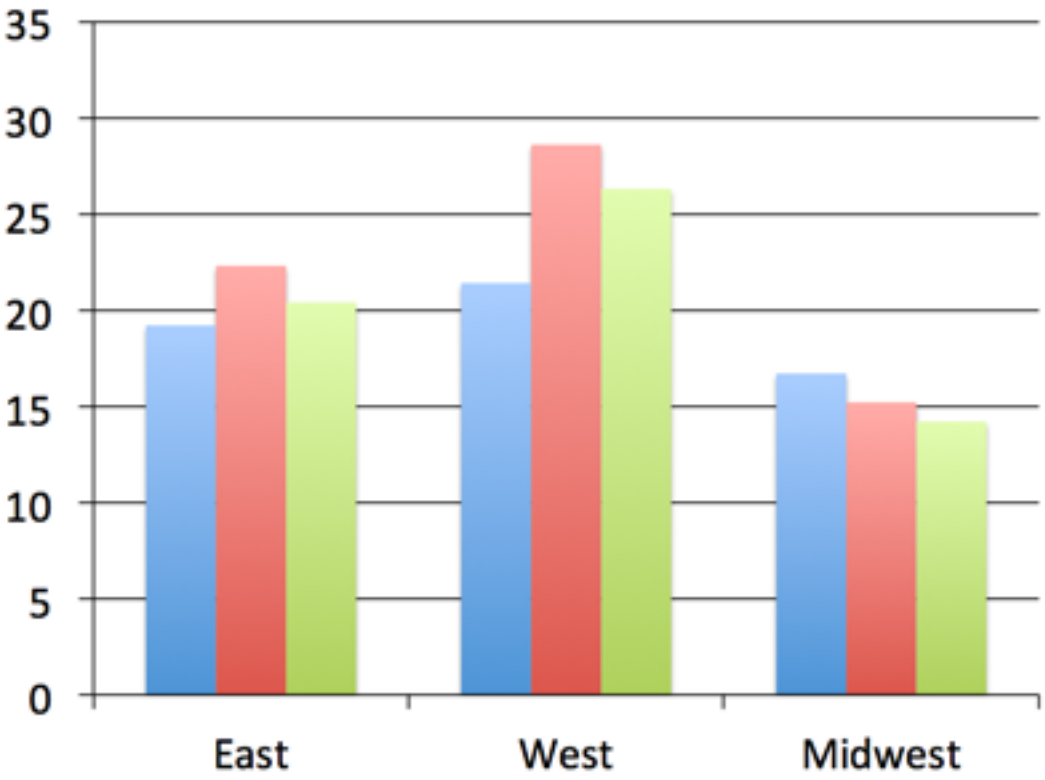
```
from pptx import Presentation
from pptx.chart.data import ChartData
from pptx.enum.chart import XL_CHART_TYPE
from pptx.util import Inches

# create presentation with 1 slide ------
prs = Presentation()
slide = prs.slides.add_slide(prs.slide_layouts[5])

# define chart data ---------------------
chart_data = ChartData()
chart_data.categories = ['East', 'West', 'Midwest']
chart_data.add_series('Series 1', (19.2, 21.4, 16.7))

# add chart to slide --------------------
x, y, cx, cy = Inches(2), Inches(2), Inches(6), Inches(4.5)
slide.shapes.add_chart(
    XL_CHART_TYPE.COLUMN_CLUSTERED, x, y, cx, cy, chart_data
)

prs.save('chart-01.pptx')
```



Customizing things a bit

The remaining code will leave out code we've already seen and only show imports, for example, when they're used for the first time, just to keep the focus on the new bits. Let's create a multi-series chart to use for these examples:

```
chart_data = ChartData()
chart_data.categories = ['East', 'West', 'Midwest']
chart_data.add_series('Q1 Sales', (19.2, 21.4, 16.7))
chart_data.add_series('Q2 Sales', (22.3, 28.6, 15.2))
chart_data.add_series('Q3 Sales', (20.4, 26.3, 14.2))

graphic_frame = slide.shapes.add_chart(
    XL_CHART_TYPE.COLUMN_CLUSTERED, x, y, cx, cy, chart_data
)

chart = graphic_frame.chart
```



Notice that we captured the shape reference returned by the *add_chart()* call as `graphic_frame` and then extracted the chart object from the graphic frame using its *chart* property. We'll need the chart reference to get to the properties we'll need in the next steps. The *add_chart()* method doesn't directly return the chart object. That's because a chart is not itself a shape. Rather it's a graphical (DrawingML) object *contained* in the graphic frame shape. Tables work this way too, also being contained in a graphic frame shape.

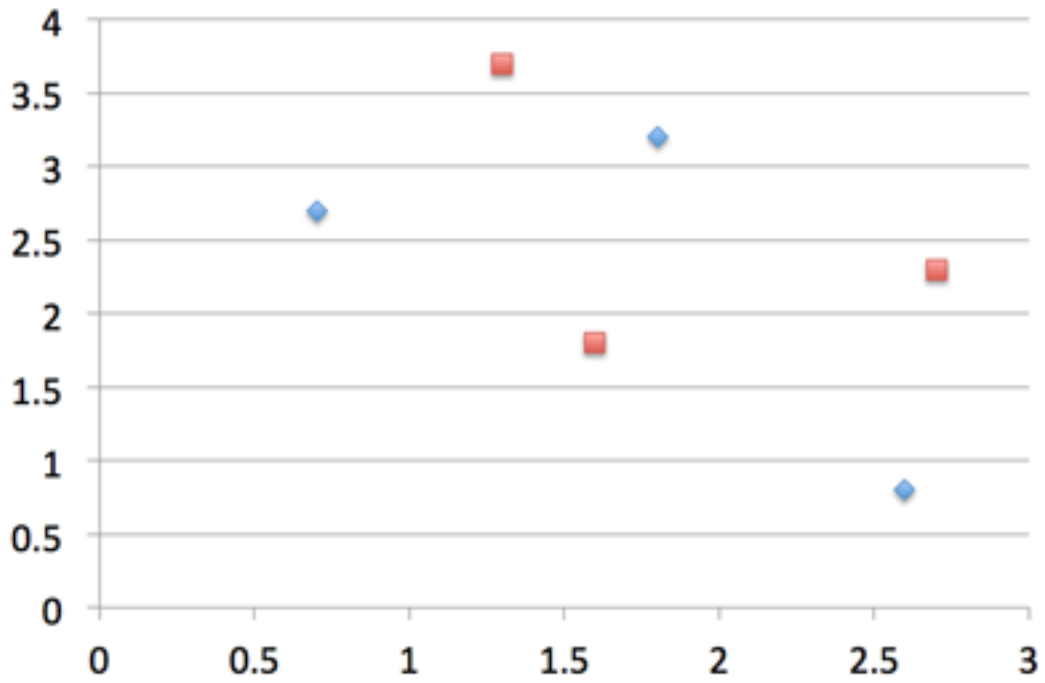
XY and Bubble charts

The charts so far use a *discrete* set of values for the independent variable (the X axis, roughly speaking). These are perfect when your values fall into a well-defined set of categories. However, there are many cases, particularly in science and engineering, where the independent variable is a continuous value, such as temperature or frequency. These are supported in PowerPoint by XY (aka. scatter) charts. A bubble chart is essentially an XY chart where the marker size is used to reflect an additional value, effectively adding a third dimension to the chart.

Because the independent variable is continuous, in general, the series do not all share the same X values. This requires a somewhat different data structure and that is provided for by distinct *XyChartData* and *BubbleChartData* objects used to specify the data behind charts of these types:

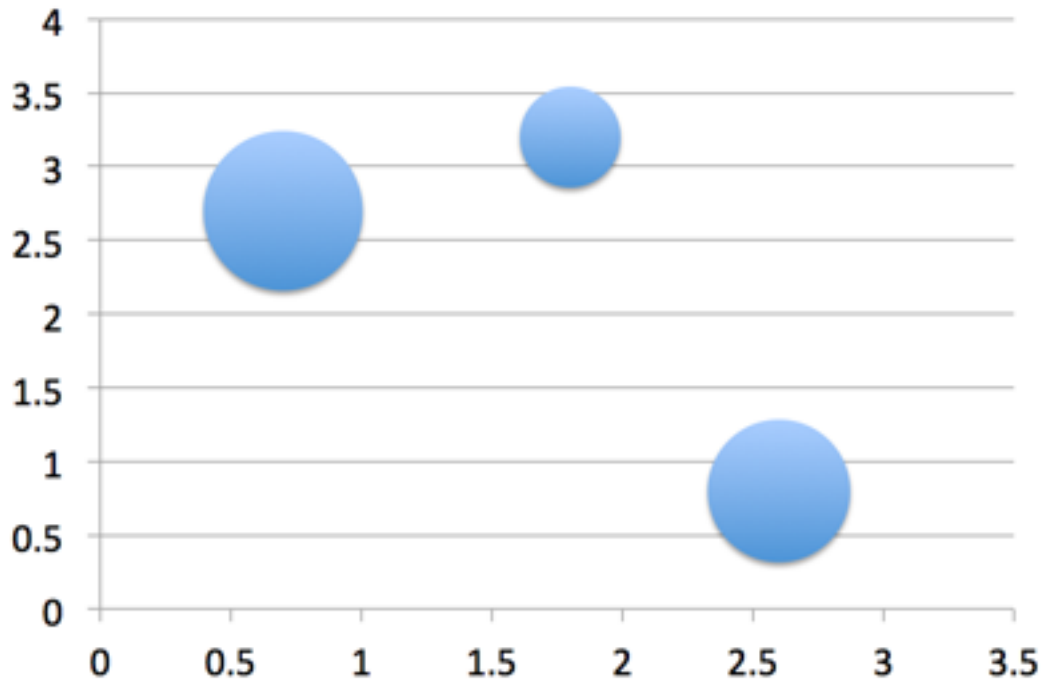
```
chart_data = XyChartData()

series_1 = chart_data.add_series('Model 1')
series_1.add_data_point(0.7, 2.7)
series_1.add_data_point(1.8, 3.2)
series_1.add_data_point(2.6, 0.8)

series_2 = chart_data.add_series('Model 2')
series_2.add_data_point(1.3, 3.7)
series_2.add_data_point(2.7, 2.3)
series_2.add_data_point(1.6, 1.8)
```

```
chart = slide.shapes.add_chart(
    XL_CHART_TYPE.XY_SCATTER, x, y, cx, cy, chart_data
).chart
```



Creation of a bubble chart is very similar, having an additional value for each data point that specifies the bubble size:

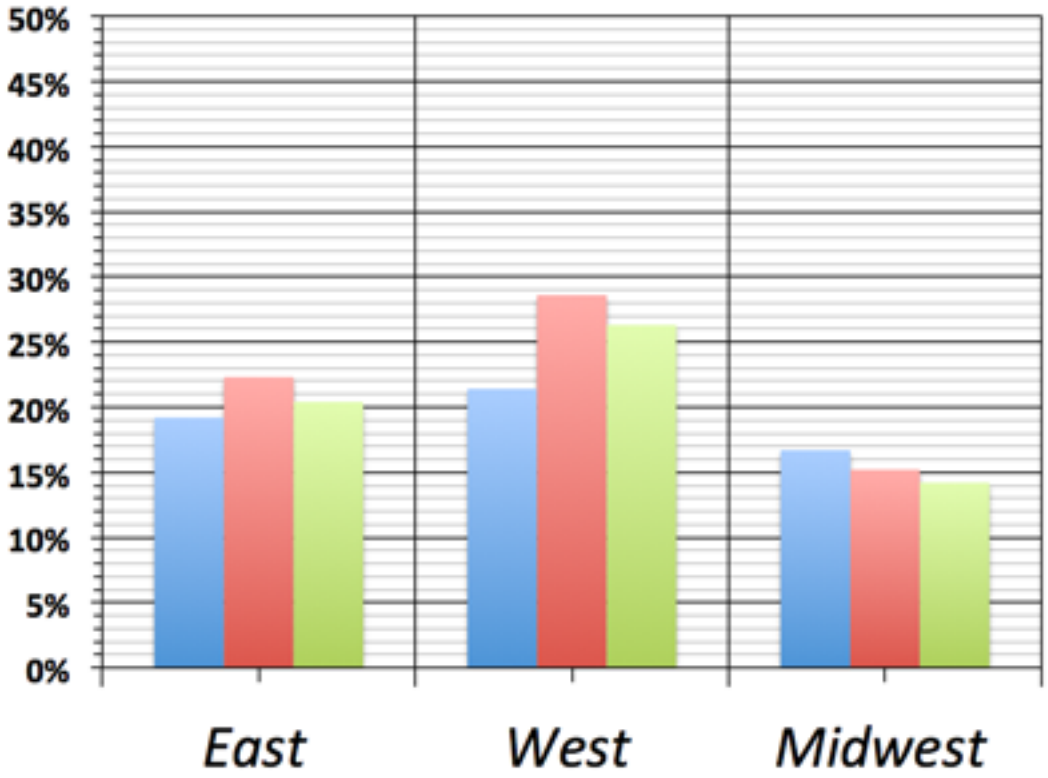
```
chart_data = BubbleChartData()

series_1 = chart_data.add_series('Series 1')
series_1.add_data_point(0.7, 2.7, 10)
series_1.add_data_point(1.8, 3.2, 4)
series_1.add_data_point(2.6, 0.8, 8)

chart = slide.shapes.add_chart(
    XL_CHART_TYPE.BUBBLE, x, y, cx, cy, chart_data
).chart
```



Axes

Let's change up the category and value axes a bit:

```
from pptx.enum.chart import XL_TICK_MARK
from pptx.util import Pt

category_axis = chart.category_axis
category_axis.has_major_gridlines = True
category_axis.minor_tick_mark = XL_TICK_MARK.OUTSIDE
category_axis.tick_labels.font.italic = True
category_axis.tick_labels.font.size = Pt(24)

value_axis = chart.value_axis
value_axis.maximum_scale = 50.0
value_axis.minor_tick_mark = XL_TICK_MARK.OUTSIDE
value_axis.has_minor_gridlines = True

tick_labels = value_axis.tick_labels
tick_labels.number_format = '0"%"'
tick_labels.font.bold = True
tick_labels.font.size = Pt(14)
```

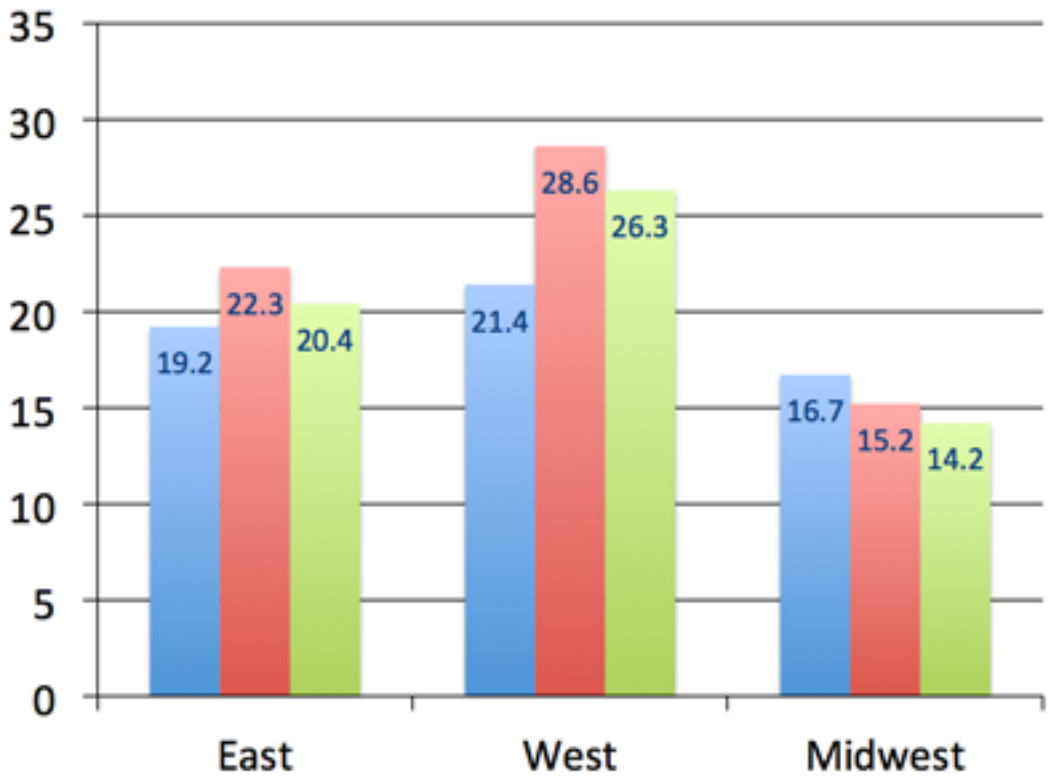


Okay, that was probably going a bit too far. But it gives us an idea of the kinds of things we can do with the value and category axes. Let's undo this part and go back to the version we had before.

Data Labels

Let's add some data labels so we can see exactly what the value for each bar is:

```
from pptx.dml.color import RGBColor
from pptx.enum.chart import XL_LABEL_POSITION

plot = chart.plots[0]
plot.has_data_labels = True
data_labels = plot.data_labels

data_labels.font.size = Pt(13)
data_labels.font.color.rgb = RGBColor(0x0A, 0x42, 0x80)
data_labels.position = XL_LABEL_POSITION.INSIDE_END
```



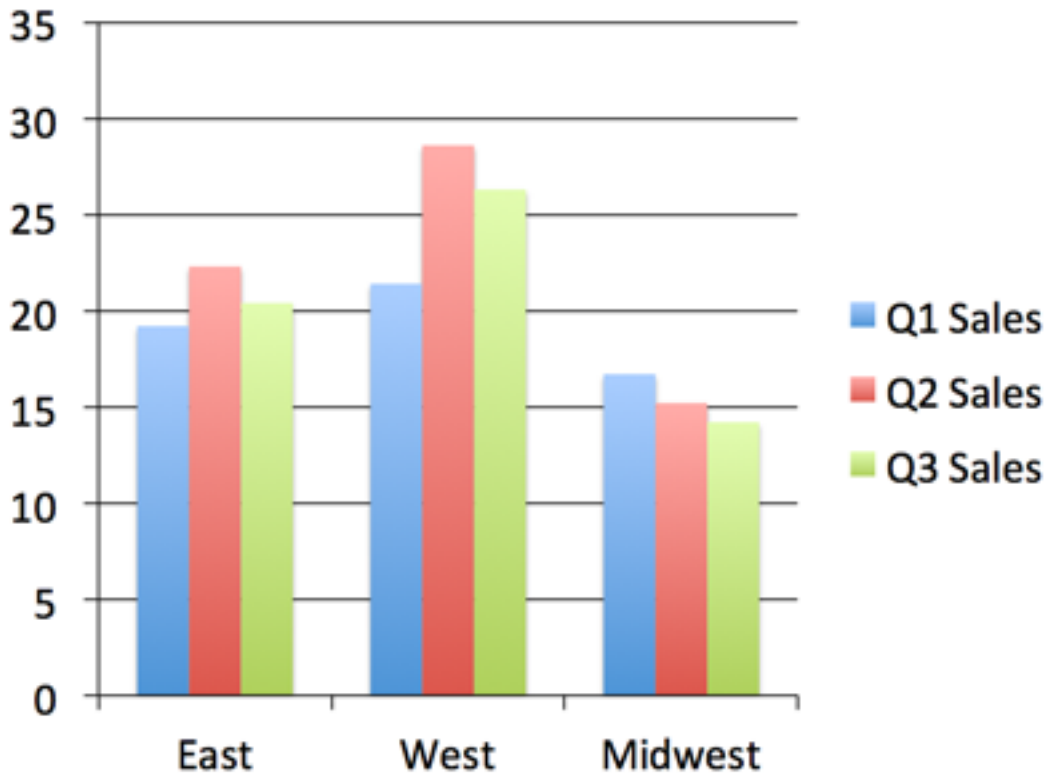
Here we needed to access a Plot object to gain access to the data labels. A plot is like a sub-chart, containing one or more series and drawn as a particular chart type, like column or line. This distinction is needed for charts that combine more than one type, like a line chart appearing on top of a column chart. A chart like this would have two plot objects, one for the series appearing as columns and the other for the lines. Most charts only have a single plot and *python-pptx* doesn't yet support creating multi-plot charts, but you can access multiple plots on a chart that already has them.

In the Microsoft API, the name *ChartGroup* is used for this object. I found that term confusing for a long time while I was learning about MS Office charts so I chose the name Plot for that object in *python-pptx*.

Legend

A legend is often useful to have on a chart, to give a name to each series and help a reader tell which one is which:

```
from pptx.enum.chart import XL_LEGEND_POSITION

chart.has_legend = True
chart.legend.position = XL_LEGEND_POSITION.RIGHT
chart.legend.include_in_layout = False
```

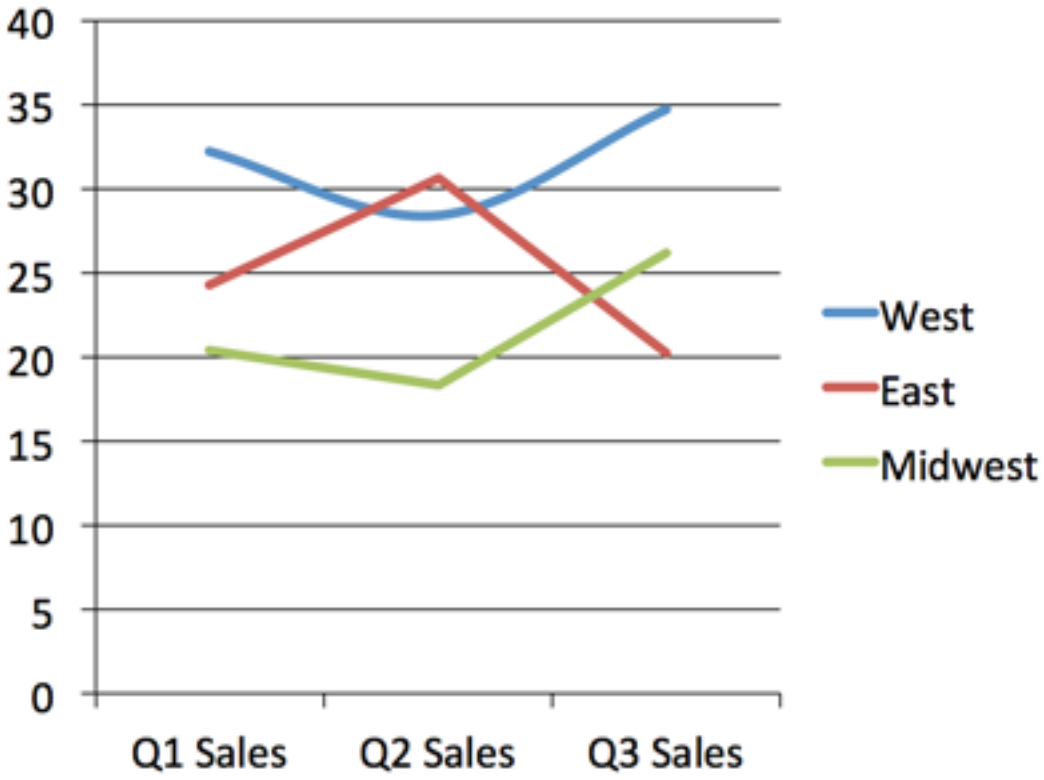


Nice! Okay, let's try some other chart types.

Line Chart

A line chart is added pretty much the same way as a bar or column chart, the main difference being the chart type provided in the `add_chart()` call:

```
chart_data = ChartData()
chart_data.categories = ['Q1 Sales', 'Q2 Sales', 'Q3 Sales']
chart_data.add_series('West',    (32.2, 28.4, 34.7))
chart_data.add_series('East',    (24.3, 30.6, 20.2))
chart_data.add_series('Midwest', (20.4, 18.3, 26.2))

x, y, cx, cy = Inches(2), Inches(2), Inches(6), Inches(4.5)
chart = slide.shapes.add_chart(
    XL_CHART_TYPE.LINE, x, y, cx, cy, chart_data
).chart

chart.has_legend = True
chart.legend.include_in_layout = False
chart.series[0].smooth = True
```

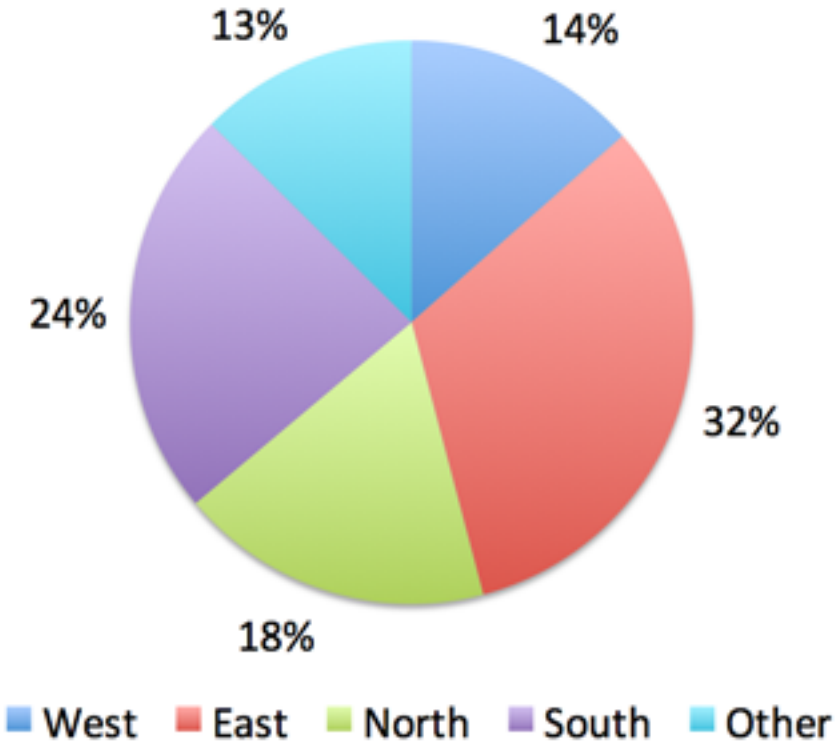


I switched the categories and series data here to better suit a line chart. You can see the line for the "West" region is *smoothed* into a curve while the other two have their points connected with straight line segments.

Pie Chart

A pie chart is a little special in that it only ever has a single series and doesn't have any axes:

```
chart_data = ChartData()
chart_data.categories = ['West', 'East', 'North', 'South', 'Other']
chart_data.add_series('Series 1', (0.135, 0.324, 0.180, 0.235, 0.126))

chart = slide.shapes.add_chart(
    XL_CHART_TYPE.PIE, x, y, cx, cy, chart_data
).chart

chart.has_legend = True
chart.legend.position = XL_LEGEND_POSITION.BOTTOM
chart.legend.include_in_layout = False

chart.plots[0].has_data_labels = True
data_labels = chart.plots[0].data_labels
data_labels.number_format = '0%'
data_labels.position = XL_LABEL_POSITION.OUTSIDE_END
```



Odds & Ends

This should be enough to get you started with adding charts to your presentation with *python-pptx*. There are more details in the API documentation for charts here: *Charts*

About colors

By default, the colors assigned to each series in a chart are the theme colors Accent 1 through Accent 6, in that order. If you have more than six series, darker and lighter versions of those same colors are used. While it's possible to assign specific colors to data points (bar, line, pie segment, etc.) for at least some chart types, the best strategy to start with is changing the theme colors in your starting "template" presentation.

Use cases

The use case that drove me to begin work on this library has been to automate the building of slides that are tedious to compose by hand. As an example, consider the task of composing a slide with an array of 10 headshot images of folks in a particular department, with the person's name and title next to their picture. After doing this a dozen times and struggling to get all the alignment and sizes to the point where my attention to detail is satisfied, well, my coding fingers got quite itchy.

However I believe a broader application will be server-side document generation on non-Windows server platforms, Linux primarily I expect. In my organization, I have found an apparently insatiable demand for PowerPoint documents as a means of communication. Once one rises beyond the level of project manager it seems the willingness to interpret text longer than a bullet point atrophies quite rapidly and PowerPoint becomes an everyday medium. I've imagined it might be pretty cool to be able to generate a "presentation-ready" deck for a salesperson that includes a particular

subset of the product catalog they could generate with a few clicks to use in a sales presentation, for example. As you come up with applications I'd love to hear about them.

Concepts

python-pptx is completely object-oriented, and in general any operations you perform with it will be on an object. The root object for a presentation is *Presentation*. API details are provided on the modules pages, but here are some basics to get you started, especially some relationships you might find surprising at first.

A presentation is loaded by constructing a new *Presentation* instance, passing in the path to a presentation to be loaded:

```
from pptx import Presentation

path = 'slide-deck-foo.pptx'
prs = Presentation(path)
```

python-pptx also contains a default template, and if you construct a *Presentation* instance without a path, a presentation based on that default template is loaded. This can be handy when you want to get started quickly, and most of the examples in this documentation use the default template.:

```
# start with default presentation
prs = Presentation()
```

Note that there is currently no distinction between templates and presentations in *python-pptx* as there is in the PowerPoint® client, there are only presentations. To use a "template" for a presentation you simply create a presentation with all the styles, logo, and layouts you want, delete all the slides (or leave some in if it suits), and then load that as your starting place.

Slide masters

A presentation has a list of slide masters and a list of slides. Let's start with a discussion of the slide masters.

One fact some find surprising (I did) is that a presentation file can have more than one slide master. It's quite uncommon in my experience to find presentations that make use of this feature, but it's entirely supported. The only time I've seen this happen is when slides from a "foreign" presentation are pasted into another deck; if you want the formatting and backgrounds from the other deck to be preserved on the pasted-in slides, the slide master and its slide layouts need to come with. Consequently, the presentation needs to maintain a list of slide masters, not just a single one, even though perhaps 99% of the time you only ever use the one. To make things a little easier for the 99% situation, you can refer to the first slide master as though it were the only one:

```
prs = Presentation()
slide_master = prs.slide_masters[0]
# is equivalent to
slide_master = prs.slide_master
```

Slide layouts

Another fact that might be surprising is that slide layouts belong to a slide master, not directly to a presentation, so normally you have to access the slide layouts via their slide master. Since this is subject to the same 99% situation described above, the slide layouts belonging to the first slide master can also be accessed directly from the presentation via syntactic sugar:

```
prs = Presentation()
title_slide_layout = prs.slide_masters[0].slide_layouts[0]
# is equivalent to:
title_slide_layout = prs.slide_layouts[0]
```

Slides

The slides in a presentation belong to the presentation object and are accessed using the `slides` attribute:

```
prs = Presentation(path)
first_slide = prs.slides[0]
```

Adding a slide

Adding a slide is accomplished by calling the `add_slide()` method on the `slides` attribute of the presentation. A slide layout must be passed in to specify the layout the new slide should take on:

```
prs = Presentation()
title_slide_layout = prs.slide_layouts[0]
new_slide = prs.slides.add_slide(title_slide_layout)
```

CHAPTER 3

Community Guide

Frequently Asked Questions

Support

We'd love to hear from you if you like *python-pptx*, want a new feature, find a bug, need help using it, or just have a word of encouragement.

The **mailing list** for *python-pptx* is python-pptx@googlegroups.com

The **issue tracker** is on github at scanny/python-pptx.

Feature requests are best broached initially on the mailing list, they can be added to the issue tracker once we've clarified the best approach, particularly the appropriate API signature.

Software Updates

Release History

0.6.7 (2017-10-30)

- Add *SlideShapes.build_freeform()*, allowing freeform shapes (such as maps) to be specified and added to a slide.
- Add support for patterned fills.
- Add *LineFormat.dash_style* to allow interrogation and setting of dashed line styles.

0.6.6 (2017-06-17)

- Add *SlideShapes.add_movie()*, allowing video media to be added to a slide.
- fix #190 Accommodate non-conforming part names having '00' index segment.

- fix #273 Accommodate non-conforming part names having no index segment.
- fix #277 ASCII/Unicode error on non-ASCII multi-level category names
- fix #279 BaseShape.id warning appearing on placeholder access.

0.6.5 (2017-03-21)

- #267 compensate for non-conforming PowerPoint behavior on c:overlay element
- compensate for non-conforming (to spec) PowerPoint behavior related to c:dLbl/c:tx that results in "can't save" error when explicit data labels are added to bubbles on a bubble chart.

0.6.4 (2017-03-17)

- add Chart.chart_title and ChartTitle object
- #263 Use Number type to test for numeric category

0.6.3 (2017-02-28)

- add DataLabel.font
- add Axis.axis_title

0.6.2 (2017-01-03)

- add support for NotesSlide (slide notes, aka. notes page)
- add support for arbitrary series ordering in XML
- add Plot.categories providing access to hierarchical categories in an existing chart.
- add support for date axes on category charts, including writing a dateAx element for the category axis when ChartData categories are date or datetime.

BACKWARD INCOMPATIBILITIES:

Some changes were made to the boilerplate XML used to create new charts. This was done to more closely adhere to the settings PowerPoint uses when creating a chart using the UI. This may result in some appearance changes in charts after upgrading. In particular:

- Chart.has_legend now defaults to True for Line charts.
- Plot.vary_by_categories now defaults to False for Line charts.

0.6.1 (2016-10-09)

- add Connector shape type

0.6.0 (2016-08-18)

- add XY chart types
- add Bubble chart types
- add Radar chart types
- add Area chart types
- add Doughnut chart types
- add Series.points and Point
- add Point.data_label
- add DataLabel.text_frame
- add DataLabel.position
- add Axis.major_gridlines
- add ChartFormat with .fill and .line
- add Axis.format (fill and line formatting)
- add ValueAxis.crosses and .crosses_at
- add Point.format (fill and line formatting)
- add Slide.slide_id
- add Slides.get() (by slide id)
- add Font.language_id
- support blank (None) data points in created charts
- add Series.marker
- add Point.marker
- add Marker.format, .style, and .size

0.5.8 (2015-11-27)

- add Shape.click_action (hyperlink on shape)
- fix: #128 Chart cat and ser names not escaped
- fix: #153 shapes.title raises on no title shape
- fix: #170 remove seek(0) from Image.from_file()

0.5.7 (2015-01-17)

- add PicturePlaceholder with .insert_picture() method
- add TablePlaceholder with .insert_table() method
- add ChartPlaceholder with .insert_chart() method
- add Picture.image property, returning Image object
- add Picture.crop_left, .crop_top, .crop_right, and .crop_bottom

- add Shape.placeholder_format and PlaceholderFormat object

BACKWARD INCOMPATIBILITIES:

Shape.shape_type is now unconditionally *MSO_SHAPE_TYPE.PLACEHOLDER* for all placeholder shapes. Previously, some placeholder shapes reported *MSO_SHAPE_TYPE.AUTO_SHAPE*, *MSO_SHAPE_TYPE.CHART*, *MSO_SHAPE_TYPE.PICTURE*, or *MSO_SHAPE_TYPE.TABLE* for that property.

0.5.6 (2014-12-06)

- fix #138 - UnicodeDecodeError in setup.py on Windows 7 Python 3.4

0.5.5 (2014-11-17)

- feature #51 - add Python 3 support

0.5.4 (2014-11-15)

- feature #43 - image native size in shapes.add_picture() is now calculated based on DPI attribute in image file, if present, defaulting to 72 dpi.
- feature #113 - Add Paragraph.space_before, Paragraph.space_after, and Paragraph.line_spacing

0.5.3 (2014-11-09)

- add experimental feature TextFrame.fit_text()

0.5.2 (2014-10-26)

- fix #127 - Shape.text_frame fails on shape having no txBody

0.5.1 (2014-09-22)

- feature #120 - add Shape.rotation
- feature #97 - add Font.underline
- issue #117 - add BMP image support
- issue #95 - add BaseShape.name setter
- issue #107 - all .text properties should return unicode, not str
- feature #106 - add .text getters to Shape, TextFrame, and Paragraph
- Rename Shape.textframe to Shape.text_frame. **Shape.textframe property (by that name) is deprecated.**

0.5.0 (2014-09-13)

- Add support for creating and manipulating bar, column, line, and pie charts
- Major refactoring of XML layer (oxml)
- Rationalized graphical object shape access **Note backward incompatibilities below**

BACKWARD INCOMPATIBILITIES:

A table is no longer treated as a shape. Rather it is a graphical object contained in a GraphicFrame shape, as are Chart and SmartArt objects.

Example:

```
table = shapes.add_table(...)

# becomes

graphic_frame = shapes.add_table(...)
table = graphic_frame.table

# or

table = shapes.add_table(...).table
```

As the enclosing shape, the id, name, shape type, position, and size are attributes of the enclosing GraphicFrame object.

The contents of a GraphicFrame shape can be identified using three available properties on a shape: has_table, has_chart, and has_smart_art. The enclosed graphical object is obtained using the properties GraphicFrame.table and GraphicFrame.chart. SmartArt is not yet supported. Accessing one of these properties on a GraphicFrame not containing the corresponding object raises an exception.

0.4.2 (2014-04-29)

- fix: issue #88 – raises on supported image file having uppercase extension
- fix: issue #89 – raises on add_slide() where non-contiguous existing ids

0.4.1 (2014-04-29)

- Rename Presentation.slidemasters to Presentation.slide_masters. Presentation.slidemasters property is deprecated.
- Rename Presentation.slidelayouts to Presentation.slide_layouts. Presentation.slidelayouts property is deprecated.
- Rename SlideMaster.slidelayouts to SlideMaster.slide_layouts. SlideMaster.slidelayouts property is deprecated.
- Rename SlideLayout.slidemaster to SlideLayout.slide_master. SlideLayout.slidemaster property is deprecated.
- Rename Slide.slidelayout to Slide.slide_layout. Slide.slidelayout property is deprecated.
- Add SlideMaster.shapes to access shapes on slide master.
- Add SlideMaster.placeholders to access placeholder shapes on slide master.
- Add _MasterPlaceholder class.
- Add _LayoutPlaceholder class with position and size inheritable from master placeholder.
- Add _SlidePlaceholder class with position and size inheritable from layout placeholder.
- Add Table.left, top, width, and height read/write properties.
- Add rudimentary GroupShape with left, top, width, and height properties.
- Add rudimentary Connector with left, top, width, and height properties.

- Add TextFrame.auto_size property.
- Add Presentation.slide_width and .slide_height read/write properties.
- Add LineFormat class providing access to read and change line color and width.
- Add AutoShape.line
- Add Picture.line
- Rationalize enumerations. **Note backward incompatibilities below**

BACKWARD INCOMPATIBILITIES:

The following enumerations were moved/renamed during the rationalization of enumerations:

- `pptx.enum.MSO_COLOR_TYPE` -> `pptx.enum.dml.MSO_COLOR_TYPE`
- `pptx.enum.MSO_FILL` -> `pptx.enum.dml.MSO_FILL`
- `pptx.enum.MSO_THEME_COLOR` -> `pptx.enum.dml.MSO_THEME_COLOR`
- `pptx.constants.MSO.ANCHOR_*` -> `pptx.enum.text.MSO_ANCHOR.*`
- `pptx.constants.MSO_SHAPE` -> `pptx.enum.shapes.MSO_SHAPE`
- `pptx.constants.PP.ALIGN_*` -> `pptx.enum.text.PP_ALIGN.*`
- `pptx.constants.MSO.{SHAPE_TYPES}` -> `pptx.enum.shapes.MSO_SHAPE_TYPE.*`

Documentation for all enumerations is available in the Enumerations section of the User Guide.

0.3.2 (2014-02-07)

- Hotfix: issue #80 generated presentations fail to load in Keynote and other Apple applications

0.3.1 (2014-01-10)

- Hotfix: failed to load certain presentations containing images with uppercase extension

0.3.0 (2013-12-12)

- Add read/write font color property supporting RGB, theme color, and inherit color types
- Add font typeface and italic support
- Add text frame margins and word-wrap
- Add support for external relationships, e.g. linked spreadsheet
- Add hyperlink support for text run in shape and table cell
- Add fill color and brightness for shape and table cell, fill can also be set to transparent (no fill)
- Add read/write position and size properties to shape and picture
- Replace PIL dependency with Pillow
- Restructure modules to better suit size of library

0.2.6 (2013-06-22)

- Add read/write access to core document properties
- Hotfix to accomodate connector shapes in _AutoShapeType
- Hotfix to allow customXml parts to load when present

0.2.5 (2013-06-11)

- Add paragraph alignment property (left, right, centered, etc.)
- Add vertical alignment within table cell (top, middle, bottom)
- Add table cell margin properties
- Add table boolean properties: first column (row header), first row (column headings), last row (for e.g. totals row), last column (for e.g. row totals), horizontal banding, and vertical banding.
- Add support for auto shape adjustment values, e.g. change radius of corner rounding on rounded rectangle, position of callout arrow, etc.

0.2.4 (2013-05-16)

- Add support for auto shapes (e.g. polygons, flowchart symbols, etc.)

0.2.3 (2013-05-05)

- Add support for table shapes
- Add indentation support to textbox shapes, enabling multi-level bullets on bullet slides.

0.2.2 (2013-03-25)

- Add support for opening and saving a presentation from/to a file-like object.
- Refactor XML handling to use lxml objectify

0.2.1 (2013-02-25)

- Add support for Python 2.6
- Add images from a stream (e.g. StringIO) in addition to a path, allowing images retrieved from a database or network resource to be inserted without saving first.
- Expand text methods to accept unicode and UTF-8 encoded 8-bit strings.
- Fix potential install bug triggered by importing `__version__` from package `__init__.py` file.

0.2.0 (2013-02-10)

First non-alpha release with basic capabilities:

- open presentation/template or use built-in default template
- add slide

- set placeholder text (e.g. bullet slides)
- add picture
- add text box

CHAPTER 4

API Documentation

Presentations

A presentation is opened using the `Presentation()` function, provided directly by the `pptx` package:

```
from pptx import Presentation
```

This function returns a *Presentation* object which is the root of a graph containing the components that constitute a presentation, e.g. slides, shapes, etc. All existing presentation components are referenced by traversing the graph and new objects are added to the graph by calling a method on that object's container. Consequently, *python-pptx* objects are generally not constructed directly.

Example:

```
# load a presentation
prs = Presentation(path_to_pptx_file)

# get reference to first shape in first slide
sp = prs.slides[0].shapes[0]

# add a picture shape to slide
pic = sld.shapes.add_picture(path, x, y, cx, cy)
```

`Presentation` function

This function is the only reference that must be imported to work with presentation files. Typical use interacts with many other classes, but there is no need to construct them as they are accessed using a property or method of their containing object.

`pptx.`**`Presentation`**(*pptx=None*)

Return a *Presentation* object loaded from *pptx*, where *pptx* can be either a path to a `.pptx` file (a string) or a file-like object. If *pptx* is missing or `None`, the built-in default presentation "template" is loaded.

`Presentation` objects

class `pptx.presentation.`**`Presentation`**

PresentationML (PML) presentation. Not intended to be constructed directly. Use *`pptx.Presentation()`* to open or create a presentation.

`core_properties`

Instance of *`CoreProperties`* holding the read/write Dublin Core document properties for this presentation.

`notes_master`

Instance of `NotesMaster` for this presentation. If the presentation does not have a notes master, one is created from a default template and returned. The same single instance is returned on each call.

`save`(*file*)

Save this presentation to *file*, where *file* can be either a path to a file (a string) or a file-like object.

`slide_height`

Height of slides in this presentation, in English Metric Units (EMU). Returns `None` if no slide width is defined. Read/write.

`slide_layouts`

Sequence of *`SlideLayout`* instances belonging to the first *`SlideMaster`* of this presentation. A presentation can have more than one slide master and each master will have its own set of layouts. This property is a convenience for the common case where the presentation has only a single slide master.

`slide_master`

First *`SlideMaster`* object belonging to this presentation. Typically, presentations have only a single slide master. This property provides simpler access in that common case.

`slide_masters`

Sequence of *`SlideMaster`* objects belonging to this presentation

`slide_width`

Width of slides in this presentation, in English Metric Units (EMU). Returns `None` if no slide width is defined. Read/write.

`slides`

`Slides` object containing the slides in this presentation.

`CoreProperties` objects

Each *`Presentation`* object has a *`CoreProperties`* object accessed via its `core_properties` attribute that provides read/write access to the so-called *core properties* for the document. The core properties are author, category, comments, content_status, created, identifier, keywords, language, last_modified_by, last_printed, modified, revision, subject, title, and version.

Each property is one of three types, `str`, `datetime.datetime`, or `int`. String properties are limited in length to 255 characters and return an empty string (‘’) if not set. Date properties are assigned and returned as `datetime.datetime` objects without timezone, i.e. in UTC. Any timezone conversions are the responsibility of the client. Date properties return `None` if not set.

python-pptx does not automatically set any of the document core properties other than to add a core properties part to a presentation that doesn’t have one (very uncommon). If *python-pptx* adds a core properties part, it contains default values for the title, last_modified_by, revision, and modified properties. Client code should change properties like revision and last_modified_by explicitly if that behavior is desired.

class `pptx.opc.coreprops.`**`CoreProperties`**

author
string – An entity primarily responsible for making the content of the resource.

category
string – A categorization of the content of this package. Example values might include: Resume, Letter, Financial Forecast, Proposal, or Technical Presentation.

comments
string – An account of the content of the resource.

content_status
string – completion status of the document, e.g. 'draft'

created
datetime – time of intial creation of the document

identifier
string – An unambiguous reference to the resource within a given context, e.g. ISBN.

keywords
string – descriptive words or short phrases likely to be used as search terms for this document

language
string – language the document is written in

last_modified_by
string – name or other identifier (such as email address) of person who last modified the document

last_printed
datetime – time the document was last printed

modified
datetime – time the document was last modified

revision
int – number of this revision, incremented by the PowerPoint® client once each time the document is saved. Note however that the revision number is not automatically incremented by *python-pptx*.

subject
string – The topic of the content of the resource.

title
string – The name given to the resource.

version
string – free-form version string

Slides

`Slides` objects

The *`Slides`* object is accessed using the *`slides`* property of *`Presentation`*. It is not intended to be constructed directly.

class `pptx.slide.`**`Slides`**
Sequence of slides belonging to an instance of *`Presentation`*, having list semantics for access to individual slides. Supports indexed access, len(), and iteration.

add_slide(*slide_layout*)
Return a newly added slide that inherits layout from *slide_layout*.

get (*slide_id*, *default=None*)
Return the slide identified by integer *slide_id* in this presentation, or *default* if not found.

index (*slide*)
Map *slide* to an integer representing its zero-based position in this slide collection. Raises `ValueError` on *slide* not present.

`Slide` objects

An individual *`Slide`* object is accessed by index from *`Slides`* or as the return value of `add_slide()`.

class `pptx.slide.`**Slide**
Slide object. Provides access to shapes and slide-level properties.

element
The lxml element proxied by this object.

has_notes_slide
Return True if this slide has a notes slide, False otherwise. A notes slide is created by *`notes_slide`* when one doesn't exist; use this property to test for a notes slide without the possible side effect of creating one.

name
String representing the internal name of this slide. Returns an empty string (‘’) if no name is assigned. Assigning an empty string or `None` to this property causes any name to be removed.

notes_slide
Return the *`NotesSlide`* instance for this slide. If the slide does not have a notes slide, one is created. The same single instance is returned on each call.

placeholders
Instance of *`SlidePlaceholders`* containing sequence of placeholder shapes in this slide.

shapes
Instance of *`SlideShapes`* containing sequence of shape objects appearing on this slide.

slide_id
The integer value that uniquely identifies this slide within this presentation. The slide id does not change if the position of this slide in the slide sequence is changed by adding, rearranging, or deleting slides.

slide_layout
`SlideLayout` object this slide inherits appearance from.

`SlideLayout` objects

class `pptx.slide.`**SlideLayout** (*element*, *part*)
Slide layout object. Provides access to placeholders, regular shapes, and slide layout-level properties.

placeholders
Instance of `LayoutPlaceholders` containing sequence of placeholder shapes in this slide layout, sorted in *idx* order.

shapes
Instance of `LayoutShapes` containing the sequence of shapes appearing on this slide layout.

slide_master
Slide master from which this slide layout inherits properties.

`SlideMaster` objects

class `pptx.slide.`**`SlideMaster`**(*element*, *part*)

Slide master object. Provides access to slide layouts. Access to placeholders, regular shapes, and slide master-level properties is inherited from `_BaseMaster`.

`slide_layouts`

Sequence of *`SlideLayout`* objects belonging to this slide master

`SlidePlaceholders` objects

class `pptx.shapes.shapetree.`**`SlidePlaceholders`**(*element*, *parent*)

Collection of placeholder shapes on a slide. Supports iteration, `len()`, and dictionary-style lookup on the *idx* value of the placeholders it contains.

`NotesSlide` objects

class `pptx.slide.`**`NotesSlide`**(*element*, *part*)

Notes slide object. Provides access to slide notes placeholder and other shapes on the notes handout page.

`element`

The lxml element proxied by this object.

`name`

String representing the internal name of this slide. Returns an empty string (‘’) if no name is assigned. Assigning an empty string or `None` to this property causes any name to be removed.

`notes_placeholder`

Return the notes placeholder on this notes slide, the shape that contains the actual notes text. Return `None` if no notes placeholder is present; while this is probably uncommon, it can happen if the notes master does not have a body placeholder, or if the notes placeholder has been deleted from the notes slide.

`notes_text_frame`

Return the text frame of the notes placeholder on this notes slide, or `None` if there is no notes placeholder. This is a shortcut to accommodate the common case of simply adding “notes” text to the notes “page”.

`part`

The package part containing this object

`placeholders`

An instance of `NotesSlidePlaceholders` containing the sequence of placeholder shapes in this notes slide.

`shapes`

An instance of `NotesSlideShapes` containing the sequence of shape objects appearing on this notes slide.

Shapes

The following classes provide access to the shapes that appear on a slide and the collections that contain them.

`SlideShapes` objects

The *`SlideShapes`* object is encountered as the `shapes` property of *`Slide`*.

class `pptx.shapes.shapetree.`**`SlideShapes`**

Sequence of shapes appearing on a slide.

The first shape in the sequence is the backmost in z-order and the last shape is topmost. Supports indexed access, len(), index(), and iteration.

`add_chart`(*chart_type*, *x*, *y*, *cx*, *cy*, *chart_data*)

Add a new chart of *chart_type* to the slide, positioned at (*x*, *y*), having size (*cx*, *cy*), and depicting *chart_data*. *chart_type* is one of the *XL_CHART_TYPE* enumeration values. *chart_data* is a *`ChartData`* object populated with the categories and series values for the chart. Note that a *`GraphicFrame`* shape object is returned, not the *`Chart`* object contained in that graphic frame shape. The chart object may be accessed using the `chart` property of the returned *`GraphicFrame`* object.

`add_connector`(*connector_type*, *begin_x*, *begin_y*, *end_x*, *end_y*)

Add a newly created connector shape to the end of this shape tree. *connector_type* is a member of the *MSO_CONNECTOR_TYPE* enumeration and the end-point values are specified as EMU values. The returned connector is of type *connector_type* and has begin and end points as specified.

`add_movie`(*movie_file*, *left*, *top*, *width*, *height*, *poster_frame_image=None*, *mime_type='video/unknown'*)

Return newly added movie shape displaying video in *movie_file*.

EXPERIMENTAL. This method has important limitations:

- The size must be specified; no auto-scaling such as that provided by `add_picture()` is performed.
- The MIME type of the video file should be specified, e.g. 'video/mp4'. The provided video file is not interrogated for its type. The MIME type *video/unknown* is used by default (and works fine in tests as of this writing).
- A poster frame image must be provided, it cannot be automatically extracted from the video file. If no poster frame is provided, the default "media loudspeaker" image will be used.

Return a newly added movie shape to the slide, positioned at (*left*, *top*), having size (*width*, *height*), and containing *movie_file*. Before the video is started, *poster_frame_image* is displayed as a placeholder for the video.

`add_picture`(*image_file*, *left*, *top*, *width=None*, *height=None*)

Add picture shape displaying image in *image_file*, where *image_file* can be either a path to a file (a string) or a file-like object.

`add_shape`(*autoshape_type_id*, *left*, *top*, *width*, *height*)

Add auto shape of type specified by *autoshape_type_id* (like `MSO_SHAPE.RECTANGLE`) and of specified size at specified position.

`add_table`(*rows*, *cols*, *left*, *top*, *width*, *height*)

Add a *`GraphicFrame`* object containing a table with the specified number of *rows* and *cols* and the specified position and size. *width* is evenly distributed between the columns of the new table. Likewise, *height* is evenly distributed between the rows. Note that the `.table` property on the returned *`GraphicFrame`* shape must be used to access the enclosed `Table` object.

`add_textbox`(*left*, *top*, *width*, *height*)

Add text box shape of specified size at specified position on slide.

`build_freeform`(*start_x=0*, *start_y=0*, *scale=1.0*)

Return *`FreeformBuilder`* object to specify a freeform shape.

The optional *start_x* and *start_y* arguments specify the starting pen position in local coordinates. They will be rounded to the nearest integer before use and each default to zero.

The optional *scale* argument specifies the size of local coordinates proportional to slide coordinates (EMU). If the vertical scale is different than the horizontal scale (local coordinate units are "rectangular"), a pair of numeric values can be provided as the *scale* argument, e.g. *scale=(1.0, 2.0)*. In this case the first number is interpreted as the horizontal (X) scale and the second as the vertical (Y) scale.

A convenient method for calculating scale is to divide a *Length* object by an equivalent count of local coordinate units, e.g. *scale = Inches(1)/1000* for 1000 local units per inch.

index(*shape*)
Return the index of *shape* in this sequence, raising `ValueError` if *shape* is not in the collection.

placeholders
Instance of *SlidePlaceholders* containing sequence of placeholder shapes in this slide.

title
The title placeholder shape on the slide or `None` if the slide has no title placeholder.

Shape objects in general

The following properties and methods are common to all shapes.

class `pptx.shapes.base.`**BaseShape**
Base class for shape objects, including *Shape*, *Picture*, and *GraphicFrame*.

click_action
An *ActionSetting* instance providing access to the mouse click behaviors defined on this shape. An *ActionSetting* object is always returned, even when no click behavior is defined on the shape.

element
Reference to the lxml element for this shape, e.g. a CT_Shape instance.

has_chart
`True` if this shape is a graphic frame containing a chart object. `False` otherwise. When `True`, the chart object can be accessed using the `.chart` property.

has_table
`True` if this shape is a graphic frame containing a table object. `False` otherwise. When `True`, the table object can be accessed using the `.table` property.

has_text_frame
`True` if this shape can contain text.

height
Read/write. Integer distance between top and bottom extents of shape in EMUs

is_placeholder
True if this shape is a placeholder. A shape is a placeholder if it has a <p:ph> element.

left
Read/write. Integer distance of the left edge of this shape from the left edge of the slide, in English Metric Units (EMU)

name
Name of this shape, e.g. 'Picture 7'

placeholder_format
A *_PlaceholderFormat* object providing access to placeholder-specific properties such as placeholder type. Raises `ValueError` on access if the shape is not a placeholder.

rotation
Read/write float. Degrees of clockwise rotation. Negative values can be assigned to indicate counter-clockwise rotation, e.g. assigning -45.0 will change setting to 315.0.

shape_id
Read-only positive integer identifying this shape.

The id of a shape is unique among all shapes on a slide.

shape_type
Unique integer identifying the type of this shape, like `MSO_SHAPE_TYPE.CHART`. Must be implemented by subclasses.

top
Read/write. Integer distance of the top edge of this shape from the top edge of the slide, in English Metric Units (EMU)

width
Read/write. Integer distance between left and right extents of shape in EMUs

`Shape` objects (AutoShapes)

The following properties and methods are defined for AutoShapes, which include text boxes and placeholders.

class `pptx.shapes.autoshape.`**Shape**
A shape that can appear on a slide. Corresponds to the `<p:sp>` element that can appear in any of the slide-type parts (slide, slideLayout, slideMaster, notesPage, notesMaster, handoutMaster).

adjustments
Read-only reference to `AdjustmentCollection` instance for this shape

auto_shape_type
Enumeration value identifying the type of this auto shape, like `MSO_SHAPE.ROUNDED_RECTANGLE`. Raises `ValueError` if this shape is not an auto shape.

fill
FillFormat instance for this shape, providing access to fill properties such as fill color.

has_text_frame
`True` if this shape can contain text. Always `True` for an AutoShape.

line
LineFormat instance for this shape, providing access to line properties such as line color.

shape_type
Unique integer identifying the type of this shape, like `MSO_SHAPE_TYPE.TEXT_BOX`.

text
Read/write. All the text in this shape as a single string. A line feed character ('\n') appears in the string for each paragraph and line break in the shape, except the last paragraph. A shape containing a single paragraph with no line breaks will produce a string having no line feed characters. Assigning a string to this property replaces all text in the shape with a single paragraph containing the assigned text. The assigned value can be a 7-bit ASCII string, a UTF-8 encoded 8-bit string, or unicode. String values are converted to unicode assuming UTF-8 encoding. Each line feed character in an assigned string is translated into a line break within the single resulting paragraph.

text_frame
TextFrame instance for this shape, containing the text of the shape and providing access to text formatting properties.

`AdjustmentCollection` objects

An AutoShape is distinctive in that it can have *adjustments*, represented in the PowerPoint user interface as small yellow diamonds that each allow a parameter of the shape, such as the angle of an arrowhead, to be adjusted. The `AdjustmentCollection` object holds these adjustment values for an AutoShape, each of which is an `Adjustment` instance.

The `AdjustmentCollection` instance for an AutoShape is accessed using the `Shape.adjustments` property (read-only).

class `pptx.shapes.autoshape.`**AdjustmentCollection**(*prstGeom*)
Sequence of `Adjustment` instances for an auto shape, each representing an available adjustment for a shape of its type. Supports `len()` and indexed access, e.g. `shape.adjustments[1] = 0.15`.

`Adjustment` objects

class `pptx.shapes.autoshape.`**Adjustment**(*name*, *def_val*, *actual=None*)
An adjustment value for an autoshape.

An adjustment value corresponds to the position of an adjustment handle on an auto shape. Adjustment handles are the small yellow diamond-shaped handles that appear on certain auto shapes and allow the outline of the shape to be adjusted. For example, a rounded rectangle has an adjustment handle that allows the radius of its corner rounding to be adjusted.

Values are `float` and generally range from 0.0 to 1.0, although the value can be negative or greater than 1.0 in certain circumstances.

effective_value
Read/write `float` representing normalized adjustment value for this adjustment. Actual values are a large-ish integer expressed in shape coordinates, nominally between 0 and 100,000. The effective value is normalized to a corresponding value nominally between 0.0 and 1.0. Intuitively this represents the proportion of the width or height of the shape at which the adjustment value is located from its starting point. For simple shapes such as a rounded rectangle, this intuitive correspondence holds. For more complicated shapes and at more extreme shape proportions (e.g. width is much greater than height), the value can become negative or greater than 1.0.

val
Denormalized effective value (expressed in shape coordinates), suitable for using in the XML.

`Connector` objects

The following properties and methods are defined for Connector shapes:

class `pptx.shapes.connector.`**Connector**
Connector (line) shape. A connector is a linear shape having end-points that can be connected to other objects (but not to other connectors). A line can be straight, have elbows, or can be curved.

begin_connect(*shape*, *cxn_pt_idx*)
EXPERIMENTAL - *The current implementation only works properly with rectangular shapes, such as pictures and rectangles. Use with other shape types may cause unexpected visual alignment of the connected end-point and could lead to a load error if cxn_pt_idx exceeds the connection point count available on the connected shape. That said, a quick test should reveal what to expect when using this method with other shape types.*

Connect the beginning of this connector to *shape* at the connection point specified by *cxn_pt_idx*. Each shape has zero or more connection points and they are identified by index, starting with 0. Generally, the

first connection point of a shape is at the top center of its bounding box and numbering proceeds counter-clockwise from there. However this is only a convention and may vary, especially with non built-in shapes.

begin_x
Return the X-position of the begin point of this connector, in English Metric Units (as a *Length* object).

begin_y
Return the Y-position of the begin point of this connector, in English Metric Units (as a *Length* object).

end_connect(*shape*, *cxn_pt_idx*)
EXPERIMENTAL - *The current implementation only works properly with rectangular shapes, such as pictures and rectangles. Use with other shape types may cause unexpected visual alignment of the connected end-point and could lead to a load error if cxn_pt_idx exceeds the connection point count available on the connected shape. That said, a quick test should reveal what to expect when using this method with other shape types.*

Connect the ending of this connector to *shape* at the connection point specified by *cxn_pt_idx*.

end_x
Return the X-position of the end point of this connector, in English Metric Units (as a *Length* object).

end_y
Return the Y-position of the end point of this connector, in English Metric Units (as a *Length* object).

FreeformBuilder objects

The following properties and methods are defined for FreeformBuilder objects. A freeform builder is used to create a shape with custom geometry:

class pptx.shapes.freeform.**FreeformBuilder**
Allows a freeform shape to be specified and created.

The initial pen position is provided on construction. From there, drawing proceeds using successive calls to draw line segments. The freeform shape may be closed by calling the `close()` method.

A shape may have more than one contour, in which case overlapping areas are "subtracted". A contour is a sequence of line segments beginning with a "move-to" operation. A move-to operation is automatically inserted in each new freeform; additional move-to ops can be inserted with the *.move_to()* method.

add_line_segments(*vertices*, *close=True*)
Add a straight line segment to each point in *vertices*.

vertices must be an iterable of (x, y) pairs (2-tuples). Each x and y value is rounded to the nearest integer before use. The optional *close* parameter determines whether the resulting contour is *closed* or left *open*.

Returns this *FreeformBuilder* object so it can be used in chained calls.

convert_to_shape(*origin_x=0*, *origin_y=0*)
Return new freeform shape positioned relative to specified offset.

origin_x and *origin_y* locate the origin of the local coordinate system in slide coordinates (EMU), perhaps most conveniently by use of a *Length* object.

Note that this method may be called more than once to add multiple shapes of the same geometry in different locations on the slide.

move_to(*x*, *y*)
Move pen to (x, y) (local coordinates) without drawing line.

Returns this *FreeformBuilder* object so it can be used in chained calls.

`Picture` objects

The following properties and methods are defined for picture shapes.

class `pptx.shapes.picture.`**`Picture`**
A picture shape, one that places an image on a slide.

Based on the *p:pic* element.

`image`
An *`Image`* object providing access to the properties and bytes of the image in this picture shape.

`shape_type`
Unique integer identifying the type of this shape, unconditionally `MSO_SHAPE_TYPE.PICTURE` in this case.

`GraphicFrame` objects

The following properties and methods are defined for graphic frame shapes. A graphic frame is the shape containing a table, chart, or smart art.

class `pptx.shapes.graphfrm.`**`GraphicFrame`**
Bases: *`pptx.shapes.base.BaseShape`*

Container shape for table, chart, smart art, and media objects. Corresponds to a `<p:graphicFrame>` element in the shape tree.

`chart`
The *`Chart`* object containing the chart in this graphic frame. Raises `ValueError` if this graphic frame does not contain a chart.

`click_action`
An *`ActionSetting`* instance providing access to the mouse click behaviors defined on this shape. An *`ActionSetting`* object is always returned, even when no click behavior is defined on the shape.

`element`
Reference to the lxml element for this shape, e.g. a CT_Shape instance.

`has_chart`
`True` if this graphic frame contains a chart object. `False` otherwise. When `True`, the chart object can be accessed using the `.chart` property.

`has_table`
`True` if this graphic frame contains a table object. `False` otherwise. When `True`, the table object can be accessed using the `.table` property.

`height`
Read/write. Integer distance between top and bottom extents of shape in EMUs

`left`
Read/write. Integer distance of the left edge of this shape from the left edge of the slide, in English Metric Units (EMU)

`name`
Name of this shape, e.g. 'Picture 7'

`rotation`
Read/write float. Degrees of clockwise rotation. Negative values can be assigned to indicate counter-clockwise rotation, e.g. assigning -45.0 will change setting to 315.0.

shape_id
Read-only positive integer identifying this shape.

The id of a shape is unique among all shapes on a slide.

table
The `Table` object contained in this graphic frame. Raises `ValueError` if this graphic frame does not contain a table.

top
Read/write. Integer distance of the top edge of this shape from the top edge of the slide, in English Metric Units (EMU)

width
Read/write. Integer distance between left and right extents of shape in EMUs

Placeholders

The following classes represent placeholder shapes. A placeholder most commonly appears on a slide, but also appears on a slide layout and a slide master. The role of a master placeholder and layout placeholder differs from that of a slide placeholder and these roles are reflected in the distinct classes for each.

There are a larger variety of slide placeholders to accomodate their more complex and varied behaviors.

`MasterPlaceholder` objects

class `pptx.shapes.placeholder.`**MasterPlaceholder**
Placeholder shape on a slide master.

auto_shape_type
Enumeration value identifying the type of this auto shape, like `MSO_SHAPE.ROUNDED_RECTANGLE`. Raises `ValueError` if this shape is not an auto shape.

click_action
An *`ActionSetting`* instance providing access to the mouse click behaviors defined on this shape. An *`ActionSetting`* object is always returned, even when no click behavior is defined on the shape.

element
Reference to the lxml element for this shape, e.g. a CT_Shape instance.

fill
`FillFormat` instance for this shape, providing access to fill properties such as fill color.

has_text_frame
`True` if this shape can contain text. Always `True` for an AutoShape.

height
Read/write. Integer distance between top and bottom extents of shape in EMUs

is_placeholder
True if this shape is a placeholder. A shape is a placeholder if it has a <p:ph> element.

left
Read/write. Integer distance of the left edge of this shape from the left edge of the slide, in English Metric Units (EMU)

line
`LineFormat` instance for this shape, providing access to line properties such as line color.

name
Name of this shape, e.g. 'Picture 7'

placeholder_format
A *_PlaceholderFormat* object providing access to placeholder-specific properties such as placeholder type. Raises `ValueError` on access if the shape is not a placeholder.

rotation
Read/write float. Degrees of clockwise rotation. Negative values can be assigned to indicate counter-clockwise rotation, e.g. assigning -45.0 will change setting to 315.0.

shape_id
Read-only positive integer identifying this shape.

The id of a shape is unique among all shapes on a slide.

text
Read/write. All the text in this shape as a single string. A line feed character ('\n') appears in the string for each paragraph and line break in the shape, except the last paragraph. A shape containing a single paragraph with no line breaks will produce a string having no line feed characters. Assigning a string to this property replaces all text in the shape with a single paragraph containing the assigned text. The assigned value can be a 7-bit ASCII string, a UTF-8 encoded 8-bit string, or unicode. String values are converted to unicode assuming UTF-8 encoding. Each line feed character in an assigned string is translated into a line break within the single resulting paragraph.

text_frame
TextFrame instance for this shape, containing the text of the shape and providing access to text formatting properties.

top
Read/write. Integer distance of the top edge of this shape from the top edge of the slide, in English Metric Units (EMU)

width
Read/write. Integer distance between left and right extents of shape in EMUs

`LayoutPlaceholder` objects

class `pptx.shapes.placeholder.`**`LayoutPlaceholder`**
Placeholder shape on a slide layout, providing differentiated behavior for slide layout placeholders, in particular, inheriting shape properties from the master placeholder having the same type, when a matching one exists.

ChartPlaceholder objects

class `pptx.shapes.placeholder.`**`ChartPlaceholder`**
Placeholder shape that can only accept a chart.

adjustments
Read-only reference to `AdjustmentCollection` instance for this shape

auto_shape_type
Enumeration value identifying the type of this auto shape, like `MSO_SHAPE.ROUNDED_RECTANGLE`. Raises `ValueError` if this shape is not an auto shape.

click_action
An *ActionSetting* instance providing access to the mouse click behaviors defined on this shape. An *ActionSetting* object is always returned, even when no click behavior is defined on the shape.

element
Reference to the lxml element for this shape, e.g. a CT_Shape instance.

fill
FillFormat instance for this shape, providing access to fill properties such as fill color.

get_or_add_ln()
Return the <a:ln> element containing the line format properties XML for this shape.

height
The effective height of this placeholder shape; its directly-applied height if it has one, otherwise the height of its parent layout placeholder.

insert_chart(*chart_type*, *chart_data*)
Return a *PlaceholderGraphicFrame* object containing a new chart of *chart_type* depicting *chart_data* and having the same position and size as this placeholder. *chart_type* is one of the *XL_CHART_TYPE* enumeration values. *chart_data* is a *ChartData* object populated with the categories and series values for the chart. Note that the new *Chart* object is not returned directly. The chart object may be accessed using the *chart* property of the returned *PlaceholderGraphicFrame* object.

is_placeholder
Boolean indicating whether this shape is a placeholder. Unconditionally True in this case.

left
The effective left of this placeholder shape; its directly-applied left if it has one, otherwise the left of its parent layout placeholder.

line
LineFormat instance for this shape, providing access to line properties such as line color.

ln
The <a:ln> element containing the line format properties such as line color and width. None if no <a:ln> element is present.

name
Name of this shape, e.g. 'Picture 7'

placeholder_format
A *_PlaceholderFormat* object providing access to placeholder-specific properties such as placeholder type. Raises ValueError on access if the shape is not a placeholder.

rotation
Read/write float. Degrees of clockwise rotation. Negative values can be assigned to indicate counter-clockwise rotation, e.g. assigning -45.0 will change setting to 315.0.

shape_id
Read-only positive integer identifying this shape.

The id of a shape is unique among all shapes on a slide.

shape_type
Member of *MSO_SHAPE_TYPE* specifying the type of this shape. Unconditionally MSO_SHAPE_TYPE.PLACEHOLDER in this case. Read-only.

text
Read/write. All the text in this shape as a single string. A line feed character ('\n') appears in the string for each paragraph and line break in the shape, except the last paragraph. A shape containing a single paragraph with no line breaks will produce a string having no line feed characters. Assigning a string to this property replaces all text in the shape with a single paragraph containing the assigned text. The assigned value can be a 7-bit ASCII string, a UTF-8 encoded 8-bit string, or unicode. String values are

converted to unicode assuming UTF-8 encoding. Each line feed character in an assigned string is translated into a line break within the single resulting paragraph.

text_frame
TextFrame instance for this shape, containing the text of the shape and providing access to text formatting properties.

top
The effective top of this placeholder shape; its directly-applied top if it has one, otherwise the top of its parent layout placeholder.

width
The effective width of this placeholder shape; its directly-applied width if it has one, otherwise the width of its parent layout placeholder.

PicturePlaceholder objects

class pptx.shapes.placeholder.**PicturePlaceholder**
Placeholder shape that can only accept a picture.

adjustments
Read-only reference to AdjustmentCollection instance for this shape

auto_shape_type
Enumeration value identifying the type of this auto shape, like MSO_SHAPE.ROUNDED_RECTANGLE. Raises ValueError if this shape is not an auto shape.

click_action
An *ActionSetting* instance providing access to the mouse click behaviors defined on this shape. An *ActionSetting* object is always returned, even when no click behavior is defined on the shape.

element
Reference to the lxml element for this shape, e.g. a CT_Shape instance.

fill
FillFormat instance for this shape, providing access to fill properties such as fill color.

get_or_add_ln()
Return the <a:ln> element containing the line format properties XML for this shape.

height
The effective height of this placeholder shape; its directly-applied height if it has one, otherwise the height of its parent layout placeholder.

insert_picture(*image_file*)
Return a *PlaceholderPicture* object depicting the image in *image_file*, which may be either a path (string) or a file-like object. The image is cropped to fill the entire space of the placeholder. A *PlaceholderPicture* object has all the properties and methods of a *Picture* shape except that the value of its shape_type property is *MSO_SHAPE_TYPE.PLACEHOLDER* instead of *MSO_SHAPE_TYPE.PICTURE*.

is_placeholder
Boolean indicating whether this shape is a placeholder. Unconditionally True in this case.

left
The effective left of this placeholder shape; its directly-applied left if it has one, otherwise the left of its parent layout placeholder.

line
LineFormat instance for this shape, providing access to line properties such as line color.

ln
The <a:ln> element containing the line format properties such as line color and width. None if no <a:ln> element is present.

name
Name of this shape, e.g. 'Picture 7'

placeholder_format
A *_PlaceholderFormat* object providing access to placeholder-specific properties such as placeholder type. Raises ValueError on access if the shape is not a placeholder.

rotation
Read/write float. Degrees of clockwise rotation. Negative values can be assigned to indicate counter-clockwise rotation, e.g. assigning -45.0 will change setting to 315.0.

shape_id
Read-only positive integer identifying this shape.

The id of a shape is unique among all shapes on a slide.

shape_type
Member of *MSO_SHAPE_TYPE* specifying the type of this shape. Unconditionally MSO_SHAPE_TYPE.PLACEHOLDER in this case. Read-only.

text
Read/write. All the text in this shape as a single string. A line feed character ('\n') appears in the string for each paragraph and line break in the shape, except the last paragraph. A shape containing a single paragraph with no line breaks will produce a string having no line feed characters. Assigning a string to this property replaces all text in the shape with a single paragraph containing the assigned text. The assigned value can be a 7-bit ASCII string, a UTF-8 encoded 8-bit string, or unicode. String values are converted to unicode assuming UTF-8 encoding. Each line feed character in an assigned string is translated into a line break within the single resulting paragraph.

text_frame
TextFrame instance for this shape, containing the text of the shape and providing access to text formatting properties.

top
The effective top of this placeholder shape; its directly-applied top if it has one, otherwise the top of its parent layout placeholder.

width
The effective width of this placeholder shape; its directly-applied width if it has one, otherwise the width of its parent layout placeholder.

TablePlaceholder objects

class pptx.shapes.placeholder.**TablePlaceholder**
Placeholder shape that can only accept a picture.

adjustments
Read-only reference to AdjustmentCollection instance for this shape

auto_shape_type
Enumeration value identifying the type of this auto shape, like MSO_SHAPE.ROUNDED_RECTANGLE. Raises ValueError if this shape is not an auto shape.

click_action
An *ActionSetting* instance providing access to the mouse click behaviors defined on this shape. An *ActionSetting* object is always returned, even when no click behavior is defined on the shape.

element
Reference to the lxml element for this shape, e.g. a CT_Shape instance.

fill
FillFormat instance for this shape, providing access to fill properties such as fill color.

get_or_add_ln()
Return the <a:ln> element containing the line format properties XML for this shape.

height
The effective height of this placeholder shape; its directly-applied height if it has one, otherwise the height of its parent layout placeholder.

insert_table(*rows*, *cols*)
Return a *PlaceholderGraphicFrame* object containing a table of *rows* rows and *cols* columns. The position and width of the table are those of the placeholder and its height is proportional to the number of rows. A *PlaceholderGraphicFrame* object has all the properties and methods of a *GraphicFrame* shape except that the value of its shape_type property is unconditionally *MSO_SHAPE_TYPE.PLACEHOLDER*. Note that the return value is not the new table but rather *contains* the new table. The table can be accessed using the *table* property of the returned *PlaceholderGraphicFrame* object.

left
The effective left of this placeholder shape; its directly-applied left if it has one, otherwise the left of its parent layout placeholder.

line
LineFormat instance for this shape, providing access to line properties such as line color.

ln
The <a:ln> element containing the line format properties such as line color and width. None if no <a:ln> element is present.

name
Name of this shape, e.g. 'Picture 7'

placeholder_format
A *_PlaceholderFormat* object providing access to placeholder-specific properties such as placeholder type. Raises ValueError on access if the shape is not a placeholder.

rotation
Read/write float. Degrees of clockwise rotation. Negative values can be assigned to indicate counter-clockwise rotation, e.g. assigning -45.0 will change setting to 315.0.

shape_id
Read-only positive integer identifying this shape.

The id of a shape is unique among all shapes on a slide.

shape_type
Member of *MSO_SHAPE_TYPE* specifying the type of this shape. Unconditionally MSO_SHAPE_TYPE.PLACEHOLDER in this case. Read-only.

text
Read/write. All the text in this shape as a single string. A line feed character ('\n') appears in the string for each paragraph and line break in the shape, except the last paragraph. A shape containing a single paragraph with no line breaks will produce a string having no line feed characters. Assigning a string to this property replaces all text in the shape with a single paragraph containing the assigned text. The assigned value can be a 7-bit ASCII string, a UTF-8 encoded 8-bit string, or unicode. String values are converted to unicode assuming UTF-8 encoding. Each line feed character in an assigned string is translated into a line break within the single resulting paragraph.

text_frame
TextFrame instance for this shape, containing the text of the shape and providing access to text formatting properties.

top
The effective top of this placeholder shape; its directly-applied top if it has one, otherwise the top of its parent layout placeholder.

width
The effective width of this placeholder shape; its directly-applied width if it has one, otherwise the width of its parent layout placeholder.

PlaceholderGraphicFrame objects

class pptx.shapes.placeholder.**PlaceholderGraphicFrame**
Placeholder shape populated with a table, chart, or smart art.

chart
The *Chart* object containing the chart in this graphic frame. Raises `ValueError` if this graphic frame does not contain a chart.

click_action
An *ActionSetting* instance providing access to the mouse click behaviors defined on this shape. An *ActionSetting* object is always returned, even when no click behavior is defined on the shape.

element
Reference to the lxml element for this shape, e.g. a CT_Shape instance.

has_chart
`True` if this graphic frame contains a chart object. `False` otherwise. When `True`, the chart object can be accessed using the `.chart` property.

has_table
`True` if this graphic frame contains a table object. `False` otherwise. When `True`, the table object can be accessed using the `.table` property.

height
Read/write. Integer distance between top and bottom extents of shape in EMUs

left
Read/write. Integer distance of the left edge of this shape from the left edge of the slide, in English Metric Units (EMU)

name
Name of this shape, e.g. 'Picture 7'

placeholder_format
A *_PlaceholderFormat* object providing access to placeholder-specific properties such as placeholder type. Raises `ValueError` on access if the shape is not a placeholder.

rotation
Read/write float. Degrees of clockwise rotation. Negative values can be assigned to indicate counter-clockwise rotation, e.g. assigning -45.0 will change setting to 315.0.

shape_id
Read-only positive integer identifying this shape.

The id of a shape is unique among all shapes on a slide.

shape_type
Unique integer identifying the type of this shape, e.g. `MSO_SHAPE_TYPE.TABLE`.

table
The `Table` object contained in this graphic frame. Raises `ValueError` if this graphic frame does not contain a table.

top
Read/write. Integer distance of the top edge of this shape from the top edge of the slide, in English Metric Units (EMU)

width
Read/write. Integer distance between left and right extents of shape in EMUs

PlaceholderPicture objects

class `pptx.shapes.placeholder.`**PlaceholderPicture**
Placeholder shape populated with a picture.

click_action
An *`ActionSetting`* instance providing access to the mouse click behaviors defined on this shape. An *`ActionSetting`* object is always returned, even when no click behavior is defined on the shape.

crop_bottom
A `float` representing the relative portion cropped from the bottom of this shape where 1.0 represents 100%. For example, 25% is represented by 0.25. Negative values are valid as are values greater than 1.0.

crop_left
A `float` representing the relative portion cropped from the left side of this shape where 1.0 represents 100%.

crop_right
A `float` representing the relative portion cropped from the right side of this shape where 1.0 represents 100%.

crop_top
A `float` representing the relative portion cropped from the top of this shape where 1.0 represents 100%.

element
Reference to the lxml element for this shape, e.g. a CT_Shape instance.

height
The effective height of this placeholder shape; its directly-applied height if it has one, otherwise the height of its parent layout placeholder.

image
An *`Image`* object providing access to the properties and bytes of the image in this picture shape.

is_placeholder
True if this shape is a placeholder. A shape is a placeholder if it has a <p:ph> element.

left
The effective left of this placeholder shape; its directly-applied left if it has one, otherwise the left of its parent layout placeholder.

line
An instance of *`LineFormat`*, providing access to the properties of the outline bordering this shape, such as its color and width.

name
Name of this shape, e.g. ‘Picture 7’

placeholder_format
A *_PlaceholderFormat* object providing access to placeholder-specific properties such as placeholder type. Raises `ValueError` on access if the shape is not a placeholder.

rotation
Read/write float. Degrees of clockwise rotation. Negative values can be assigned to indicate counter-clockwise rotation, e.g. assigning -45.0 will change setting to 315.0.

shape_id
Read-only positive integer identifying this shape.

The id of a shape is unique among all shapes on a slide.

shape_type
Member of *MSO_SHAPE_TYPE* specifying the type of this shape. Unconditionally `MSO_SHAPE_TYPE.PLACEHOLDER` in this case. Read-only.

top
The effective top of this placeholder shape; its directly-applied top if it has one, otherwise the top of its parent layout placeholder.

width
The effective width of this placeholder shape; its directly-applied width if it has one, otherwise the width of its parent layout placeholder.

_PlaceholderFormat objects

class `pptx.shapes.base.`**_PlaceholderFormat**
Accessed via the *placeholder_format* property of a placeholder shape, provides properties specific to placeholders, such as the placeholder type.

element
The *p:ph* element proxied by this object.

idx
Integer placeholder 'idx' attribute.

type
Placeholder type, a member of the *PP_PLACEHOLDER_TYPE* enumeration, e.g. PP_PLACEHOLDER.CHART

Table-related objects

Table objects

A *Table* object is added to a slide using the *add_table()* method on *SlideShapes*.

class `pptx.shapes.table.`**Table**
A table shape. Not intended to be constructed directly, use `Slide.shapes.add_table()` to add a table to a slide.

cell(*row_idx*, *col_idx*)
Return table cell at *row_idx*, *col_idx* location. Indexes are zero-based, e.g. cell(0, 0) is the top, left cell.

columns
Read-only reference to collection of *_Column* objects representing the table's columns. *_Column* objects are accessed using list notation, e.g. `col = tbl.columns[0]`.

first_col
Read/write boolean property which, when true, indicates the first column should be formatted differently, as for a side-heading column at the far left of the table.

first_row
Read/write boolean property which, when true, indicates the first row should be formatted differently, e.g. for column headings.

horz_banding
Read/write boolean property which, when true, indicates the rows of the table should appear with alternating shading.

last_col
Read/write boolean property which, when true, indicates the last column should be formatted differently, as for a row totals column at the far right of the table.

last_row
Read/write boolean property which, when true, indicates the last row should be formatted differently, as for a totals row at the bottom of the table.

rows
Read-only reference to collection of *_Row* objects representing the table's rows. *_Row* objects are accessed using list notation, e.g. `col = tbl.rows[0]`.

vert_banding
Read/write boolean property which, when true, indicates the columns of the table should appear with alternating shading.

_Column objects

class pptx.shapes.table.**_Column**
Table column

width
Width of column in EMU.

_Row objects

class pptx.shapes.table.**_Row**
Table row

cells
Read-only reference to collection of cells in row. An individual cell is referenced using list notation, e.g. `cell = row.cells[0]`.

height
Height of row in EMU.

_Cell objects

A *_Cell* object represents a single table cell at a particular row/column location in the table. *_Cell* objects are not constructed directly. A reference to a *_Cell* object is obtained using the *Table.cell()* method, specifying the cell's row/column location.

class pptx.shapes.table.**_Cell**(*tc*, *parent*)
Table cell

fill
FillFormat instance for this cell, providing access to fill properties such as foreground color.

margin_left
Read/write integer value of left margin of cell as a *Length* value object. If assigned `None`, the default value is used, 0.1 inches for left and right margins and 0.05 inches for top and bottom.

margin_right
Right margin of cell.

margin_top
Top margin of cell.

margin_bottom
Bottom margin of cell.

text
Write-only. Assignment to *text* replaces all text currently contained in the cell, resulting in a text frame containing exactly one paragraph, itself containing a single run. The assigned value can be a 7-bit ASCII string, a UTF-8 encoded 8-bit string, or unicode. String values are converted to unicode assuming UTF-8 encoding.

text_frame
TextFrame instance containing the text that appears in the cell.

vertical_anchor
Vertical anchor of this table cell, determines the vertical alignment of text in the cell. Value is like `MSO_ANCHOR.MIDDLE`. Can be `None`, meaning the cell has no vertical anchor setting and its effective value is inherited from a higher-level object.

ChartData objects

A ChartData object is used to specify the data depicted in a chart. It is used when creating a new chart and when replacing the data for an existing chart. Most charts are created using a *CategoryChartData* object, however the data for XY and bubble chart types is different enough that those each require a distinct chart data object.

class `pptx.chart.data.`**ChartData**(*number_format=u'General'*)
ChartData is simply an alias for *CategoryChartData* and may be removed in a future release. All new development should use *CategoryChartData* for creating or replacing the data in chart types other than XY and Bubble.

class `pptx.chart.data.`**CategoryChartData**(*number_format=u'General'*)
Accumulates data specifying the categories and series values for a chart and acts as a proxy for the chart data table that will be written to an Excel worksheet. Used as a parameter in `shapes.add_chart()` and `Chart.replace_data()`.

This object is suitable for use with category charts, i.e. all those having a discrete set of label values (categories) as the range of their independent variable (X-axis) values. Unlike the ChartData types for charts supporting a continuous range of independent variable values (such as XyChartData), CategoryChartData has a single collection of category (X) values and each data point in its series specifies only the Y value. The corresponding X value is inferred by its position in the sequence.

add_category(*label*)
Return a newly created *Category* object having *label* and appended to the end of the category collection for this chart. *label* can be a string, a number, a datetime.date, or datetime.datetime object. All category labels in a chart must be the same type. All category labels in a chart having multi-level categories must be strings.

add_series(*name*, *values=()*, *number_format=None*)
Add a series to this data set entitled *name* and having the data points specified by *values*, an iterable of numeric values. *number_format* specifies how the series values will be displayed, and may be a string, e.g. '#,##0', or an integer in the range 0-22 or 37-49, signifying one of the built-in Excel number formats. The valid integer values and their meaning are documented on the *Excel Number Formats* page.

categories
A *Categories* object providing access to the hierarchy of category objects for this chart data. Assigning an iterable of category labels (strings, numbers, or dates) replaces the *Categories* object with a new one containing a category for each label in the sequence.

Creating a chart from chart data having date categories will cause the chart to have a *DateAxis* for its category axis.

values_ref(*series*)
The Excel worksheet reference to the values for *series* (not including the column heading).

number_format
The formatting template string, e.g. '#,##0.0', that determines how X and Y values are formatted in this chart and in the Excel spreadsheet. A number format specified on a series will override this value for that series. Likewise, a distinct number format can be specified for a particular data point within a series.

class pptx.chart.data.**Categories**
A sequence of *Category* objects, also having certain hierarchical graph behaviors for support of multi-level (nested) categories.

add_category(*label*)
Return a newly created *Category* object having *label* and appended to the end of this category sequence. *label* can be a string, a number, a datetime.date, or datetime.datetime object. All category labels in a chart must be the same type. All category labels in a chart having multi-level categories must be strings.

Creating a chart from chart data having date categories will cause the chart to have a *DateAxis* for its category axis.

are_dates
Return `True` if the first category in this collection has a date label (as opposed to str or numeric). A date label is one of type datetime.date or datetime.datetime. Returns `False` otherwise, including when this category collection is empty. It also returns False when this category collection is hierarchical, because hierarchical categories can only be written as string labels.

are_numeric
Return `True` if the first category in this collection has a numeric label (as opposed to a string label), including if that value is a datetime.date or datetime.datetime object (as those are converted to integers for storage in Excel). Returns `False` otherwise, including when this category collection is empty. It also returns False when this category collection is hierarchical, because hierarchical categories can only be written as string labels.

depth
The number of hierarchy levels in this category graph. Returns 0 if it contains no categories.

index(*category*)
The offset of *category* in the overall sequence of leaf categories. A non-leaf category gets the index of its first sub-category.

leaf_count
The number of leaf-level categories in this hierarchy. The return value is the same as that of *len()* only when the hierarchy is single level.

levels
A generator of (idx, label) sequences representing the category hierarchy from the bottom up. The first

level contains all leaf categories, and each subsequent is the next level up.

number_format
Read/write. Return a string representing the number format used in Excel to format these category values, e.g. '0.0' or 'mm/dd/yyyy'. This string is only relevant when the categories are numeric or date type, although it returns 'General' without error when the categories are string labels. Assigning `None` causes the default number format to be used, based on the type of the category labels.

class pptx.chart.data.**Category**(*label*, *parent*)
A chart category, primarily having a label to be displayed on the category axis, but also able to be configured in a hierarchy for support of multi-level category charts.

add_sub_category(*label*)
Return a newly created *Category* object having *label* and appended to the end of the sub-category sequence for this category.

label
The value that appears on the axis for this category. The label can be a string, a number, or a datetime.date or datetime.datetime object.

numeric_str_val(*date_1904=False*)
The string representation of the numeric (or date) label of this category, suitable for use in the XML *c:pt* element for this category. The optional *date_1904* parameter specifies the epoch used for calculating Excel date numbers.

sub_categories
The sequence of child categories for this category.

class pptx.chart.data.**XyChartData**(*number_format=u'General'*)
A specialized ChartData object suitable for use with an XY (aka. scatter) chart. Unlike ChartData, it has no category sequence. Rather, each data point of each series specifies both an X and a Y value.

add_series(*name*, *number_format=None*)
Return an *XySeriesData* object newly created and added at the end of this sequence, identified by *name* and values formatted with *number_format*.

number_format
The formatting template string, e.g. '#,##0.0', that determines how X and Y values are formatted in this chart and in the Excel spreadsheet. A number format specified on a series will override this value for that series. Likewise, a distinct number format can be specified for a particular data point within a series.

class pptx.chart.data.**BubbleChartData**(*number_format=u'General'*)
A specialized ChartData object suitable for use with a bubble chart. A bubble chart is essentially an XY chart where the markers are scaled to provide a third quantitative dimension to the exhibit.

add_series(*name*, *number_format=None*)
Return a *BubbleSeriesData* object newly created and added at the end of this sequence, and having series named *name* and values formatted with *number_format*.

number_format
The formatting template string, e.g. '#,##0.0', that determines how X and Y values are formatted in this chart and in the Excel spreadsheet. A number format specified on a series will override this value for that series. Likewise, a distinct number format can be specified for a particular data point within a series.

class pptx.chart.data.**XySeriesData**(*chart_data*, *name*, *number_format*)
The data specific to a particular XY chart series. It provides access to the series label, the series data points, and an optional number format to be applied to each data point not having a specified number format.

The sequence of data points in an XY series is significant; lines are plotted following the sequence of points, even if that causes a line segment to "travel backward" (implying a multi-valued function). The data points are not automatically sorted into increasing order by X value.

add_data_point(*x*, *y*, *number_format=None*)
Return an XyDataPoint object newly created with values *x* and *y*, and appended to this sequence.

index
Zero-based integer indicating the sequence position of this series in its chart. For example, the second of three series would return *1*.

name
The name of this series, e.g. 'Series 1'. This name is used as the column heading for the y-values of this series and may also appear in the chart legend and perhaps other chart locations.

number_format
The formatting template string that determines how a number in this series is formatted, both in the chart and in the Excel spreadsheet; for example '#,##0.0'. If not specified for this series, it is inherited from the parent chart data object.

x_values
A sequence containing the X value of each datapoint in this series, in data point order.

y_values
A sequence containing the Y value of each datapoint in this series, in data point order.

class pptx.chart.data.**BubbleSeriesData**(*chart_data*, *name*, *number_format*)
The data specific to a particular Bubble chart series. It provides access to the series label, the series data points, and an optional number format to be applied to each data point not having a specified number format.

The sequence of data points in a bubble chart series is maintained throughout the chart building process because a data point has no unique identifier and can only be retrieved by index.

add_data_point(*x*, *y*, *size*, *number_format=None*)
Append a new BubbleDataPoint object having the values *x*, *y*, and *size*. The optional *number_format* is used to format the Y value. If not provided, the number format is inherited from the series data.

bubble_sizes
A sequence containing the bubble size for each datapoint in this series, in data point order.

data_point_offset
The integer count of data points that appear in all chart series prior to this one.

index
Zero-based integer indicating the sequence position of this series in its chart. For example, the second of three series would return *1*.

name
The name of this series, e.g. 'Series 1'. This name is used as the column heading for the y-values of this series and may also appear in the chart legend and perhaps other chart locations.

number_format
The formatting template string that determines how a number in this series is formatted, both in the chart and in the Excel spreadsheet; for example '#,##0.0'. If not specified for this series, it is inherited from the parent chart data object.

x_values
A sequence containing the X value of each datapoint in this series, in data point order.

y_values
A sequence containing the Y value of each datapoint in this series, in data point order.

Charts

python-pptx provides an API for adding and manipulating charts. A chart object, like a table, is not a shape. Rather it is a graphical object contained in a *GraphicFrame* shape. The shape API, such as position, size, shape id, and name, are provided by the graphic frame shape. The chart itself is accessed using the `chart` property on the graphic frame shape.

`Chart` objects

The *Chart* object is the root of a generally hierarchical graph of component objects that together provide access to the properties and methods required to specify and format a chart.

class `pptx.chart.chart.`**`Chart`**(*chartSpace*, *chart_part*)

A chart object.

`category_axis`

The category axis of this chart. In the case of an XY or Bubble chart, this is the X axis. Raises `ValueError` if no category axis is defined (as is the case for a pie chart, for example).

`chart_style`

Read/write integer index of chart style used to format this chart. Range is from 1 to 48. Value is `None` if no explicit style has been assigned, in which case the default chart style is used. Assigning `None` causes any explicit setting to be removed. The integer index corresponds to the style's position in the chart style gallery in the PowerPoint UI.

`chart_title`

A *ChartTitle* object providing access to title properties.

Calling this property is destructive in the sense it adds a chart title element (*c:title*) to the chart XML if one is not already present. Use `has_title` to test for presence of a chart title non-destructively.

`chart_type`

Read-only *XL_CHART_TYPE* enumeration value specifying the type of this chart. If the chart has two plots, for example, a line plot overlayed on a bar plot, the type reported is for the first (back-most) plot.

`has_legend`

Read/write boolean, `True` if the chart has a legend. Assigning `True` causes a legend to be added to the chart if it doesn't already have one. Assigning False removes any existing legend definition along with any existing legend settings.

`has_title`

Read/write boolean, specifying whether this chart has a title.

Assigning `True` causes a title to be added if not already present. Assigning `False` removes any existing title along with its text and settings.

`legend`

A *Legend* object providing access to the properties of the legend for this chart.

`plots`

The sequence of plots in this chart. A plot, called a *chart group* in the Microsoft API, is a distinct sequence of one or more series depicted in a particular charting type. For example, a chart having a series plotted as a line overlaid on three series plotted as columns would have two plots; the first corresponding to the three column series and the second to the line series. Plots are sequenced in the order drawn, i.e. back-most to front-most. Supports *len()*, membership (e.g. `p in plots`), iteration, slicing, and indexed access (e.g. `plot = plots[i]`).

replace_data(*chart_data*)
Use the categories and series values in the *ChartData* object *chart_data* to replace those in the XML and Excel worksheet for this chart.

series
A SeriesCollection object containing all the series in this chart. When the chart has multiple plots, all the series for the first plot appear before all those for the second, and so on. Series within a plot have an explicit ordering and appear in that sequence.

value_axis
The *ValueAxis* object providing access to properties of the value axis of this chart. Raises ValueError if the chart has no value axis.

class pptx.chart.chart.**ChartTitle**
Provides properties for manipulating a chart title.

format
ChartFormat object providing access to line and fill formatting.

Return the *ChartFormat* object providing shape formatting properties for this chart title, such as its line color and fill.

has_text_frame
Read/write Boolean specifying whether this title has a text frame.

Return True if this chart title has a text frame, and False otherwise. Assigning True causes a text frame to be added if not already present. Assigning False causes any existing text frame to be removed along with its text and formatting.

text_frame
TextFrame instance for this chart title.

Return a *TextFrame* instance allowing read/write access to the text of this chart title and its text formatting properties. Accessing this property is destructive in the sense it adds a text frame if one is not present. Use has_text_frame to test for the presence of a text frame non-destructively.

Legend objects

A legend provides a visual key relating each series of data points to their assigned meaning by mapping a color, line type, or point shape to each series name. A legend is optional, but there can be at most one. Most aspects of a legend are determined automatically, but aspects of its position may be specified via the API.

class pptx.chart.chart.**Legend**
Represents the legend in a chart. A chart can have at most one legend.

font
The *Font* object that provides access to the text properties for this legend, such as bold, italic, etc.

horz_offset
Adjustment of the x position of the legend from its default. Expressed as a float between -1.0 and 1.0 representing a fraction of the chart width. Negative values move the legend left, positive values move it to the right. None if no setting is specified.

include_in_layout
True if legend should be located inside plot area.

Read/write boolean specifying whether legend should be placed inside the plot area. In many cases this will cause it to be superimposed on the chart itself. Assigning None to this property causes any *c:overlay* element to be removed, which is interpreted the same as True. This use case should rarely be required and assigning a boolean value is recommended.

position
Read/write *XL_LEGEND_POSITION* enumeration value specifying the general region of the chart in which to place the legend.

`Axis` objects

A chart typically has two axes, a category axis and a value axis. In general, one of these is horizontal and the other is vertical, where which is which depends on the chart type. For example, the category axis is horizontal on a column chart, but vertical on a bar chart.

A chart where the independent variable is in a continuous (numeric) range, such as an XY/scatter chart, does not have a category axis. Rather it has two value axes.

A category is perhaps most commonly a string label, such as `'East'` or `'Revenue'`; however a category can also be a number or a date (although all categories in a chart must be the same type).

When a chart's categories are dates, the category axis is generally, but not necessarily a *DateAxis* object.

A Chart may have zero to four axes. A pie chart, for example, has neither a category nor a value axis.

class `pptx.chart.axis._BaseAxis`
Base class for chart axis objects. All axis objects share these properties.

axis_title
An *AxisTitle* object providing access to title properties.

Calling this property is destructive in the sense that it adds an axis title element (*c:title*) to the axis XML if one is not already present. Use `has_title` to test for presence of axis title non-destructively.

format
The *ChartFormat* object providing access to the shape formatting properties of this axis, such as its line color and fill.

has_major_gridlines
Read/write boolean value specifying whether this axis has gridlines at its major tick mark locations. Assigning `True` to this property causes major gridlines to be displayed. Assigning `False` causes them to be removed.

has_minor_gridlines
Read/write boolean value specifying whether this axis has gridlines at its minor tick mark locations. Assigning `True` to this property causes minor gridlines to be displayed. Assigning `False` causes them to be removed.

has_title
Read/write boolean specifying whether this axis has a title.

`True` if this axis has a title, `False` otherwise. Assigning `True` causes an axis title to be added if not already present. Assigning `False` causes any existing title to be deleted.

major_gridlines
The *MajorGridlines* object representing the major gridlines for this axis.

major_tick_mark
Read/write *XL_TICK_MARK* value specifying the type of major tick mark to display on this axis.

maximum_scale
Read/write float value specifying the upper limit of the value range for this axis, the number at the top or right of the vertical or horizontal value scale, respectively. The value `None` indicates the upper limit should be determined automatically based on the range of data point values associated with the axis.

minimum_scale
Read/write float value specifying lower limit of value range, the number at the bottom or left of the value scale. `None` if no minimum scale has been set. The value `None` indicates the lower limit should be determined automatically based on the range of data point values associated with the axis.

minor_tick_mark
Read/write *XL_TICK_MARK* value specifying the type of minor tick mark for this axis.

tick_labels
The *TickLabels* instance providing access to axis tick label formatting properties. Tick labels are the numbers appearing on a value axis or the category names appearing on a category axis.

tick_label_position
Read/write *XL_TICK_LABEL_POSITION* value specifying where the tick labels for this axis should appear.

visible
Read/write. `True` if axis is visible, `False` otherwise.

class `pptx.chart.axis.`**CategoryAxis**
A category axis of a chart.

category_type
A member of *XL_CATEGORY_TYPE* specifying the scale type of this axis. Unconditionally `CATEGORY_SCALE` for a *CategoryAxis* object.

class `pptx.chart.axis.`**DateAxis**
A category axis with dates as its category labels and having some special display behaviors such as making length of equal periods equal and normalizing month start dates despite unequal month lengths.

category_type
A member of *XL_CATEGORY_TYPE* specifying the scale type of this axis. Unconditionally `TIME_SCALE` for a *DateAxis* object.

class `pptx.chart.axis.`**AxisTitle**
Provides properties for manipulating axis title.

format
ChartFormat object providing access to shape formatting.

Return the *ChartFormat* object providing shape formatting properties for this axis title, such as its line color and fill.

has_text_frame
Read/write Boolean specifying presence of a text frame.

Return `True` if this axis title has a text frame, and `False` otherwise. Assigning `True` causes a text frame to be added if not already present. Assigning `False` causes any existing text frame to be removed along with any text contained in the text frame.

text_frame
TextFrame instance for this axis title.

Return a *TextFrame* instance allowing read/write access to the text of this axis title and its text formatting properties. Accessing this property is destructive as it adds a new text frame if not already present.

Value Axes

Some axis properties are only relevant to value axes, in particular, those related to numeric values rather than text category labels.

class pptx.chart.axis.**ValueAxis**
An axis having continuous (as opposed to discrete) values. The vertical axis is generally a value axis, however both axes of an XY-type chart are value axes.

crosses
Member of *XL_AXIS_CROSSES* enumeration specifying the point on this axis where the other axis crosses, such as auto/zero, minimum, or maximum. Returns *XL_AXIS_CROSSES.CUSTOM* when a specific numeric crossing point (e.g. 1.5) is defined.

crosses_at
Numeric value on this axis at which the perpendicular axis crosses. Returns `None` if no crossing value is set.

major_unit
The float number of units between major tick marks on this value axis. `None` corresponds to the 'Auto' setting in the UI, and specifies the value should be calculated by PowerPoint based on the underlying chart data.

minor_unit
The float number of units between minor tick marks on this value axis. `None` corresponds to the 'Auto' setting in the UI, and specifies the value should be calculated by PowerPoint based on the underlying chart data.

`MajorGridlines` objects

Gridlines are the vertical and horizontal lines that extend major tick marks of an axis across the chart to ease comparison of a data point with the axis divisions.

class pptx.chart.axis.**MajorGridlines**
Provides access to the properties of the major gridlines appearing on an axis.

format
The *ChartFormat* object providing access to the shape formatting properties of this data point, such as line and fill.

`TickLabels` objects

Tick labels are the numbers appearing on a value axis or the category names appearing on a category axis. Certain formatting options are available for changing how these labels are displayed.

class pptx.chart.axis.**TickLabels**
A service class providing access to formatting of axis tick mark labels.

font
The *Font* object that provides access to the text properties for these tick labels, such as bold, italic, etc.

number_format
Read/write string (e.g. "$#,##0.00") specifying the format for the numbers on this axis. The syntax for these strings is the same as it appears in the PowerPoint or Excel UI. Returns 'General' if no number format has been set. Note that this format string has no effect on rendered tick labels when `number_format_is_linked()` is `True`. Assigning a format string to this property automatically sets `number_format_is_linked()` to `False`.

number_format_is_linked
Read/write boolean specifying whether number formatting should be taken from the source spreadsheet rather than the value of `number_format()`.

offset
Read/write int value in range 0-1000 specifying the spacing between the tick mark labels and the axis as a percentange of the default value. 100 if no label offset setting is present.

_BasePlot objects

A *plot* is a group of series all depicted using the same charting type, e.g. bar, column, line, etc. Most charts have only a single plot; however, a chart may have multiple, as in where a line plot appears overlaid on a bar plot in the same chart. In the Microsoft API, this concept has the name *chart group*. The term *plot* was chosen for *python-pptx* to avoid the common mistake of understanding a chart group to be a group of chart objects.

Certain properties must be set at the plot level. Some of those properties are not present on plots of all chart types. For example, `gap_width` is only present on a bar or column plot.

class `pptx.chart.plot.`**_BasePlot**
A distinct plot that appears in the plot area of a chart. A chart may have more than one plot, in which case they appear as superimposed layers, such as a line plot appearing on top of a bar chart.

categories
Returns a *Categories* sequence object containing a *Category* object for each of the category labels associated with this plot. The *Category* class derives from `str`, so the returned value can be treated as a simple sequence of strings for the common case where all you need is the labels in the order they appear on the chart. *Categories* provides additional properties for dealing with hierarchical categories when required.

chart
The *Chart* object containing this plot.

data_labels
DataLabels instance providing properties and methods on the collection of data labels associated with this plot.

has_data_labels
Read/write boolean, `True` if the series has data labels. Assigning `True` causes data labels to be added to the plot. Assigning False removes any existing data labels.

series
A sequence of `Series` objects representing the series in this plot, in the order they appear in the plot.

vary_by_categories
Read/write boolean value specifying whether to use a different color for each of the points in this plot. Only effective when there is a single series; PowerPoint automatically varies color by series when more than one series is present.

BarPlot objects

The following properties are only present on bar-type plots, which includes both bar and column charts.

class `pptx.chart.plot.`**BarPlot**
A bar chart-style plot.

gap_width
Width of gap between bar(s) of each category, as an integer percentage of the bar width. The default value for a new bar chart is 150, representing 150% or 1.5 times the width of a single bar.

overlap
Read/write int value in range -100..100 specifying a percentage of the bar width by which to overlap adjacent bars in a multi-series bar chart. Default is 0. A setting of -100 creates a gap of a full bar width

and a setting of 100 causes all the bars in a category to be superimposed. A stacked bar plot has overlap of 100 by default.

`BubblePlot` objects

The following properties are only present on bubble-type plots.

class `pptx.chart.plot.`**`BubblePlot`**
A bubble chart plot.

`bubble_scale`
An integer between 0 and 300 inclusive indicating the percentage of the default size at which bubbles should be displayed. Assigning `None` produces the same behavior as assigning *100*.

`Categories` objects

Category plots provide access to a *`Categories`* object with their *.categories* property.

class `pptx.chart.category.`**`Categories`**
A sequence of *`Category`* objects, each representing a category label on the chart. Provides properties for dealing with hierarchical categories.

`depth`
Return an integer representing the number of hierarchical levels in this category collection. Returns 1 for non-hierarchical categories and 0 if no categories are present (generally meaning no series are present).

`flattened_labels`
Return a sequence of tuples, each containing the flattened hierarchy of category labels for a leaf category. Each tuple is in parent -> child order, e.g. `('US', 'CA', 'San Francisco')`, with the leaf category appearing last. If this categories collection is non-hierarchical, each tuple will contain only a leaf category label. If the plot has no series (and therefore no categories), an empty tuple is returned.

`levels`
Return a sequence of *`CategoryLevel`* objects representing the hierarchy of this category collection. The sequence is empty when the category collection is not hierarchical, that is, contains only leaf-level categories. The levels are ordered from the leaf level to the root level; so the first level will contain the same categories as this category collection.

`Category` objects

class `pptx.chart.category.`**`Category`**
An extension of *str* that provides the category label as its string value, and additional attributes representing other aspects of the category.

`idx`
Return an integer representing the index reference of this category. For a leaf node, the index identifies the category. For a parent (or other ancestor) category, the index specifies the first leaf category that ancestor encloses.

`label`
Return the label of this category as a string.

`CategoryLevel` objects

class `pptx.chart.category.`**`CategoryLevel`**

A sequence of *`Category`* objects representing a single level in a hierarchical category collection. This object is only used when the categories are hierarchical, meaning they have more than one level and higher level categories group those at lower levels.

`DataLabels` objects

A *data label* is text that labels a particular data point, usually with its value, allowing the point to be interpreted more clearly than just visually comparing its marker with its axis.

A *`DataLabels`* object is not a collection, such as a sequence, and it does not provide access to individual data points. Rather, it provides properties that allow all the data labels in its scope to be formatted at once.

class `pptx.chart.datalabel.`**`DataLabels`**

Collection of data labels associated with a plot, and perhaps with a series or data point, although the latter two are not yet implemented.

`font`

The *`Font`* object that provides access to the text properties for these data labels, such as bold, italic, etc.

`number_format`

Read/write string specifying the format for the numbers on this set of data labels. Returns 'General' if no number format has been set. Note that this format string has no effect on rendered data labels when `number_format_is_linked()` is `True`. Assigning a format string to this property automatically sets `number_format_is_linked()` to `False`.

`number_format_is_linked`

Read/write boolean specifying whether number formatting should be taken from the source spreadsheet rather than the value of `number_format()`.

`position`

Read/write *XL_DATA_LABEL_POSITION* enumeration value specifying the position of the data labels with respect to their data point, or `None` if no position is specified. Assigning `None` causes PowerPoint to choose the default position, which varies by chart type.

class `pptx.chart.datalabel.`**`DataLabel`**

The data label associated with an individual data point.

`font`

The *`Font`* object providing text formatting for this data label.

This font object is used to customize the appearance of automatically inserted text, such as the data point value. The font applies to the entire data label. More granular control of the appearance of custom data label text is controlled by a font object on runs in the text frame.

`has_text_frame`

Return `True` if this data label has a text frame (implying it has custom data label text), and `False` otherwise. Assigning `True` causes a text frame to be added if not already present. Assigning `False` causes any existing text frame to be removed along with any text contained in the text frame.

`position`

Read/write *XL_DATA_LABEL_POSITION* member specifying the position of this data label with respect to its data point, or `None` if no position is specified. Assigning `None` causes PowerPoint to choose the default position, which varies by chart type.

text_frame
TextFrame instance for this data label, containing the text of the data label and providing access to its text formatting properties.

Series objects

A *series* is a sequence of data points that represent a coherent set of observations across each of the categories in the chart. For example, on a chart having regional categories "West", "East", and "Mid-west", a series might be "Q1 Sales" and have values 42, 120, and 34. The series in this case coheres around the first quarter time period.

In general, the type (class) of a series object depends upon the chart type. The following properties are available on series objects of all types.

class pptx.chart.series.**_BaseSeries**
Base class for *BarSeries* and other series classes.

format
The *ChartFormat* instance for this series, providing access to shape properties such as fill and line.

index
The zero-based integer index of this series as reported in its *c:ser/c:idx* element.

name
The string label given to this series, appears as the title of the column for this series in the Excel worksheet. It also appears as the label for this series in the legend.

BarSeries objects

These properties are available on a series belonging to a bar-type plot.

class pptx.chart.series.**BarSeries**
A data point series belonging to a bar plot.

invert_if_negative
`True` if a point having a value less than zero should appear with a fill different than those with a positive value. `False` if the fill should be the same regardless of the bar's value. When `True`, a bar with a solid fill appears with white fill; in a bar with gradient fill, the direction of the gradient is reversed, e.g. dark -> light instead of light -> dark. The term "invert" here should be understood to mean "invert the *direction* of the *fill gradient*".

format
The *ChartFormat* instance for this series, providing access to shape properties such as fill and line.

index
The zero-based integer index of this series as reported in its *c:ser/c:idx* element.

name
The string label given to this series, appears as the title of the column for this series in the Excel worksheet. It also appears as the label for this series in the legend.

points
The *CategoryPoints* object providing access to individual data points in this series.

values
Read-only. A sequence containing the float values for this series, in the order they appear on the chart.

`LineSeries` objects

These properties are available on a series belonging to a line-type plot.

class `pptx.chart.series.`**LineSeries**
A data point series belonging to a line plot.

smooth
Read/write boolean specifying whether to use curve smoothing to form the line connecting the data points in this series into a continuous curve. If `False`, a series of straight line segments are used to connect the points.

format
The *ChartFormat* instance for this series, providing access to shape properties such as fill and line.

index
The zero-based integer index of this series as reported in its *c:ser/c:idx* element.

marker
The `Marker` instance for this series, providing access to data point marker properties such as fill and line. Setting these properties determines the appearance of markers for all points in this series that are not overridden by settings at the point level.

name
The string label given to this series, appears as the title of the column for this series in the Excel worksheet. It also appears as the label for this series in the legend.

points
The *CategoryPoints* object providing access to individual data points in this series.

values
Read-only. A sequence containing the float values for this series, in the order they appear on the chart.

`XySeries` objects

These properties are available on series belonging to an XY plot.

class `pptx.chart.series.`**XySeries**
A data point series belonging to an XY (scatter) plot.

iter_values()
Generate each float Y value in this series, in the order they appear on the chart. A value of *None* represents a missing Y value (corresponding to a blank Excel cell).

points
The *XyPoints* object providing access to individual data points in this series.

values
Read-only. A sequence containing the float values for this series, in the order they appear on the chart.

format
The *ChartFormat* instance for this series, providing access to shape properties such as fill and line.

index
The zero-based integer index of this series as reported in its *c:ser/c:idx* element.

marker
The `Marker` instance for this series, providing access to data point marker properties such as fill and line. Setting these properties determines the appearance of markers for all points in this series that are not overridden by settings at the point level.

name
The string label given to this series, appears as the title of the column for this series in the Excel worksheet. It also appears as the label for this series in the legend.

`Point` objects

An XY or bubble chart has a `points` attribute providing access to a sequence of *`Point`* objects. That sequence supports iteration, indexed access, and `len()`.

class `pptx.chart.point.`**CategoryPoints**
Sequence providing access to individual *`Point`* objects, each representing the visual properties of a data point in the specified category series.

class `pptx.chart.point.`**BubblePoints**
Sequence providing access to the individual data points in a `BubbleSeries` object.

class `pptx.chart.point.`**XyPoints**
Sequence providing access to the individual data points in an *`XySeries`* object.

class `pptx.chart.point.`**Point**
Provides access to the properties of an individual data point in a series, such as the visual properties of its marker and the text and font of its data label.

data_label
The *`DataLabel`* object representing the label on this data point.

format
The *`ChartFormat`* object providing access to the shape formatting properties of this data point, such as line and fill.

marker
The `Marker` instance for this point, providing access to the visual properties of the data point marker, such as fill and line. Setting these properties overrides any value set at the series level.

Text-related objects

`TextFrame` objects

class `pptx.text.text.`**TextFrame**
The part of a shape that contains its text. Not all shapes have a text frame. Corresponds to the `<p:txBody>` element that can appear as a child element of `<p:sp>`. Not intended to be constructed directly.

add_paragraph()
Return new *`_Paragraph`* instance appended to the sequence of paragraphs contained in this text frame.

auto_size
The type of automatic resizing that should be used to fit the text of this shape within its bounding box when the text would otherwise extend beyond the shape boundaries. May be `None`, `MSO_AUTO_SIZE.NONE`, `MSO_AUTO_SIZE.SHAPE_TO_FIT_TEXT`, or `MSO_AUTO_SIZE.TEXT_TO_FIT_SHAPE`.

clear()
Remove all paragraphs except one empty one.

fit_text(*font_family='Calibri'*, *max_size=18*, *bold=False*, *italic=False*, *font_file=None*)
Make the text in this text frame fit entirely within the bounds of its shape by setting word wrap on and applying the "best-fit" font size to all the text it contains. *`TextFrame.auto_size`* is set to `MSO_AUTO_SIZE.NONE`. The font size will not be set larger than *max_size* points. If the path to a

matching TrueType font is provided as *font_file*, that font file will be used for the font metrics. If *font_file* is `None`, best efforts are made to locate a font file with matchhing *font_family*, *bold*, and *italic* installed on the current system (usually succeeds if the font is installed).

margin_bottom
Length value representing the inset of text from the bottom text frame border. *pptx.util.Inches()* provides a convenient way of setting the value, e.g. `text_frame.margin_bottom = Inches(0.05)`.

margin_left
Inset of text from left text frame border as *Length* value.

margin_right
Inset of text from right text frame border as *Length* value.

margin_top
Inset of text from top text frame border as *Length* value.

paragraphs
Immutable sequence of *_Paragraph* instances corresponding to the paragraphs in this text frame. A text frame always contains at least one paragraph.

text
Read/write. All the text in this text frame as a single unicode string. A line feed character ('\n') appears in the string for each paragraph and line break in the shape, except the last paragraph. A shape containing a single paragraph with no line breaks will produce a string having no line feed characters. Assignment replaces all text in the text frame with a single paragraph containing the assigned text. The assigned value can be a 7-bit ASCII string, a UTF-8 encoded 8-bit string, or unicode. String values are converted to unicode assuming UTF-8 encoding. Each line feed character in the assigned string is translated to a line break.

vertical_anchor
Read/write member of *MSO_VERTICAL_ANCHOR* enumeration or `None`, representing the vertical alignment of text in this text frame. `None` indicates the effective value should be inherited from this object's style hierarchy.

word_wrap
Read-write setting determining whether lines of text in this shape are wrapped to fit within the shape's width. Valid values are True, False, or None. True and False turn word wrap on and off, respectively. Assigning None to word wrap causes any word wrap setting to be removed from the text frame, causing it to inherit this setting from its style hierarchy.

`Font` objects

The *Font* object is encountered as a property of *_Run*, *_Paragraph*, and in future other presentation text objects.

class `pptx.text.text.`**Font**
Character properties object, providing font size, font name, bold, italic, etc. Corresponds to `<a:rPr>` child element of a run. Also appears as `<a:defRPr>` and `<a:endParaRPr>` in paragraph and `<a:defRPr>` in list style elements.

bold
Get or set boolean bold value of *Font*, e.g. `paragraph.font.bold = True`. If set to `None`, the bold setting is cleared and is inherited from an enclosing shape's setting, or a setting in a style or master. Returns None if no bold attribute is present, meaning the effective bold value is inherited from a master or the theme.

color
The *ColorFormat* instance that provides access to the color settings for this font.

fill
FillFormat instance for this font, providing access to fill properties such as fill color.

italic
Get or set boolean italic value of *Font* instance, with the same behaviors as bold with respect to None values.

language_id
Get or set the language id of this *Font* instance. The language id is a member of the *MSO_LANGUAGE_ID* enumeration. Assigning `None` removes any language setting, the same behavior as assigning *MSO_LANGUAGE_ID.NONE*.

name
Get or set the typeface name for this *Font* instance, causing the text it controls to appear in the named font, if a matching font is found. Returns `None` if the typeface is currently inherited from the theme. Setting it to `None` removes any override of the theme typeface.

size
Read/write *Length* value or `None`, indicating the font height in English Metric Units (EMU). `None` indicates the font size should be inherited from its style hierarchy, such as a placeholder or document defaults (usually 18pt). *Length* is a subclass of `int` having properties for convenient conversion into points or other length units. Likewise, the *pptx.util.Pt* class allows convenient specification of point values:

```
>> font.size = Pt(24)
>> font.size
304800
>> font.size.pt
24.0
```

underline
Read/write. `True`, `False`, `None`, or a member of the *MSO_TEXT_UNDERLINE_TYPE* enumeration indicating the underline setting for this font. `None` is the default and indicates the underline setting should be inherited from the style hierarchy, such as from a placeholder. `True` indicates single underline. `False` indicates no underline. Other settings such as double and wavy underlining are indicated with members of the *MSO_TEXT_UNDERLINE_TYPE* enumeration.

_Paragraph objects

class pptx.text.text.**_Paragraph**
Paragraph object. Not intended to be constructed directly.

add_run()
Return a new run appended to the runs in this paragraph.

alignment
Horizontal alignment of this paragraph, represented by either a member of the enumeration *PP_PARAGRAPH_ALIGNMENT* or `None`. The value `None` indicates the paragraph should 'inherit' its effective value from its style hierarchy. Assigning `None` removes any explicit setting, causing its inherited value to be used.

clear()
Remove all content from this paragraph. Paragraph properties are preserved. Content includes runs, line breaks, and fields.

font
Font object containing default character properties for the runs in this paragraph. These character prop-

erties override default properties inherited from parent objects such as the text frame the paragraph is contained in and they may be overridden by character properties set at the run level.

level
Read-write integer indentation level of this paragraph, having a range of 0-8 inclusive. 0 represents a top-level paragraph and is the default value. Indentation level is most commonly encountered in a bulleted list, as is found on a word bullet slide.

line_spacing
Numeric or *Length* value specifying the space between baselines in successive lines of this paragraph. A value of `None` indicates no explicit value is assigned and its effective value is inherited from the paragraph's style hierarchy. A numeric value, e.g. *2* or *1.5*, indicates spacing is applied in multiples of line heights. A *Length* value such as `Pt(12)` indicates spacing is a fixed height. The *Pt* value class is a convenient way to apply line spacing in units of points.

runs
Immutable sequence of *_Run* objects corresponding to the runs in this paragraph.

space_after
Length value specifying the spacing to appear between this paragraph and the subsequent paragraph. A value of `None` indicates no explicit value is assigned and its effective value is inherited from the paragraph's style hierarchy. *Length* objects provide convenience properties, such as `.pt` and `.inches`, that allow easy conversion to various length units.

space_before
Length value specifying the spacing to appear between this paragraph and the prior paragraph. A value of `None` indicates no explicit value is assigned and its effective value is inherited from the paragraph's style hierarchy. *Length* objects provide convenience properties, such as `.pt` and `.cm`, that allow easy conversion to various length units.

text
Read/write. A single unicode string containing all the text in this paragraph. Formed by concatenating the text in each run and field making up the paragraph, adding a line feed character ('\n') for each line break element encountered. Assignment causes all content in the paragraph to be replaced by a single run containing the assigned text. Each line feed character in the assigned text is replaced with a line break. The assigned value can be a 7-bit ASCII string, a UTF-8 encoded 8-bit string, or unicode. String values are converted to unicode assuming UTF-8 encoding.

_Run objects

class `pptx.text.text._Run`
Text run object. Corresponds to `<a:r>` child element in a paragraph.

font
Font instance containing run-level character properties for the text in this run. Character properties can be and perhaps most often are inherited from parent objects such as the paragraph and slide layout the run is contained in. Only those specifically overridden at the run level are contained in the font object.

hyperlink
`_Hyperlink` instance acting as proxy for any `<a:hlinkClick>` element under the run properties element. Created on demand, the hyperlink object is available whether an `<a:hlinkClick>` element is present or not, and creates or deletes that element as appropriate in response to actions on its methods and attributes.

text
Read/write. A unicode string containing the text in this run. Assignment replaces all text in the run. The assigned value can be a 7-bit ASCII string, a UTF-8 encoded 8-bit string, or unicode. String values are converted to unicode assuming UTF-8 encoding.

Click Action-related Objects

The following classes represent click and hover mouse actions, typically a hyperlink. Other actions such as navigating to another slide in the presentation or running a macro are also possible.

`ActionSetting` objects

class `pptx.action.`**`ActionSetting`**

Properties that specify how a shape or text run reacts to mouse actions during a slide show.

`action`

A member of the *PP_ACTION_TYPE* enumeration, such as *PP_ACTION.HYPERLINK*, indicating the type of action that will result when the specified shape or text is clicked or the mouse pointer is positioned over the shape during a slide show.

`hyperlink`

A *`Hyperlink`* object representing the hyperlink action defined on this click or hover mouse event. A *`Hyperlink`* object is always returned, even if no hyperlink or other click action is defined.

`part`

The package part containing this object

`target_slide`

A reference to the slide in this presentation that is the target of the slide jump action in this shape. Slide jump actions include *PP_ACTION.FIRST_SLIDE*, *LAST_SLIDE*, *NEXT_SLIDE*, *PREVIOUS_SLIDE*, and *NAMED_SLIDE*. Returns `None` for all other actions. In particular, the *LAST_SLIDE_VIEWED* action and the *PLAY* (start other presentation) actions are not supported.

`Hyperlink` objects

class `pptx.action.`**`Hyperlink`**

Represents a hyperlink action on a shape or text run.

`address`

Read/write. The URL of the hyperlink. URL can be on http, https, mailto, or file scheme; others may work. Returns `None` if no hyperlink is defined, including when another action such as *RUN_MACRO* is defined on the object. Assigning `None` removes any action defined on the object, whether it is a hyperlink action or not.

`part`

The package part containing this object

DrawingML objects

Low-level drawing elements like fill and color that appear repeatedly in various aspects of shapes.

`ChartFormat` objects

class `pptx.dml.chtfmt.`**`ChartFormat`**(*element*)

The *`ChartFormat`* object provides access to visual shape properties for chart elements like `Axis`, `Series`, and *`MajorGridlines`*. It has two properties, `fill` and `line`, which return a *`FillFormat`* and

LineFormat object respectively. The *ChartFormat* object is provided by the `format` property on the target axis, series, etc.

fill
FillFormat instance for this object, providing access to fill properties such as fill color.

line
The *LineFormat* object providing access to the visual properties of this object, such as line color and line style.

`FillFormat` objects

class `pptx.dml.fill.`**FillFormat**(*eg_fill_properties_parent*, *fill_obj*)
Provides access to the current fill properties object and provides methods to change the fill type.

back_color
Return a *ColorFormat* object representing background color.

This property is only applicable to pattern fills and lines.

background()
Sets the fill type to noFill, i.e. transparent.

fore_color
Return a *ColorFormat* instance representing the foreground color of this fill.

pattern
Return member of *MSO_PATTERN_TYPE* indicating fill pattern.

Raises `TypeError` when fill is not patterned (call *fill.patterned()* first). Returns `None` if no pattern has been set; PowerPoint may display the default *PERCENT_5* pattern in this case. Assigning `None` will remove any explicit pattern setting, although relying on the default behavior is discouraged and may produce rendering differences across client applications.

patterned()
Selects the pattern fill type.

Note that calling this method does not by itself set a foreground or background color of the pattern. Rather it enables subsequent assignments to properties like fore_color to set the pattern and colors.

solid()
Sets the fill type to solid, i.e. a solid color. Note that calling this method does not set a color or by itself cause the shape to appear with a solid color fill; rather it enables subsequent assignments to properties like fore_color to set the color.

type
Return a value from the *MSO_FILL_TYPE* enumeration corresponding to the type of this fill.

`LineFormat` objects

class `pptx.dml.line.`**LineFormat**(*parent*)
Provides access to line properties such as color, style, and width.

A LineFormat object is typically accessed via the `.line` property of a shape such as *Shape* or *Picture*.

color
The *ColorFormat* instance that provides access to the color settings for this line. Essentially a shortcut for `line.fill.fore_color`. As a side-effect, accessing this property causes the line fill type to

be set to `MSO_FILL.SOLID`. If this sounds risky for your use case, use `line.fill.type` to non-destructively discover the existing fill type.

dash_style
Return value indicating line style.

Returns a member of *MSO_LINE_DASH_STYLE* indicating line style, or `None` if no explicit value has been set. When no explicit value has been set, the line dash style is inherited from the style hierarchy.

Assigning `None` removes any existing explicitly-defined dash style.

fill
FillFormat instance for this line, providing access to fill properties such as foreground color.

width
The width of the line expressed as an integer number of *English Metric Units*. The returned value is an instance of *Length*, a value class having properties such as *.inches*, *.cm*, and *.pt* for converting the value into convenient units.

ColorFormat objects

class pptx.dml.color.**ColorFormat**(*eg_colorChoice_parent*, *color*)
Provides access to color settings such as RGB color, theme color, and luminance adjustments.

brightness
Read/write float value between -1.0 and 1.0 indicating the brightness adjustment for this color, e.g. -0.25 is 25% darker and 0.4 is 40% lighter. 0 means no brightness adjustment.

rgb
RGBColor value of this color, or None if no RGB color is explicitly defined for this font. Setting this value to an *RGBColor* instance causes its type to change to MSO_COLOR_TYPE.RGB. If the color was a theme color with a brightness adjustment, the brightness adjustment is removed when changing it to an RGB color.

theme_color
Theme color value of this color, one of those defined in the MSO_THEME_COLOR enumeration, e.g. MSO_THEME_COLOR.ACCENT_1. Raises AttributeError on access if the color is not type `MSO_COLOR_TYPE.SCHEME`. Assigning a value in `MSO_THEME_COLOR` causes the color's type to change to `MSO_COLOR_TYPE.SCHEME`.

type
Read-only. A value from *MSO_COLOR_TYPE*, either RGB or SCHEME, corresponding to the way this color is defined, or None if no color is defined at the level of this font.

RGBColor objects

class pptx.dml.color.**RGBColor**
Immutable value object defining a particular RGB color.

classmethod from_string(*rgb_hex_str*)
Return a new instance from an RGB color hex string like `'3C2F80'`.

Image

An image depicted in a *Picture* shape can be accessed using its *image* property. The *Image* object provides access to detailed properties of the image itself, including the bytes of the image file itself.

`Image` objects

The *Image* object is encountered as the `image` property of *Picture*.

class `pptx.parts.image.`**Image**
Immutable value object representing an image such as a JPEG, PNG, or GIF.

blob
The binary image bytestream of this image.

content_type
MIME-type of this image, e.g. `'image/jpeg'`.

dpi
A (horz_dpi, vert_dpi) 2-tuple specifying the dots-per-inch resolution of this image. A default value of (72, 72) is used if the dpi is not specified in the image file.

ext
Canonical file extension for this image e.g. `'png'`. The returned extension is all lowercase and is the canonical extension for the content type of this image, regardless of what extension may have been used in its filename, if any.

filename
The filename from the path from which this image was loaded, if loaded from the filesystem. `None` if no filename was used in loading, such as when loaded from an in-memory stream.

sha1
SHA1 hash digest of the image blob

size
A (width, height) 2-tuple specifying the dimensions of this image in pixels.

Exceptions

Exceptions used with python-pptx.

The base exception class is PythonPptxError.

exception `pptx.exc.`**PythonPptxError**
Generic error class.

exception `pptx.exc.`**PackageNotFoundError**
Raised when a package cannot be found at the specified path.

exception `pptx.exc.`**InvalidXmlError**
Raised when a value is encountered in the XML that is not valid according to the schema.

`util` Module

Utility functions and classes that come in handy when working with PowerPoint and Open XML.

class `pptx.util.`**Length**
Bases: `int`

Base class for length classes Inches, Emu, Cm, Mm, Pt, and Px. Provides properties for converting length values to convenient units.

inches
Floating point length in inches

centipoints
Integer length in hundredths of a point (1/7200 inch). Used internally because PowerPoint stores font size in centipoints.

cm
Floating point length in centimeters

emu
Integer length in English Metric Units

mm
Floating point length in millimeters

pt
Floating point length in points

class pptx.util.**Inches**
Bases: *pptx.util.Length*

Convenience constructor for length in inches

class pptx.util.**Centipoints**
Bases: *pptx.util.Length*

Convenience constructor for length in hundredths of a point

class pptx.util.**Cm**
Bases: *pptx.util.Length*

Convenience constructor for length in centimeters

class pptx.util.**Emu**
Bases: *pptx.util.Length*

Convenience constructor for length in english metric units

class pptx.util.**Mm**
Bases: *pptx.util.Length*

Convenience constructor for length in millimeters

class pptx.util.**Pt**
Bases: *pptx.util.Length*

Convenience value class for specifying a length in points

Enumerations

Documentation for the various enumerations used for *python-pptx* property settings can be found here:

MSO_AUTO_SHAPE_TYPE

Specifies a type of AutoShape, e.g. DOWN_ARROW

Alias: `MSO_SHAPE`

Example:

```
from pptx.enum.shapes import MSO_SHAPE
from pptx.util import Inches

left = top = width = height = Inches(1.0)
slide.shapes.add_shape(
    MSO_SHAPE.ROUNDED_RECTANGLE, left, top, width, height
)
```

ACTION_BUTTON_BACK_OR_PREVIOUS Back or Previous button. Supports mouse-click and mouse-over actions

ACTION_BUTTON_BEGINNING Beginning button. Supports mouse-click and mouse-over actions.

ACTION_BUTTON_CUSTOM Button with no default picture or text. Supports mouse-click and mouse-over action.

ACTION_BUTTON_DOCUMENT Document button. Supports mouse-click and mouse-over actions.

ACTION_BUTTON_END End button. Supports mouse-click and mouse-over actions.

ACTION_BUTTON_FORWARD_OR_NEXT Forward or Next button. Supports mouse-click and mouse-over actions.

ACTION_BUTTON_HELP Help button. Supports mouse-click and mouse-over actio.

ACTION_BUTTON_HOME Home button. Supports mouse-click and mouse-over actio.

ACTION_BUTTON_INFORMATION Information button. Supports mouse-click and mouse-over actions.

ACTION_BUTTON_MOVIE Movie button. Supports mouse-click and mouse-over actions.

ACTION_BUTTON_RETURN Return button. Supports mouse-click and mouse-over actions.

ACTION_BUTTON_SOUND Sound button. Supports mouse-click and mouse-over actions.

ARC Arc

BALLOON Rounded Rectangular Callout

BENT_ARROW Block arrow that follows a curved 90-degree angle

BENT_UP_ARROW Block arrow that follows a sharp 90-degree angle. Points up by default.

BEVEL Bevel

BLOCK_ARC Block arc

CAN Can

CHART_PLUS Chart Plus

CHART_STAR Chart Star

CHART_X Chart X

CHEVRON Chevron

CHORD Geometric chord shape

CIRCULAR_ARROW Block arrow that follows a curved 180-degree angle

CLOUD Cloud

CLOUD_CALLOUT Cloud callout

CORNER Corner

CORNER_TABS Corner Tabs

CROSS Cross

CUBE Cube

CURVED_DOWN_ARROW Block arrow that curves down

CURVED_DOWN_RIBBON Ribbon banner that curves down

CURVED_LEFT_ARROW Block arrow that curves left

CURVED_RIGHT_ARROW Block arrow that curves right

CURVED_UP_ARROW Block arrow that curves up

CURVED_UP_RIBBON Ribbon banner that curves up

DECAGON Decagon

DIAGONAL_STRIPE Diagonal Stripe

DIAMOND Diamond

DODECAGON Dodecagon

DONUT Donut

DOUBLE_BRACE Double brace

DOUBLE_BRACKET Double bracket

DOUBLE_WAVE Double wave

DOWN_ARROW Block arrow that points down

DOWN_ARROW_CALLOUT Callout with arrow that points down

DOWN_RIBBON Ribbon banner with center area below ribbon ends

EXPLOSION1 Explosion

EXPLOSION2 Explosion

FLOWCHART_ALTERNATE_PROCESS Alternate process flowchart symbol

FLOWCHART_CARD Card flowchart symbol

FLOWCHART_COLLATE Collate flowchart symbol

FLOWCHART_CONNECTOR Connector flowchart symbol

FLOWCHART_DATA Data flowchart symbol

FLOWCHART_DECISION Decision flowchart symbol

FLOWCHART_DELAY Delay flowchart symbol

FLOWCHART_DIRECT_ACCESS_STORAGE Direct access storage flowchart symbol

FLOWCHART_DISPLAY Display flowchart symbol

FLOWCHART_DOCUMENT Document flowchart symbol

FLOWCHART_EXTRACT Extract flowchart symbol

FLOWCHART_INTERNAL_STORAGE Internal storage flowchart symbol

FLOWCHART_MAGNETIC_DISK Magnetic disk flowchart symbol

FLOWCHART_MANUAL_INPUT Manual input flowchart symbol

FLOWCHART_MANUAL_OPERATION Manual operation flowchart symbol

FLOWCHART_MERGE Merge flowchart symbol

FLOWCHART_MULTIDOCUMENT Multi-document flowchart symbol

FLOWCHART_OFFLINE_STORAGE Offline Storage

FLOWCHART_OFFPAGE_CONNECTOR Off-page connector flowchart symbol

FLOWCHART_OR "Or" flowchart symbol

FLOWCHART_PREDEFINED_PROCESS Predefined process flowchart symbol

FLOWCHART_PREPARATION Preparation flowchart symbol

FLOWCHART_PROCESS Process flowchart symbol

FLOWCHART_PUNCHED_TAPE Punched tape flowchart symbol

FLOWCHART_SEQUENTIAL_ACCESS_STORAGE Sequential access storage flowchart symbol

FLOWCHART_SORT Sort flowchart symbol

FLOWCHART_STORED_DATA Stored data flowchart symbol

FLOWCHART_SUMMING_JUNCTION Summing junction flowchart symbol

FLOWCHART_TERMINATOR Terminator flowchart symbol

FOLDED_CORNER Folded corner

FRAME Frame

FUNNEL Funnel

GEAR_6 Gear 6

GEAR_9 Gear 9

HALF_FRAME Half Frame

HEART Heart

HEPTAGON Heptagon

HEXAGON Hexagon

HORIZONTAL_SCROLL Horizontal scroll

ISOSCELES_TRIANGLE Isosceles triangle

LEFT_ARROW Block arrow that points left

LEFT_ARROW_CALLOUT Callout with arrow that points left

LEFT_BRACE Left brace

LEFT_BRACKET Left bracket

LEFT_CIRCULAR_ARROW Left Circular Arrow

LEFT_RIGHT_ARROW Block arrow with arrowheads that point both left and right

LEFT_RIGHT_ARROW_CALLOUT Callout with arrowheads that point both left and right

LEFT_RIGHT_CIRCULAR_ARROW Left Right Circular Arrow

LEFT_RIGHT_RIBBON Left Right Ribbon

LEFT_RIGHT_UP_ARROW Block arrow with arrowheads that point left, right, and up

LEFT_UP_ARROW Block arrow with arrowheads that point left and up

LIGHTNING_BOLT Lightning bolt

LINE_CALLOUT_1 Callout with border and horizontal callout line

LINE_CALLOUT_1_ACCENT_BAR Callout with vertical accent bar

LINE_CALLOUT_1_BORDER_AND_ACCENT_BAR Callout with border and vertical accent bar

LINE_CALLOUT_1_NO_BORDER Callout with horizontal line

LINE_CALLOUT_2 Callout with diagonal straight line

LINE_CALLOUT_2_ACCENT_BAR Callout with diagonal callout line and accent bar

LINE_CALLOUT_2_BORDER_AND_ACCENT_BAR Callout with border, diagonal straight line, and accent bar

LINE_CALLOUT_2_NO_BORDER Callout with no border and diagonal callout line

LINE_CALLOUT_3 Callout with angled line

LINE_CALLOUT_3_ACCENT_BAR Callout with angled callout line and accent bar

LINE_CALLOUT_3_BORDER_AND_ACCENT_BAR Callout with border, angled callout line, and accent bar

LINE_CALLOUT_3_NO_BORDER Callout with no border and angled callout line

LINE_CALLOUT_4 Callout with callout line segments forming a U-shape

LINE_CALLOUT_4_ACCENT_BAR Callout with accent bar and callout line segments forming a U-shape

LINE_CALLOUT_4_BORDER_AND_ACCENT_BAR Callout with border, accent bar, and callout line segments forming a U-shape

LINE_CALLOUT_4_NO_BORDER Callout with no border and callout line segments forming a U-shape.

LINE_INVERSE Straight Connector

MATH_DIVIDE Division

MATH_EQUAL Equal

MATH_MINUS Minus

MATH_MULTIPLY Multiply

MATH_NOT_EQUAL Not Equal

MATH_PLUS Plus

MOON Moon

NO_SYMBOL “No” symbol

NON_ISOSCELES_TRAPEZOID Non-isosceles Trapezoid

NOTCHED_RIGHT_ARROW Notched block arrow that points right

OCTAGON Octagon

OVAL Oval

OVAL_CALLOUT Oval-shaped callout

PARALLELOGRAM Parallelogram

PENTAGON Pentagon

PIE Pie

PIE_WEDGE Pie

PLAQUE Plaque

PLAQUE_TABS Plaque Tabs

QUAD_ARROW Block arrows that point up, down, left, and right

QUAD_ARROW_CALLOUT Callout with arrows that point up, down, left, and right

RECTANGLE Rectangle

RECTANGULAR_CALLOUT Rectangular callout

REGULAR_PENTAGON Pentagon

RIGHT_ARROW Block arrow that points right

RIGHT_ARROW_CALLOUT Callout with arrow that points right

RIGHT_BRACE Right brace

RIGHT_BRACKET Right bracket

RIGHT_TRIANGLE Right triangle

ROUND_1_RECTANGLE Round Single Corner Rectangle

ROUND_2_DIAG_RECTANGLE Round Diagonal Corner Rectangle

ROUND_2_SAME_RECTANGLE Round Same Side Corner Rectangle

ROUNDED_RECTANGLE Rounded rectangle

ROUNDED_RECTANGULAR_CALLOUT Rounded rectangle-shaped callout

SMILEY_FACE Smiley face

SNIP_1_RECTANGLE Snip Single Corner Rectangle

SNIP_2_DIAG_RECTANGLE Snip Diagonal Corner Rectangle

SNIP_2_SAME_RECTANGLE Snip Same Side Corner Rectangle

SNIP_ROUND_RECTANGLE Snip and Round Single Corner Rectangle

SQUARE_TABS Square Tabs

STAR_10_POINT 10-Point Star

STAR_12_POINT 12-Point Star

STAR_16_POINT 16-point star

STAR_24_POINT 24-point star

STAR_32_POINT 32-point star

STAR_4_POINT 4-point star

STAR_5_POINT 5-point star

STAR_6_POINT 6-Point Star

STAR_7_POINT 7-Point Star

STAR_8_POINT 8-point star

STRIPED_RIGHT_ARROW Block arrow that points right with stripes at the tail

SUN Sun

SWOOSH_ARROW Swoosh Arrow

TEAR Teardrop

TRAPEZOID Trapezoid

U_TURN_ARROW Block arrow forming a U shape

UP_ARROW Block arrow that points up

UP_ARROW_CALLOUT Callout with arrow that points up

UP_DOWN_ARROW Block arrow that points up and down

UP_DOWN_ARROW_CALLOUT Callout with arrows that point up and down

UP_RIBBON Ribbon banner with center area above ribbon ends

VERTICAL_SCROLL Vertical scroll

WAVE Wave

MSO_AUTO_SIZE

Determines the type of automatic sizing allowed.

The following names can be used to specify the automatic sizing behavior used to fit a shape's text within the shape bounding box, for example:

```
from pptx.enum.text import MSO_AUTO_SIZE

shape.text_frame.auto_size = MSO_AUTO_SIZE.TEXT_TO_FIT_SHAPE
```

The word-wrap setting of the text frame interacts with the auto-size setting to determine the specific auto-sizing behavior.

Note that `TextFrame.auto_size` can also be set to `None`, which removes the auto size setting altogether. This causes the setting to be inherited, either from the layout placeholder, in the case of a placeholder shape, or from the theme.

NONE No automatic sizing of the shape or text will be done. Text can freely extend beyond the horizontal and vertical edges of the shape bounding box.

SHAPE_TO_FIT_TEXT The shape height and possibly width are adjusted to fit the text. Note this setting interacts with the TextFrame.word_wrap property setting. If word wrap is turned on, only the height of the shape will be adjusted; soft line breaks will be used to fit the text horizontally.

TEXT_TO_FIT_SHAPE The font size is reduced as necessary to fit the text within the shape.

MIXED Return value only; indicates a combination of automatic sizing schemes are used.

MSO_COLOR_TYPE

Specifies the color specification scheme

Example:

```
from pptx.enum.dml import MSO_COLOR_TYPE

assert shape.fill.fore_color.type == MSO_COLOR_TYPE.SCHEME
```

RGB Color is specified by an *RGBColor* value

SCHEME Color is one of the preset theme colors

HSL Color is specified using Hue, Saturation, and Luminosity values

PRESET Color is specified using a named built-in color

SCRGB Color is an scRGB color, a wide color gamut RGB color space

SYSTEM Color is one specified by the operating system, such as the window background color.

MSO_CONNECTOR_TYPE

Specifies a type of connector.

Alias: `MSO_CONNECTOR`

Example:

```
from pptx.enum.shapes import MSO_CONNECTOR
from pptx.util import Cm

shapes = prs.slides[0].shapes
connector = shapes.add_connector(
    MSO_CONNECTOR.STRAIGHT, Cm(2), Cm(2), Cm(10), Cm(10)
)
assert connector.left.cm == 2
```

CURVE Curved connector.

ELBOW Elbow connector.

STRAIGHT Straight line connector.

MIXED Return value only; indicates a combination of other states.

MSO_FILL_TYPE

Specifies the type of bitmap used for the fill of a shape.

Alias: `MSO_FILL`

Example:

```
from pptx.enum.dml import MSO_FILL

assert shape.fill.type == MSO_FILL.SOLID
```

BACKGROUND The shape is transparent, such that whatever is behind the shape shows through. Often this is the slide background, but if a visible shape is behind, that will show through.

GRADIENT Shape is filled with a gradient

GROUP Shape is part of a group and should inherit the fill properties of the group.

PATTERNED Shape is filled with a pattern

PICTURE Shape is filled with a bitmapped image

SOLID Shape is filled with a solid color

TEXTURED Shape is filled with a texture

MSO_LANGUAGE_ID

Specifies the language identifier.

Example:

```
from pptx.enum.lang import MSO_LANGUAGE_ID

font.language_id = MSO_LANGUAGE_ID.POLISH
```

MIXED More than one language in specified range.

NONE No language specified.

AFRIKAANS The Afrikaans language.

ALBANIAN The Albanian language.

AMHARIC The Amharic language.

ARABIC The Arabic language.

ARABIC_ALGERIA The Arabic Algeria language.

ARABIC_BAHRAIN The Arabic Bahrain language.

ARABIC_EGYPT The Arabic Egypt language.

ARABIC_IRAQ The Arabic Iraq language.

ARABIC_JORDAN The Arabic Jordan language.

ARABIC_KUWAIT The Arabic Kuwait language.

ARABIC_LEBANON The Arabic Lebanon language.

ARABIC_LIBYA The Arabic Libya language.

ARABIC_MOROCCO The Arabic Morocco language.

ARABIC_OMAN The Arabic Oman language.

ARABIC_QATAR The Arabic Qatar language.

ARABIC_SYRIA The Arabic Syria language.

ARABIC_TUNISIA The Arabic Tunisia language.

ARABIC_UAE The Arabic UAE language.

ARABIC_YEMEN The Arabic Yemen language.

ARMENIAN The Armenian language.

ASSAMESE The Assamese language.

AZERI_CYRILLIC The Azeri Cyrillic language.

AZERI_LATIN The Azeri Latin language.

BASQUE The Basque language.

BELGIAN_DUTCH The Belgian Dutch language.

BELGIAN_FRENCH The Belgian French language.

BENGALI The Bengali language.

BOSNIAN The Bosnian language.

BOSNIAN_BOSNIA_HERZEGOVINA_CYRILLIC The Bosnian Bosnia Herzegovina Cyrillic language.

BOSNIAN_BOSNIA_HERZEGOVINA_LATIN The Bosnian Bosnia Herzegovina Latin language.

BRAZILIAN_PORTUGUESE The Brazilian Portuguese language.

BULGARIAN The Bulgarian language.

BURMESE The Burmese language.

BYELORUSSIAN The Byelorussian language.

CATALAN The Catalan language.

CHEROKEE The Cherokee language.

CHINESE_HONG_KONG_SAR The Chinese Hong Kong SAR language.

CHINESE_MACAO_SAR The Chinese Macao SAR language.

CHINESE_SINGAPORE The Chinese Singapore language.

CROATIAN The Croatian language.

CZECH The Czech language.

DANISH The Danish language.

DIVEHI The Divehi language.

DUTCH The Dutch language.

EDO The Edo language.

ENGLISH_AUS The English AUS language.

ENGLISH_BELIZE The English Belize language.

ENGLISH_CANADIAN The English Canadian language.

ENGLISH_CARIBBEAN The English Caribbean language.

ENGLISH_INDONESIA The English Indonesia language.

ENGLISH_IRELAND The English Ireland language.

ENGLISH_JAMAICA The English Jamaica language.

ENGLISH_NEW_ZEALAND The English NewZealand language.

ENGLISH_PHILIPPINES The English Philippines language.

ENGLISH_SOUTH_AFRICA The English South Africa language.

ENGLISH_TRINIDAD_TOBAGO The English Trinidad Tobago language.

ENGLISH_UK The English UK language.

ENGLISH_US The English US language.

ENGLISH_ZIMBABWE The English Zimbabwe language.

ESTONIAN The Estonian language.

FAEROESE The Faeroese language.

FARSI The Farsi language.

FILIPINO The Filipino language.

FINNISH The Finnish language.

FRANCH_CONGO_DRC The French Congo DRC language.

FRENCH The French language.

FRENCH_CAMEROON The French Cameroon language.

FRENCH_CANADIAN The French Canadian language.

FRENCH_COTED_IVOIRE The French Coted Ivoire language.

FRENCH_HAITI The French Haiti language.

FRENCH_LUXEMBOURG The French Luxembourg language.

FRENCH_MALI The French Mali language.

FRENCH_MONACO The French Monaco language.

FRENCH_MOROCCO The French Morocco language.

FRENCH_REUNION The French Reunion language.

FRENCH_SENEGAL The French Senegal language.

FRENCH_WEST_INDIES The French West Indies language.

FRISIAN_NETHERLANDS The Frisian Netherlands language.

FULFULDE The Fulfulde language.

GAELIC_IRELAND The Gaelic Ireland language.

GAELIC_SCOTLAND The Gaelic Scotland language.

GALICIAN The Galician language.

GEORGIAN The Georgian language.

GERMAN The German language.

GERMAN_AUSTRIA The German Austria language.

GERMAN_LIECHTENSTEIN The German Liechtenstein language.

GERMAN_LUXEMBOURG The German Luxembourg language.

GREEK The Greek language.

GUARANI The Guarani language.

GUJARATI The Gujarati language.

HAUSA The Hausa language.

HAWAIIAN The Hawaiian language.

HEBREW The Hebrew language.

HINDI The Hindi language.

HUNGARIAN The Hungarian language.

IBIBIO The Ibibio language.

ICELANDIC The Icelandic language.

IGBO The Igbo language.

INDONESIAN The Indonesian language.

INUKTITUT The Inuktitut language.

ITALIAN The Italian language.

JAPANESE The Japanese language.

KANNADA The Kannada language.

KANURI The Kanuri language.

KASHMIRI The Kashmiri language.

KASHMIRI_DEVANAGARI The Kashmiri Devanagari language.

KAZAKH The Kazakh language.

KHMER The Khmer language.

KIRGHIZ The Kirghiz language.

KONKANI The Konkani language.

KOREAN The Korean language.

KYRGYZ The Kyrgyz language.

LAO The Lao language.

LATIN The Latin language.

LATVIAN The Latvian language.

LITHUANIAN The Lithuanian language.

MACEDONINAN_FYROM The Macedonian FYROM language.

MALAY_BRUNEI_DARUSSALAM The Malay Brunei Darussalam language.

MALAYALAM The Malayalam language.

MALAYSIAN The Malaysian language.

MALTESE The Maltese language.

MANIPURI The Manipuri language.

MAORI The Maori language.

MARATHI The Marathi language.

MEXICAN_SPANISH The Mexican Spanish language.

MONGOLIAN The Mongolian language.

NEPALI The Nepali language.

NO_PROOFING No proofing.

NORWEGIAN_BOKMOL The Norwegian Bokmol language.

NORWEGIAN_NYNORSK The Norwegian Nynorsk language.

ORIYA The Oriya language.

OROMO The Oromo language.

PASHTO The Pashto language.

POLISH The Polish language.

PORTUGUESE The Portuguese language.

PUNJABI The Punjabi language.

QUECHUA_BOLIVIA The Quechua Bolivia language.

QUECHUA_ECUADOR The Quechua Ecuador language.

QUECHUA_PERU The Quechua Peru language.

RHAETO_ROMANIC The Rhaeto Romanic language.

ROMANIAN The Romanian language.

ROMANIAN_MOLDOVA The Romanian Moldova language.

RUSSIAN The Russian language.

RUSSIAN_MOLDOVA The Russian Moldova language.

SAMI_LAPPISH The Sami Lappish language.

SANSKRIT The Sanskrit language.

SEPEDI The Sepedi language.

SERBIAN_BOSNIA_HERZEGOVINA_CYRILLIC The Serbian Bosnia Herzegovina Cyrillic language.

SERBIAN_BOSNIA_HERZEGOVINA_LATIN The Serbian Bosnia Herzegovina Latin language.

SERBIAN_CYRILLIC The Serbian Cyrillic language.

SERBIAN_LATIN The Serbian Latin language.

SESOTHO The Sesotho language.

SIMPLIFIED_CHINESE The Simplified Chinese language.

SINDHI The Sindhi language.

SINDHI_PAKISTAN The Sindhi Pakistan language.

SINHALESE The Sinhalese language.

SLOVAK The Slovak language.

SLOVENIAN The Slovenian language.

SOMALI The Somali language.

SORBIAN The Sorbian language.

SPANISH The Spanish language.

SPANISH_ARGENTINA The Spanish Argentina language.

SPANISH_BOLIVIA The Spanish Bolivia language.

SPANISH_CHILE The Spanish Chile language.

SPANISH_COLOMBIA The Spanish Colombia language.

SPANISH_COSTA_RICA The Spanish Costa Rica language.

SPANISH_DOMINICAN_REPUBLIC The Spanish Dominican Republic language.

SPANISH_ECUADOR The Spanish Ecuador language.

SPANISH_EL_SALVADOR The Spanish El Salvador language.

SPANISH_GUATEMALA The Spanish Guatemala language.

SPANISH_HONDURAS The Spanish Honduras language.

SPANISH_MODERN_SORT The Spanish Modern Sort language.

SPANISH_NICARAGUA The Spanish Nicaragua language.

SPANISH_PANAMA The Spanish Panama language.

SPANISH_PARAGUAY The Spanish Paraguay language.

SPANISH_PERU The Spanish Peru language.

SPANISH_PUERTO_RICO The Spanish Puerto Rico language.

SPANISH_URUGUAY The Spanish Uruguay language.

SPANISH_VENEZUELA The Spanish Venezuela language.

SUTU The Sutu language.

SWAHILI The Swahili language.

SWEDISH The Swedish language.

SWEDISH_FINLAND The Swedish Finland language.

SWISS_FRENCH The Swiss French language.

SWISS_GERMAN The Swiss German language.

SWISS_ITALIAN The Swiss Italian language.

SYRIAC The Syriac language.

TAJIK The Tajik language.

TAMAZIGHT The Tamazight language.

TAMAZIGHT_LATIN The Tamazight Latin language.

TAMIL The Tamil language.

TATAR The Tatar language.

TELUGU The Telugu language.

THAI The Thai language.

TIBETAN The Tibetan language.

TIGRIGNA_ERITREA The Tigrigna Eritrea language.

TIGRIGNA_ETHIOPIC The Tigrigna Ethiopic language.

TRADITIONAL_CHINESE The Traditional Chinese language.

TSONGA The Tsonga language.

TSWANA The Tswana language.

TURKISH The Turkish language.

TURKMEN The Turkmen language.

UKRAINIAN The Ukrainian language.

URDU The Urdu language.

UZBEK_CYRILLIC The Uzbek Cyrillic language.

UZBEK_LATIN The Uzbek Latin language.

VENDA The Venda language.

VIETNAMESE The Vietnamese language.

WELSH The Welsh language.

XHOSA The Xhosa language.

YI The Yi language.

YIDDISH The Yiddish language.

YORUBA The Yoruba language.

ZULU The Zulu language.

MSO_LINE_DASH_STYLE

Specifies the dash style for a line.

Alias: `MSO_LINE`

Example:

```
from pptx.enum.dml import MSO_LINE

shape.line.dash_style == MSO_LINE.DASH_DOT_DOT
```

DASH Line consists of dashes only.

DASH_DOT Line is a dash-dot pattern.

DASH_DOT_DOT Line is a dash-dot-dot pattern.

LONG_DASH Line consists of long dashes.

LONG_DASH_DOT Line is a long dash-dot pattern.

ROUND_DOT Line is made up of round dots.

SOLID Line is solid.

SQUARE_DOT Line is made up of square dots.

DASH_STYLE_MIXED Not supported.

MSO_PATTERN_TYPE

Specifies the fill pattern used in a shape.

Alias: `MSO_PATTERN`

Example:

```
from pptx.enum.dml import MSO_PATTERN

fill = shape.fill
fill.patterned()
fill.pattern == MSO_PATTERN.WAVE
```

CROSS Cross

DARK_DOWNWARD_DIAGONAL Dark Downward Diagonal

DARK_HORIZONTAL Dark Horizontal

DARK_UPWARD_DIAGONAL Dark Upward Diagonal

DARK_VERTICAL Dark Vertical

DASHED_DOWNWARD_DIAGONAL Dashed Downward Diagonal

DASHED_HORIZONTAL Dashed Horizontal

DASHED_UPWARD_DIAGONAL Dashed Upward Diagonal

DASHED_VERTICAL Dashed Vertical

DIAGONAL_BRICK Diagonal Brick

DIAGONAL_CROSS Diagonal Cross

DIVOT Pattern Divot

DOTTED_DIAMOND Dotted Diamond

DOTTED_GRID Dotted Grid

DOWNWARD_DIAGONAL Downward Diagonal

HORIZONTAL Horizontal

HORIZONTAL_BRICK Horizontal Brick

LARGE_CHECKER_BOARD Large Checker Board

LARGE_CONFETTI Large Confetti

LARGE_GRID Large Grid

LIGHT_DOWNWARD_DIAGONAL Light Downward Diagonal

LIGHT_HORIZONTAL Light Horizontal

LIGHT_UPWARD_DIAGONAL Light Upward Diagonal

LIGHT_VERTICAL Light Vertical

NARROW_HORIZONTAL Narrow Horizontal

NARROW_VERTICAL Narrow Vertical

OUTLINED_DIAMOND Outlined Diamond

PERCENT_10 10% of the foreground color.

PERCENT_20 20% of the foreground color.

PERCENT_25 25% of the foreground color.

PERCENT_30 30% of the foreground color.

PERCENT_40 40% of the foreground color.

PERCENT_5 5% of the foreground color.

PERCENT_50 50% of the foreground color.

PERCENT_60 60% of the foreground color.

PERCENT_70 70% of the foreground color.

PERCENT_75 75% of the foreground color.

PERCENT_80 80% of the foreground color.

PERCENT_90 90% of the foreground color.

PLAID Plaid

SHINGLE Shingle

SMALL_CHECKER_BOARD Small Checker Board

SMALL_CONFETTI Small Confetti

SMALL_GRID Small Grid

SOLID_DIAMOND Solid Diamond

SPHERE Sphere

TRELLIS Trellis

UPWARD_DIAGONAL Upward Diagonal

VERTICAL Vertical

WAVE Wave

WEAVE Weave

WIDE_DOWNWARD_DIAGONAL Wide Downward Diagonal

WIDE_UPWARD_DIAGONAL Wide Upward Diagonal

ZIG_ZAG Zig Zag

MIXED Mixed pattern.

MSO_SHAPE_TYPE

Specifies the type of a shape

Alias: `MSO`

Example:

```
from pptx.enum.shapes import MSO_SHAPE_TYPE

assert shape.type == MSO_SHAPE_TYPE.PICTURE
```

AUTO_SHAPE AutoShape

CALLOUT Callout shape

CANVAS Drawing canvas

CHART Chart, e.g. pie chart, bar chart

COMMENT Comment

DIAGRAM Diagram

EMBEDDED_OLE_OBJECT Embedded OLE object

FORM_CONTROL Form control

FREEFORM Freeform

GROUP Group shape

IGX_GRAPHIC SmartArt graphic

INK Ink

INK_COMMENT Ink Comment

LINE Line

LINKED_OLE_OBJECT Linked OLE object

LINKED_PICTURE Linked picture

MEDIA Media

OLE_CONTROL_OBJECT OLE control object

PICTURE Picture

PLACEHOLDER Placeholder

SCRIPT_ANCHOR Script anchor

TABLE Table

TEXT_BOX Text box

TEXT_EFFECT Text effect

WEB_VIDEO Web video

MIXED Mixed shape types

MSO_TEXT_UNDERLINE_TYPE

Indicates the type of underline for text. Used with *Font.underline* to specify the style of text underlining.

Alias: MSO_UNDERLINE

Example:

```
from pptx.enum.text import MSO_UNDERLINE

run.font.underline = MSO_UNDERLINE.DOUBLE_LINE
```

NONE Specifies no underline.

DASH_HEAVY_LINE Specifies a dash underline.

DASH_LINE Specifies a dash line underline.

DASH_LONG_HEAVY_LINE Specifies a long heavy line underline.

DASH_LONG_LINE Specifies a dashed long line underline.

DOT_DASH_HEAVY_LINE Specifies a dot dash heavy line underline.

DOT_DASH_LINE Specifies a dot dash line underline.

DOT_DOT_DASH_HEAVY_LINE Specifies a dot dot dash heavy line underline.

DOT_DOT_DASH_LINE Specifies a dot dot dash line underline.

DOTTED_HEAVY_LINE Specifies a dotted heavy line underline.

DOTTED_LINE Specifies a dotted line underline.

DOUBLE_LINE Specifies a double line underline.

HEAVY_LINE Specifies a heavy line underline.

SINGLE_LINE Specifies a single line underline.

WAVY_DOUBLE_LINE Specifies a wavy double line underline.

WAVY_HEAVY_LINE Specifies a wavy heavy line underline.

WAVY_LINE Specifies a wavy line underline.

WORDS Specifies underlining words.

MIXED Specifies a mixed of underline types.

MSO_THEME_COLOR_INDEX

Indicates the Office theme color, one of those shown in the color gallery on the formatting ribbon.

Alias: `MSO_THEME_COLOR`

Example:

```
from pptx.enum.dml import MSO_THEME_COLOR

shape.fill.solid()
shape.fill.fore_color.theme_color == MSO_THEME_COLOR.ACCENT_1
```

NOT_THEME_COLOR Indicates the color is not a theme color.

ACCENT_1 Specifies the Accent 1 theme color.

ACCENT_2 Specifies the Accent 2 theme color.

ACCENT_3 Specifies the Accent 3 theme color.

ACCENT_4 Specifies the Accent 4 theme color.

ACCENT_5 Specifies the Accent 5 theme color.

ACCENT_6 Specifies the Accent 6 theme color.

BACKGROUND_1 Specifies the Background 1 theme color.

BACKGROUND_2 Specifies the Background 2 theme color.

DARK_1 Specifies the Dark 1 theme color.

DARK_2 Specifies the Dark 2 theme color.

FOLLOWED_HYPERLINK Specifies the theme color for a clicked hyperlink.

HYPERLINK Specifies the theme color for a hyperlink.

LIGHT_1 Specifies the Light 1 theme color.

LIGHT_2 Specifies the Light 2 theme color.

TEXT_1 Specifies the Text 1 theme color.

TEXT_2 Specifies the Text 2 theme color.

MIXED Indicates multiple theme colors are used, such as in a group shape.

MSO_VERTICAL_ANCHOR

Specifies the vertical alignment of text in a text frame. Used with the `.vertical_anchor` property of the *TextFrame* object. Note that the `vertical_anchor` property can also have the value None, indicating there is no directly specified vertical anchor setting and its effective value is inherited from its placeholder if it has one or from the theme. None may also be assigned to remove an explicitly specified vertical anchor setting.

TOP Aligns text to top of text frame and inherits its value from its layout placeholder or theme.

MIDDLE Centers text vertically

BOTTOM Aligns text to bottom of text frame

MIXED Return value only; indicates a combination of the other states.

PP_ACTION_TYPE

Specifies the type of a mouse action (click or hover action).

Alias: `PP_ACTION`

Example:

```
from pptx.enum.action import PP_ACTION

assert shape.click_action.action == PP_ACTION.HYPERLINK
```

END_SHOW Slide show ends.

FIRST_SLIDE Returns to the first slide.

HYPERLINK Hyperlink.

LAST_SLIDE Moves to the last slide.

LAST_SLIDE_VIEWED Moves to the last slide viewed.

NAMED_SLIDE Moves to slide specified by slide number.

NAMED_SLIDE_SHOW Runs the slideshow.

NEXT_SLIDE Moves to the next slide.

NONE No action is performed.

OPEN_FILE Opens the specified file.

OLE_VERB OLE Verb.

PLAY Begins the slideshow.

PREVIOUS_SLIDE Moves to the previous slide.

RUN_MACRO Runs a macro.

RUN_PROGRAM Runs a program.

PP_MEDIA_TYPE

Indicates the OLE media type.

Example:

```
from pptx.enum.shapes import PP_MEDIA_TYPE

movie = slide.shapes[0]
assert movie.media_type == PP_MEDIA_TYPE.MOVIE
```

MOVIE Video media such as MP4.

OTHER Other media types

SOUND Audio media such as MP3.

MIXED Return value only; indicates multiple media types.

PP_PARAGRAPH_ALIGNMENT

Specifies the horizontal alignment for one or more paragraphs.

Alias: `PP_ALIGN`

Example:

```
from pptx.enum.text import PP_ALIGN

shape.paragraphs[0].alignment = PP_ALIGN.CENTER
```

CENTER Center align

DISTRIBUTE Evenly distributes e.g. Japanese characters from left to right within a line

JUSTIFY Justified, i.e. each line both begins and ends at the margin with spacing between words adjusted such that the line exactly fills the width of the paragraph.

JUSTIFY_LOW Justify using a small amount of space between words.

LEFT Left aligned

RIGHT Right aligned

THAI_DISTRIBUTE Thai distributed

MIXED Return value only; indicates multiple paragraph alignments are present in a set of paragraphs.

PP_PLACEHOLDER_TYPE

Specifies one of the 18 distinct types of placeholder.

Alias: `PP_PLACEHOLDER`

Example:

```
from pptx.enum.shapes import PP_PLACEHOLDER

placeholder = slide.placeholders[0]
assert placeholder.type == PP_PLACEHOLDER.TITLE
```

BITMAP Bitmap

BODY Body

CENTER_TITLE Center Title

CHART Chart

DATE Date

FOOTER Footer

HEADER Header

MEDIA_CLIP Media Clip

OBJECT Object

ORG_CHART Organization Chart

PICTURE Picture

SLIDE_NUMBER Slide Number

SUBTITLE Subtitle

TABLE Table

TITLE Title

VERTICAL_BODY Vertical Body

VERTICAL_OBJECT Vertical Object

VERTICAL_TITLE Vertical Title

MIXED Return value only; multiple placeholders of differing types.

XL_AXIS_CROSSES

Specifies the point on the specified axis where the other axis crosses.

Example:

```
from pptx.enum.chart import XL_AXIS_CROSSES

value_axis.crosses = XL_AXIS_CROSSES.MAXIMUM
```

AUTOMATIC The axis crossing point is set automatically, often at zero.

CUSTOM The *.crosses_at* property specifies the axis crossing point.

MAXIMUM The axis crosses at the maximum value.

MINIMUM The axis crosses at the minimum value.

XL_CATEGORY_TYPE

Specifies the type of the category axis.

Example:

```
from pptx.enum.chart import XL_CATEGORY_TYPE

date_axis = chart.category_axis
assert date_axis.category_type == XL_CATEGORY_TYPE.TIME_SCALE
```

AUTOMATIC_SCALE The application controls the axis type.

CATEGORY_SCALE Axis groups data by an arbitrary set of categories

TIME_SCALE Axis groups data on a time scale of days, months, or years.

XL_CHART_TYPE

Specifies the type of a chart.

Example:

```
from pptx.enum.chart import XL_CHART_TYPE

assert chart.chart_type == XL_CHART_TYPE.BAR_STACKED
```

THREE_D_AREA 3D Area.

THREE_D_AREA_STACKED 3D Stacked Area.

THREE_D_AREA_STACKED_100 100% Stacked Area.

THREE_D_BAR_CLUSTERED 3D Clustered Bar.

THREE_D_BAR_STACKED 3D Stacked Bar.

THREE_D_BAR_STACKED_100 3D 100% Stacked Bar.

THREE_D_COLUMN 3D Column.

THREE_D_COLUMN_CLUSTERED 3D Clustered Column.

THREE_D_COLUMN_STACKED 3D Stacked Column.

THREE_D_COLUMN_STACKED_100 3D 100% Stacked Column.

THREE_D_LINE 3D Line.

THREE_D_PIE 3D Pie.

THREE_D_PIE_EXPLODED Exploded 3D Pie.

AREA Area

AREA_STACKED Stacked Area.

AREA_STACKED_100 100% Stacked Area.

BAR_CLUSTERED Clustered Bar.

BAR_OF_PIE Bar of Pie.

BAR_STACKED Stacked Bar.

BAR_STACKED_100 100% Stacked Bar.

BUBBLE Bubble.

BUBBLE_THREE_D_EFFECT Bubble with 3D effects.

COLUMN_CLUSTERED Clustered Column.

COLUMN_STACKED Stacked Column.

COLUMN_STACKED_100 100% Stacked Column.

CONE_BAR_CLUSTERED Clustered Cone Bar.

CONE_BAR_STACKED Stacked Cone Bar.

CONE_BAR_STACKED_100 100% Stacked Cone Bar.

CONE_COL 3D Cone Column.

CONE_COL_CLUSTERED Clustered Cone Column.

CONE_COL_STACKED Stacked Cone Column.

CONE_COL_STACKED_100 100% Stacked Cone Column.

CYLINDER_BAR_CLUSTERED Clustered Cylinder Bar.

CYLINDER_BAR_STACKED Stacked Cylinder Bar.

CYLINDER_BAR_STACKED_100 100% Stacked Cylinder Bar.

CYLINDER_COL 3D Cylinder Column.

CYLINDER_COL_CLUSTERED Clustered Cone Column.

CYLINDER_COL_STACKED Stacked Cone Column.

CYLINDER_COL_STACKED_100 100% Stacked Cylinder Column.

DOUGHNUT Doughnut.

DOUGHNUT_EXPLODED Exploded Doughnut.

LINE Line.

LINE_MARKERS Line with Markers.

LINE_MARKERS_STACKED Stacked Line with Markers.

LINE_MARKERS_STACKED_100 100% Stacked Line with Markers.

LINE_STACKED Stacked Line.

LINE_STACKED_100 100% Stacked Line.

PIE Pie.

PIE_EXPLODED Exploded Pie.

PIE_OF_PIE Pie of Pie.

PYRAMID_BAR_CLUSTERED Clustered Pyramid Bar.

PYRAMID_BAR_STACKED Stacked Pyramid Bar.

PYRAMID_BAR_STACKED_100 100% Stacked Pyramid Bar.

PYRAMID_COL 3D Pyramid Column.

PYRAMID_COL_CLUSTERED Clustered Pyramid Column.

PYRAMID_COL_STACKED Stacked Pyramid Column.

PYRAMID_COL_STACKED_100 100% Stacked Pyramid Column.

RADAR Radar.

RADAR_FILLED Filled Radar.

RADAR_MARKERS Radar with Data Markers.

STOCK_HLC High-Low-Close.

STOCK_OHLC Open-High-Low-Close.

STOCK_VHLC Volume-High-Low-Close.

STOCK_VOHLC Volume-Open-High-Low-Close.

SURFACE 3D Surface.

SURFACE_TOP_VIEW Surface (Top View).

SURFACE_TOP_VIEW_WIREFRAME Surface (Top View wireframe).

SURFACE_WIREFRAME 3D Surface (wireframe).

XY_SCATTER Scatter.

XY_SCATTER_LINES Scatter with Lines.

XY_SCATTER_LINES_NO_MARKERS Scatter with Lines and No Data Markers.

XY_SCATTER_SMOOTH Scatter with Smoothed Lines.

XY_SCATTER_SMOOTH_NO_MARKERS Scatter with Smoothed Lines and No Data Markers.

XL_DATA_LABEL_POSITION

Specifies where the data label is positioned.

Example:

```
from pptx.enum.chart import XL_LABEL_POSITION

data_labels = chart.plots[0].data_labels
data_labels.position = XL_LABEL_POSITION.OUTSIDE_END
```

ABOVE The data label is positioned above the data point.

BELOW The data label is positioned below the data point.

BEST_FIT Word sets the position of the data label.

CENTER The data label is centered on the data point or inside a bar or a pie slice.

INSIDE_BASE The data label is positioned inside the data point at the bottom edge.

INSIDE_END The data label is positioned inside the data point at the top edge.

LEFT The data label is positioned to the left of the data point.

MIXED Data labels are in multiple positions.

OUTSIDE_END The data label is positioned outside the data point at the top edge.

RIGHT The data label is positioned to the right of the data point.

XL_LEGEND_POSITION

Specifies the position of the legend on a chart.

Example:

```
from pptx.enum.chart import XL_LEGEND_POSITION

chart.has_legend = True
chart.legend.position = XL_LEGEND_POSITION.BOTTOM
```

BOTTOM Below the chart.

CORNER In the upper-right corner of the chart border.

CUSTOM A custom position.

LEFT Left of the chart.

RIGHT Right of the chart.

TOP Above the chart.

XL_MARKER_STYLE

Specifies the marker style for a point or series in a line chart, scatter chart, or radar chart.

Example:

```
from pptx.enum.chart import XL_MARKER_STYLE

series.marker.style = XL_MARKER_STYLE.CIRCLE
```

AUTOMATIC Automatic markers

CIRCLE Circular markers

DASH Long bar markers

DIAMOND Diamond-shaped markers

DOT Short bar markers

NONE No markers

PICTURE Picture markers

PLUS Square markers with a plus sign

SQUARE Square markers

STAR Square markers with an asterisk

TRIANGLE Triangular markers

X Square markers with an X

XL_TICK_LABEL_POSITION

Specifies the position of tick-mark labels on a chart axis.

Example:

```
from pptx.enum.chart import XL_TICK_LABEL_POSITION

category_axis = chart.category_axis
category_axis.tick_label_position = XL_TICK_LABEL_POSITION.LOW
```

HIGH Top or right side of the chart.

LOW Bottom or left side of the chart.

NEXT_TO_AXIS Next to axis (where axis is not at either side of the chart).

NONE No tick labels.

XL_TICK_MARK

Specifies a type of axis tick for a chart.

Example:

```
from pptx.enum.chart import XL_TICK_MARK

chart.value_axis.minor_tick_mark = XL_TICK_MARK.INSIDE
```

CROSS Tick mark crosses the axis

INSIDE Tick mark appears inside the axis

NONE No tick mark

OUTSIDE Tick mark appears outside the axis

Excel Number Formats

The following integer values can be used in `ChartData.add_series()` calls to specify a numeric display format to use for the values of that series.

Example:

```
chart_data = ChartData()
# 10 signifies a percentange with two digits after the decimal point
chart_data.add_series('Series 1', (0.25, 0.45, 0.3), 10)
```

Information on string number format codes (such as '#,##0') can be found on this web page.

Value	Type	Format String
0	General	General
1	Decimal	0
2	Decimal	0.00
3	Decimal	#,##0
4	Decimal	#,##0.00
5	Currency	$#,##0;$-#,##0
6	Currency	$#,##0;[Red]$-#,##0
7	Currency	$#,##0.00;$-#,##0.00
8	Currency	$#,##0.00;[Red]$-#,##0.00
9	Percentage	0%
10	Percentage	0.00%
11	Scientific	0.00E+00
12	Fraction	# ?/?
13	Fraction	# /
14	Date	m/d/yy
15	Date	d-mmm-yy
16	Date	d-mmm
17	Date	mmm-yy
18	Time	h:mm AM/PM
19	Time	h:mm:ss AM/PM
20	Time	h:mm
21	Time	h:mm:ss
22	Time	m/d/yy h:mm
37	Currency	#,##0;-#,##0
38	Currency	#,##0;[Red]-#,##0
39	Currency	#,##0.00;-#,##0.00
40	Currency	#,##0.00;[Red]-#,##0.00
41	Accounting	_ * #,##0_ ;_ * "_ ;_ @_
42	Accounting	_ $* #,##0_ ;_ $* "_ ;_ @_
43	Accounting	_ * #,##0.00_ ;_ * "??_ ;_ @_
44	Accounting	_ $* #,##0.00_ ;_ $* "??_ ;_ @_
45	Time	mm:ss
46	Time	h :mm:ss
47	Time	mm:ss.0
48	Scientific	##0.0E+00
49	Text	@

CHAPTER 5

Contributor Guide

Running the test suite

python-pptx has a robust test suite, comprising over 600 tests at the time of writing, both at the acceptance test and unit test levels. `pytest` is used for unit tests, with help from the excellent `mock` library. `behave` is used for acceptance tests.

You can run the tests from the source working directory using the following commands:

```
$ py.test
============================ test session starts ============================
platform darwin -- Python 2.7.5 -- pytest-2.4.2
plugins: cov
collected 301 items

tests/test_oxml.py ....................................
tests/test_presentation.py ...................
tests/test_spec.py ........
tests/test_text.py ........................
tests/test_util.py ..............
tests/opc/test_packaging.py ..................................................
tests/opc/test_rels.py ..............
tests/parts/test_coreprops.py .
tests/parts/test_image.py ...........
tests/parts/test_part.py ........
tests/parts/test_slides.py ..................
tests/shapes/test_autoshape.py .................
tests/shapes/test_picture.py .
tests/shapes/test_placeholder.py .
tests/shapes/test_shape.py ...........
tests/shapes/test_shapetree.py ..............
tests/shapes/test_table.py ....................................

====================== 301 passed in 2.00 seconds ======================
```

```
$ behave
Feature: Add a text box to a slide
  In order to accommodate a requirement for free-form text on a slide
  As a presentation developer
  I need the ability to place a text box on a slide

  Scenario: Add a text box to a slide
    Given I have a reference to a blank slide
    When I add a text box to the slide's shape collection
    And I save the presentation
    Then the text box appears in the slide

# ... more output ...

13 features passed, 0 failed, 0 skipped
30 scenarios passed, 0 failed, 0 skipped
120 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m1.3s
```

Understanding `xmlchemy`

`xmlchemy` is an object-XML mapping layer somewhat reminiscent of SQLAlchemy, hence the name. Mapping XML elements to objects is not nearly as challenging as mapping to a relational database, so this layer is substantially more modest. Nevertheless, it provides a powerful and very useful abstraction layer around `lxml`, particularly well-suited to providing access to the broad schema of XML elements involved in the Open XML standard.

Additional topics to add ...

- understanding complex types in Open XML
- understanding attribute definitions Open XML
- understanding simple types in Open XML

Adding support for a new element type

- add a new custom element mapping to `pptx.oxml.__init__`
- add a new custom element class in the appropriate `pptx.oxml` subpackage module
- Add element definition members to the class
- Add attribute definition members to the class
- Add simple type definitions to `pptx.oxml.simpletype`

Example

```
from pptx.oxml.xmlchemy import BaseOxmlElement


class CT_Foobar(BaseOxmlElement):
    """
```

```
    Custom element class corresponding to ``CT_Foobar`` complex type
    definition in pml.xsd or other Open XML schema.
    """
    hlink = ZeroOrOne('a:hlink', successors=('a:rtl', 'a:extLst'))
    eg_fillProperties = ZeroOrOneChoice(
        (Choice('a:noFill'), Choice('a:solidFill'), Choice('a:gradFill'),
         Choice('a:blipFill'), Choice('a:pattFill')),
        successors=(
            'a:effectLst', 'a:effectDag', 'a:highlight', 'a:uLnTx',
            'a:uLn', 'a:uFillTx' 'a:extLst'
        )
    )

    sz = OptionalAttribute(
        'sz', ST_SimpleType, default=ST_SimpleType.OPTION
    )
    anchor = OptionalAttribute('i', XsdBoolean)
    rId = RequiredAttribute('r:id', XsdString)
```

Protocol

```
>>> assert isinstance(foobar, CT_Foobar)
>>> foobar.hlink
None
>>> hlink = foobar._add_hlink()
>>> hlink
<pptx.oxml.xyz.CT_Hyperlink object at 0x10ab4b2d0>
>>> assert foobar.hlink is hlink

>>> foobar.eg_fillProperties
None
>>> foobar.solidFill
None
>>> solidFill = foobar.get_or_change_to_solidFill()
>>> solidFill
<pptx.oxml.xyz.CT_SolidFill object at 0x10ab4b2d0>
>>> assert foobar.eg_fillProperties is solidFill
>>> assert foobar.solidFill is solidFill
>>> foobar.remove_eg_fillProperties()
>>> foobar.eg_fillProperties
None
>>> foobar.solidFill
None
```

`OneAndOnlyOne` element declaration

The `OneAndOnlyOne` callable generates the API for a required child element:

```
childElement = OneAndOnlyOne('ns:localTagName')
```

Unlike the other element declarations, the call does not include a `successors` argument. Since no API for inserting a new element is generated, a successors list is not required.

Generated API

childElement property (read-only) Holds a reference to the child element object. Raises `InvalidXmlError` on access if the required child element is not present.

Protocol

```
>>> foobar.childElement
<pptx.oxml.xyz.CT_ChildElement object at 0x10ab4b2d0>
```

`RequiredAttribute` attribute declaration

```
reqAttr = RequiredAttribute('reqAtr', ST_SimpleType)
```

Generated API

childElement property (read/write) Referencing the property returns the type-converted value of the attribute as determined by the `from_xml()` method of the simple type class appearing in the declaration (e.g. ST_SimpleType above). Assignments to the property are validated by the `validate()` method of the simple type class, potentially raising `TypeError` or `ValueError`. Values are assigned in their natural Python type and are encoded to the appropriate string value by the `to_xml()` method of the simple type class.

`ZeroOrOne` element declaration

```
childElement = ZeroOrOne(
    'ns:localTagName', successors=('ns:abc', 'ns:def')
)
```

Generated API

childElement property (read-only) Holds a reference to the child element object, or None if the element is not present.

get_or_add_childElement() method Returns the child element object, newly added if not present.

_add_childElement() empty element adder method Returns a newly added empty child element having the declared tag name. Adding is unconditional and assumes the element is not already present. This method is called by the `get_or_add_childElement()` method as needed and may be called by a hand-coded `add_childElement()` method as needed. May be overridden to produce customized behavior.

_new_childElement() empty element creator method Returns a new "loose" child element of the declared tag name. Called by `_add_childElement()` to obtain a new child element, it may be overridden to customize the element creation process.

_insert_childElement(childElement) element inserter method Returns the passed `childElement` after inserting it before any successor elements, as listed in the `successors` argument of the declaration. Called by `_add_childElement()` to insert the new element it creates using `_new_childElement()`.

_remove_childElement() element remover method Removes all instances of the child element. Does not raise an error if no matching child elements are present.

Development Practices

Release procedure

- merge outstanding feature branch(es) into develop
  - `$ git checkout develop`
  - `$ git merge --no-ff {feature-branch}`
- complete release updates
  - update version number in *pptx/__init__.py*
  - update setup.py (shouldn't usually be any changes)
  - update README.py, in particular, the release history
  - update doc/index.rst
  - confirm docs compile without errors
  - run all tests (behave, nosetests, tox)
  - create trial distribution (make clean sdist)
  - `git commit -m 'Release v0.2.2'`
  - merge develop into master
    - `$ git checkout master`
    - `$ git merge --no-ff develop`
  - create tag for new version
    - `$ git tag -a v0.2.5 -m 'Release version 0.2.5'`
- release uploaded to PyPI
  - upload: `make upload`
- synchronize local repo with github
  - `$ git push scanny develop`
  - `$ git push scanny master`
  - `$ git push --tags`
- docs regenerated
  - trigger regeneration of docs on RTD.org

Creating a hand-modified package

- remove file, e.g. /docProps/core.xml
- remove reference from [Content_Types].xml
- remove relationship(s) from _rels/.rels or wherever they are

Repackage:

```
rm -f ../no-core-props.pptx && zip -Dqr ../no-core-props.pptx .
```

Procedure – Adding a new feature

- issue added to github issue tracker
- git feature branch created
- working analysis documented
- acceptance test failing (not just raising exceptions)
- recursively, outside in:
  - unit test failing
  - next level method(s) written
  - unit test passing
- all tests passing
  - unit
  - acceptance
  - tox
  - visual confirmation of behavior in PowerPoint
- documentation updated as required
  - API additions
  - example code
- feature branch committed, rebased if required
- feature branch merged into develop
  - `git flow feature finish paragraph-level`
- changes pushed to github
- issue closed

Outside-in layers

- API wrapper method (if applicable)
- Internal API method
- objectify manipulation layer
- perhaps others

Creating slide images for documentation

- Desired slide created using a test script
- Zoom slide to 100% with: click on slide, View > Zoom > Zoom... > 100% > OK
- Screenshot file on desktop using Cmd-Shift-4, Space, click
- Load into PhotoShop and crop, keeping dark gray border
- Save as PNG, scaled to 280 x 210px

- Completed image saved in `doc/_static/img/`

Acceptance testing with `behave`

... using *behave* for now for acceptance testing ...

Installation

```
pip install behave
```

Tutorial

The behave tutorial is well worth working through.

And this more detailed set of examples and tutorials is great for getting the practicalities down.

`behave` Resources

- INVEST in Good Stories, and SMART Tasks
- The Secret Ninja Cucumber Scrolls
- Behavior Driven Outside In Development Explained, Part 1
- The behave website contains excellent documentation on installing and using behave.

Vision

A robust, full-featured, and well-documented general-purpose library for manipulating Open XML PowerPoint files.

- **robust** - High reliability driven by a comprehensive test suite.
- **full-featured** - Anything that the file format will allow can be accomplished via the API. (Note that visions often take some time to fulfill completely :).
- **well-documented** - I don't know about you, but I find it hard to remember what I was thinking yesterday if I don't write it down. That's not a problem for most of my thinking, but when it comes to how I set up an object hierarchy to interact, it can be a big time-waster. So I like it when things are nicely laid out in black-and-white. Other folks seem to like that too :).
- **general-purpose** - Applicability to all conceivable purposes is valued over being especially well-suited to any particular purpose. Particular purposes can always be accomplished by building a wrapper library of your own. Serving general purposes from a particularized library is not so easy.
- **create AND manipulate** - While this library will perhaps most commonly be used for *writing* .pptx files, it will also be suitable for *reading* .pptx files and inspecting and manipulating their contents. I could see that coming in handy for full-text indexing, removing speaker notes, changing out templates, adding dynamically generated slides to static boilerplate, that sort of thing.

Analysis

Documentation of studies undertaken in support of API and code design.

Feature Analysis

Roughly in reverse chronological order of creation:

Slides

Notes Slide

A slide may have an associated notes page. The notes it contains are displayed under the slide in the PowerPoint UI when in edit mode. The slide notes are also shown in presenter mode, are displayed in Notes Pages view, and are printed on those notes pages.

Internally, a notes page is referred to as a *notes slide*. This is sensible because it is actually a specialized instance of a slide. It contains shapes, many of which are placeholders, and allows inserting of new shapes such as auto shapes, tables, and charts (although it is probably not common). Much of the functionality for a notes slide is inherited from existing base classes.

A notes slide is created using the notes master as a template. A presentation has no notes master when created from a template in PowerPoint. One is created according to a PowerPoint-internal preset default the first time it is needed, which is generally when a notes slide is created. It's possible one can also be created by entering the Notes Master view and almost certainly is created by editing the master found there (haven't tried it though). A presentation can have at most one notes master.

On creation, certain placeholders (slide image, notes, slide number) are copied from the notes master onto the new notes slide (if they have not been removed from the master). These "cloned" placeholders inherit position, size, and formatting from their corresponding notes master placeholder. If the position, size, or formatting of a notes slide placeholder is changed, the changed property is no long inherited (unchanged properties, however, continue to be inherited).

The remaining three placeholders that can appear on the notes master (date, header text, footer text) can optionally be made to appear on an individual notes slide using the Header and Footer... dialog. This dialog also has a button allowing all notes slides to be updated with the selected options. There appears to be some heuristic logic that automatically propagates these settings to notes for new slides as they are created.

A notes slide is not automatically created for each slide. Rather it is created the first time it is needed, perhaps most commonly when notes text is typed into the notes area under the slide in the UI.

Candidate Protocol

A notes slide is created on first access if one doesn't exist. Consequently, Slide.has_notes_slide is provided to detect the presence of an existing notes slide, avoiding creation of unwanted notes slide objects:

```
>>> slide.has_notes_slide
False
```

Because a notes slide is created on first access, Slide.notes_slide unconditionally returns a *NotesSlide* object. Once created, the same instance is returned on each call:

```
>>> notes_slide = slide.notes_slide
>>> notes_slide
<pptx.notes.NotesSlide object at 0x10698c1e0>
>>> slide.has_notes_slide
True
```

Like any slide (slide, slide layout, slide master, etc.), a notes slide has shapes and placeholders, as well as other standard properties:

```
>>> notes_slide.shapes
<pptx.shapes.shapetree.NotesSlideShapes object at 0x10698c1e0>
>>> notes_slide.placeholders
<pptx.shapes.shapetree.NotesSlidePlaceholders object at 0x10698f622>
```

The distinctive characteristic of a notes slide is the notes it contains. These notes are contained in the text frame of the notes placeholder and are manipulated using the same properties and methods as any shape textframe:

```
>>> text_frame = notes_slide.notes_text_frame
>>> text_frame.text = 'foobar'
>>> text_frame.text
'foobar'
>>> text_frame.add_paragraph('barfoo')
>>> text_frame.text
'foobar\nbarfoo'
```

PowerPoint behavior

- Notes page has its own view (View > Notes Page)
  - Notes can be edited from there too
  - It has a hide background graphics option and the other background settings
  - Notes have richer edit capabilities (such as ruler for tabs and indents) in the Notes view.
- A notes slide, once created, doesn't disappear when its notes content is deleted.
- NotesSlide placeholders inherit properties from the NotesMaster placeholder *of the same type*.
- For some reason PowerPoint adds a second theme for the NotesMaster, not sure what that's all about. I'll just add the default theme2.xml, then folks can avoid that if they encounter problems just by pre-loading a notesMaster of their own in their template.

Example XML

Empty notes page element:

```
<p:notes xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
         xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main"
         xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships
↪">
  <p:cSld>
    <p:spTree>
      <p:nvGrpSpPr>
        <p:cNvPr id="1" name=""/>
        <p:cNvGrpSpPr/>
        <p:nvPr/>
      </p:nvGrpSpPr>
      <p:grpSpPr>
        <a:xfrm>
          <a:off x="0" y="0"/>
          <a:ext cx="0" cy="0"/>
          <a:chOff x="0" y="0"/>
```

```
          <a:chExt cx="0" cy="0"/>
        </a:xfrm>
      </p:grpSpPr>
    </p:spTree>
  </p:cSld>
  <p:clrMapOvr>
    <a:masterClrMapping/>
  </p:clrMapOvr>
</p:notes>
```

Default notes page populated with three base placeholders:

```
<p:notes
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <p:cSld>
    <p:spTree>
      <p:nvGrpSpPr>
        <p:cNvPr id="1" name=""/>
        <p:cNvGrpSpPr/>
        <p:nvPr/>
      </p:nvGrpSpPr>
      <p:grpSpPr>
        <a:xfrm>
          <a:off x="0" y="0"/>
          <a:ext cx="0" cy="0"/>
          <a:chOff x="0" y="0"/>
          <a:chExt cx="0" cy="0"/>
        </a:xfrm>
      </p:grpSpPr>
      <p:sp>
        <p:nvSpPr>
          <p:cNvPr id="2" name="Slide Image Placeholder 1"/>
          <p:cNvSpPr>
            <a:spLocks noGrp="1" noRot="1" noChangeAspect="1"/>
          </p:cNvSpPr>
          <p:nvPr>
            <p:ph type="sldImg"/>
          </p:nvPr>
        </p:nvSpPr>
        <p:spPr/>
      </p:sp>
      <p:sp>
        <p:nvSpPr>
          <p:cNvPr id="3" name="Notes Placeholder 2"/>
          <p:cNvSpPr>
            <a:spLocks noGrp="1"/>
          </p:cNvSpPr>
          <p:nvPr>
            <p:ph type="body" idx="1"/>
          </p:nvPr>
        </p:nvSpPr>
        <p:spPr/>
        <p:txBody>
          <a:bodyPr/>
          <a:lstStyle/>
```

```
          <a:p>
            <a:r>
              <a:rPr lang="en-US" smtClean="0"/>
              <a:t>Notes</a:t>
            </a:r>
            <a:endParaRPr lang="en-US" dirty="0"/>
          </a:p>
        </p:txBody>
      </p:sp>
      <p:sp>
        <p:nvSpPr>
          <p:cNvPr id="4" name="Slide Number Placeholder 3"/>
          <p:cNvSpPr>
            <a:spLocks noGrp="1"/>
          </p:cNvSpPr>
          <p:nvPr>
            <p:ph type="sldNum" sz="quarter" idx="10"/>
          </p:nvPr>
        </p:nvSpPr>
        <p:spPr/>
        <p:txBody>
          <a:bodyPr/>
          <a:lstStyle/>
          <a:p>
            <a:fld id="{64BF21E3-C0F6-2742-868D-4CDFA1962D8A}" type="slidenum">
              <a:rPr lang="en-US" smtClean="0"/>
              <a:t>1</a:t>
            </a:fld>
            <a:endParaRPr lang="en-US"/>
          </a:p>
        </p:txBody>
      </p:sp>
    </p:spTree>
    <p:extLst>
      <p:ext uri="{BB962C8B-B14F-4D97-AF65-F5344CB8AC3E}">
        <p14:creationId xmlns:p14="http://schemas.microsoft.com/office/powerpoint/
↪2010/main" val="347586568"/>
      </p:ext>
    </p:extLst>
  </p:cSld>
  <p:clrMapOvr>
    <a:masterClrMapping/>
  </p:clrMapOvr>
</p:notes>
```

Related Schema Definitions

The root element of a notesSlide part is a *p:notes* element:

```
<xsd:element name="notes" type="CT_NotesSlide"/>

<xsd:complexType name="CT_NotesSlide">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="cSld"      type="CT_CommonSlideData"/>
    <xsd:element name="clrMapOvr" type="a:CT_ColorMappingOverride" minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionListModify"    minOccurs="0"/>
```

```
  </xsd:sequence>
  <xsd:attribute name="showMasterSp"     type="xsd:boolean" default="true"/>
  <xsd:attribute name="showMasterPhAnim" type="xsd:boolean" default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_CommonSlideData">
  <xsd:sequence>
    <xsd:element name="bg"          type="CT_Background"       minOccurs="0"/>
    <xsd:element name="spTree"      type="CT_GroupShape"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element name="controls"    type="CT_ControlList"      minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="optional" default=""/>
</xsd:complexType>
```

Notes Master

A presentation may have a notes master part (zero or one). The notes master determines the background elements and default layout of a notes page.

In the PowerPoint UI, a notes item is called a "notes page"; internally however, it is referred to as a "notes slide", is based on the Slide object, and shares many behaviors with it such as containing shapes and inheriting from a master.

A notes slide is created on-first use, perhaps most frequently by typing text into the notes pane. When a new notes slide is created, it is based on the notes master. If no notes master is yet present, a new one is created from an internal PowerPoint-preset default and added to the package.

When a notes slide is created from the notes master, certain placeholders are partially "cloned" from the master. All other placeholders and shapes remain only on the master. The cloning process only creates placeholder shapes on the notes slide, but does not copy position or size information or other formatting. By default, position, size, and other formatting is inherited from the corresponding master placeholder. This achieves consistency through a presentation.

If position, size, etc. is changed on a NotesSlide, the new position overrides that on the master. This override is property-by-property.

A notes slide maintains a relationship with the master it was created from. This relationship is traversed to access the master placeholder properties for inheritance purposes.

A master can contain at most one each of six different placeholders: slide image, notes (textbox), header, footer, date, and slide number. The header, footer, and date placeholders are not copied to the notes slide by default. These three placeholders can be individually shown or hidden (copied to or deleted from the notes slide actually) using the Insert > Header and Footer... menu option/dialog. This dialog is also available from the Page Setup dialog.

All other items on the notes master, such as background color/texture/image and any additional text boxes or other shapes such as a logo, constitute "background graphics" and are shown by default. The may be hidden or re-shown on a notes slide-by-slide basis using the Format Background... (context or menu bar) menu item.

Candidate Protocol

A NotesMaster part is created on first access when not yet present:

```
>>> notes_master = presentation.notes_master
>>> notes_master
<pptx.slide.NotesMaster object at 0x10698c1e0>
```

It provides access to its placeholders and its shapes (which include placeholders):

```
>>> notes_master.shapes
<pptx.shapes.shapetree.MasterShapes object at 0x10698d140>
>>> notes_master.placeholders
<pptx.shapes.shapetree.MasterPlaceholders object at 0x1069902f0>
```

These placeholders and other shapes can be manipulated as usual.

Understanding Placeholders implementation

- SlidePlaceholders is a direct implementation, not subclassing Shapes

Call stacks:

- **_BaseSlide.placeholders**

  => BasePlaceholders() => _BaseShapes + _is_member_elm() override

- **NotesMaster.placeholders**

  => _BaseMaster.placeholders

  => MasterPlaceholders()

  => BasePlaceholders() + get() + _shape_factory() override => _BaseShapes + _is_member_elm() override

PowerPoint behavior

- A NotesSlide placeholder inherits properties from the NotesMaster placeholder *of the same type* (there can be at most one).
- For some reason PowerPoint adds a second theme for the NotesMaster, not sure what that's all about. I'll just add the default theme2.xml, then folks can avoid that if there are problems just by pre-loading a notesMaster of their own in their template.

Related Schema Definitions

The root element of a notesSlide part is a *p:notes* element:

```
<xsd:element name="notesMaster" type="CT_NotesMaster"/>

<xsd:complexType name="CT_NotesMaster">
  <xsd:sequence>
    <xsd:element name="cSld"       type="CT_CommonSlideData"/>
    <xsd:element name="clrMap"     type="a:CT_ColorMapping"/>
    <xsd:element name="hf"         type="CT_HeaderFooter"        minOccurs="0"/>
    <xsd:element name="notesStyle" type="a:CT_TextListStyle"     minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_CommonSlideData">
  <xsd:sequence>
```

```
    <xsd:element name="bg"          type="CT_Background"       minOccurs="0"/>
    <xsd:element name="spTree"      type="CT_GroupShape"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element name="controls"    type="CT_ControlList"      minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="optional" default=""/>
</xsd:complexType>

<xsd:element name="theme" type="CT_OfficeStyleSheet"/>

<xsd:complexType name="CT_OfficeStyleSheet">
  <xsd:sequence>
    <xsd:element name="themeElements"     type="CT_BaseStyles"/>
    <xsd:element name="objectDefaults"    type="CT_ObjectStyleDefaults"    minOccurs=
↪"0"/>
    <xsd:element name="extraClrSchemeLst" type="CT_ColorSchemeList"        minOccurs=
↪"0"/>
    <xsd:element name="custClrLst"        type="CT_CustomColorList"        minOccurs=
↪"0"/>
    <xsd:element name="extLst"            type="CT_OfficeArtExtensionList" minOccurs=
↪"0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" default=""/>
</xsd:complexType>
```

Base Slide

A slide is the fundamental visual content container in a presentation, that content taking the form of shape objects. The SlideMaster and SlideLayout objects are also slides and the three share common behaviors. They each also have distinctive behaviors. The focus of this page is the common slide characteristics.

Name

For the moment, the only shared attribute is name, stored in the *p:sld/p:cSld/@name* attribute.

While this attribute is populated by PowerPoint in slide layouts, it is commonly unpopulated in slides and slide masters. When a slide has no explicit name, PowerPoint uses the default name 'Slide {n}', where *n* is the sequence number of the slide in the current presentation. This name is not written to the *p:cSlide/@name* attribute. In the outline pane, PowerPoint uses the slide title if there is one.

Slide names may come into things when doing animations. Otherwise they don't show up in the commonly-used parts of the UI.

XML specimens

Example slide contents:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<p:sld
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
```

```
    >
  <p:cSld name="Overview">
    <p:spTree>
      ...
    </p:spTree>
  </p:cSld>
  <p:clrMapOvr>
    <a:masterClrMapping/>
  </p:clrMapOvr>
</p:sld>
```

Schema excerpt

```
<xsd:element name="sld" type="CT_Slide"/>

<xsd:complexType name="CT_Slide">  <!-- denormalized -->
  <xsd:sequence minOccurs="1" maxOccurs="1">
    <xsd:element name="cSld"       type="CT_CommonSlideData"/>
    <xsd:element name="clrMapOvr"  type="a:CT_ColorMappingOverride" minOccurs="0"/>
    <xsd:element name="transition" type="CT_SlideTransition"       minOccurs="0"/>
    <xsd:element name="timing"     type="CT_SlideTiming"           minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_ExtensionListModify"   minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="showMasterSp"     type="xsd:boolean" default="true"/>
  <xsd:attribute name="showMasterPhAnim" type="xsd:boolean" default="true"/>
  <xsd:attribute name="show"             type="xsd:boolean" default="true"/>
</xsd:complexType>

<xsd:element name="sldLayout" type="CT_SlideLayout"/>

<xsd:complexType name="CT_SlideLayout">
  <xsd:sequence>
    <xsd:element name="cSld"       type="CT_CommonSlideData"/>
    <xsd:element name="clrMapOvr"  type="a:CT_ColorMappingOverride" minOccurs="0"/>
    <xsd:element name="transition" type="CT_SlideTransition"       minOccurs="0"/>
    <xsd:element name="timing"     type="CT_SlideTiming"           minOccurs="0"/>
    <xsd:element name="hf"         type="CT_HeaderFooter"          minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_ExtensionListModify"   minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="showMasterSp"     type="xsd:boolean"        default="true"/>
  <xsd:attribute name="showMasterPhAnim" type="xsd:boolean"        default="true"/>
  <xsd:attribute name="matchingName"     type="xsd:string"         default=""/>
  <xsd:attribute name="type"             type="ST_SlideLayoutType" default="cust"/>
  <xsd:attribute name="preserve"         type="xsd:boolean"        default="false"/>
  <xsd:attribute name="userDrawn"        type="xsd:boolean"        default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_SlideMaster">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="cSld"          type="CT_CommonSlideData"/>
    <xsd:element name="clrMap"        type="a:CT_ColorMapping"/>
    <xsd:element name="sldLayoutIdLst" type="CT_SlideLayoutIdList"     minOccurs="0"/>
    <xsd:element name="transition"    type="CT_SlideTransition"       minOccurs="0"/>
    <xsd:element name="timing"        type="CT_SlideTiming"           minOccurs="0"/>
    <xsd:element name="hf"            type="CT_HeaderFooter"          minOccurs="0"/>
    <xsd:element name="txStyles"      type="CT_SlideMasterTextStyles" minOccurs="0"/>
```

```
    <xsd:element name="extLst"         type="CT_ExtensionListModify"   minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="preserve" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_CommonSlideData">
  <xsd:sequence>
    <xsd:element name="bg"          type="CT_Background"       minOccurs="0"/>
    <xsd:element name="spTree"      type="CT_GroupShape"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element name="controls"    type="CT_ControlList"      minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="optional" default=""/>
</xsd:complexType>

<xsd:complexType name="CT_ColorMappingOverride">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="masterClrMapping"   type="CT_EmptyElement"/>
      <xsd:element name="overrideClrMapping" type="CT_ColorMapping"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ColorMapping">
  <xsd:sequence>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="bg1"      type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="tx1"      type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="bg2"      type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="tx2"      type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent1"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent2"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent3"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent4"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent5"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent6"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="hlink"    type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="folHlink" type="ST_ColorSchemeIndex" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_EmptyElement"/>
```

Slide

A slide is the fundamental visual content container in a presentation, that content taking the form of shape objects.

The slides in a presentation are owned by the presentation object. In particular, the unique identifier of a slide, the slide id, is assigned and managed by the presentation part, and is not recorded in the slide XML.

A slide master and slide layout are both closely related to slide and the three share the majority of their properties and behaviors.

Slide ID

PowerPoint assigns a unique integer identifier to a slide when it is created. Note that this identifier is only present in the presentation part, and maps to a relationship ID; it is not recorded in the XML of the slide itself. Therefore the identifier is only unique within a presentation. The ID takes the form of an integer starting at 256 and incrementing by one for each new slide. Changing the ordering of the slides does not change the id. The id of a deleted slide is not reused, although I'm not sure whether it's clever enough not to reuse the id of the last added slide when it's been deleted as there doesn't seem to be any record in the XML of the max value assigned.

XML specimens

Example presentation XML showing sldIdLst:

```
<p:presentation
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <p:sldMasterIdLst>
    <p:sldMasterId id="2147483648" r:id="rId1"/>
  </p:sldMasterIdLst>

  <p:sldIdLst>
    <p:sldId r:id="rId7" id="256"/>
  </p:sldIdLst>

  <p:sldSz cx="12192000" cy="6858000"/>
  <p:notesSz cx="6858000" cy="9144000"/>
</p:presentation>
```

Example slide contents:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<p:sld
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <p:cSld>
    <p:spTree>
      <p:nvGrpSpPr>
        <p:cNvPr id="1" name=""/>
        <p:cNvGrpSpPr/>
        <p:nvPr/>
      </p:nvGrpSpPr>
      <p:grpSpPr/>
      <p:graphicFrame>
        <p:nvGraphicFramePr>
          <p:cNvPr id="2" name="Chart 1"/>
          <p:cNvGraphicFramePr>
            <a:graphicFrameLocks noGrp="1"/>
          </p:cNvGraphicFramePr>
          <p:nvPr/>
        </p:nvGraphicFramePr>
        <p:xfrm>
          <a:off x="3801533" y="956733"/>
```

```
          <a:ext cx="8128000" cy="5418667"/>
        </p:xfrm>
        <a:graphic>
          <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/chart">
            <c:chart xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"␣
↪r:id="rId2"/>
          </a:graphicData>
        </a:graphic>
      </p:graphicFrame>
    </p:spTree>
  </p:cSld>
  <p:clrMapOvr>
    <a:masterClrMapping/>
  </p:clrMapOvr>
</p:sld>
```

Schema excerpt

```
<xsd:element name="sld" type="CT_Slide"/>

<xsd:complexType name="CT_Slide">  <!-- denormalized -->
  <xsd:sequence minOccurs="1" maxOccurs="1">
    <xsd:element name="cSld"       type="CT_CommonSlideData"/>
    <xsd:element name="clrMapOvr"  type="a:CT_ColorMappingOverride" minOccurs="0"/>
    <xsd:element name="transition" type="CT_SlideTransition"       minOccurs="0"/>
    <xsd:element name="timing"     type="CT_SlideTiming"           minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_ExtensionListModify"   minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="showMasterSp"     type="xsd:boolean" default="true"/>
  <xsd:attribute name="showMasterPhAnim" type="xsd:boolean" default="true"/>
  <xsd:attribute name="show"             type="xsd:boolean" default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_Presentation">
  <!-- ... -->
  <xsd:element name="sldIdLst" type="CT_SlideIdList" minOccurs="0"/>
  <!-- ... -->
</xsd:complexType>

<xsd:complexType name="CT_SlideIdList">
  <xsd:sequence>
    <xsd:element name="sldId" type="CT_SlideIdListEntry" minOccurs="0" maxOccurs=
↪"unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SlideIdListEntry">
  <xsd:sequence>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="ST_SlideId" use="required"/>
  <xsd:attribute ref="r:id" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_CommonSlideData">
  <xsd:sequence>
```

```
    <xsd:element name="bg"          type="CT_Background"       minOccurs="0"/>
    <xsd:element name="spTree"      type="CT_GroupShape"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element name="controls"    type="CT_ControlList"      minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="optional" default=""/>
</xsd:complexType>

<xsd:simpleType name="ST_SlideId">
  <xsd:restriction base="xsd:unsignedInt">
    <xsd:minInclusive value="256"/>
    <xsd:maxExclusive value="2147483648"/>
  </xsd:restriction>
</xsd:simpleType>
```

Slide Master

A slide master acts as a "parent" for zero or more slide layouts, providing an inheritance base for placeholders and other slide and shape properties.

Schema excerpt

```
<xsd:element name="sldMaster" type="CT_SlideMaster"/>

<xsd:complexType name="CT_SlideMaster">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="cSld"          type="CT_CommonSlideData"/>
    <xsd:element name="clrMap"        type="a:CT_ColorMapping"/>
    <xsd:element name="sldLayoutIdLst" type="CT_SlideLayoutIdList"     minOccurs="0"/>
    <xsd:element name="transition"    type="CT_SlideTransition"       minOccurs="0"/>
    <xsd:element name="timing"        type="CT_SlideTiming"           minOccurs="0"/>
    <xsd:element name="hf"            type="CT_HeaderFooter"          minOccurs="0"/>
    <xsd:element name="txStyles"      type="CT_SlideMasterTextStyles" minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionListModify"   minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="preserve" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_ColorMapping">
  <xsd:sequence>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="bg1"      type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="tx1"      type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="bg2"      type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="tx2"      type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent1"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent2"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent3"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent4"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent5"  type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="accent6"  type="ST_ColorSchemeIndex" use="required"/>
```

```
  <xsd:attribute name="hlink"    type="ST_ColorSchemeIndex" use="required"/>
  <xsd:attribute name="folHlink" type="ST_ColorSchemeIndex" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_SlideLayoutIdList">
  <xsd:sequence>
    <xsd:element name="sldLayoutId" type="CT_SlideLayoutIdListEntry" minOccurs="0"␣
↪maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SlideLayoutIdListEntry">
  <xsd:sequence>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id"  type="ST_SlideLayoutId"/>
  <xsd:attribute ref="r:id" type="ST_RelationshipId" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_RelationshipId">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

Slide Layout

A slide layout acts as an property inheritance base for zero or more slides. This provides a certain amount of separation between formatting and content and contributes to visual consistency across the slides of a presentation.

Schema excerpt

```
<xsd:element name="sldLayout" type="CT_SlideLayout"/>

<xsd:complexType name="CT_SlideLayout">
  <xsd:sequence>
    <xsd:element name="cSld"       type="CT_CommonSlideData"/>
    <xsd:element name="clrMapOvr"  type="a:CT_ColorMappingOverride" minOccurs="0"/>
    <xsd:element name="transition" type="CT_SlideTransition"       minOccurs="0"/>
    <xsd:element name="timing"     type="CT_SlideTiming"           minOccurs="0"/>
    <xsd:element name="hf"         type="CT_HeaderFooter"          minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_ExtensionListModify"   minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="showMasterSp"     type="xsd:boolean"       default="true"/>
  <xsd:attribute name="showMasterPhAnim" type="xsd:boolean"       default="true"/>
  <xsd:attribute name="matchingName"     type="xsd:string"        default=""/>
  <xsd:attribute name="type"             type="ST_SlideLayoutType" default="cust"/>
  <xsd:attribute name="preserve"         type="xsd:boolean"       default="false"/>
  <xsd:attribute name="userDrawn"        type="xsd:boolean"       default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_CommonSlideData">
  <xsd:sequence>
    <xsd:element name="bg"          type="CT_Background"       minOccurs="0"/>
    <xsd:element name="spTree"      type="CT_GroupShape"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
```

```
    <xsd:element name="controls"    type="CT_ControlList"      minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="optional" default=""/>
</xsd:complexType>
```

Chart

Chart - Chart Title

A chart can have a title. The title is a rich text container, and can contain arbitrary text with arbitrary formatting (font, size, color, etc.). There is little but one thing to distinquish a chart title from an independent text box; its position is automatically adjusted by the chart to account for resizing and movement.

A title is visible whenever present. The only way to "hide" it is to delete it, along with its contents.

Although it will not yet be supported, the chart title can be specified in the XML as a cell reference in the Excel worksheet. In general, any constructive operations on the title will remove this.

Proposed Scope

- Chart.has_title
- Chart.chart_title
- ChartTitle.has_text_frame
- ChartTitle.text_frame
- ChartTitle.format

Candidate protocol

Chart title presence

The presence of a chart title is reported by `Chart.has_title`. Reading this property is non-destructive. Starting with a newly-created chart, which has no title:

```
>>> chart = shapes.add_chart(...).chart
>>> chart.has_title
False
```

Assigning `True` to `.has_title` causes an empty title element to be added along with its text frame elements (when not already present). This assignment is idempotent, such that assigning `True` when a chart title is present leaves the chart unchanged:

```
>>> chart.has_title = True
>>> chart.has_title
True
```

Assigning `False` to `.has_title` removes the title element from the XML along with its contents. This assignment is also idempotent:

```
>>> chart.has_title = False
>>> chart.has_title
False
```

Chart title access

Access to the chart title is provided by `Chart.chart_title`. This property always provides a *ChartTitle* object; a new one is created if not present. This behavior produces cleaner code for the common "get or add" case; `Chart.has_title` is provided to avoid this potentially destructive behavior when required:

```
>>> chart = shapes.add_chart(...).chart
>>> chart.has_title
False
>>> chart.chart_title
<pptx.chart.ChartTitle object at 0x65432fd>
>>> chart.has_title
True
```

ChartTitle.text_frame

The `ChartTitle` object can contain either a text frame or an Excel cell reference (`<c:strRef>`). However, the only operation on the *c:strRef* element the library will support (for now anyway) is to delete it when adding a text frame.

`ChartTitle.has_text_frame` is used to determine whether a text frame is present. Assigning `True` to `.has_text_frame` causes any Excel reference to be removed and an empty text frame to be inserted. If a text frame is already present, no changes are made:

```
>>> chart_title.has_text_frame
False
>>> chart_title.has_text_frame = True
>>> chart_title.has_text_frame
True
```

Assigning `False` to `.has_text_frame` removes the text frame element from the XML along with its contents. This assignment is also idempotent:

```
>>> chart.has_text_frame = False
>>> chart.has_text_frame
False
```

The text frame can be accessed using `ChartTitle.text_frame`. This property always provides a *TextFrame* object; one is added if not present and any `c:strRef` element is removed:

```
>>> chart.has_text_frame
False
>>> chart_title.text_frame
<pptx.text.text.TextFrame object at 0x65432fe>
>>> chart.has_text_frame
True
```

XML semantics

- `c:autoTitleDeleted` set True has no effect on the visibility of a default chart title (no actual text, 'placeholder' display: 'Chart Title'. It also seems to have no effect on an actual title, having a text frame and actual text.

XML specimens

Default when clicking *Chart Title* > *Title Above Chart* from ribbon (before changing the text of the title):

```
<c:chart>
  <c:title>
    <c:layout/>
    <c:overlay val="0"/>
  </c:title>
  <c:autoTitleDeleted val="0"/>
  <c:plotArea>
    ...
  </c:plotArea>
</c:chart>
```

Text 'Foobar' typed into chart title just after adding it from ribbon:

```
<c:title>
  <c:tx>
    <c:rich>
      <a:bodyPr/>
      <a:lstStyle/>
      <a:p>
        <a:pPr>
          <a:defRPr/>
        </a:pPr>
        <a:r>
          <a:rPr lang="en-US" dirty="0" smtClean="0"/>
          <a:t>Foobar</a:t>
        </a:r>
        <a:endParaRPr lang="en-US" dirty="0"/>
      </a:p>
    </c:rich>
  </c:tx>
  <c:layout/>
  <c:overlay val="0"/>
</c:title>
```

Related Schema Definitions

```
<xsd:complexType name="CT_Chart">
  <xsd:sequence>
    <xsd:element name="title"            type="CT_Title"         minOccurs="0"/>
    <xsd:element name="autoTitleDeleted" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="pivotFmts"        type="CT_PivotFmts"     minOccurs="0"/>
    <xsd:element name="view3D"           type="CT_View3D"        minOccurs="0"/>
    <xsd:element name="floor"            type="CT_Surface"       minOccurs="0"/>
    <xsd:element name="sideWall"         type="CT_Surface"       minOccurs="0"/>
```

```
    <xsd:element name="backWall"         type="CT_Surface"       minOccurs="0"/>
    <xsd:element name="plotArea"         type="CT_PlotArea"/>
    <xsd:element name="legend"           type="CT_Legend"        minOccurs="0"/>
    <xsd:element name="plotVisOnly"      type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="dispBlanksAs"     type="CT_DispBlanksAs"  minOccurs="0"/>
    <xsd:element name="showDLblsOverMax" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Title">
  <xsd:sequence>
    <xsd:element name="tx"      type="CT_Tx"                minOccurs="0"/>
    <xsd:element name="layout"  type="CT_Layout"            minOccurs="0"/>
    <xsd:element name="overlay" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="spPr"    type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"    type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Tx">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element name="strRef" type="CT_StrRef"/>
      <xsd:element name="rich"   type="a:CT_TextBody"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Layout">
  <xsd:sequence>
    <xsd:element name="manualLayout" type="CT_ManualLayout"  minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextBody">
  <xsd:sequence>
    <xsd:element name="bodyPr"   type="CT_TextBodyProperties"/>
    <xsd:element name="lstStyle" type="CT_TextListStyle"      minOccurs="0"/>
    <xsd:element name="p"        type="CT_TextParagraph"      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Axis Title

A chart axis can have a title. Theoretically, the title text can be drawn from a cell in the spreadsheet behind the chart; however, there is no mechanism for specifying this from the PowerPoint 2011 UI (Mac) and there's no mention of such a procedure I could find on search. So only the "manually applied" axis title text will be discussed here, other than how to remove any elements associated with "linked" text, however they may have gotten there.

The title is a rich text container, and can contain arbitrary text with arbitrary formatting (font, size, color, etc.). There is little but one thing to distinquish an axis title from an independent text box; its position is automatically adjusted by the chart to account for resizing and movement.

An axis title is visible whenever present. The only way to "hide" it is to delete it, along with its contents.

Although it will not be supported, it appears that axis title text can be specified in the XML as a cell reference in the Excel worksheet. This is a so-called "linked" title, and in general, any constructive operations on the axis title will remove any linked title present.

Proposed Scope

Completed

- Axis.has_title
- Axis.axis_title
- AxisTitle.has_text_frame
- AxisTitle.text_frame
- AxisTitle.format

Pending

- AxisTitle.orientation (this may require text frame rotation)
- XL_ORIENTATION enumeration

Candidate protocol

Typical usage

I expect the most typical use is simply to set the text of the axis title:

```
>>> axis = shapes.add_chart(...).chart.value_axis
>>> axis.axis_title.text_frame.text = 'Foobar'
```

Axis title access

`Axis.has_axis_title` is used to non-destructively test for the presence of an axis title and may also be used to add an axis title.

An axis on a newly created chart has no axis title:

```
>>> axis = shapes.add_chart(...).chart.value_axis
>>> axis.has_title
False
```

Assigning `True` to `.has_title` causes an empty axis title element to be added along with its text frame elements (when not already present):

```
>>> axis.has_title = True
>>> axis.has_title
True
```

`Axis.axis_title` is used to access the `AxisTitle` object for an axis. It will always return an `AxisTitle` object, but it may be destructive in the sense that it adds an axis title if there is none:

```
>>> axis = shapes.add_chart(...).chart.value_axis
>>> axis.has_title
False
>>> axis.axis_title
<pptx.chart.axis.AxisTitle object at 0x65432fd>
>>> axis.has_title
True
```

Assigning `False` to `.has_title` removes the title element from the XML along with its contents:

```
>>> axis.has_title = False
>>> axis.has_title
False
```

Assigning `None` to `Axis.axis_title` has the same effect (not sure we'll actually implement this as a priority):

```
>>> axis.has_title
True
>>> axis.axis_title = None
>>> axis.has_title
False
```

AxisTitle.text_frame

According to the schema and the MS API, the `AxisTitle` object can contain either a text frame or an Excel cell reference (`<c:strRef>`). However, the only operation on the *c:strRef* element the library will support (for now anyway) is to delete it when adding a text frame.

A newly added axis title will already have a text frame, but for the sake of completeness, `AxisTitle.has_text_frame` will allow the client to test for, add, and remove an axis title text frame. Assigning `True` to `.has_text_frame` causes any Excel reference (`<c:strRef>`) element to be removed and an empty text frame to be inserted. If a text frame is already present, no changes are made:

```
>>> axis_title.has_text_frame
False
>>> axis_title.has_text_frame = True
>>> axis_title.has_text_frame
True
```

The text frame can be accessed using `AxisTitle.text_frame`. This call always returns a text frame object, newly created if not already present:

```
>>> axis_title.has_text_frame
False
>>> axis_title.text_frame
<pptx.text.text.TextFrame object at 0x65432fe>
>>> axis_title.has_text_frame
True
```

AxisTitle.orientation

By default, the PowerPoint UI adds an axis title for a vertical axis at 90° counterclockwise rotation. The MS API provides for rotation to be specified as an integer number of degrees between -90 and 90. Positive angles are interpreted

as counterclockwise from the horizontal. Orientation can also be specified as one of the members of the *XlOrientation* enumeration. The enumeration includes values for horizontal, 90° (upward), -90° (downward), and (vertically) stacked:

```
>>> axis = shapes.add_chart(...).chart.value_axis
>>> axis_title.orientation
90
>>> axis_title.orientation = XL_ORIENTATION.HORIZONTAL
>>> axis_title.orientation
0
```

MS API

Axis object

Axis.AxisTitle Provides access to the AxisTitle object for this axis.

Axis.HasTitle Getting indicates presence of axis title. Setting ensures presence or absence of axis title.

AxisTitle object

AxisTitle.Format Provides access to fill and line formatting.

AxisTitle.FormulaLocal Returns or sets the cell reference for the axis title text.

AxisTitle.HorizontalAlignment Not terrifically useful AFAICT unless title extends to multiple lines.

AxisTitle.IncludeInLayout Might not be available via UI; no such option present on PowerPoint 2011 for Mac.

AxisTitle.Orientation An integer value from –90 to 90 degrees or one of the XlOrientation constants.

AxisTitle.Text Returns or sets the axis title text. Setting removes any existing directly-applied formatting, but not title-level formatting.

AxisTitle.VerticalAlignment Perhaps not terrifically useful since the textbox is automatically positioned and sized, so no difference is visible in the typical cases.

PowerPoint UI Behaviors

- To add an axis title from the PowerPoint UI:

  Chart Layout (ribbon) > *Axis Titles* > *Vertical Axis Title* > *Rotated Title*

- The default title "Axis Title" appears when no text has been entered.
- The default orientation of a vertical axis title inserted by the UI is rotated 90 degrees counterclockwise. This is initially (before text is present) implemented using the *c:txPr* element. That element is removed when explicit title text is added.

XlOrientation Enumeration

https://msdn.microsoft.com/en-us/library/office/ff746480.aspx

xlDownward (-4170) Text runs downward.

xlHorizontal (-4128) Text runs horizontally.

xlUpward (-4171) Text runs upward.

xlVertical (-4166) Text runs downward and is centered in the cell.

Specimen XML

Add axis title in UI (but don't set text):

```
<c:valAx>
  <!-- ... -->
  <c:majorGridlines/>

  <c:title>
    <c:layout/>
    <c:overlay val="0"/>
    <c:txPr>
      <a:bodyPr rot="-5400000" vert="horz"/>
      <a:lstStyle/>
      <a:p>
        <a:pPr>
          <a:defRPr/>
        </a:pPr>
        <a:endParaRPr lang="en-US"/>
      </a:p>
    </c:txPr>
  </c:title>

  <c:numFmt formatCode="General" sourceLinked="1"/>
  <!-- ... -->
</c:valAx>
```

Edit text directly in UI. Note that *c:txPr* element is removed when text is added:

```
<c:title>
  <c:tx>
    <c:rich>
      <a:bodyPr rot="-5400000" vert="horz"/>
      <a:lstStyle/>
      <a:p>
        <a:pPr>
          <a:defRPr/>
        </a:pPr>
        <a:r>
          <a:rPr lang="en-US" dirty="0" smtClean="0"/>
          <a:t>Foobar</a:t>
        </a:r>
        <a:endParaRPr lang="en-US" dirty="0"/>
      </a:p>
    </c:rich>
  </c:tx>
  <c:layout/>
  <c:overlay val="0"/>
</c:title>
```

Related Schema Definitions

```
<xsd:group name="EG_AxShared">
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="crosses"   type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_Title">
  <xsd:sequence>
    <xsd:element name="tx"      type="CT_Tx"                minOccurs="0"/>
    <xsd:element name="layout"  type="CT_Layout"            minOccurs="0"/>
    <xsd:element name="overlay" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="spPr"    type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"    type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Tx">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element name="strRef" type="CT_StrRef"/>
      <xsd:element name="rich"   type="a:CT_TextBody"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Layout">
  <xsd:sequence>
    <xsd:element name="manualLayout" type="CT_ManualLayout"  minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ShapeProperties">
  <xsd:sequence>
    <xsd:element name="xfrm"               type="CT_Transform2D"
↪minOccurs="0"/>
    <xsd:group   ref ="EG_Geometry"
↪minOccurs="0"/>
```

```
    <xsd:group   ref ="EG_FillProperties"                                    ␣
↪minOccurs="0"/>
    <xsd:element name="ln"                 type="CT_LineProperties"          ␣
↪minOccurs="0"/>
    <xsd:group   ref ="EG_EffectProperties"                                  ␣
↪minOccurs="0"/>
    <xsd:element name="scene3d"            type="CT_Scene3D"                 ␣
↪minOccurs="0"/>
    <xsd:element name="sp3d"               type="CT_Shape3D"                 ␣
↪minOccurs="0"/>
    <xsd:element name="extLst"             type="CT_OfficeArtExtensionList"␣
↪minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode"/>
</xsd:complexType>

<xsd:complexType name="CT_TextBody">  <!-- text frame -->
  <xsd:sequence>
    <xsd:element name="bodyPr"   type="CT_TextBodyProperties"/>
    <xsd:element name="lstStyle" type="CT_TextListStyle"      minOccurs="0"/>
    <xsd:element name="p"        type="CT_TextParagraph"      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextBodyProperties">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="prstTxWarp"  type="CT_PresetTextShape"        minOccurs="0"/>
    <xsd:choice minOccurs="0">      <!-- EG_TextAutofit -->
      <xsd:element name="noAutofit"   type="CT_TextNoAutofit"/>
      <xsd:element name="normAutofit" type="CT_TextNormalAutofit"/>
      <xsd:element name="spAutoFit"   type="CT_TextShapeAutofit"/>
    </xsd:choice>
    <xsd:element name="scene3d"     type="CT_Scene3D"                minOccurs="0"/>
    <xsd:choice minOccurs="0">      <!-- EG_Text3D -->
      <xsd:element name="sp3d"        type="CT_Shape3D"/>
      <xsd:element name="flatTx"      type="CT_FlatText"/>
    </xsd:choice>
    <xsd:element name="extLst"      type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="rot"              type="ST_Angle"/>
  <xsd:attribute name="spcFirstLastPara" type="xsd:boolean"/>
  <xsd:attribute name="vertOverflow"     type="ST_TextVertOverflowType"/>
  <xsd:attribute name="horzOverflow"     type="ST_TextHorzOverflowType"/>
  <xsd:attribute name="vert"             type="ST_TextVerticalType"/>
  <xsd:attribute name="wrap"             type="ST_TextWrappingType"/>
  <xsd:attribute name="lIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="tIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="rIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="bIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="numCol"           type="ST_TextColumnCount"/>
  <xsd:attribute name="spcCol"           type="ST_PositiveCoordinate32"/>
  <xsd:attribute name="rtlCol"           type="xsd:boolean"/>
  <xsd:attribute name="fromWordArt"      type="xsd:boolean"/>
  <xsd:attribute name="anchor"           type="ST_TextAnchoringType"/>
  <xsd:attribute name="anchorCtr"        type="xsd:boolean"/>
  <xsd:attribute name="forceAA"          type="xsd:boolean"/>
  <xsd:attribute name="upright"          type="xsd:boolean" default="false"/>
  <xsd:attribute name="compatLnSpc"      type="xsd:boolean"/>
```

```
</xsd:complexType>

<xsd:complexType name="CT_TextListStyle">
  <xsd:sequence>
    <xsd:element name="defPPr"  type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl1pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl2pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl3pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl4pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl5pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl6pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl7pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl8pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl9pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList"  minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ST_TextVerticalType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="horz"/>
    <xsd:enumeration value="vert"/>
    <xsd:enumeration value="vert270"/>
    <xsd:enumeration value="wordArtVert"/>
    <xsd:enumeration value="eaVert"/>
    <xsd:enumeration value="mongolianVert"/>
    <xsd:enumeration value="wordArtVertRtl"/>
  </xsd:restriction>
</xsd:simpleType>
```

Chart - Data Labels

On a PowerPoint chart, data points may be labeled as an aid to readers. Typically, the label is the value of the data point, but a data label may have any combination of its series name, category name, and value. A number format may also be applied to the value displayed.

Object access

The *DataLabels* object is not a collection of *DataLabel* objects. *DataLabels* controls the formatting of the data labels for a whole series, so "global" settings in a way of thinking. An individual *DataLabel* object is accessed using it's associated *Point* object:

```
data_label = chart.plot[0].series[0].points[0].data_label
```

The two object types share many attributes.

DataLabel.font

Proposed protocol:

```
>>> font = data_label.font
>>> font.color.rgb = RGBColor(0x3C, 0xEB, 0x27)
```

Position

There are nine choices for where a data label may be positioned relative to its data point marker, although the available options depend on the chart type. The options are specified using the *XL_DATA_LABEL_POSITION* enumeration.

XML Semantics. The default position when no `<c:dLblPos>` element is present (common) depends on the chart type:

barChart (clustered)	OUTSIDE_END
bar3DChart (clustered)	OUTSIDE_END
barChart (stacked)	CENTER
barChart (percent stacked)	CENTER
bar3DChart (stacked)	CENTER
bar3DChart (percent stacked)	CENTER
pieChart	BEST_FIT
pie3DChart	BEST_FIT
ofPieChart	BEST_FIT
areaChart	CENTER
area3DChart	CENTER
doughnutChart	CENTER
radarChart	OUTSIDE_END
all others	RIGHT

http://msdn.microsoft.com/en-us/library/ff535061(v=office.12).aspx

Proposed protocol:

```
>>> data_labels = plot.data_labels
>>> data_labels.position
OUTSIDE_END (2)
>>> data_labels.position = XL_DATA_LABEL_POSITION.INSIDE_END
>>> data_labels.position
INSIDE_END (3)
```

MS API

The following properties from the MS API DataLabel object are of most interest:

Delete() Deletes the DataLabel object, which might be useful for "undoing" data label overrides and restoring inherited behaviors.

AutoText True if the object automatically generates appropriate text based on context. Read/write Boolean.

Caption Returns or sets the data label text. Read/write String.

Format Returns the line, fill, and effect formatting for the object. Read-only ChartFormat.

NumberFormat Returns or sets the format code for the object. Read/write String.

NumberFormatLinked True if the number format is linked to the cells (so that the number format changes in the labels when it changes in the cells). Read/write Boolean.

Orientation Returns or sets the text orientation. Read/write Long in range -90 to 90 (degrees).

Position Returns or sets the position of the data label. Read/write XlDataLabelPosition.

Separator Returns or sets the separator used for the data labels on a chart. Used when more than one value is shown, like category+value. Read/write Variant.

Shadow Returns or sets a value that indicates whether the object has a shadow. Read/write Boolean.

ShowBubbleSize True to show the bubble size for the data labels on a chart. False to hide the bubble size. Read/write Boolean.

ShowCategoryName True to display the category name for the data labels on a chart. False to hide the category name. Read/write Boolean.

ShowLegendKey True if the data label legend key is visible. This is a small square of the series color shown in the legend, and appears adjacent to the data label. Read/write Boolean.

ShowPercentage True to display the percentage value for the data labels on a chart. False to hide the value. This might only be relevant for pie charts and similar. Read/write Boolean.

ShowSeriesName True to show the series name for the data labels on a chart. False to hide the series name. Read/write Boolean.

ShowValue True to display a specified chart's data label values. False to hide the values. Read/write Boolean.

Text Returns or sets the text for the specified object. Read/write String.

VerticalAlignment Returns or sets the vertical alignment of the specified object. Read/write Variant.

PowerPoint behavior

- A default PowerPoint bar chart does not display data labels, but it does have a `<c:dLbls>` child element on its `<c:barChart>` element.
- Data labels are added to a chart in the UI by selecting the *Data Labels* drop-down menu in the Chart Layout ribbon. The options include setting the contents of the data label, its position relative to the point, and bringing up the *Format Data Labels* dialog.
- The default number format, when no `<c:numFmt>` child element appears, is equivalent to `<c:numFmt formatCode="General" sourceLinked="1"/>`

XML Semantics

- A `<c:dLbls>` element at the plot level (e.g. `<c:barChart>`) is overridden completely by a `<c:dLbls>` element at the series level. Unless overridden, a `<c:dLbls>` element at the plot level determines the content and formatting for data labels on all the plot's series.
- A `<c:dLbl>` element appears as a child of `<c:dLbls>` only when the data label for its associated data point has overrides.

XML specimens

The `<c:dLbls>` element is available on a plot (e.g. `<c:barChart>`), a series (`<c:ser>`), and perhaps elsewhere.

Default `<c:dLbls>` element added by PowerPoint when selecting Data Labels > Value from the Chart Layout ribbon:

```
<c:dLbls>
  <c:showLegendKey val="0"/>
  <c:showVal val="1"/>
  <c:showCatName val="0"/>
  <c:showSerName val="0"/>
  <c:showPercent val="0"/>
  <c:showBubbleSize val="0"/>
</c:dLbls>
```

A <c:dLbls> element specifying the labels should appear in 10pt Bold Italic Arial Narrow, color Accent 6, 25% Darker:

```
<c:dLbls>
  <c:txPr>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr>
        <a:defRPr sz="1000" b="1" i="1">
          <a:solidFill>
            <a:schemeClr val="accent6">
              <a:lumMod val="75000"/>
            </a:schemeClr>
          </a:solidFill>
          <a:latin typeface="Arial Narrow"/>
        </a:defRPr>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:showLegendKey val="0"/>
  <c:showVal val="1"/>
  <c:showCatName val="0"/>
  <c:showSerName val="0"/>
  <c:showPercent val="0"/>
  <c:showBubbleSize val="0"/>
</c:dLbls>
```

A <c:dLbls> element having an individual point override for font color and outline box:

```
<c:dLbls>
  <c:dLbl>
    <c:idx val="0"/>
    <c:spPr>
      <!-- ouline color (Red) -->
      <a:ln>
        <a:solidFill>
          <a:srgbClr val="FF0000"/>
        </a:solidFill>
      </a:ln>
    </c:spPr>
    <c:txPr>
      <a:bodyPr/>
      <a:lstStyle/>
      <a:p>
        <a:pPr>
          <!-- font color (Green) -->
          <a:defRPr>
            <a:solidFill>
              <a:srgbClr val="00FF00"/>
            </a:solidFill>
          </a:defRPr>
        </a:pPr>
        <a:endParaRPr lang="en-US"/>
      </a:p>
    </c:txPr>
```

```
    <c:showLegendKey val="0"/>
    <c:showCatName val="0"/>
    <c:showSerName val="0"/>
  </c:dLbl>
  <c:spPr/>
  <!-- ... -->
</c:dLbls>
```

Related Schema Definitions

```
<xsd:complexType name="CT_DLbls">
  <xsd:sequence>
    <xsd:element name="dLbl" type="CT_DLbl" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:choice>
      <xsd:element name="delete"      type="CT_Boolean"/>
      <xsd:group   ref="Group_DLbls"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="Group_DLbls">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="numFmt"          type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"            type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="dLblPos"         type="CT_DLblPos"           minOccurs="0"/>
    <xsd:element name="showLegendKey"   type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showVal"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showCatName"     type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showSerName"     type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showPercent"     type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showBubbleSize"  type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="separator"       type="xsd:string"           minOccurs="0"/>
    <xsd:element name="showLeaderLines" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="leaderLines"     type="CT_ChartLines"        minOccurs="0"/>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_DLbl">
  <xsd:sequence>
    <xsd:element name="idx" type="CT_UnsignedInt"/>
    <xsd:choice>
      <xsd:element name="delete"     type="CT_Boolean"/>
      <xsd:group   ref="Group_DLbl"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="Group_DLbl"> <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="layout"         type="CT_Layout"            minOccurs="0"/>
    <xsd:element name="tx"             type="CT_Tx"                minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
```

```
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="dLblPos"        type="CT_DLblPos"           minOccurs="0"/>
    <xsd:element name="showLegendKey"  type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showVal"        type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showCatName"    type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showSerName"    type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showPercent"    type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showBubbleSize" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="separator"      type="xsd:string"           minOccurs="0"/>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_DLblPos">
  <xsd:attribute name="val" type="ST_DLblPos" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_NumFmt">
  <xsd:attribute name="formatCode"   type="xsd:string"  use="required"/>
  <xsd:attribute name="sourceLinked" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_TextBody">
  <xsd:sequence>
    <xsd:element name="bodyPr"   type="CT_TextBodyProperties"/>
    <xsd:element name="lstStyle" type="CT_TextListStyle" minOccurs="0"/>
    <xsd:element name="p"        type="CT_TextParagraph" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Tx">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="strRef" type="CT_StrRef"/>
      <xsd:element name="rich"   type="a:CT_TextBody"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ST_DLblPos">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="bestFit"/>
    <xsd:enumeration value="b"/>
    <xsd:enumeration value="ctr"/>
    <xsd:enumeration value="inBase"/>
    <xsd:enumeration value="inEnd"/>
    <xsd:enumeration value="l"/>
    <xsd:enumeration value="outEnd"/>
    <xsd:enumeration value="r"/>
    <xsd:enumeration value="t"/>
  </xsd:restriction>
</xsd:simpleType>
```

Chart - Date Axis

A date axis maintains a linear relationship between axis distance and elapsed time in days, months, or years, even if the data items do not include values for each base time unit. This is in contrast to a category axis, which uses the

independent axis values themselves to determine the discrete set of categories.

The date axis does this by essentially creating a "logical category" for each base time unit in a range, whether the data contains a value for that time unit or not.

Variations in the number of days in a month (or year) is also taken care of automatically. This would otherwise cause month starts to appear at days not the first of the month.

This does not make the category axis continuous, however the "grain" of the discrete units can be quite small in comparison to the axis length. The individual base unit divisions can but are not typically represented graphically. The granularity of major and minor divisions is specified as a property on the date axis.

MS API

- Axis.CategoryType = XL_CATEGORY_TYPE.TIME
- Axis.BaseUnit = XL_TIME_UNIT.DAYS

Enumerations

XL_CATEGORY_TYPE

- .AUTOMATIC (-4105)
- .CATEGORY (2)
- .TIME (3)

XL_TIME_UNIT https://msdn.microsoft.com/en-us/library/office/ff746136.aspx * .DAYS (0) * .MONTHS (1) * .YEARS (2)

XML Semantics

Changing category axis between *c:catAx* and *c:dateAx*

In the MS API, DateAxis.CategoryType is used to discover the category type, but is also used to change an axis from a date axis to a category axis and back. Note the corresponding property in *python-pptx* is read-only for now.

Changing from a category to date axis leaves the base, major, and minor unit elements in place. They apply again (instead of the defaults) if the axis is changed back:

```
<c:baseTimeUnit val="days"/>
<c:majorUnit val="1.0"/>
<c:majorTimeUnit val="months"/>
<c:minorUnit val="1.0"/>
```

Choice of date axis

PowerPoint automatically uses a *c:dateAx* element if the category labels in Excel are dates (numbers with date formatting).

python-pptx uses a *c:dateAx* element if the category labels in the chart data object are datetime.date or datetime.datetime objects.

Note that *python-pptx* does not change the category axis type when using *Chart.replace_data()*.

A date axis is only available on an area, bar (including column), or line chart.

Multi-level categories and date axis

Multi-level categories are mutually exclusive with a date axis.

A *c:multiLvlStrRef* element is the only one that can enclose a multi-level category hierarchy. There is no provision for any or all the levels under this element to be other than string values (as the 'Str' in its name implies).

Automatic axis type

The *c:auto* element under *c:catAx* or *c:dateAx* controls whether the category axis changes automatically between category and date types.

Specimen XML

Series having date categories:

```
<c:ser>
  <c:idx val="0"/>
  <c:order val="0"/>
  <c:tx>
    <c:strRef>
      <c:f>Sheet1!$B$1</c:f>
      <c:strCache>
        <c:ptCount val="1"/>
        <c:pt idx="0">
          <c:v>Buzz</c:v>
        </c:pt>
      </c:strCache>
    </c:strRef>
  </c:tx>
  <c:marker>
    <c:symbol val="none"/>
  </c:marker>
  <c:cat>
    <c:numRef>
      <c:f>Sheet1!$A$2:$A$520</c:f>
      <c:numCache>
        <c:formatCode>mm\-dd\-yyyy</c:formatCode>
        <c:ptCount val="3"/>
        <c:pt idx="0">
          <c:v>42156.0</c:v>
        </c:pt>
        <c:pt idx="1">
          <c:v>42157.0</c:v>
        </c:pt>
        <c:pt idx="2">
          <c:v>42158.0</c:v>
        </c:pt>
      </c:numCache>
    </c:numRef>
  </c:cat>
  <c:val>
    <c:numRef>
```

```
      <c:f>Sheet1!$B$2:$B$520</c:f>
      <c:numCache>
        <c:formatCode>0.0</c:formatCode>
        <c:ptCount val="3"/>
        <c:pt idx="0">
          <c:v>19.65943065775559</c:v>
        </c:pt>
        <c:pt idx="1">
          <c:v>20.13705095574664</c:v>
        </c:pt>
        <c:pt idx="2">
          <c:v>19.48264757927654</c:v>
        </c:pt>
      </c:numCache>
    </c:numRef>
  </c:val>
```

Plot area having date axis:

```
<c:plotArea>
  <c:lineChart>
    <c:ser>
      <c:idx val="0"/>
      <c:order val="0"/>
      <c:tx>
        <c:strRef>
          <c:f>Sheet1!$B$1</c:f>
          <c:strCache>
            <c:ptCount val="1"/>
            <c:pt idx="0">
              <c:v>Series 1</c:v>
            </c:pt>
          </c:strCache>
        </c:strRef>
      </c:tx>
      <c:cat>
        <c:numRef>
          <c:f>Sheet1!$A$2:$A$4</c:f>
          <c:numCache>
            <c:formatCode>d\-mmm</c:formatCode>
            <c:ptCount val="3"/>
            <c:pt idx="0">
              <c:v>42370.0</c:v>
            </c:pt>
            <c:pt idx="1">
              <c:v>42371.0</c:v>
            </c:pt>
            <c:pt idx="2">
              <c:v>42372.0</c:v>
            </c:pt>
          </c:numCache>
        </c:numRef>
      </c:cat>
      <c:val>
        <c:numRef>
          <c:f>Sheet1!$B$2:$B$4</c:f>
          <c:numCache>
            <c:formatCode>General</c:formatCode>
```

```
            <c:ptCount val="3"/>
            <c:pt idx="0">
              <c:v>4.3</c:v>
            </c:pt>
            <c:pt idx="1">
              <c:v>2.5</c:v>
            </c:pt>
            <c:pt idx="2">
              <c:v>3.5</c:v>
            </c:pt>
          </c:numCache>
        </c:numRef>
      </c:val>
      <c:smooth val="0"/>
    </c:ser>
    <c:axId val="2142588392"/>
    <c:axId val="2106388088"/>
  </c:lineChart>
  <c:dateAx>
    <c:axId val="2142588392"/>
    <c:scaling>
      <c:orientation val="minMax"/>
    </c:scaling>
    <c:delete val="0"/>
    <c:axPos val="b"/>
    <c:numFmt formatCode="d\-mmm" sourceLinked="1"/>
    <c:majorTickMark val="out"/>
    <c:minorTickMark val="none"/>
    <c:tickLblPos val="nextTo"/>
    <c:crossAx val="2106388088"/>
    <c:crosses val="autoZero"/>
    <c:auto val="1"/>
    <c:lblOffset val="100"/>
    <c:baseTimeUnit val="days"/>
  </c:dateAx>
  <c:valAx>
    <c:axId val="2106388088"/>
    <c:scaling>
      <c:orientation val="minMax"/>
    </c:scaling>
    <c:delete val="0"/>
    <c:axPos val="l"/>
    <c:majorGridlines/>
    <c:numFmt formatCode="General" sourceLinked="1"/>
    <c:majorTickMark val="out"/>
    <c:minorTickMark val="none"/>
    <c:tickLblPos val="nextTo"/>
    <c:crossAx val="2142588392"/>
    <c:crosses val="autoZero"/>
    <c:crossBetween val="between"/>
  </c:valAx>
</c:plotArea>
```

References

- Understanding Date-Based Axis Versus Category-Based Axis in Trend Charts http://www.quepublishing.com/articles/article.aspx?p=1642672&seqNum=2

Related Schema Definitions

```
<xsd:complexType name="CT_PlotArea">
  <xsd:sequence>
    <!-- 17 others -->
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="valAx"  type="CT_ValAx"/>
      <xsd:element name="catAx"  type="CT_CatAx"/>
      <xsd:element name="dateAx" type="CT_DateAx"/>
      <xsd:element name="serAx"  type="CT_SerAx"/>
    </xsd:choice>
    <xsd:element name="dTable" type="CT_DTable"            minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_CatAx">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"       minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"       minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"            minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"           minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"       minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"       minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice                                               minOccurs="0">
      <xsd:element name="crosses"      type="CT_Crosses"/>
      <xsd:element name="crossesAt"    type="CT_Double"/>
    </xsd:choice>
    <xsd:element name="auto"           type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="lblAlgn"        type="CT_LblAlgn"          minOccurs="0"/>
    <xsd:element name="lblOffset"      type="CT_LblOffset"        minOccurs="0"/>
    <xsd:element name="tickLblSkip"    type="CT_Skip"             minOccurs="0"/>
    <xsd:element name="tickMarkSkip"   type="CT_Skip"             minOccurs="0"/>
    <xsd:element name="noMultiLvlLbl"  type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DateAx">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
```

```
    <xsd:element name="delete"          type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"           type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"           type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"          type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"      type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"            type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"         type="CT_UnsignedInt"/>
    <xsd:choice                                                     minOccurs="0">
      <xsd:element name="crosses"       type="CT_Crosses"/>
      <xsd:element name="crossesAt"     type="CT_Double"/>
    </xsd:choice>
    <xsd:element name="auto"           type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="lblOffset"      type="CT_LblOffset"     minOccurs="0"/>
    <xsd:element name="baseTimeUnit"   type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="majorUnit"      type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="majorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="minorUnit"      type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="minorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ValAx">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="axId"            type="CT_UnsignedInt"/>
    <xsd:element name="scaling"         type="CT_Scaling"/>
    <xsd:element name="delete"          type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"           type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"           type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"          type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"      type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"            type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"         type="CT_UnsignedInt"/>
    <xsd:choice                                                     minOccurs="0">
      <xsd:element name="crosses"    type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
    <xsd:element name="crossBetween"   type="CT_CrossBetween"      minOccurs="0"/>
    <xsd:element name="majorUnit"      type="CT_AxisUnit"          minOccurs="0"/>
    <xsd:element name="minorUnit"      type="CT_AxisUnit"          minOccurs="0"/>
    <xsd:element name="dispUnits"      type="CT_DispUnits"         minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SerAx">
  <xsd:sequence>
    <xsd:group   ref="EG_AxShared"/>
```

```
    <xsd:element name="tickLblSkip"  type="CT_Skip"          minOccurs="0"/>
    <xsd:element name="tickMarkSkip" type="CT_Skip"          minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_AxShared">
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="crosses"   type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_AxisUnit">
  <xsd:attribute name="val" type="ST_AxisUnit" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_ChartLines">
  <xsd:sequence>
    <xsd:element name="spPr" type="a:CT_ShapeProperties" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Crosses">
  <xsd:attribute name="val" type="ST_Crosses" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_Scaling">
  <xsd:sequence>
    <xsd:element name="logBase"     type="CT_LogBase"       minOccurs="0"/>
    <xsd:element name="orientation" type="CT_Orientation"   minOccurs="0"/>
    <xsd:element name="max"         type="CT_Double"        minOccurs="0"/>
    <xsd:element name="min"         type="CT_Double"        minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumFmt">
  <xsd:attribute name="formatCode"   type="xsd:string"  use="required"/>
  <xsd:attribute name="sourceLinked" type="xsd:boolean"/>
</xsd:complexType>
```

```
<xsd:complexType name="CT_TickLblPos">
  <xsd:attribute name="val" type="ST_TickLblPos" default="nextTo"/>
</xsd:complexType>

<xsd:complexType name="CT_TickMark">
  <xsd:attribute name="val" type="ST_TickMark" default="cross"/>
</xsd:complexType>

<xsd:complexType name="CT_TimeUnit">
  <xsd:attribute name="val" type="ST_TimeUnit" default="days"/>
</xsd:complexType>

<xsd:complexType name="CT_Boolean">
  <xsd:attribute name="val" type="xsd:boolean" default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_Double">
  <xsd:attribute name="val" type="xsd:double" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_AxisUnit">
  <xsd:restriction base="xsd:double">
    <xsd:minExclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_Crosses">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="autoZero"/>
    <xsd:enumeration value="max"/>
    <xsd:enumeration value="min"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_TickLblPos">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="high"/>
    <xsd:enumeration value="low"/>
    <xsd:enumeration value="nextTo"/>
    <xsd:enumeration value="none"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_TickMark">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="cross"/>
    <xsd:enumeration value="in"/>
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="out"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_TimeUnit">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="days"/>
    <xsd:enumeration value="months"/>
    <xsd:enumeration value="years"/>
  </xsd:restriction>
```

```
</xsd:simpleType>
```

Chart - Categories

- A category can be `None`, meaning the category exists, but there is no label for it.
  - The string value of it can be ‘’, since it needs to be a string. But the .label value will be `None`.

Caveats

- Categories are read from the first series in the plot. In all normal situations I’ve ever encountered, all series have exactly the same set of categories.

  I’m not sure why the category labels are redundantly applied to each series, but the fact that they are leads me to believe I must be missing something; I don’t believe they would repeat them like that if there wasn’t a situation in which they could be different between two series in the same plot.

  Consequently, there might be some edge cases where this won’t work quite as expected. At the same time, I expect it will work exactly as expected for 99.9% of the cases, maybe 99.99%.

- Categories are not enforced to be strictly hierarchical. So if a higher-level category “breaks” (gets a new value) without also a new value at the next level, the results might be a little weird. Like changing from USA to Canada, but leaving NY as the state (by leaving that cell blank).

  Basically, the meaning of a structure like this is ambiguous. PowerPoint interprets it without error and python-pptx interprets it without error, but the results might not exactly match up. PowerPoint seems to take the senior category break and show the intermediate category as blank or ‘None’. But I suppose it could be interpreted as the senior category just splits a subordinate one; it’s open to some question.

Multilevel Categories

A category chart can have more than one level of categories, where each individual category is nested in a higher-level category.

https://www.youtube.com/watch?v=KbyQpzA7fLo

Acceptance Tests

cht-plot-props.feature:

```
Scenario: Get CategoryPlot.categories
  Given a category plot
   Then plot.categories is a Categories object
```

cht-categories.feature:

```
Scenario: Categories.__len__()
  Given a Categories object having 3 categories
   Then len(categories) is 3

Scenario: Categories.__getitem__()
  Given a Categories object having 3 categories
   Then categories[2] is a Category object
```

```
Scenario: Categories.__iter__()
  Given a Categories object having 3 categories
   Then iterating categories produces 3 Category objects

Scenario: Categories.depth
  Given a Categories object having 3 levels of categories
   Then categories.depth is 3

Scenario: Categories.flattened
  Given a Categories object having 3 levels of categories
   Then categories.flattened is tuple of 3-tuples
```

cht-category.feature:

```
Scenario: Category derives from string
  Given a Category object
   Then isinstance(category, str) is True
```

XML semantics

- The hierarchy of the levels is indicated by their document order.
- The scope of non-leaf category entries is indicated by the 'idx' attribute value. A non-leaf category spans from the leaf node having the matching idx to the last leaf node not contained in its subsequent sibling.
- The idx value on each *<c:pt>* element identifies the element for possible overrides, like manual positioning or deletion (hiding). It may also key it to the values in the series and/or other items; the spec is silent on these details.
- I can't find a way to set the *c:noMultiLvlLbl* element truthy using the UI. I suspect this is only an Excel option.
- The *c:lvl* element does not appear when there is only a single level of categories. Also in that case, a *c:strCache* element contains the *c:pt* elements rather than a *c:multiLvlStrCache* element.

17. What behavior is produced by a truthy value in *c:catAx/c:noMultiLvlLbl/@val* when there are multiple levels of categories defined?

XML specimens

Single-level categories (common case):

```
<c:cat>
  <c:strRef>
    <c:f>Sheet1!$A$2:$A$5</c:f>
    <c:strCache>
      <c:ptCount val="4"/>
      <c:pt idx="0">
        <c:v>Category 1</c:v>
      </c:pt>
      <c:pt idx="1">
        <c:v>Category 2</c:v>
      </c:pt>
      <c:pt idx="2">
        <c:v>Category 3</c:v>
      </c:pt>
```

```
      <c:pt idx="3">
        <c:v>Category 4</c:v>
      </c:pt>
    </c:strCache>
  </c:strRef>
</c:cat>
```

Multi-level categories:

```
<c:cat>
   <c:multiLvlStrRef>
     <c:f>Sheet1!$C$1:$J$3</c:f>
     <c:multiLvlStrCache>
       <c:ptCount val="8"/>
       <c:lvl>
         <c:pt idx="0">
           <c:v>county one</c:v>
         </c:pt>
         <c:pt idx="1">
           <c:v>county two</c:v>
         </c:pt>
         <c:pt idx="2">
           <c:v>county one</c:v>
         </c:pt>
         <c:pt idx="3">
           <c:v>county two</c:v>
         </c:pt>
         <c:pt idx="4">
           <c:v>county one</c:v>
         </c:pt>
         <c:pt idx="5">
           <c:v>county two</c:v>
         </c:pt>
         <c:pt idx="6">
           <c:v>country one</c:v>
         </c:pt>
         <c:pt idx="7">
           <c:v>county two</c:v>
         </c:pt>
       </c:lvl>
       <c:lvl>
         <c:pt idx="0">
           <c:v>city one</c:v>
         </c:pt>
         <c:pt idx="2">
           <c:v>city two </c:v>
         </c:pt>
         <c:pt idx="4">
           <c:v>city one</c:v>
         </c:pt>
         <c:pt idx="6">
           <c:v>city two </c:v>
         </c:pt>
       </c:lvl>
       <c:lvl>
         <c:pt idx="0">
           <c:v>UK</c:v>
         </c:pt>
```

```
          <c:pt idx="4">
            <c:v>US</c:v>
          </c:pt>
        </c:lvl>
      </c:multiLvlStrCache>
    </c:multiLvlStrRef>
  </c:cat>


<c:catAx>
  ...
  <c:noMultiLvlLbl val="0"/>
</c:catAx>
```

Related Schema Definitions

A `<c:cat>` element is a child of a `<c:ser>` (series) element and is of the `CT_AxDataSource` type:

```
<xsd:complexType name="CT_AxDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="multiLvlStrRef" type="CT_MultiLvlStrRef"/>
      <xsd:element name="numRef"         type="CT_NumRef"/>
      <xsd:element name="numLit"         type="CT_NumData"/>
      <xsd:element name="strRef"         type="CT_StrRef"/>
      <xsd:element name="strLit"         type="CT_StrData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_MultiLvlStrRef">
  <xsd:sequence>
    <xsd:element name="f"                type="xsd:string"/>
    <xsd:element name="multiLvlStrCache" type="CT_MultiLvlStrData" minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList"   minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_MultiLvlStrData">
  <xsd:sequence>
    <xsd:element name="ptCount" type="CT_UnsignedInt"   minOccurs="0"/>
    <xsd:element name="lvl"     type="CT_Lvl"           minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="extLst"  type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Lvl">
  <xsd:sequence>
    <xsd:element name="pt" type="CT_StrVal" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_StrVal">
  <xsd:sequence>
    <xsd:element name="v" type="s:ST_Xstring"/>
```

```
  </xsd:sequence>
  <xsd:attribute name="idx" type="xsd:unsignedInt" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_Xstring">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

Chart - Markers

A marker is a small geometric shape that explicitly indicates a data point position on a line-oriented chart. Line, XY, and Radar are the line-oriented charts. Other chart types do not have markers.

PowerPoint behavior

At series level, UI provides property controls for:

- Marker Fill
  - Solid
  - Gradient
  - Picture or Texture
  - Pattern
- Marker Line
  - Solid
    - RGBColor (probably HSB too, etc.)
    - Transparency
  - Gradient
  - Weights & Arrows
- Marker Style
  - Choice of enumerated shapes
  - Size

XL_MARKER_STYLE enumeration

https://msdn.microsoft.com/en-us/library/bb241374(v=office.12).aspx

xlMarkerStyleAutomatic	-4105	Automatic markers
xlMarkerStyleCircle	8	Circular markers
xlMarkerStyleDash	-4115	Long bar markers
xlMarkerStyleDiamond	2	Diamond-shaped markers
xlMarkerStyleDot	-4118	Short bar markers
xlMarkerStyleNone	-4142	No markers
xlMarkerStylePicture	-4147	Picture markers
xlMarkerStylePlus	9	Square markers with a plus sign
xlMarkerStyleSquare	1	Square markers
xlMarkerStyleStar	5	Square markers with an asterisk
xlMarkerStyleTriangle	3	Triangular markers
xlMarkerStyleX	-4168	Square markers with an X

XML specimens

Marker properties set at the series level, all markers for the series:

```
<c:ser>
  <!-- ... -->
  <c:marker>
    <c:spPr>
      <a:solidFill>
        <a:schemeClr val="accent4">
          <a:lumMod val="60000"/>
          <a:lumOff val="40000"/>
        </a:schemeClr>
      </a:solidFill>
      <a:ln>
        <a:solidFill>
          <a:schemeClr val="accent6">
            <a:alpha val="51000"/>
          </a:schemeClr>
        </a:solidFill>
      </a:ln>
    </c:spPr>
  </c:marker>
  <!-- ... -->
</c:ser>
```

Marker properties set on an individual point:

```
<c:ser>
  <!-- ... -->
  <c:dPt>
    <c:idx val="0"/>
    <c:marker>
      <c:symbol val="circle"/>
      <c:size val="16"/>
      <c:spPr>
        <a:gradFill flip="none" rotWithShape="1">
          <a:gsLst>
            <a:gs pos="0">
              <a:schemeClr val="accent2"/>
            </a:gs>
            <a:gs pos="80000">
              <a:srgbClr val="FFFFFF">
```

```
              <a:alpha val="61000"/>
            </a:srgbClr>
          </a:gs>
        </a:gsLst>
        <a:path path="circle">
          <a:fillToRect l="50000" t="50000" r="50000" b="50000"/>
        </a:path>
        <a:tileRect/>
      </a:gradFill>
      <a:ln w="12700">
        <a:solidFill>
          <a:schemeClr val="accent6"/>
        </a:solidFill>
        <a:prstDash val="sysDot"/>
      </a:ln>
      <a:effectLst>
        <a:outerShdw blurRad="50800" dist="38100" dir="14160000" algn="tl"␣
↪rotWithShape="0">
          <a:schemeClr val="accent6">
            <a:lumMod val="75000"/>
            <a:alpha val="43000"/>
          </a:schemeClr>
        </a:outerShdw>
      </a:effectLst>
    </c:spPr>
  </c:marker>
  <c:bubble3D val="0"/>
  <c:spPr>
    <a:effectLst>
      <a:outerShdw blurRad="50800" dist="38100" dir="14160000" algn="tl"␣
↪rotWithShape="0">
        <a:schemeClr val="accent6">
          <a:lumMod val="75000"/>
          <a:alpha val="43000"/>
        </a:schemeClr>
      </a:outerShdw>
    </a:effectLst>
  </c:spPr>
 </c:dPt>
 <!-- ... -->
</c:ser>
```

Related Schema Definitions

```
<xsd:complexType name="CT_LineSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"       type="CT_UnsignedInt"/>
    <xsd:element name="order"     type="CT_UnsignedInt"/>
    <xsd:element name="tx"        type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"      type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker"    type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"       type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"     type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline" type="CT_Trendline"         minOccurs="0" maxOccurs=
↪"unbounded"/>
```

```
    <xsd:element name="errBars"   type="CT_ErrBars"           minOccurs="0"/>
    <xsd:element name="cat"       type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"       type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="smooth"    type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ScatterSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"         type="CT_UnsignedInt"/>
    <xsd:element name="order"       type="CT_UnsignedInt"/>
    <xsd:element name="tx"          type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"        type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker"      type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"         type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"       type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"   type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"     type="CT_ErrBars"           minOccurs="0"␣
↪maxOccurs="2"/>
    <xsd:element name="xVal"        type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="yVal"        type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="smooth"      type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_RadarSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"    type="CT_UnsignedInt"/>
    <xsd:element name="order"  type="CT_UnsignedInt"/>
    <xsd:element name="tx"     type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker" type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"    type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"  type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="cat"    type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"    type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DPt">
  <xsd:sequence>
    <xsd:element name="idx"              type="CT_UnsignedInt"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="marker"           type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="bubble3D"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="explosion"        type="CT_UnsignedInt"       minOccurs="0"/>
    <xsd:element name="spPr"             type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="pictureOptions"   type="CT_PictureOptions"    minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="CT_Marker">
  <xsd:sequence>
    <xsd:element name="symbol" type="CT_MarkerStyle"       minOccurs="0"/>
    <xsd:element name="size"   type="CT_MarkerSize"        minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_MarkerStyle">
  <xsd:attribute name="val" type="ST_MarkerStyle" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_MarkerSize">
  <xsd:attribute name="val" type="ST_MarkerSize" default="5"/>
</xsd:complexType>

<xsd:simpleType name="ST_MarkerSize">
  <xsd:restriction base="xsd:unsignedByte">
    <xsd:minInclusive value="2"/>
    <xsd:maxInclusive value="72"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_MarkerStyle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="circle"/>
    <xsd:enumeration value="dash"/>
    <xsd:enumeration value="diamond"/>
    <xsd:enumeration value="dot"/>
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="picture"/>
    <xsd:enumeration value="plus"/>
    <xsd:enumeration value="square"/>
    <xsd:enumeration value="star"/>
    <xsd:enumeration value="triangle"/>
    <xsd:enumeration value="x"/>
    <xsd:enumeration value="auto"/>
  </xsd:restriction>
</xsd:simpleType>
```

Charts - Overview

Adding a new chart - steps and tests

1. Analysis - add chart type page with schema for each new chart element and see if there's anything distinctive about the chart type series
2. feature/sld-add-chart .. add new chart types, steps/chart.py
3. pptx/chart/xmlwriter.py add new type to ChartXmlWriter
4. pptx/chart/xmlwriter.py add _AreaChartXmlWriter, one per new element type
5. pptx/chart/series.py add AreaSeries, one per new element type

There are 73 different possible chart types, but only 16 distinct XML chart-type elements. The implementation effort is largely proportional to the number of new XML chart-type elements.

Here is an accounting of the implementation status of the 73 chart types:

29 - supported for creation so far

areaChart

- AREA
- AREA_STACKED
- AREA_STACKED_100

barChart

- BAR_CLUSTERED
- BAR_STACKED
- BAR_STACKED_100
- COLUMN_CLUSTERED
- COLUMN_STACKED
- COLUMN_STACKED_100

bubbleChart

- BUBBLE
- BUBBLE_THREE_D_EFFECT

doughnutChart

- DOUGHNUT
- DOUGHNUT_EXPLODED

lineChart

- LINE
- LINE_MARKERS
- LINE_MARKERS_STACKED
- LINE_MARKERS_STACKED_100
- LINE_STACKED
- LINE_STACKED_100

pieChart

- PIE
- PIE_EXPLODED

radarChart

- RADAR
- RADAR_FILLED
- RADAR_MARKERS

scatterChart

- XY_SCATTER
- XY_SCATTER_LINES
- XY_SCATTER_LINES_NO_MARKERS
- XY_SCATTER_SMOOTH
- XY_SCATTER_SMOOTH_NO_MARKERS

44 remaining:

area3DChart

- THREE_D_AREA
- THREE_D_AREA_STACKED
- THREE_D_AREA_STACKED_100

bar3DChart (28 types)

- THREE_D_BAR_CLUSTERED
- THREE_D_BAR_STACKED
- THREE_D_BAR_STACKED_100
- THREE_D_COLUMN
- THREE_D_COLUMN_CLUSTERED
- THREE_D_COLUMN_STACKED
- THREE_D_COLUMN_STACKED_100
- CONE_BAR_CLUSTERED
- CONE_BAR_STACKED
- CONE_BAR_STACKED_100
- CONE_COL

- CONE_COL_CLUSTERED
- CONE_COL_STACKED
- CONE_COL_STACKED_100
- CYLINDER_BAR_CLUSTERED
- CYLINDER_BAR_STACKED
- CYLINDER_BAR_STACKED_100
- CYLINDER_COL
- CYLINDER_COL_CLUSTERED
- CYLINDER_COL_STACKED
- CYLINDER_COL_STACKED_100
- PYRAMID_BAR_CLUSTERED
- PYRAMID_BAR_STACKED
- PYRAMID_BAR_STACKED_100
- PYRAMID_COL
- PYRAMID_COL_CLUSTERED
- PYRAMID_COL_STACKED
- PYRAMID_COL_STACKED_100

line3DChart

- THREE_D_LINE

pie3DChart

- THREE_D_PIE
- THREE_D_PIE_EXPLODED

ofPieChart

- BAR_OF_PIE
- PIE_OF_PIE

stockChart

- STOCK_HLC
- STOCK_OHLC
- STOCK_VHLC
- STOCK_VOHLC

surfaceChart

- SURFACE
- SURFACE_WIREFRAME

surface3DChart

- SURFACE_TOP_VIEW
- SURFACE_TOP_VIEW_WIREFRAME

Chart parts glossary

data point (point) An individual numeric value, represented by a bar, point, column, or pie slice.

data series (series) A group of related data points. For example, the columns of a series will all be the same color.

category axis (X axis) The horizontal axis of a two-dimensional or three-dimensional chart.

value axis (Y axis) The vertical axis of a two-dimensional or three-dimensional chart.

depth axis (Z axis) The front-to-back axis of a three-dimensional chart.

grid lines Horizontal or vertical lines that may be added to an axis to aid comparison of a data point to an axis value.

legend A key that explains which data series each color or pattern represents.

floor The bottom of a three-dimensional chart.

walls The background of a chart. Three-dimensional charts have a back wall and a side wall, which can be formatted separately.

data labels Numeric labels on each data point. A data label can represent the actual value or a percentage.

axis title Explanatory text label associated with an axis

data table A optional tabular display within the *plot area* of the values on which the chart is based. Not to be confused with the Excel worksheet holding the chart values.

chart title A label explaining the overall purpose of the chart.

chart area Overall chart object, containing the chart and all its auxiliary pieces such as legends and titles.

plot area Region of the chart area that contains the actual plots, bounded by but not including the axes. May contain more than one plot, each with its own distinct set of series. A plot is known as a *chart group* in the MS API.

axis ... may be either a *category axis* or a *value axis* ... on a two-dimensional chart, either the horizontal (x) axis or the vertical (y) axis. A 3-dimensional chart also has a depth (z) axis. Pie, doughnut, and radar charts have a radial axis.

How many axes do each of the different chart types have?

series categories ...

series values ...

Chart types

- column
  - 2-D column
    - clustered column
    - stacked column
    - 100% stacked column
  - 3-D column
    - 3-D clustered column
    - 3-D stacked column
    - 3-D 100% stacked column
    - 3-D column
  - cylinder
  - cone
  - pyramid
- line
  - 2-D line
  - 3-D line
- pie
  - 2-D pie
  - 3-D pie
- bar
  - 2-D bar
  - 3-D bar
  - cylinder
  - cone
  - pyramid
- area
- scatter
- other
  - stock (e.g. open-high-low-close)
  - surface
  - doughnut
  - bubble
  - radar

Related Schema Definitions

```
<!-- homonym <c:chart> element in graphicData element -->

<xsd:element name="chart" type="CT_RelId"/>

<xsd:complexType name="CT_RelId">
  <xsd:attribute ref="r:id" use="required"/>
</xsd:complexType>


<!-- elements in chartX.xml part -->

<xsd:element name="chartSpace" type="CT_ChartSpace"/>

<xsd:complexType name="CT_ChartSpace">
  <xsd:sequence>
    <xsd:element name="date1904"       type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="lang"           type="CT_TextLanguageID"    minOccurs="0"/>
    <xsd:element name="roundedCorners" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="style"          type="CT_Style"             minOccurs="0"/>
    <xsd:element name="clrMapOvr"      type="a:CT_ColorMapping"    minOccurs="0"/>
    <xsd:element name="pivotSource"    type="CT_PivotSource"       minOccurs="0"/>
    <xsd:element name="protection"     type="CT_Protection"        minOccurs="0"/>
    <xsd:element name="chart"          type="CT_Chart"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="externalData"   type="CT_ExternalData"      minOccurs="0"/>
    <xsd:element name="printSettings"  type="CT_PrintSettings"     minOccurs="0"/>
    <xsd:element name="userShapes"     type="CT_RelId"             minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Chart">
  <xsd:sequence>
    <xsd:element name="title"            type="CT_Title"         minOccurs="0"/>
    <xsd:element name="autoTitleDeleted" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="pivotFmts"        type="CT_PivotFmts"     minOccurs="0"/>
    <xsd:element name="view3D"           type="CT_View3D"        minOccurs="0"/>
    <xsd:element name="floor"            type="CT_Surface"       minOccurs="0"/>
    <xsd:element name="sideWall"         type="CT_Surface"       minOccurs="0"/>
    <xsd:element name="backWall"         type="CT_Surface"       minOccurs="0"/>
    <xsd:element name="plotArea"         type="CT_PlotArea"/>
    <xsd:element name="legend"           type="CT_Legend"        minOccurs="0"/>
    <xsd:element name="plotVisOnly"      type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="dispBlanksAs"     type="CT_DispBlanksAs"  minOccurs="0"/>
    <xsd:element name="showDLblsOverMax" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_PlotArea">
  <xsd:sequence>
    <xsd:element name="layout" type="CT_Layout" minOccurs="0" maxOccurs="1"/>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element name="areaChart"      type="CT_AreaChart"/>
      <xsd:element name="area3DChart"    type="CT_Area3DChart"/>
```

```
      <xsd:element name="lineChart"      type="CT_LineChart"/>
      <xsd:element name="line3DChart"    type="CT_Line3DChart"/>
      <xsd:element name="stockChart"     type="CT_StockChart"/>
      <xsd:element name="radarChart"     type="CT_RadarChart"/>
      <xsd:element name="scatterChart"   type="CT_ScatterChart"/>
      <xsd:element name="pieChart"       type="CT_PieChart"/>
      <xsd:element name="pie3DChart"     type="CT_Pie3DChart"/>
      <xsd:element name="doughnutChart"  type="CT_DoughnutChart"/>
      <xsd:element name="barChart"       type="CT_BarChart"/>
      <xsd:element name="bar3DChart"     type="CT_Bar3DChart"/>
      <xsd:element name="ofPieChart"     type="CT_OfPieChart"/>
      <xsd:element name="surfaceChart"   type="CT_SurfaceChart"/>
      <xsd:element name="surface3DChart" type="CT_Surface3DChart"/>
      <xsd:element name="bubbleChart"    type="CT_BubbleChart"/>
    </xsd:choice>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="valAx"  type="CT_ValAx"/>
      <xsd:element name="catAx"  type="CT_CatAx"/>
      <xsd:element name="dateAx" type="CT_DateAx"/>
      <xsd:element name="serAx"  type="CT_SerAx"/>
    </xsd:choice>
    <xsd:element name="dTable" type="CT_DTable"            minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_CatAx">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"       minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"       minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"            minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"           minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"       minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"       minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice                                               minOccurs="0">
      <xsd:element name="crosses"      type="CT_Crosses"/>
      <xsd:element name="crossesAt"    type="CT_Double"/>
    </xsd:choice>
    <xsd:element name="auto"           type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="lblAlgn"        type="CT_LblAlgn"          minOccurs="0"/>
    <xsd:element name="lblOffset"      type="CT_LblOffset"        minOccurs="0"/>
    <xsd:element name="tickLblSkip"    type="CT_Skip"             minOccurs="0"/>
    <xsd:element name="tickMarkSkip"   type="CT_Skip"             minOccurs="0"/>
    <xsd:element name="noMultiLvlLbl"  type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ValAx">  <!-- denormalized -->
```

```
  <xsd:sequence>
    <xsd:element name="axId"          type="CT_UnsignedInt"/>
    <xsd:element name="scaling"       type="CT_Scaling"/>
    <xsd:element name="delete"        type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="axPos"         type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"      minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"      minOccurs="0"/>
    <xsd:element name="title"         type="CT_Title"            minOccurs="0"/>
    <xsd:element name="numFmt"        type="CT_NumFmt"           minOccurs="0"/>
    <xsd:element name="majorTickMark" type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="minorTickMark" type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="tickLblPos"    type="CT_TickLblPos"       minOccurs="0"/>
    <xsd:element name="spPr"          type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"          type="a:CT_TextBody"       minOccurs="0"/>
    <xsd:element name="crossAx"       type="CT_UnsignedInt"/>
    <xsd:choice                                                  minOccurs="0">
      <xsd:element name="crosses"   type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
    <xsd:element name="crossBetween"  type="CT_CrossBetween"     minOccurs="0"/>
    <xsd:element name="majorUnit"     type="CT_AxisUnit"         minOccurs="0"/>
    <xsd:element name="minorUnit"     type="CT_AxisUnit"         minOccurs="0"/>
    <xsd:element name="dispUnits"     type="CT_DispUnits"        minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Boolean">
  <xsd:attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_Double">
  <xsd:attribute name="val" type="xsd:double" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_Scaling">
  <xsd:sequence>
    <xsd:element name="logBase"     type="CT_LogBase"       minOccurs="0"/>
    <xsd:element name="orientation" type="CT_Orientation"   minOccurs="0"/>
    <xsd:element name="max"         type="CT_Double"        minOccurs="0"/>
    <xsd:element name="min"         type="CT_Double"        minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumFmt">
  <xsd:attribute name="formatCode"   type="s:ST_Xstring" use="required"/>
  <xsd:attribute name="sourceLinked" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_TickMark">
  <xsd:attribute name="val" type="ST_TickMark" default="cross"/>
</xsd:complexType>

<xsd:simpleType name="ST_TickMark">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="cross"/>
    <xsd:enumeration value="in"/>
```

```
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="out"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_DLbls">
  <xsd:sequence>
    <xsd:element name="dLbl" type="CT_DLbl" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:choice>
      <xsd:element name="delete" type="CT_Boolean"/>
      <xsd:group   ref="Group_DLbls"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DLbl">
  <xsd:sequence>
    <xsd:element name="idx" type="CT_UnsignedInt"/>
    <xsd:choice>
      <xsd:element name="delete" type="CT_Boolean"/>
      <xsd:group   ref="Group_DLbl"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="Group_DLbls">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="numFmt"          type="CT_NumFmt"           minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"            type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="dLblPos"         type="CT_DLblPos"          minOccurs="0"/>
    <xsd:element name="showLegendKey"   type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="showVal"         type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="showCatName"     type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="showSerName"     type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="showPercent"     type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="showBubbleSize"  type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="separator"       type="xsd:string"          minOccurs="0"/>
    <xsd:element name="showLeaderLines" type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="leaderLines"     type="CT_ChartLines"       minOccurs="0"/>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_Style">
  <xsd:attribute name="val" type="ST_Style" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_Style">
  <xsd:restriction base="xsd:unsignedByte">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="48"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_Legend">
  <xsd:sequence>
```

```
    <xsd:element name="legendPos"   type="CT_LegendPos"         minOccurs="0"/>
    <xsd:element name="legendEntry" type="CT_LegendEntry"       minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="layout"      type="CT_Layout"            minOccurs="0"/>
    <xsd:element name="overlay"     type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="spPr"        type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"        type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_LegendPos">
  <xsd:attribute name="val" type="ST_LegendPos" default="r"/>
</xsd:complexType>

<xsd:simpleType name="ST_LegendPos">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="b"/>
    <xsd:enumeration value="tr"/>
    <xsd:enumeration value="l"/>
    <xsd:enumeration value="r"/>
    <xsd:enumeration value="t"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_DLblPos">
  <xsd:attribute name="val" type="ST_DLblPos" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_DLblPos">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="bestFit"/>
    <xsd:enumeration value="b"/>
    <xsd:enumeration value="ctr"/>
    <xsd:enumeration value="inBase"/>
    <xsd:enumeration value="inEnd"/>
    <xsd:enumeration value="l"/>
    <xsd:enumeration value="outEnd"/>
    <xsd:enumeration value="r"/>
    <xsd:enumeration value="t"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_LblOffset">
  <xsd:attribute name="val" type="ST_LblOffset" default="100%"/>
</xsd:complexType>

<xsd:simpleType name="ST_LblOffset">
  <xsd:union memberTypes="ST_LblOffsetPercent ST_LblOffsetUShort"/>
</xsd:simpleType>

<xsd:simpleType name="ST_LblOffsetUShort">
  <xsd:restriction base="xsd:unsignedShort">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="1000"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="ST_LblOffsetPercent">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="0*(([0-9])|([1-9][0-9])|([1-9][0-9][0-9])|1000)%"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_Overlap">
  <xsd:attribute name="val" type="ST_Overlap" default="0%"/>
</xsd:complexType>

<xsd:simpleType name="ST_Overlap">
  <xsd:union memberTypes="ST_OverlapPercent ST_OverlapByte"/>
</xsd:simpleType>

<xsd:simpleType name="ST_OverlapPercent">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="(-?0*(([0-9])|([1-9][0-9])|100))%"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_OverlapByte">
  <xsd:restriction base="xsd:byte">
    <xsd:minInclusive value="-100"/>
    <xsd:maxInclusive value="100"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_Layout">
  <xsd:sequence>
    <xsd:element name="manualLayout" type="CT_ManualLayout"  minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ManualLayout">
  <xsd:sequence>
    <xsd:element name="layoutTarget" type="CT_LayoutTarget"  minOccurs="0"/>
    <xsd:element name="xMode"        type="CT_LayoutMode"    minOccurs="0"/>
    <xsd:element name="yMode"        type="CT_LayoutMode"    minOccurs="0"/>
    <xsd:element name="wMode"        type="CT_LayoutMode"    minOccurs="0"/>
    <xsd:element name="hMode"        type="CT_LayoutMode"    minOccurs="0"/>
    <xsd:element name="x"            type="CT_Double"        minOccurs="0"/>
    <xsd:element name="y"            type="CT_Double"        minOccurs="0"/>
    <xsd:element name="w"            type="CT_Double"        minOccurs="0"/>
    <xsd:element name="h"            type="CT_Double"        minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_LayoutMode">
  <xsd:attribute name="val" type="ST_LayoutMode" default="factor"/>
</xsd:complexType>

<xsd:simpleType name="ST_LayoutMode">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="edge"/>
    <xsd:enumeration value="factor"/>
  </xsd:restriction>
```

```
</xsd:simpleType>
```

Area Chart

The area chart is similar to a stacked line chart where the area between the lines is filled in.

XML specimens

XML for default Area chart:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<c:chartSpace
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <c:date1904 val="0"/>
  <c:lang val="en-US"/>
  <c:roundedCorners val="0"/>
  <mc:AlternateContent xmlns:mc="http://schemas.openxmlformats.org/markup-
↪compatibility/2006">
    <mc:Choice xmlns:c14="http://schemas.microsoft.com/office/drawing/2007/8/2/chart"␣
↪Requires="c14">
      <c14:style val="118"/>
    </mc:Choice>
    <mc:Fallback>
      <c:style val="18"/>
    </mc:Fallback>
  </mc:AlternateContent>
  <c:chart>
    <c:autoTitleDeleted val="0"/>
    <c:plotArea>
      <c:layout/>
      <c:areaChart>
        <c:grouping val="standard"/>
        <c:varyColors val="0"/>
        <c:ser>
          <c:idx val="0"/>
          <c:order val="0"/>
          <c:tx>
            <c:strRef>
              <c:f>Sheet1!$B$1</c:f>
              <c:strCache>
                <c:ptCount val="1"/>
                <c:pt idx="0">
                  <c:v>Series 1</c:v>
                </c:pt>
              </c:strCache>
            </c:strRef>
          </c:tx>
          <c:cat>
            <c:numRef>
              <c:f>Sheet1!$A$2:$A$6</c:f>
              <c:numCache>
                <c:formatCode>m/d/yy</c:formatCode>
```

```
            <c:ptCount val="5"/>
            <c:pt idx="0">
              <c:v>37261.0</c:v>
            </c:pt>
            <c:pt idx="1">
              <c:v>37262.0</c:v>
            </c:pt>
            <c:pt idx="2">
              <c:v>37263.0</c:v>
            </c:pt>
            <c:pt idx="3">
              <c:v>37264.0</c:v>
            </c:pt>
            <c:pt idx="4">
              <c:v>37265.0</c:v>
            </c:pt>
          </c:numCache>
        </c:numRef>
      </c:cat>
      <c:val>
        <c:numRef>
          <c:f>Sheet1!$B$2:$B$6</c:f>
          <c:numCache>
            <c:formatCode>General</c:formatCode>
            <c:ptCount val="5"/>
            <c:pt idx="0">
              <c:v>32.0</c:v>
            </c:pt>
            <c:pt idx="1">
              <c:v>32.0</c:v>
            </c:pt>
            <c:pt idx="2">
              <c:v>28.0</c:v>
            </c:pt>
            <c:pt idx="3">
              <c:v>12.0</c:v>
            </c:pt>
            <c:pt idx="4">
              <c:v>15.0</c:v>
            </c:pt>
          </c:numCache>
        </c:numRef>
      </c:val>
    </c:ser>
    <c:ser>
      <c:idx val="1"/>
      <c:order val="1"/>
      <c:tx>
        <c:strRef>
          <c:f>Sheet1!$C$1</c:f>
          <c:strCache>
            <c:ptCount val="1"/>
            <c:pt idx="0">
              <c:v>Series 2</c:v>
            </c:pt>
          </c:strCache>
        </c:strRef>
      </c:tx>
```

```
          <c:cat>
            <c:numRef>
              <c:f>Sheet1!$A$2:$A$6</c:f>
              <c:numCache>
                <c:formatCode>m/d/yy</c:formatCode>
                <c:ptCount val="5"/>
                <c:pt idx="0">
                  <c:v>37261.0</c:v>
                </c:pt>
                <c:pt idx="1">
                  <c:v>37262.0</c:v>
                </c:pt>
                <c:pt idx="2">
                  <c:v>37263.0</c:v>
                </c:pt>
                <c:pt idx="3">
                  <c:v>37264.0</c:v>
                </c:pt>
                <c:pt idx="4">
                  <c:v>37265.0</c:v>
                </c:pt>
              </c:numCache>
            </c:numRef>
          </c:cat>
          <c:val>
            <c:numRef>
              <c:f>Sheet1!$C$2:$C$6</c:f>
              <c:numCache>
                <c:formatCode>General</c:formatCode>
                <c:ptCount val="5"/>
                <c:pt idx="0">
                  <c:v>12.0</c:v>
                </c:pt>
                <c:pt idx="1">
                  <c:v>12.0</c:v>
                </c:pt>
                <c:pt idx="2">
                  <c:v>12.0</c:v>
                </c:pt>
                <c:pt idx="3">
                  <c:v>21.0</c:v>
                </c:pt>
                <c:pt idx="4">
                  <c:v>28.0</c:v>
                </c:pt>
              </c:numCache>
            </c:numRef>
          </c:val>
        </c:ser>
        <c:dLbls>
          <c:showLegendKey val="0"/>
          <c:showVal val="0"/>
          <c:showCatName val="0"/>
          <c:showSerName val="0"/>
          <c:showPercent val="0"/>
          <c:showBubbleSize val="0"/>
        </c:dLbls>
        <c:axId val="-2088330696"/>
```

```
        <c:axId val="-2081266648"/>
      </c:areaChart>
      <c:dateAx>
        <c:axId val="-2088330696"/>
        <c:scaling>
          <c:orientation val="minMax"/>
        </c:scaling>
        <c:delete val="0"/>
        <c:axPos val="b"/>
        <c:numFmt formatCode="m/d/yy" sourceLinked="1"/>
        <c:majorTickMark val="out"/>
        <c:minorTickMark val="none"/>
        <c:tickLblPos val="nextTo"/>
        <c:crossAx val="-2081266648"/>
        <c:crosses val="autoZero"/>
        <c:auto val="1"/>
        <c:lblOffset val="100"/>
        <c:baseTimeUnit val="days"/>
      </c:dateAx>
      <c:valAx>
        <c:axId val="-2081266648"/>
        <c:scaling>
          <c:orientation val="minMax"/>
        </c:scaling>
        <c:delete val="0"/>
        <c:axPos val="l"/>
        <c:majorGridlines/>
        <c:numFmt formatCode="General" sourceLinked="1"/>
        <c:majorTickMark val="out"/>
        <c:minorTickMark val="none"/>
        <c:tickLblPos val="nextTo"/>
        <c:crossAx val="-2088330696"/>
        <c:crosses val="autoZero"/>
        <c:crossBetween val="midCat"/>
      </c:valAx>
    </c:plotArea>
    <c:legend>
      <c:legendPos val="r"/>
      <c:layout/>
      <c:overlay val="0"/>
    </c:legend>
    <c:plotVisOnly val="1"/>
    <c:dispBlanksAs val="zero"/>
    <c:showDLblsOverMax val="0"/>
  </c:chart>
  <c:txPr>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr>
        <a:defRPr sz="1800"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:externalData r:id="rId1">
    <c:autoUpdate val="0"/>
  </c:externalData>
```

```
</c:chartSpace>
```

Related Schema Definitions

```
<xsd:complexType name="CT_AreaChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="grouping"   type="CT_Grouping"      minOccurs="0"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_AreaSer"       minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Grouping">
  <xsd:attribute name="val" type="ST_Grouping" default="standard"/>
</xsd:complexType>

<xsd:simpleType name="ST_Grouping">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="percentStacked"/>
    <xsd:enumeration value="standard"/>
    <xsd:enumeration value="stacked"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_AreaSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"             type="CT_UnsignedInt"/>
    <xsd:element name="order"           type="CT_UnsignedInt"/>
    <xsd:element name="tx"              type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="pictureOptions"  type="CT_PictureOptions"    minOccurs="0"/>
    <xsd:element name="dPt"             type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"           type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"       type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"         type="CT_ErrBars"           minOccurs="0"␣
↪maxOccurs="2"/>
    <xsd:element name="cat"             type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"             type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="extLst"          type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ChartLines">
  <xsd:sequence>
    <xsd:element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

Doughnut Chart

A doughnut chart is similar in many ways to a pie chart, except there is a "hole" in the middle. It can accept multiple series, which appear as concentric "rings".

XML specimens

XML for default doughnut chart:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<c:chartSpace
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <c:date1904 val="0"/>
  <c:lang val="en-US"/>
  <c:roundedCorners val="0"/>
  <mc:AlternateContent xmlns:mc="http://schemas.openxmlformats.org/markup-
↪compatibility/2006">
    <mc:Choice xmlns:c14="http://schemas.microsoft.com/office/drawing/2007/8/2/chart"␣
↪Requires="c14">
      <c14:style val="118"/>
    </mc:Choice>
    <mc:Fallback>
      <c:style val="18"/>
    </mc:Fallback>
  </mc:AlternateContent>
  <c:chart>
    <c:autoTitleDeleted val="0"/>
    <c:plotArea>
      <c:layout/>
      <c:doughnutChart>
        <c:varyColors val="1"/>
        <c:ser>
          <c:idx val="0"/>
          <c:order val="0"/>
          <c:tx>
            <c:strRef>
              <c:f>Sheet1!$B$1</c:f>
              <c:strCache>
                <c:ptCount val="1"/>
                <c:pt idx="0">
                  <c:v>Sales</c:v>
                </c:pt>
              </c:strCache>
            </c:strRef>
          </c:tx>
          <c:cat>
            <c:strRef>
              <c:f>Sheet1!$A$2:$A$5</c:f>
              <c:strCache>
                <c:ptCount val="4"/>
                <c:pt idx="0">
                  <c:v>1st Qtr</c:v>
                </c:pt>
                <c:pt idx="1">
```

```
                <c:v>2nd Qtr</c:v>
              </c:pt>
              <c:pt idx="2">
                <c:v>3rd Qtr</c:v>
              </c:pt>
              <c:pt idx="3">
                <c:v>4th Qtr</c:v>
              </c:pt>
            </c:strCache>
          </c:strRef>
        </c:cat>
        <c:val>
          <c:numRef>
            <c:f>Sheet1!$B$2:$B$5</c:f>
            <c:numCache>
              <c:formatCode>General</c:formatCode>
              <c:ptCount val="4"/>
              <c:pt idx="0">
                <c:v>8.2</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>3.2</c:v>
              </c:pt>
              <c:pt idx="2">
                <c:v>1.4</c:v>
              </c:pt>
              <c:pt idx="3">
                <c:v>1.2</c:v>
              </c:pt>
            </c:numCache>
          </c:numRef>
        </c:val>
      </c:ser>
      <c:ser>
        <c:idx val="1"/>
        <c:order val="1"/>
        <c:tx>
          <c:strRef>
            <c:f>Sheet1!$C$1</c:f>
            <c:strCache>
              <c:ptCount val="1"/>
              <c:pt idx="0">
                <c:v>Revenue</c:v>
              </c:pt>
            </c:strCache>
          </c:strRef>
        </c:tx>
        <c:cat>
          <c:strRef>
            <c:f>Sheet1!$A$2:$A$5</c:f>
            <c:strCache>
              <c:ptCount val="4"/>
              <c:pt idx="0">
                <c:v>1st Qtr</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>2nd Qtr</c:v>
              </c:pt>
```

```
              <c:pt idx="2">
                <c:v>3rd Qtr</c:v>
              </c:pt>
              <c:pt idx="3">
                <c:v>4th Qtr</c:v>
              </c:pt>
            </c:strCache>
          </c:strRef>
        </c:cat>
        <c:val>
          <c:numRef>
            <c:f>Sheet1!$C$2:$C$5</c:f>
            <c:numCache>
              <c:formatCode>General</c:formatCode>
              <c:ptCount val="4"/>
              <c:pt idx="0">
                <c:v>1000.0</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>2000.0</c:v>
              </c:pt>
              <c:pt idx="2">
                <c:v>3000.0</c:v>
              </c:pt>
              <c:pt idx="3">
                <c:v>4000.0</c:v>
              </c:pt>
            </c:numCache>
          </c:numRef>
        </c:val>
      </c:ser>
      <c:dLbls>
        <c:showLegendKey val="0"/>
        <c:showVal val="0"/>
        <c:showCatName val="0"/>
        <c:showSerName val="0"/>
        <c:showPercent val="0"/>
        <c:showBubbleSize val="0"/>
        <c:showLeaderLines val="1"/>
      </c:dLbls>
      <c:firstSliceAng val="0"/>
      <c:holeSize val="50"/>
    </c:doughnutChart>
  </c:plotArea>
  <c:legend>
    <c:legendPos val="r"/>
    <c:layout/>
    <c:overlay val="0"/>
  </c:legend>
  <c:plotVisOnly val="1"/>
  <c:dispBlanksAs val="gap"/>
  <c:showDLblsOverMax val="0"/>
</c:chart>
<c:txPr>
  <a:bodyPr/>
  <a:lstStyle/>
  <a:p>
    <a:pPr>
```

```
        <a:defRPr sz="1800"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:externalData r:id="rId1">
    <c:autoUpdate val="0"/>
  </c:externalData>
</c:chartSpace>
```

Related Schema Definitions

```
<xsd:complexType name="CT_DoughnutChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="varyColors"    type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"           type="CT_PieSer"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"         type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="firstSliceAng" type="CT_FirstSliceAng" minOccurs="0"/>
    <xsd:element name="holeSize"      type="CT_HoleSize"      minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_PieSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"       type="CT_UnsignedInt"/>
    <xsd:element name="order"     type="CT_UnsignedInt"/>
    <xsd:element name="tx"        type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"      type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="explosion" type="CT_UnsignedInt"       minOccurs="0"/>
    <xsd:element name="dPt"       type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"     type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="cat"       type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"       type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Chart - Points

A chart may be understood as a graph of a function, formed by rendering a finite number of ordered pairs (e.g. (x, y)) in one of several graphical formats such as bar, pie, or bubble.

A functon can be expressed as:

```
y = f(x)
```

where *x* is the input to the function and *y* is the output. *x* is known more formally as the function's *argument*, and *y* as its *value*. The set of possible input (x) values is the *domain* of the function. The set of possible output (y) values is the function's *range*.

In PowerPoint, the function is perhaps rarely mathematical; perhaps more commonly, the x and y values are correlated

observations, facts to be communicated graphically as a set. Still these terms borrowed from mathematics provide convenient language for describing aspects of PowerPoint charts.

PowerPoint charts can be divided broadly into *category* types, which take a discrete argument, and *XY* (aka. scatter) types which take a continuous argument. For category types, the argument is one of a discrete set of categories, expressed as a label, such as "Blue", "Red", and "Green". The argument for an XY type is a real number. In both cases, the *value* of each data point is a real number.

The argument is commonly associated with the horizontal axis of the chart and the value with the vertical axis; these would be the X and Y axis respectively on an XY chart. However the 2D orientation of the axes is reversed in some char types (bar charts in particular) and takes other forms in chart types such as radar and doughnut. In the MS API, the axis associated with the argument is known as the *category axis* and the other is the *value axis*. This terminology is used across all chart types (including XY) as a matter of convention.

Series

A chart can have one or more series. A series is a set of ordered pairs that are related in some way and meant to be distinguished graphically on the chart, often so they can be compared. For example, one series could represent Q1 financial results, depicted as a blue line, and a second series represent Q2 financial results with a red line.

In a standard chart, all series share the same domain (e.g. the same X axis). In a category chart, all series also share the same set of categories. In an XY chart, the domain values are continuous; so in general, each data point in each XY series will in have a distinct argument (x value).

In the MS API, there are multiple members on Series related to data points:

- Values
- XValues
- Points

Note that *Categories* is not in this set. `CategoryCollection` is a member of `ChartGroup` (named Plot in *python-pptx*) in the MS API.

Values is the Excel range address at which the series values are stored. A set of constant values can be assigned in the MS API but this is not supported in *python-pptx*.

XValues is the Excel range address at which the series arguments are stored when the chart is one of the XY types. X values replace categories for an XY chart.

Points is a sequence of Point objects, each of which provides access to the formatting of the graphical representation of the data point. Notably, it does not allow changing the category, x or y value, or size (for a bubble chart).

Every chart type supported by MS Office charts has the concept of data points.

Informally, a data point is an atomic element of the content depicted by the chart.

MS Office charts do not have a firm distinction for data points.

In all cases, points belong to a *series*.

There is some distinction between a data point for a category chart and one for an XY (scatter) chart.

The API

Functions

```
data_point_count = min(
    xVal.ptCount_val, yVal.ptCount_val, bubbleSize.ptCount_val
)

invariant(ptCount_val > max(pt.idx))

@property
def pt_v(self, parent, idx):
    pts = parent.xpath('.//c:pt[@val=%s]' % idx)
    if pts is None:
        return None
    pt = pts[0]
    return pt.v

points[i] ~= Point(xVal.pts[i], yVal.pts[i], bubbleSize.pts[i])
```

Point membership hypothesis:

- Visual alignment is not significant. A range may be anywhere in relation to the other data point ranges.
- Sequence is low row/col to high row/col in the Excel range, regardless of the "direction" in which the cells are selected. This seems to be enforced by a validation "correction" of the range string in the UI 'Select Data' dialog.
- The number (count) of x, y, or size values is the number of cells in the range. This corresponds directly to ptCount/@val for that data source sequence.
- The number of points in the series is the minimum of the x, y, and size value counts (ptCount).
- The ordered set of values for a point is formed by simple indexing within each value sequence. For example:

  ```
  point[3] = (x_values[3], y_values[3], sizes[3])
  ```

  When any value sequence in the set runs out of elements, no further points are formed.

  All values of each data source sequence are written to the XML; values are not truncated because they lack a counter part in one of the other sequences. Consequently, the ptCount values will not necessarily match.

Experiment (IronPython):

- Create a chart with three X-values and four Y-values. .XValues has three and .Values has four members. How many points are in Series.Points?

Observations:

- 4 X, 4 Y, 4 size – 4 points
- 3 X, 4 Y, 4 size – 3 points
- 4 X, 3 Y, 4 size – 3 points
- 4 X, 4 Y, 3 size – 3 points

Points with blank (y) values still count as a point. Points with blank (y) values still count as a point.

Explain how ...

Hypothesis: An x, y, or size value index always starts at zero, at the beginning of the range, and increments to ptCount-1.

Hypothesis: ptCount is always based on the number of cells in the Excel range, including blank cells at the start and end.

ptCount and pt behavior on range including blank cell at end.

xVal//ptCount can be different than yVal//ptCount

Related Schema Definitions

Point-related elements are stored under *c:ser*:

```
<xsd:complexType name="CT_ScatterStyle">
  <xsd:attribute name="val" type="ST_ScatterStyle" default="marker"/>
</xsd:complexType>

<xsd:complexType name="CT_ScatterSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"         type="CT_UnsignedInt"/>
    <xsd:element name="order"       type="CT_UnsignedInt"/>
    <xsd:element name="tx"          type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"        type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker"      type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"         type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"       type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"   type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"     type="CT_ErrBars"           minOccurs="0"␣
↪maxOccurs="2"/>
    <xsd:element name="xVal"        type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="yVal"        type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="smooth"      type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_AxDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="multiLvlStrRef" type="CT_MultiLvlStrRef"/>
      <xsd:element name="numRef"         type="CT_NumRef"/>
      <xsd:element name="numLit"         type="CT_NumData"/>
      <xsd:element name="strRef"         type="CT_StrRef"/>
      <xsd:element name="strLit"         type="CT_StrData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DLbls">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="dLbl"            type="CT_DLbl"             minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="numFmt"          type="CT_NumFmt"           minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"            type="a:CT_TextBody"       minOccurs="0"/>
    <xsd:element name="dLblPos"         type="CT_DLblPos"          minOccurs="0"/>
    <xsd:element name="showLegendKey"   type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="showVal"         type="CT_Boolean"          minOccurs="0"/>
```

```
    <xsd:element name="showCatName"     type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showSerName"     type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showPercent"     type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showBubbleSize"  type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="separator"       type="xsd:string"           minOccurs="0"/>
    <xsd:element name="showLeaderLines" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="leaderLines"     type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="extLst"          type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DLbl">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"           type="CT_UnsignedInt"/>
    <xsd:element name="layout"        type="CT_Layout"            minOccurs="0"/>
    <xsd:element name="tx"            type="CT_Tx"                minOccurs="0"/>
    <xsd:element name="numFmt"        type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="spPr"          type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"          type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="dLblPos"       type="CT_DLblPos"           minOccurs="0"/>
    <xsd:element name="showLegendKey" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showVal"       type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showCatName"   type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showSerName"   type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showPercent"   type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="showBubbleSize" type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="separator"     type="xsd:string"           minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DPt">
  <xsd:sequence>
    <xsd:element name="idx"             type="CT_UnsignedInt"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="marker"          type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="bubble3D"        type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="explosion"       type="CT_UnsignedInt"       minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="pictureOptions"  type="CT_PictureOptions"    minOccurs="0"/>
    <xsd:element name="extLst"          type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Marker">
  <xsd:sequence>
    <xsd:element name="symbol" type="CT_MarkerStyle"       minOccurs="0"/>
    <xsd:element name="size"   type="CT_MarkerSize"        minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_MarkerStyle">
  <xsd:attribute name="val" type="ST_MarkerStyle" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_MarkerSize">
```

```
  <xsd:attribute name="val" type="ST_MarkerSize" default="5"/>
</xsd:complexType>

<xsd:complexType name="CT_NumData">
  <xsd:sequence>
    <xsd:element name="formatCode" type="s:ST_Xstring"     minOccurs="0"/>
    <xsd:element name="ptCount"    type="CT_UnsignedInt"   minOccurs="0"/>
    <xsd:element name="pt"         type="CT_NumVal"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="numRef" type="CT_NumRef"/>
      <xsd:element name="numLit" type="CT_NumData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumRef">
  <xsd:sequence>
    <xsd:element name="f"        type="xsd:string"/>
    <xsd:element name="numCache" type="CT_NumData"       minOccurs="0"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ST_MarkerSize">
  <xsd:restriction base="xsd:unsignedByte">
    <xsd:minInclusive value="2"/>
    <xsd:maxInclusive value="72"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_MarkerStyle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="circle"/>
    <xsd:enumeration value="dash"/>
    <xsd:enumeration value="diamond"/>
    <xsd:enumeration value="dot"/>
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="picture"/>
    <xsd:enumeration value="plus"/>
    <xsd:enumeration value="square"/>
    <xsd:enumeration value="star"/>
    <xsd:enumeration value="triangle"/>
    <xsd:enumeration value="x"/>
    <xsd:enumeration value="auto"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_ScatterStyle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="line"/>
```

```
    <xsd:enumeration value="lineMarker"/>
    <xsd:enumeration value="marker"/>
    <xsd:enumeration value="smooth"/>
    <xsd:enumeration value="smoothMarker"/>
  </xsd:restriction>
</xsd:simpleType>
```

Radar Chart

A radar chart is essentially a line chart wrapped around on itself.

It's worth a try to see if inheriting from LineChart would work.

PowerPoint UI

- Insert > Chart ... > Other > Radar
- Default Radar uses 5 categories, 2 series, points connected with lines, no data point markers. Other radar options can add markers or fill.
- Layout seems an awful lot like a line chart
- Supports data labels, but only with data point value, not custom text (from UI).

XML semantics

- There is one *c:ser* element for each series. Each *c:ser* element contains both the categories and the values (even though that is often redundant).
- ? What happens if the category items in a radar chart don't match? Inside are *c:xVal*, *c:yVal*, and *c:bubbleSize*, each containing the set of points for that "series".

XML specimen

XML for default radar chart (simplified):

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<c:chartSpace
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <c:chart>
    <c:autoTitleDeleted val="0"/>
    <c:plotArea>
      <c:layout/>
      <c:radarChart>
        <c:radarStyle val="marker"/>
        <c:varyColors val="0"/>
        <c:ser>
          <c:idx val="0"/>
          <c:order val="0"/>
          <c:tx>
```

```
            <c:strRef>
              <c:f>Sheet1!$B$1</c:f>
              <c:strCache>
                <c:ptCount val="1"/>
                <c:pt idx="0">
                  <c:v>Series 1</c:v>
                </c:pt>
              </c:strCache>
            </c:strRef>
          </c:tx>
          <c:marker>
            <c:symbol val="none"/>
          </c:marker>
          <c:cat>
            <c:numRef>
              <c:f>Sheet1!$A$2:$A$6</c:f>
              <c:numCache>
                <c:formatCode>m/d/yy</c:formatCode>
                <c:ptCount val="5"/>
                <c:pt idx="0">
                  <c:v>37261.0</c:v>
                </c:pt>
                <c:pt idx="1">
                  <c:v>37262.0</c:v>
                </c:pt>
                <c:pt idx="2">
                  <c:v>37263.0</c:v>
                </c:pt>
                <c:pt idx="3">
                  <c:v>37264.0</c:v>
                </c:pt>
                <c:pt idx="4">
                  <c:v>37265.0</c:v>
                </c:pt>
              </c:numCache>
            </c:numRef>
          </c:cat>
          <c:val>
            <c:numRef>
              <c:f>Sheet1!$B$2:$B$6</c:f>
              <c:numCache>
                <c:formatCode>General</c:formatCode>
                <c:ptCount val="5"/>
                <c:pt idx="0">
                  <c:v>32.0</c:v>
                </c:pt>
                <c:pt idx="1">
                  <c:v>32.0</c:v>
                </c:pt>
                <c:pt idx="2">
                  <c:v>28.0</c:v>
                </c:pt>
                <c:pt idx="3">
                  <c:v>12.0</c:v>
                </c:pt>
                <c:pt idx="4">
                  <c:v>15.0</c:v>
                </c:pt>
```

```
            </c:numCache>
          </c:numRef>
        </c:val>
      </c:ser>
      <c:ser>
        <c:idx val="1"/>
        <c:order val="1"/>
        <c:tx>
          <c:strRef>
            <c:f>Sheet1!$C$1</c:f>
            <c:strCache>
              <c:ptCount val="1"/>
              <c:pt idx="0">
                <c:v>Series 2</c:v>
              </c:pt>
            </c:strCache>
          </c:strRef>
        </c:tx>
        <c:marker>
          <c:symbol val="none"/>
        </c:marker>
        <c:cat>
          <c:numRef>
            <c:f>Sheet1!$A$2:$A$6</c:f>
            <c:numCache>
              <c:formatCode>m/d/yy</c:formatCode>
              <c:ptCount val="5"/>
              <c:pt idx="0">
                <c:v>37261.0</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>37262.0</c:v>
              </c:pt>
              <c:pt idx="2">
                <c:v>37263.0</c:v>
              </c:pt>
              <c:pt idx="3">
                <c:v>37264.0</c:v>
              </c:pt>
              <c:pt idx="4">
                <c:v>37265.0</c:v>
              </c:pt>
            </c:numCache>
          </c:numRef>
        </c:cat>
        <c:val>
          <c:numRef>
            <c:f>Sheet1!$C$2:$C$6</c:f>
            <c:numCache>
              <c:formatCode>General</c:formatCode>
              <c:ptCount val="5"/>
              <c:pt idx="0">
                <c:v>12.0</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>12.0</c:v>
              </c:pt>
              <c:pt idx="2">
```

```
            <c:v>12.0</c:v>
           </c:pt>
           <c:pt idx="3">
            <c:v>21.0</c:v>
           </c:pt>
           <c:pt idx="4">
            <c:v>28.0</c:v>
           </c:pt>
          </c:numCache>
         </c:numRef>
        </c:val>
       </c:ser>
       <c:dLbls>
        <c:showLegendKey val="0"/>
        <c:showVal val="0"/>
        <c:showCatName val="0"/>
        <c:showSerName val="0"/>
        <c:showPercent val="0"/>
        <c:showBubbleSize val="0"/>
       </c:dLbls>
       <c:axId val="2079682968"/>
       <c:axId val="2079686056"/>
      </c:radarChart>
      <c:catAx>
       <c:axId val="2079682968"/>
       <c:scaling>
        <c:orientation val="minMax"/>
       </c:scaling>
       <c:delete val="0"/>
       <c:axPos val="b"/>
       <c:majorGridlines/>
       <c:numFmt formatCode="m/d/yy" sourceLinked="1"/>
       <c:majorTickMark val="out"/>
       <c:minorTickMark val="none"/>
       <c:tickLblPos val="nextTo"/>
       <c:crossAx val="2079686056"/>
       <c:crosses val="autoZero"/>
       <c:auto val="1"/>
       <c:lblAlgn val="ctr"/>
       <c:lblOffset val="100"/>
       <c:noMultiLvlLbl val="0"/>
      </c:catAx>
      <c:valAx>
       <c:axId val="2079686056"/>
       <c:scaling>
        <c:orientation val="minMax"/>
       </c:scaling>
       <c:delete val="0"/>
       <c:axPos val="l"/>
       <c:majorGridlines/>
       <c:numFmt formatCode="General" sourceLinked="1"/>
       <c:majorTickMark val="cross"/>
       <c:minorTickMark val="none"/>
       <c:tickLblPos val="nextTo"/>
       <c:crossAx val="2079682968"/>
       <c:crosses val="autoZero"/>
       <c:crossBetween val="between"/>
      </c:valAx>
```

```
    </c:plotArea>
    <c:legend>
      <c:legendPos val="r"/>
      <c:layout/>
      <c:overlay val="0"/>
    </c:legend>
    <c:plotVisOnly val="1"/>
    <c:dispBlanksAs val="gap"/>
    <c:showDLblsOverMax val="0"/>
  </c:chart>
  <c:txPr>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr>
        <a:defRPr sz="1800"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:externalData r:id="rId1">
    <c:autoUpdate val="0"/>
  </c:externalData>
</c:chartSpace>
```

MS API Protocol

Make a radar chart:

```
Public Sub MakeGraph()

    Dim title       As String = "Title"
    Dim x_label     As String = "X Axis Label"
    Dim y_label     As String = "Y Axis Label"
    Dim new_chart   As Chart

    ' Make the chart.
    Set new_chart = Charts.Add()
    ActiveChart.ChartType = xlRadar
    ActiveChart.SetSourceData _
        Source:=sheet.Range("A1:A" & UBound(values)), _
        PlotBy:=xlColumns

    ' Set the chart's title abd axis labels.
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = title

        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = x_label

        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = y_label
    End With
End Sub
```

Related Schema Definitions

Radar chart elements:

```
<xsd:complexType name="CT_RadarChart">
  <xsd:sequence>
    <xsd:element name="radarStyle" type="CT_RadarStyle"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_RadarSer"      minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_RadarStyle">
  <xsd:attribute name="val" type="ST_RadarStyle" default="standard"/>
</xsd:complexType>

<xsd:complexType name="CT_RadarSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"    type="CT_UnsignedInt"/>
    <xsd:element name="order"  type="CT_UnsignedInt"/>
    <xsd:element name="tx"     type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker" type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"    type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"  type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="cat"    type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"    type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SerTx">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="strRef" type="CT_StrRef"/>
      <xsd:element name="v"      type="s:ST_Xstring"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Marker">
  <xsd:sequence>
    <xsd:element name="symbol" type="CT_MarkerStyle"       minOccurs="0"/>
    <xsd:element name="size"   type="CT_MarkerSize"        minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DPt">
  <xsd:sequence>
    <xsd:element name="idx"             type="CT_UnsignedInt"/>
```

```
    <xsd:element name="invertIfNegative" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="marker"           type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="bubble3D"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="explosion"        type="CT_UnsignedInt"       minOccurs="0"/>
    <xsd:element name="spPr"             type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="pictureOptions"   type="CT_PictureOptions"    minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ST_RadarStyle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="standard"/>
    <xsd:enumeration value="marker"/>
    <xsd:enumeration value="filled"/>
  </xsd:restriction>
</xsd:simpleType>
```

Bubble Chart

A bubble chart is an extension of an X-Y or scatter chart. Containing an extra (third) series, the size of the bubble is proportional to the value in the third series.

The bubble is proportioned to its value by one of two methods selected by the client:

- Bubble *width* is proportional to value (linear)
- Bubble *area* is proportional to value (quadratic)

By default, the scale of the bubbles is determined by setting the largest bubble equal to a "default bubble size" equal to roughly 25% of the height or width of the chart area, whichever is less. The default bubble size can be scaled using a property setting to obtain proportionally larger or smaller bubbles.

PowerPoint UI

To create a bubble chart by hand in PowerPoint:

1. Carts ribbon > Other > Bubble

A three-row, three-column Excel worksheet opens and the default chart appears.

XML semantics

- I think this is going to get into blank cells if multiple series are required. Might be worth checking out how using NA() possibly differs as to how it appears in the XML.
- There is a single *c:ser* element. Inside are *c:xVal*, *c:yVal*, and *c:bubbleSize*, each containing the set of points for that "series".

XML specimen

XML for default bubble chart:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<c:chartSpace
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <c:chart>
    <c:plotArea>
      <c:bubbleChart>
        <c:varyColors val="0"/>
        <c:ser>
          <c:idx val="0"/>
          <c:order val="0"/>
          <c:tx>
            <c:strRef>
              <c:f>Sheet1!$B$1</c:f>
              <c:strCache>
                <c:ptCount val="1"/>
                <c:pt idx="0">
                  <c:v>Y-Value 1</c:v>
                </c:pt>
              </c:strCache>
            </c:strRef>
          </c:tx>
          <c:invertIfNegative val="0"/>
          <c:xVal>
            <c:numRef>
              <c:f>Sheet1!$A$2:$A$4</c:f>
              <c:numCache>
                <c:formatCode>General</c:formatCode>
                <c:ptCount val="3"/>
                <c:pt idx="0">
                  <c:v>0.7</c:v>
                </c:pt>
                <c:pt idx="1">
                  <c:v>1.8</c:v>
                </c:pt>
                <c:pt idx="2">
                  <c:v>2.6</c:v>
                </c:pt>
              </c:numCache>
            </c:numRef>
          </c:xVal>
          <c:yVal>
            <c:numRef>
              <c:f>Sheet1!$B$2:$B$4</c:f>
              <c:numCache>
                <c:formatCode>General</c:formatCode>
                <c:ptCount val="3"/>
                <c:pt idx="0">
                  <c:v>2.7</c:v>
                </c:pt>
                <c:pt idx="1">
                  <c:v>3.2</c:v>
                </c:pt>
                <c:pt idx="2">
                  <c:v>0.8</c:v>
                </c:pt>
```

```
            </c:numCache>
          </c:numRef>
        </c:yVal>
        <c:bubbleSize>
          <c:numRef>
            <c:f>Sheet1!$C$2:$C$4</c:f>
            <c:numCache>
              <c:formatCode>General</c:formatCode>
              <c:ptCount val="3"/>
              <c:pt idx="0">
                <c:v>10.0</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>4.0</c:v>
              </c:pt>
              <c:pt idx="2">
                <c:v>8.0</c:v>
              </c:pt>
            </c:numCache>
          </c:numRef>
        </c:bubbleSize>
        <c:bubble3D val="0"/>
      </c:ser>
      <c:dLbls>
        <c:showLegendKey val="0"/>
        <c:showVal val="0"/>
        <c:showCatName val="0"/>
        <c:showSerName val="0"/>
        <c:showPercent val="0"/>
        <c:showBubbleSize val="0"/>
      </c:dLbls>
      <c:bubbleScale val="100"/>
      <c:showNegBubbles val="0"/>
      <c:axId val="2110171512"/>
      <c:axId val="2110299944"/>
    </c:bubbleChart>
    <c:valAx>
      <c:axId val="2110171512"/>
      <c:scaling>
        <c:orientation val="minMax"/>
      </c:scaling>
      <c:delete val="0"/>
      <c:axPos val="b"/>
      <c:numFmt formatCode="General" sourceLinked="1"/>
      <c:majorTickMark val="out"/>
      <c:minorTickMark val="none"/>
      <c:tickLblPos val="nextTo"/>
      <c:crossAx val="2110299944"/>
      <c:crosses val="autoZero"/>
      <c:crossBetween val="midCat"/>
    </c:valAx>
    <c:valAx>
      <c:axId val="2110299944"/>
      <c:scaling>
        <c:orientation val="minMax"/>
      </c:scaling>
      <c:delete val="0"/>
      <c:axPos val="l"/>
```

```
        <c:majorGridlines/>
        <c:numFmt formatCode="General" sourceLinked="1"/>
        <c:majorTickMark val="out"/>
        <c:minorTickMark val="none"/>
        <c:tickLblPos val="nextTo"/>
        <c:crossAx val="2110171512"/>
        <c:crosses val="autoZero"/>
        <c:crossBetween val="midCat"/>
      </c:valAx>
    </c:plotArea>
    <c:legend>
      <c:legendPos val="r"/>
      <c:layout/>
      <c:overlay val="0"/>
    </c:legend>
    <c:plotVisOnly val="1"/>
    <c:dispBlanksAs val="gap"/>
    <c:showDLblsOverMax val="0"/>
  </c:chart>
  <c:txPr>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr>
        <a:defRPr sz="1800"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:externalData r:id="rId1">
    <c:autoUpdate val="0"/>
  </c:externalData>
</c:chartSpace>
```

MS API Protocol

Create (unconventional) multi-series bubble chart in Excel:

```
If (selection.Columns.Count <> 4 Or selection.Rows.Count < 3) Then
    MsgBox "Selection must have 4 columns and at least 2 rows"
    Exit Sub
End If

Dim bubbleChart As ChartObject
Set bubbleChart = ActiveSheet.ChartObjects.Add(
      Left:=selection.Left, Width:=600, Top:=selection.Top, Height:=400
)
bubbleChart.chart.ChartType = xlBubble
Dim r As Integer
For r = 2 To selection.Rows.Count
    With bubbleChart.chart.SeriesCollection.NewSeries
        .Name = "=" & selection.Cells(r, 1).Address(External:=True)
        .XValues = selection.Cells(r, 2).Address(External:=True)
        .Values = selection.Cells(r, 3).Address(External:=True)
        .BubbleSizes = selection.Cells(r, 4).Address(External:=True)
    End With
```

```
Next

bubbleChart.chart.SetElement (msoElementPrimaryCategoryAxisTitleAdjacentToAxis)
bubbleChart.chart.Axes(xlCategory, xlPrimary).AxisTitle.Text = "=" & selection.
↪Cells(1, 2).Address(External:=True)

bubbleChart.chart.SetElement (msoElementPrimaryValueAxisTitleRotated)
bubbleChart.chart.Axes(xlValue, xlPrimary).AxisTitle.Text = "=" & selection.Cells(1,␣
↪3).Address(External:=True)

bubbleChart.chart.SetElement (msoElementPrimaryCategoryGridLinesMajor)
bubbleChart.chart.Axes(xlCategory).MinimumScale = 0
```

Related Schema Definitions

Bubble chart elements:

```
<xsd:complexType name="CT_BubbleChart">
  <xsd:sequence>
    <xsd:element name="varyColors"     type="CT_Boolean"        minOccurs="0"/>
    <xsd:element name="ser"            type="CT_BubbleSer"      minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"          type="CT_DLbls"          minOccurs="0"/>
    <xsd:element name="bubble3D"       type="CT_Boolean"        minOccurs="0"/>
    <xsd:element name="bubbleScale"    type="CT_BubbleScale"    minOccurs="0"/>
    <xsd:element name="showNegBubbles" type="CT_Boolean"        minOccurs="0"/>
    <xsd:element name="sizeRepresents" type="CT_SizeRepresents" minOccurs="0"/>
    <xsd:element name="axId"           type="CT_UnsignedInt"    minOccurs="2"␣
↪maxOccurs="2"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"  minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_BubbleSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"              type="CT_UnsignedInt"/>
    <xsd:element name="order"            type="CT_UnsignedInt"/>
    <xsd:element name="tx"               type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"             type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="dPt"              type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"            type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"        type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"          type="CT_ErrBars"           minOccurs="0"␣
↪maxOccurs="2"/>
    <xsd:element name="xVal"             type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="yVal"             type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="bubbleSize"       type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="bubble3D"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_AxDataSource">
```

```
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="multiLvlStrRef" type="CT_MultiLvlStrRef"/>
      <xsd:element name="numRef"         type="CT_NumRef"/>
      <xsd:element name="numLit"         type="CT_NumData"/>
      <xsd:element name="strRef"         type="CT_StrRef"/>
      <xsd:element name="strLit"         type="CT_StrData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_BubbleScale">
  <xsd:attribute name="val" type="ST_BubbleScale" default="100%"/>
</xsd:complexType>

<xsd:complexType name="CT_NumData">
  <xsd:sequence>
    <xsd:element name="formatCode" type="s:ST_Xstring"     minOccurs="0"/>
    <xsd:element name="ptCount"    type="CT_UnsignedInt"   minOccurs="0"/>
    <xsd:element name="pt"         type="CT_NumVal"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="numRef" type="CT_NumRef"/>
      <xsd:element name="numLit" type="CT_NumData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumRef">
  <xsd:sequence>
    <xsd:element name="f"        type="xsd:string"/>
    <xsd:element name="numCache" type="CT_NumData"       minOccurs="0"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SizeRepresents">
  <xsd:attribute name="val" type="ST_SizeRepresents" default="area"/>
</xsd:complexType>

<xsd:complexType name="CT_Trendline">
  <xsd:sequence>
    <xsd:element name="name"          type="xsd:string"           minOccurs="0"/>
    <xsd:element name="spPr"          type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="trendlineType" type="CT_TrendlineType"/>
    <xsd:element name="order"         type="CT_Order"             minOccurs="0"/>
    <xsd:element name="period"        type="CT_Period"            minOccurs="0"/>
    <xsd:element name="forward"       type="CT_Double"            minOccurs="0"/>
    <xsd:element name="backward"      type="CT_Double"            minOccurs="0"/>
    <xsd:element name="intercept"     type="CT_Double"            minOccurs="0"/>
    <xsd:element name="dispRSqr"      type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="dispEq"        type="CT_Boolean"           minOccurs="0"/>
```

```
    <xsd:element name="trendlineLbl"  type="CT_TrendlineLbl"      minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ST_BubbleScale">
  <xsd:union memberTypes="ST_BubbleScalePercent ST_BubbleScaleUInt"/>
</xsd:simpleType>

<xsd:simpleType name="ST_BubbleScalePercent">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="0*(([0-9])|([1-9][0-9])|([1-2][0-9][0-9])|300)%"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_BubbleScaleUInt">
  <xsd:restriction base="xsd:unsignedInt">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="300"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_SizeRepresents">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="area"/>
    <xsd:enumeration value="w"/>
  </xsd:restriction>
</xsd:simpleType>
```

References

- https://blogs.msdn.microsoft.com/tomholl/2011/03/27/creating-multi-series-bubble-charts-in-excel/
- http://peltiertech.com/Excel/ChartsHowTo/HowToBubble.html
- http://peltiertech.com/Excel/Charts/ControlBubbleSizes.html

X-Y Chart

An X-Y chart, also known as a *scatter* chart, depending on the version of PowerPoint, is distinguished by having two value axes rather than one category axis and one value axis.

XML specimen

XML for default XY chart:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<c:chartSpace
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <c:chart>
    <c:autoTitleDeleted val="0"/>
```

```
    <c:plotArea>
      <c:layout/>
      <c:scatterChart>
        <c:scatterStyle val="lineMarker"/>
        <c:varyColors val="0"/>
        <c:ser>
          <c:idx val="0"/>
          <c:order val="0"/>
          <c:tx>
            <c:strRef>
              <c:f>Sheet1!$B$1</c:f>
              <c:strCache>
                <c:ptCount val="1"/>
                <c:pt idx="0">
                  <c:v>Red Bull</c:v>
                </c:pt>
              </c:strCache>
            </c:strRef>
          </c:tx>
          <c:spPr>
            <a:ln w="47625">
              <a:noFill/>
            </a:ln>
          </c:spPr>
          <c:xVal>
            <c:numRef>
              <c:f>Sheet1!$A$2:$A$7</c:f>
              <c:numCache>
                <c:formatCode>General</c:formatCode>
                <c:ptCount val="6"/>
                <c:pt idx="0">
                  <c:v>0.7</c:v>
                </c:pt>
                <c:pt idx="1">
                  <c:v>1.8</c:v>
                </c:pt>
                <c:pt idx="2">
                  <c:v>2.6</c:v>
                </c:pt>
                <c:pt idx="3">
                  <c:v>0.8</c:v>
                </c:pt>
                <c:pt idx="4">
                  <c:v>1.7</c:v>
                </c:pt>
                <c:pt idx="5">
                  <c:v>2.5</c:v>
                </c:pt>
              </c:numCache>
            </c:numRef>
          </c:xVal>
          <c:yVal>
            <c:numRef>
              <c:f>Sheet1!$B$2:$B$7</c:f>
              <c:numCache>
                <c:formatCode>General</c:formatCode>
                <c:ptCount val="6"/>
                <c:pt idx="0">
```

```
            <c:v>2.7</c:v>
           </c:pt>
           <c:pt idx="1">
            <c:v>3.2</c:v>
           </c:pt>
           <c:pt idx="2">
            <c:v>0.8</c:v>
           </c:pt>
          </c:numCache>
         </c:numRef>
        </c:yVal>
        <c:smooth val="0"/>
       </c:ser>
       <c:ser>
        <c:idx val="1"/>
        <c:order val="1"/>
        <c:tx>
         <c:strRef>
          <c:f>Sheet1!$C$1</c:f>
          <c:strCache>
           <c:ptCount val="1"/>
           <c:pt idx="0">
            <c:v>Monster</c:v>
           </c:pt>
          </c:strCache>
         </c:strRef>
        </c:tx>
        <c:spPr>
         <a:ln w="47625">
          <a:noFill/>
         </a:ln>
        </c:spPr>
        <c:xVal>
         <c:numRef>
          <c:f>Sheet1!$A$2:$A$7</c:f>
          <c:numCache>
           <c:formatCode>General</c:formatCode>
           <c:ptCount val="6"/>
           <c:pt idx="0">
            <c:v>0.7</c:v>
           </c:pt>
           <c:pt idx="1">
            <c:v>1.8</c:v>
           </c:pt>
           <c:pt idx="2">
            <c:v>2.6</c:v>
           </c:pt>
           <c:pt idx="3">
            <c:v>0.8</c:v>
           </c:pt>
           <c:pt idx="4">
            <c:v>1.7</c:v>
           </c:pt>
           <c:pt idx="5">
            <c:v>2.5</c:v>
           </c:pt>
          </c:numCache>
         </c:numRef>
```

```
          </c:xVal>
          <c:yVal>
            <c:numRef>
              <c:f>Sheet1!$C$2:$C$7</c:f>
              <c:numCache>
                <c:formatCode>General</c:formatCode>
                <c:ptCount val="6"/>
                <c:pt idx="3">
                  <c:v>3.2</c:v>
                </c:pt>
                <c:pt idx="4">
                  <c:v>4.3</c:v>
                </c:pt>
                <c:pt idx="5">
                  <c:v>1.2</c:v>
                </c:pt>
              </c:numCache>
            </c:numRef>
          </c:yVal>
          <c:smooth val="0"/>
        </c:ser>
        <c:dLbls>
          <c:showLegendKey val="0"/>
          <c:showVal val="0"/>
          <c:showCatName val="0"/>
          <c:showSerName val="0"/>
          <c:showPercent val="0"/>
          <c:showBubbleSize val="0"/>
        </c:dLbls>
        <c:axId val="-2128940872"/>
        <c:axId val="-2129643912"/>
      </c:scatterChart>
      <c:valAx>
        <c:axId val="-2128940872"/>
        <c:scaling>
          <c:orientation val="minMax"/>
        </c:scaling>
        <c:delete val="0"/>
        <c:axPos val="b"/>
        <c:numFmt formatCode="General" sourceLinked="1"/>
        <c:majorTickMark val="out"/>
        <c:minorTickMark val="none"/>
        <c:tickLblPos val="nextTo"/>
        <c:crossAx val="-2129643912"/>
        <c:crosses val="autoZero"/>
        <c:crossBetween val="midCat"/>
      </c:valAx>
      <c:valAx>
        <c:axId val="-2129643912"/>
        <c:scaling>
          <c:orientation val="minMax"/>
        </c:scaling>
        <c:delete val="0"/>
        <c:axPos val="l"/>
        <c:majorGridlines/>
        <c:numFmt formatCode="General" sourceLinked="1"/>
        <c:majorTickMark val="out"/>
        <c:minorTickMark val="none"/>
```

```
        <c:tickLblPos val="nextTo"/>
        <c:crossAx val="-2128940872"/>
        <c:crosses val="autoZero"/>
        <c:crossBetween val="midCat"/>
      </c:valAx>
    </c:plotArea>
    <c:legend>
      <c:legendPos val="r"/>
      <c:layout/>
      <c:overlay val="0"/>
    </c:legend>
    <c:plotVisOnly val="1"/>
    <c:dispBlanksAs val="gap"/>
    <c:showDLblsOverMax val="0"/>
  </c:chart>
  <c:txPr>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr>
        <a:defRPr sz="1800"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:externalData r:id="rId1">
    <c:autoUpdate val="0"/>
  </c:externalData>
</c:chartSpace>
```

Related Schema Definitions

XY (scatter) chart elements:

```
<xsd:complexType name="CT_PlotArea">
  <xsd:sequence>
    <xsd:element name="layout" type="CT_Layout" minOccurs="0" maxOccurs="1"/>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element name="areaChart"      type="CT_AreaChart"/>
      <xsd:element name="area3DChart"    type="CT_Area3DChart"/>
      <xsd:element name="lineChart"      type="CT_LineChart"/>
      <xsd:element name="line3DChart"    type="CT_Line3DChart"/>
      <xsd:element name="stockChart"     type="CT_StockChart"/>
      <xsd:element name="radarChart"     type="CT_RadarChart"/>
      <xsd:element name="scatterChart"   type="CT_ScatterChart"/>
      <xsd:element name="pieChart"       type="CT_PieChart"/>
      <xsd:element name="pie3DChart"     type="CT_Pie3DChart"/>
      <xsd:element name="doughnutChart"  type="CT_DoughnutChart"/>
      <xsd:element name="barChart"       type="CT_BarChart"/>
      <xsd:element name="bar3DChart"     type="CT_Bar3DChart"/>
      <xsd:element name="ofPieChart"     type="CT_OfPieChart"/>
      <xsd:element name="surfaceChart"   type="CT_SurfaceChart"/>
      <xsd:element name="surface3DChart" type="CT_Surface3DChart"/>
      <xsd:element name="bubbleChart"    type="CT_BubbleChart"/>
    </xsd:choice>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
```

```
      <xsd:element name="valAx"  type="CT_ValAx"/>
      <xsd:element name="catAx"  type="CT_CatAx"/>
      <xsd:element name="dateAx" type="CT_DateAx"/>
      <xsd:element name="serAx"  type="CT_SerAx"/>
    </xsd:choice>
    <xsd:element name="dTable" type="CT_DTable"            minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ScatterChart">
  <xsd:sequence>
    <xsd:element name="scatterStyle" type="CT_ScatterStyle"/>
    <xsd:element name="varyColors"   type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"          type="CT_ScatterSer"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"        type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="axId"         type="CT_UnsignedInt"   minOccurs="2" maxOccurs=
↪"2"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ScatterStyle">
  <xsd:attribute name="val" type="ST_ScatterStyle" default="marker"/>
</xsd:complexType>

<xsd:complexType name="CT_ScatterSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"         type="CT_UnsignedInt"/>
    <xsd:element name="order"       type="CT_UnsignedInt"/>
    <xsd:element name="tx"          type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"        type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker"      type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"         type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"       type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"   type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"     type="CT_ErrBars"           minOccurs="0"␣
↪maxOccurs="2"/>
    <xsd:element name="xVal"        type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="yVal"        type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="smooth"      type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_AxDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="multiLvlStrRef" type="CT_MultiLvlStrRef"/>
      <xsd:element name="numRef"         type="CT_NumRef"/>
      <xsd:element name="numLit"         type="CT_NumData"/>
      <xsd:element name="strRef"         type="CT_StrRef"/>
      <xsd:element name="strLit"         type="CT_StrData"/>
    </xsd:choice>
```

```
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Marker">
  <xsd:sequence>
    <xsd:element name="symbol" type="CT_MarkerStyle"       minOccurs="0"/>
    <xsd:element name="size"   type="CT_MarkerSize"        minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_MarkerStyle">
  <xsd:attribute name="val" type="ST_MarkerStyle" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_MarkerSize">
  <xsd:attribute name="val" type="ST_MarkerSize" default="5"/>
</xsd:complexType>

<xsd:complexType name="CT_NumData">
  <xsd:sequence>
    <xsd:element name="formatCode" type="s:ST_Xstring"     minOccurs="0"/>
    <xsd:element name="ptCount"    type="CT_UnsignedInt"   minOccurs="0"/>
    <xsd:element name="pt"         type="CT_NumVal"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="numRef" type="CT_NumRef"/>
      <xsd:element name="numLit" type="CT_NumData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumRef">
  <xsd:sequence>
    <xsd:element name="f"        type="xsd:string"/>
    <xsd:element name="numCache" type="CT_NumData"       minOccurs="0"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ST_MarkerSize">
  <xsd:restriction base="xsd:unsignedByte">
    <xsd:minInclusive value="2"/>
    <xsd:maxInclusive value="72"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_MarkerStyle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="circle"/>
    <xsd:enumeration value="dash"/>
```

```
    <xsd:enumeration value="diamond"/>
    <xsd:enumeration value="dot"/>
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="picture"/>
    <xsd:enumeration value="plus"/>
    <xsd:enumeration value="square"/>
    <xsd:enumeration value="star"/>
    <xsd:enumeration value="triangle"/>
    <xsd:enumeration value="x"/>
    <xsd:enumeration value="auto"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_ScatterStyle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="line"/>
    <xsd:enumeration value="lineMarker"/>
    <xsd:enumeration value="marker"/>
    <xsd:enumeration value="smooth"/>
    <xsd:enumeration value="smoothMarker"/>
  </xsd:restriction>
</xsd:simpleType>
```

Line Chart

A line chart is one of the fundamental plot types.

XML Semantics

- c:lineChart/c:marker{val=0|1} doesn't seem to have any effect one way or the other. Default line charts inserted using PowerPoint always have it set to 1 (True). But even if it's set to 0 or removed, markers still appear. Hypothesis is that c:lineChart/c:ser/c:marker/c:symbol{val=none} and the like are the operative settings.

XML specimens

Minimal working XML for a line plot. Note this does not reference values in a spreadsheet.:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<c:chartSpace xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
              xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
              xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/
↪relationships">
  <c:chart>
    <c:plotArea>
      <c:lineChart>
        <c:grouping val="standard"/>
        <c:ser>
          <c:idx val="0"/>
          <c:order val="0"/>
          <c:tx>
            <c:v>Series 1</c:v>
          </c:tx>
```

```
          <c:marker>
            <c:symbol val="none"/>
          </c:marker>
          <c:cat>
            <c:strLit>
              <c:ptCount val="2"/>
              <c:pt idx="0">
                <c:v>Category 1</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>Category 2</c:v>
              </c:pt>
            </c:strLit>
          </c:cat>
          <c:val>
            <c:numLit>
              <c:ptCount val="2"/>
              <c:pt idx="0">
                <c:v>4.3</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>2.5</c:v>
              </c:pt>
            </c:numLit>
          </c:val>
          <c:smooth val="0"/>
        </c:ser>
        <c:axId val="-2044963928"/>
        <c:axId val="-2071826904"/>
      </c:lineChart>
      <c:catAx>
        <c:axId val="-2044963928"/>
        <c:scaling/>
        <c:delete val="0"/>
        <c:axPos val="b"/>
        <c:crossAx val="-2071826904"/>
      </c:catAx>
      <c:valAx>
        <c:axId val="-2071826904"/>
        <c:scaling/>
        <c:delete val="0"/>
        <c:axPos val="l"/>
        <c:crossAx val="-2044963928"/>
      </c:valAx>
    </c:plotArea>
  </c:chart>
</c:chartSpace>
```

Related Schema Definitions

Line chart elements:

```
<xsd:complexType name="CT_LineChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="grouping"   type="CT_Grouping"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
```

```
    <xsd:element name="ser"        type="CT_LineSer"       minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="hiLowLines" type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="upDownBars" type="CT_UpDownBars"    minOccurs="0"/>
    <xsd:element name="marker"     type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="smooth"     type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Grouping">
  <xsd:attribute name="val" type="ST_Grouping" default="standard"/>
</xsd:complexType>

<xsd:simpleType name="ST_Grouping">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="percentStacked"/>
    <xsd:enumeration value="standard"/>
    <xsd:enumeration value="stacked"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_LineSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"       type="CT_UnsignedInt"/>
    <xsd:element name="order"     type="CT_UnsignedInt"/>
    <xsd:element name="tx"        type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"      type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker"    type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"       type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"     type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline" type="CT_Trendline"         minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="errBars"   type="CT_ErrBars"           minOccurs="0"/>
    <xsd:element name="cat"       type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"       type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="smooth"    type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Marker">
  <xsd:sequence>
    <xsd:element name="symbol" type="CT_MarkerStyle"       minOccurs="0"/>
    <xsd:element name="size"   type="CT_MarkerSize"        minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Pie Chart

A pie chart is one of the fundamental plot types.

XML specimens

Minimal working XML for a pie plot. Note this does not reference values in a spreadsheet.:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<c:chartSpace
  xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
  xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
  xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships">
  <c:chart>
    <c:plotArea>
      <c:pieChart>
        <c:ser>
          <c:idx val="0"/>
          <c:order val="0"/>
          <c:tx>
            <c:v>Sales</c:v>
          </c:tx>
          <c:cat>
            <c:strLit>
              <c:ptCount val="4"/>
              <c:pt idx="0">
                <c:v>1st Qtr</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>2nd Qtr</c:v>
              </c:pt>
              <c:pt idx="2">
                <c:v>3rd Qtr</c:v>
              </c:pt>
              <c:pt idx="3">
                <c:v>4th Qtr</c:v>
              </c:pt>
            </c:strLit>
          </c:cat>
          <c:val>
            <c:numLit>
              <c:ptCount val="4"/>
              <c:pt idx="0">
                <c:v>8.2</c:v>
              </c:pt>
              <c:pt idx="1">
                <c:v>3.2</c:v>
              </c:pt>
              <c:pt idx="2">
                <c:v>1.4</c:v>
              </c:pt>
              <c:pt idx="3">
                <c:v>1.2</c:v>
              </c:pt>
            </c:numLit>
          </c:val>
        </c:ser>
```

```
      </c:pieChart>
    </c:plotArea>
  </c:chart>
</c:chartSpace>
```

Related Schema Definitions

Pie chart elements:

```
<xsd:complexType name="CT_PieChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="varyColors"    type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"           type="CT_PieSer"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"         type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="firstSliceAng" type="CT_FirstSliceAng" minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_PieSer">
  <xsd:sequence>
    <xsd:element name="idx"       type="CT_UnsignedInt"/>
    <xsd:element name="order"     type="CT_UnsignedInt"/>
    <xsd:element name="tx"        type="CT_SerTx"              minOccurs="0"/>
    <xsd:element name="spPr"      type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="explosion" type="CT_UnsignedInt"       minOccurs="0"/>
    <xsd:element name="dPt"       type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"     type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="cat"       type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"       type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SerTx">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="strRef" type="CT_StrRef"/>
      <xsd:element name="v"      type="s:ST_Xstring"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_AxDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="multiLvlStrRef" type="CT_MultiLvlStrRef"/>
      <xsd:element name="numRef"         type="CT_NumRef"/>
      <xsd:element name="numLit"         type="CT_NumData"/>
      <xsd:element name="strRef"         type="CT_StrRef"/>
      <xsd:element name="strLit"         type="CT_StrData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="CT_NumDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="numRef" type="CT_NumRef"/>
      <xsd:element name="numLit" type="CT_NumData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumRef">
  <xsd:sequence>
    <xsd:element name="f"        type="xsd:string"/>
    <xsd:element name="numCache" type="CT_NumData"       minOccurs="0"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumData">
  <xsd:sequence>
    <xsd:element name="formatCode" type="s:ST_Xstring"     minOccurs="0"/>
    <xsd:element name="ptCount"    type="CT_UnsignedInt"   minOccurs="0"/>
    <xsd:element name="pt"         type="CT_NumVal"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_StrRef">
  <xsd:sequence>
    <xsd:element name="f"        type="xsd:string"/>
    <xsd:element name="strCache" type="CT_StrData"       minOccurs="0"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_StrData">
  <xsd:sequence>
    <xsd:element name="ptCount" type="CT_UnsignedInt"   minOccurs="0"/>
    <xsd:element name="pt"      type="CT_StrVal"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="extLst"  type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ST_Xstring">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="CT_NumVal">
  <xsd:sequence>
    <xsd:element name="v" type="s:ST_Xstring"/>
  </xsd:sequence>
  <xsd:attribute name="idx"        type="xsd:unsignedInt" use="required"/>
  <xsd:attribute name="formatCode" type="s:ST_Xstring"/>
</xsd:complexType>
```

Bar Chart

The bar chart is a fundamental plot type, used for both column and bar charts by specifying the bar direction. It is also used for the clustered, stacked, and stacked 100% varieties by specifying grouping and overlap.

Gap Width

A gap appears between the bar or clustered bars for each category on a bar chart. The default width for this gap is 150% of the bar width. It can be set between 0 and 500% of the bar width. In the MS API this is set using the property *ChartGroup.GapWidth*.

Proposed protocol:

```
>>> assert isinstance(bar_plot, BarPlot)
>>> bar_plot.gap_width
150
>>> bar_plot.gap_width = 300
>>> bar_plot.gap_width
300
>>> bar_plot.gap_width = 700
ValueError: gap width must be in range 0-500 (percent)
```

Overlap

In a bar chart having two or more series, the bars for each category are clustered together for ready comparison. By default, these bars are directly adjacent to each other, visually "touching".

The bars can be made to overlap each other or have a space between them using the *overlap* property. Its values range between -100 and 100, representing the percentage of the bar width by which to overlap adjacent bars. A setting of -100 creates a gap of a full bar width and a setting of 100 causes all the bars in a category to be superimposed. The default value is 0.

Proposed protocol:

```
>>> bar_plot = chart.plots[0]
>>> assert isinstance(bar_plot, BarPlot)
>>> bar_plot.overlap
0
>>> bar_plot.overlap = -50
>>> bar_plot.overlap
-50
>>> bar_plot.overlap = 200
ValueError: overlap must be in range -100..100 (percent)
```

XML specimens

Minimal working XML for a single-series column plot. Note this does not reference values in a spreadsheet.:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<c:chartSpace xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
              xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
              xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/
↪relationships">
```

```
  <c:chart>
    <c:plotArea>
      <c:barChart>
        <c:barDir val="col"/>
        <c:grouping val="clustered"/>
        <c:ser>
          <c:idx val="0"/>
          <c:order val="0"/>
          <c:cat>
            <c:strLit>
              <c:ptCount val="1"/>
              <c:pt idx="0">
                <c:v>Foo</c:v>
              </c:pt>
            </c:strLit>
          </c:cat>
          <c:val>
            <c:numLit>
              <c:ptCount val="1"/>
              <c:pt idx="0">
                <c:v>4.3</c:v>
              </c:pt>
            </c:numLit>
          </c:val>
        </c:ser>
        <c:dLbls>
          <c:showLegendKey val="0"/>
          <c:showVal val="0"/>
          <c:showCatName val="0"/>
          <c:showSerName val="0"/>
          <c:showPercent val="0"/>
          <c:showBubbleSize val="0"/>
        </c:dLbls>
        <c:gapWidth val="300"/>
        <c:overlap val="-20"/>
        <c:axId val="-2068027336"/>
        <c:axId val="-2113994440"/>
      </c:barChart>
      <c:catAx>
        <c:axId val="-2068027336"/>
        <c:scaling/>
        <c:delete val="0"/>
        <c:axPos val="b"/>
        <c:crossAx val="-2113994440"/>
      </c:catAx>
      <c:valAx>
        <c:axId val="-2113994440"/>
        <c:scaling/>
        <c:delete val="0"/>
        <c:axPos val="l"/>
        <c:crossAx val="-2068027336"/>
      </c:valAx>
    </c:plotArea>
  </c:chart>
</c:chartSpace>
```

Related Schema Definitions

```
<xsd:complexType name="CT_BarChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="barDir"     type="CT_BarDir"/>
    <xsd:element name="grouping"   type="CT_BarGrouping"   minOccurs="0"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_BarSer"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="gapWidth"   type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="overlap"    type="CT_Overlap"       minOccurs="0"/>
    <xsd:element name="serLines"   type="CT_ChartLines"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_BarSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"              type="CT_UnsignedInt"/>
    <xsd:element name="order"            type="CT_UnsignedInt"/>
    <xsd:element name="tx"               type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"             type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="pictureOptions"   type="CT_PictureOptions"    minOccurs="0"/>
    <xsd:element name="dPt"              type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"            type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"        type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"          type="CT_ErrBars"           minOccurs="0"/>
    <xsd:element name="cat"              type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"              type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="shape"            type="CT_Shape"             minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<!-- gap-width -->

<xsd:complexType name="CT_GapAmount">
  <xsd:attribute name="val" type="ST_GapAmount" default="150%"/>
</xsd:complexType>

<xsd:simpleType name="ST_GapAmount">
  <xsd:union memberTypes="ST_GapAmountPercent ST_GapAmountUShort"/>
</xsd:simpleType>

<xsd:simpleType name="ST_GapAmountPercent">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="0*(([0-9])|([1-9][0-9])|([1-4][0-9][0-9])|500)%"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_GapAmountUShort">
```

```
  <xsd:restriction base="xsd:unsignedShort">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="500"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- overlap -->

<xsd:complexType name="CT_Overlap">
  <xsd:attribute name="val" type="ST_Overlap" default="0%"/>
</xsd:complexType>

<xsd:simpleType name="ST_Overlap">
  <xsd:union memberTypes="ST_OverlapPercent ST_OverlapByte"/>
</xsd:simpleType>

<xsd:simpleType name="ST_OverlapPercent">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="(-?0*(([0-9])|([1-9][0-9])|100))%"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_OverlapByte">
  <xsd:restriction base="xsd:byte">
    <xsd:minInclusive value="-100"/>
    <xsd:maxInclusive value="100"/>
  </xsd:restriction>
</xsd:simpleType>
```

Chart - Plots

Within the `c:chartSpace/c:chart/c:plotArea` element may occur 0..n *xChart* elements, each containing 0..n series.

xChart here is a placeholder name for the various plot elements, each of which ends with *Chart*:

- `<c:barChart>`
- `<c:lineChart>`
- `<c:pieChart>`
- `<c:areaChart>`
- etc. ...

In the Microsoft API, the term *chart group* is used for the concept these elements represent. Rather than a group of charts, their role is perhaps better described as a *series group*. The terminology is a bit vexed when it comes down to details. The term *plot* was chosen for the purposes of this library.

The reason the concept of a plot is required is that more than one type of plotting may appear on a single chart. For example, a line chart can appear on top of a column chart. To be more precise, one or more series can be plotted as lines superimposed on one or more series plotted as columns.

The two sets of series both belong to a single chart, and there is only a single data source to define all the series in the chart. The Excel workbook that provides the chart data uses only a single worksheet.

Individual series can be assigned a different *chart type* using the PowerPoint UI; this operation causes them to be placed in a distinct *xChart* element of the corresponding type. During this operation, PowerPoint adds *xChart* elements in

an order that seems to correspond to logical viewing order, without respect to the sequence of the series chosen. It appears area plots are placed in back, bar next, and line in front. This prevents plots of one type from obscuring the others.

Note that not all combinations of chart types are possible. I've seen area overlaid with column overlaid with line. Bar cannot be combined with line, which seems sensible because their axes locations differ (switched horizontal/vertical).

Feature Summary

Most plot-level properties are particular to a chart type. One so far is shared by plots of almost all chart types.

- **Plot.vary_by_categories** – Read/write boolean determining whether point markers for each category have a different color. A point marker here means a bar for a bar chart, a pie section, etc. This setting is only operative when the plot has a single series. A plot with multiple series uses varying colors to distinguish series rather than categories.

Candidate protocol

Plot.vary_by_categories:

```
>>> plot = chart.plots[0]
>>> plot.vary_by_categories
True
>>> plot.vary_by_categories = False
>>> plot.vary_by_categories
False
```

XML Semantics

- The value of `c:xChart/c:varyColors` defaults to `True` when the element is not present or when the element is present but its `val` attribute is not.

Related Schema Definitions

```
<xsd:complexType name="CT_AreaChart">
  <xsd:sequence>
    <xsd:group    ref="EG_AreaChartShared"/>
    <xsd:element name="axId"   type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2"/>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Area3DChart">
  <xsd:sequence>
    <xsd:group    ref="EG_AreaChartShared"/>
    <xsd:element name="gapDepth" type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="axId"     type="CT_UnsignedInt"   minOccurs="2" maxOccurs="3"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_AreaChartShared">
```

```
  <xsd:sequence>
    <xsd:element name="grouping"   type="CT_Grouping"   minOccurs="0"/>
    <xsd:element name="varyColors" type="CT_Boolean"    minOccurs="0"/>
    <xsd:element name="ser"        type="CT_AreaSer"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"      minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines" minOccurs="0"/>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_Grouping">
  <xsd:attribute name="val" type="ST_Grouping" default="standard"/>
</xsd:complexType>

<xsd:simpleType name="ST_Grouping">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="percentStacked"/>
    <xsd:enumeration value="standard"/>
    <xsd:enumeration value="stacked"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_BarChart">
  <xsd:sequence>
    <xsd:group    ref="EG_BarChartShared"/>
    <xsd:element name="gapWidth" type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="overlap"  type="CT_Overlap"       minOccurs="0"/>
    <xsd:element name="serLines" type="CT_ChartLines"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="axId"     type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Bar3DChart">
  <xsd:sequence>
    <xsd:group    ref="EG_BarChartShared"/>
    <xsd:element name="gapWidth" type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="gapDepth" type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="shape"    type="CT_Shape"         minOccurs="0"/>
    <xsd:element name="axId"     type="CT_UnsignedInt"   minOccurs="2" maxOccurs="3"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_BarChartShared">
  <xsd:sequence>
    <xsd:element name="barDir"     type="CT_BarDir"/>
    <xsd:element name="grouping"   type="CT_BarGrouping" minOccurs="0"/>
    <xsd:element name="varyColors" type="CT_Boolean"     minOccurs="0"/>
    <xsd:element name="ser"        type="CT_BarSer"      minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"       minOccurs="0"/>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_BarDir">
  <xsd:attribute name="val" type="ST_BarDir" default="col"/>
```

```
</xsd:complexType>

<xsd:simpleType name="ST_BarDir">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="bar"/>
    <xsd:enumeration value="col"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_BarGrouping">
  <xsd:attribute name="val" type="ST_BarGrouping" default="clustered"/>
</xsd:complexType>

<xsd:simpleType name="ST_BarGrouping">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="percentStacked"/>
    <xsd:enumeration value="clustered"/>
    <xsd:enumeration value="standard"/>
    <xsd:enumeration value="stacked"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_Shape">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="cone"/>
    <xsd:enumeration value="coneToMax"/>
    <xsd:enumeration value="box"/>
    <xsd:enumeration value="cylinder"/>
    <xsd:enumeration value="pyramid"/>
    <xsd:enumeration value="pyramidToMax"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_LineChart">
  <xsd:sequence>
    <xsd:group ref="EG_LineChartShared" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="hiLowLines" type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="upDownBars" type="CT_UpDownBars"    minOccurs="0"/>
    <xsd:element name="marker"     type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="smooth"     type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_LineChartShared">
  <xsd:sequence>
    <xsd:element name="grouping"   type="CT_Grouping"/>
    <xsd:element name="varyColors" type="CT_Boolean"    minOccurs="0"/>
    <xsd:element name="ser"        type="CT_LineSer"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"      minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines" minOccurs="0"/>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_PieChart">
```

```
  <xsd:sequence>
    <xsd:group    ref="EG_PieChartShared"/>
    <xsd:element name="firstSliceAng" type="CT_FirstSliceAng" minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_PieChartShared">
  <xsd:sequence>
    <xsd:element name="varyColors" type="CT_Boolean" minOccurs="0"/>
    <xsd:element name="ser"        type="CT_PieSer"  minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"   minOccurs="0"/>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_PieSer">
  <xsd:sequence>
    <xsd:group    ref="EG_SerShared"/>
    <xsd:element name="explosion" type="CT_UnsignedInt"   minOccurs="0"/>
    <xsd:element name="dPt"       type="CT_DPt"           minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"     type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="cat"       type="CT_AxDataSource"  minOccurs="0"/>
    <xsd:element name="val"       type="CT_NumDataSource" minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_SerShared">
  <xsd:sequence>
    <xsd:element name="idx"   type="CT_UnsignedInt"/>
    <xsd:element name="order" type="CT_UnsignedInt"/>
    <xsd:element name="tx"    type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"  type="a:CT_ShapeProperties" minOccurs="0"/>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_RadarChart">
  <xsd:sequence>
    <xsd:element name="radarStyle" type="CT_RadarStyle"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_RadarSer"      minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_RadarStyle">
  <xsd:attribute name="val" type="ST_RadarStyle" default="standard"/>
</xsd:complexType>

<xsd:simpleType name="ST_RadarStyle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="standard"/>
```

```
    <xsd:enumeration value="marker"/>
    <xsd:enumeration value="filled"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_ScatterChart">
  <xsd:sequence>
    <xsd:element name="scatterStyle" type="CT_ScatterStyle"/>
    <xsd:element name="varyColors"   type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"          type="CT_ScatterSer"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"        type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="axId"         type="CT_UnsignedInt"   minOccurs="2" maxOccurs=
↪"2"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ScatterStyle">
  <xsd:attribute name="val" type="ST_ScatterStyle" default="marker"/>
</xsd:complexType>

<xsd:simpleType name="ST_ScatterStyle">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="line"/>
    <xsd:enumeration value="lineMarker"/>
    <xsd:enumeration value="marker"/>
    <xsd:enumeration value="smooth"/>
    <xsd:enumeration value="smoothMarker"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_StockChart">
  <xsd:sequence>
    <xsd:element name="ser"        type="CT_LineSer"       minOccurs="3" maxOccurs="4
↪"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="hiLowLines" type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="upDownBars" type="CT_UpDownBars"    minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SurfaceChart">
  <xsd:sequence>
    <xsd:group    ref="EG_SurfaceChartShared"/>
    <xsd:element name="axId"   type="CT_UnsignedInt"   minOccurs="2" maxOccurs="3"/>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Surface3DChart">
  <xsd:sequence>
    <xsd:group ref="EG_SurfaceChartShared"/>
```

```
    <xsd:element name="axId"   type="CT_UnsignedInt"   minOccurs="3" maxOccurs="3"/>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_SurfaceChartShared">
  <xsd:sequence>
    <xsd:element name="wireframe" type="CT_Boolean"    minOccurs="0"/>
    <xsd:element name="ser"       type="CT_SurfaceSer" minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="bandFmts"  type="CT_BandFmts"   minOccurs="0"/>
  </xsd:sequence>
</xsd:group>
```

Chart - Legend

A chart may have a legend. The legend may be placed within the plot area or alongside it. The legend has a fill and font and may be positioned on the top, right, bottom, left, in the upper-right corner, or in a custom position. Individual legend entries may be custom specified, but that is not yet in scope.

Candidate protocol

```
>>> chart.has_legend
False
>>> chart.has_legend = True
>>> chart.has_legend
True
>>> legend = chart.legend
>>> legend
<pptx.chart.chart.Legend object at 0xdeadbeef1>

>>> legend.font
<pptx.text.Font object at 0xdeadbeef2>

>>> legend.horz_offset
0.0
>>> legend.horz_offset = 0.2
>>> legend.horz_offset
0.2

>>> legend.include_in_layout
True
>>> legend.include_in_layout = False
>>> legend.include_in_layout
False

>>> legend.position
XL_LEGEND_POSITION.RIGHT (-4152)
>>> legend.position = XL_LEGEND_POSITION.BOTTOM
>>> legend.position
XL_LEGEND_POSITION.BOTTOM (-4107)
```

Feature Summary

- *Chart.has_legend* – Read/write boolean property
- *Chart.legend* – Read-only *Legend* object or None
- *Legend.horz_offset* – Read/write float (-1.0 -> 1.0).
- *Legend.include_in_layout* – Read/write boolean.
- *Legend.position* – Read/write *XL_LEGEND_POSITION*
- *Legend.font* – Read-only *Font* object.

Enumerations

- *XL_LEGEND_POSITION*

Microsoft API

Chart.HasLegend True if the chart has a legend. Read/write Boolean.

Chart.Legend Returns the legend for the chart. Read-only Legend.

Legend.IncludeInLayout True if a legend will occupy the chart layout space when a chart layout is being determined. The default is True. Read/write Boolean.

Legend.Position Returns or sets the position of the legend on the chart. Read/write XlLegendPosition.

XML specimens

Example legend XML:

```
<c:legend>
  <c:legendPos val="t"/>
  <c:layout>
    <c:manualLayout>
      <c:xMode val="edge"/>
      <c:yMode val="edge"/>
      <c:x val="0.321245570866142"/>
      <c:y val="0.025"/>
      <c:w val="0.532508858267717"/>
      <c:h val="0.0854055118110236"/>
    </c:manualLayout>
  </c:layout>
  <c:overlay val="1"/>
  <c:spPr>
    <a:solidFill>
      <a:schemeClr val="accent6">
        <a:lumMod val="20000"/>
        <a:lumOff val="80000"/>
      </a:schemeClr>
    </a:solidFill>
  </c:spPr>
  <c:txPr>
    <a:bodyPr/>
    <a:lstStyle/>
```

```
    <a:p>
      <a:pPr>
        <a:defRPr sz="1600" b="0" i="1" baseline="0"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
</c:legend>
```

Legend having horz_offset == 0.42:

```
<c:legend>
  <c:legendPos val="r"/>
  <c:layout>
    <c:manualLayout>
      <c:xMode val="factor"/>
      <c:yMode val="factor"/>
      <c:x val="0.42"/>
    </c:manualLayout>
  </c:layout>
  <c:overlay val="0"/>
</c:legend>
```

Related Schema Definitions

```
<xsd:complexType name="CT_Legend">
  <xsd:sequence>
    <xsd:element name="legendPos"   type="CT_LegendPos"         minOccurs="0"/>
    <xsd:element name="legendEntry" type="CT_LegendEntry"       minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="layout"      type="CT_Layout"            minOccurs="0"/>
    <xsd:element name="overlay"     type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="spPr"        type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"        type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_LegendPos">
  <xsd:attribute name="val" type="ST_LegendPos" default="r"/>
</xsd:complexType>

<xsd:simpleType name="ST_LegendPos">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="b"/>
    <xsd:enumeration value="tr"/>
    <xsd:enumeration value="l"/>
    <xsd:enumeration value="r"/>
    <xsd:enumeration value="t"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_LegendEntry">
  <xsd:sequence>
    <xsd:element name="idx" type="CT_UnsignedInt"/>
    <xsd:choice>
```

```
      <xsd:element name="delete" type="CT_Boolean"/>
      <xsd:group    ref="EG_LegendEntryData"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Layout">
  <xsd:sequence>
    <xsd:element name="manualLayout" type="CT_ManualLayout"  minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_LayoutTarget">
  <xsd:attribute name="val" type="ST_LayoutTarget" default="outer"/>
</xsd:complexType>

<xsd:simpleType name="ST_LayoutTarget">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="inner"/>
    <xsd:enumeration value="outer"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_LayoutMode">
  <xsd:attribute name="val" type="ST_LayoutMode" default="factor"/>
</xsd:complexType>

<xsd:simpleType name="ST_LayoutMode">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="edge"/>
    <xsd:enumeration value="factor"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_ManualLayout">
  <xsd:sequence>
    <xsd:element name="layoutTarget" type="CT_LayoutTarget"  minOccurs="0"/>
    <xsd:element name="xMode"        type="CT_LayoutMode"    minOccurs="0"/>
    <xsd:element name="yMode"        type="CT_LayoutMode"    minOccurs="0"/>
    <xsd:element name="wMode"        type="CT_LayoutMode"    minOccurs="0"/>
    <xsd:element name="hMode"        type="CT_LayoutMode"    minOccurs="0"/>
    <xsd:element name="x"            type="CT_Double"        minOccurs="0"/>
    <xsd:element name="y"            type="CT_Double"        minOccurs="0"/>
    <xsd:element name="w"            type="CT_Double"        minOccurs="0"/>
    <xsd:element name="h"            type="CT_Double"        minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

SlideShapes.add_chart()

A chart is added to a slide similarly to adding any other shape. Note that a chart is not itself a shape; the item returned by *.add_shape()* is a *GraphicFrame* shape which contains a *Chart* object. The actual chart object is accessed using the `chart` attribute on the graphic frame that contains it.

Adding a chart requires three items, a chart type, the position and size desired, and a *ChartData* object specifying the categories and series values for the new chart.

Protocol

Creating a new chart:

```
>>> chart_data = ChartData()
>>> chart_data.categories = 'Foo', 'Bar'
>>> chart_data.add_series('Series 1', (1.2, 2.3))
>>> chart_data.add_series('Series 2', (3.4, 4.5))

>>> x, y, cx, cy = Inches(2), Inches(2), Inches(6), Inches(4)
>>> graphic_frame = shapes.add_chart(
>>>     XL_CHART_TYPE.COLUMN_CLUSTERED, x, y, cx, cy, chart_data
>>> )
>>> chart = graphicFrame.chart
```

Specimen XML

Chart in a graphic frame:

```
<p:graphicFrame>
  <p:nvGraphicFramePr>
    <p:cNvPr id="2" name="Chart 1"/>
    <p:cNvGraphicFramePr/>
    <p:nvPr>
      <p:extLst>
        <p:ext uri="{D42A27DB-BD31-4B8C-83A1-F6EECF244321}">
          <p14:modId xmlns:p14="http://schemas.microsoft.com/office/powerpoint/2010/
↪main"
                     val="11227583"/>
        </p:ext>
      </p:extLst>
    </p:nvPr>
  </p:nvGraphicFramePr>
  <p:xfrm>
    <a:off x="1524000" y="1397000"/>
    <a:ext cx="6096000" cy="4064000"/>
  </p:xfrm>
  <a:graphic>
    <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/chart">
      <c:chart xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
               xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/
↪relationships"
               r:id="rId2"/>
    </a:graphicData>
  </a:graphic>
</p:graphicFrame>
```

Related Schema Definitions

A `<p:graphicFrame>` element appears in a `CT_GroupShape` element, typically a `<p:spTree>` (shape tree) element:

```
<xsd:complexType name="CT_GroupShape">
  <xsd:sequence>
    <xsd:element name="nvGrpSpPr" type="CT_GroupShapeNonVisual"/>
    <xsd:element name="grpSpPr"   type="a:CT_GroupShapeProperties"/>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="sp"           type="CT_Shape"/>
      <xsd:element name="grpSp"        type="CT_GroupShape"/>
      <xsd:element name="graphicFrame" type="CT_GraphicalObjectFrame"/>
      <xsd:element name="cxnSp"        type="CT_Connector"/>
      <xsd:element name="pic"          type="CT_Picture"/>
      <xsd:element name="contentPart"  type="CT_Rel"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Graphic frame-related elements:

```
<xsd:complexType name="CT_GraphicalObjectFrame">
  <xsd:sequence>
    <xsd:element name="nvGraphicFramePr" type="CT_GraphicalObjectFrameNonVisual"/>
    <xsd:element name="xfrm"             type="a:CT_Transform2D"/>
    <xsd:element ref="a:graphic"/>  <!-- type="CT_GraphicalObject" -->
    <xsd:element name="extLst"           type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="a:ST_BlackWhiteMode"/>
</xsd:complexType>

<xsd:complexType name="CT_GraphicalObjectFrameNonVisual">
  <xsd:sequence>
    <xsd:element name="cNvPr"            type="a:CT_NonVisualDrawingProps"/>
    <xsd:element name="cNvGraphicFramePr" type="a:CT_NonVisualGraphicFrameProperties"/
↪>
    <xsd:element name="nvPr"             type="CT_ApplicationNonVisualDrawingProps"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_GraphicalObject">
  <xsd:sequence>
    <xsd:element name="graphicData" type="CT_GraphicalObjectData"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_GraphicalObjectData">
  <xsd:sequence>
    <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="strict"/>
  </xsd:sequence>
  <xsd:attribute name="uri" type="xsd:token" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="hlinkClick" type="CT_Hyperlink"             minOccurs="0"/>
    <xsd:element name="hlinkHover" type="CT_Hyperlink"             minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id"     type="ST_DrawingElementId" use="required"/>
  <xsd:attribute name="name"   type="xsd:string"          use="required"/>
```

```
  <xsd:attribute name="descr"  type="xsd:string"          default=""/>
  <xsd:attribute name="hidden" type="xsd:boolean"         default="false"/>
  <xsd:attribute name="title"  type="xsd:string"          default=""/>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualGraphicFrameProperties">
  <xsd:sequence>
    <xsd:element name="graphicFrameLocks" type="CT_GraphicalObjectFrameLocking"␣
↪minOccurs="0"/>
    <xsd:element name="extLst"            type="CT_OfficeArtExtensionList"     ␣
↪minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_GraphicalObjectFrameLocking">
  <xsd:sequence>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="noGrp"          type="xsd:boolean" default="false"/>
  <xsd:attribute name="noDrilldown"    type="xsd:boolean" default="false"/>
  <xsd:attribute name="noSelect"       type="xsd:boolean" default="false"/>
  <xsd:attribute name="noChangeAspect" type="xsd:boolean" default="false"/>
  <xsd:attribute name="noMove"         type="xsd:boolean" default="false"/>
  <xsd:attribute name="noResize"       type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_ApplicationNonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="ph"          type="CT_Placeholder"      minOccurs="0"/>
    <xsd:group   ref="a:EG_Media"                              minOccurs="0"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="isPhoto"   type="xsd:boolean" default="false"/>
  <xsd:attribute name="userDrawn" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:group name="EG_Media">
  <xsd:choice>
    <xsd:element name="audioCd"       type="CT_AudioCD"/>
    <xsd:element name="wavAudioFile"  type="CT_EmbeddedWAVAudioFile"/>
    <xsd:element name="audioFile"     type="CT_AudioFile"/>
    <xsd:element name="videoFile"     type="CT_VideoFile"/>
    <xsd:element name="quickTimeFile" type="CT_QuickTimeFile"/>
  </xsd:choice>
</xsd:group>
```

Chart Data

The data behind a chart–its category labels and its series names and values–turns out to be a pivotal object in both the construction of a new chart and updating the data behind an existing chart.

The first role of *ChartData* is to act as a data transfer object, allowing the data for a chart to be accumulated over multiple calls such as *.add_series()*. This avoids the need to assemble and send a complex nested structure of primitives to method calls.

In addition to this, *ChartData* also takes on the role of broker to `ChartXmlWriter` and `WorkbookWriter`

which know how to assemble the <c:chartSpace> XML and Excel workbook for a chart, respectively. This is sensible because neither of these objects can operate without a chart data instance to provide the data they need and doing so concentrates the coupling to the latter two objects into one place.

Protocol

A *ChartData* object is constructed directly as needed, and used for either creating a new chart or for replacing the data behind an existing one.

Creating a new chart:

```
>>> chart_data = ChartData()
>>> chart_data.categories = 'Foo', 'Bar'
>>> chart_data.add_series('Series 1', (1.2, 2.3))
>>> chart_data.add_series('Series 2', (3.4, 4.5))

>>> x, y, cx, cy = Inches(2), Inches(2), Inches(6), Inches(4)
>>> graphic_frame = shapes.add_chart(
>>>     XL_CHART_TYPE.COLUMN_CLUSTERED, x, y, cx, cy, chart_data
>>> )
>>> chart = graphicFrame.chart
```

Changing the data behind an existing chart:

```
>>> chart_data = ChartData()
>>> chart_data.categories = 'Foobar', 'Barbaz', 'Bazfoo'
>>> chart_data.add_series('New Series 1', (5.6, 6.7, 7.8))
>>> chart_data.add_series('New Series 2', (2.3, 3.4, 4.5))
>>> chart_data.add_series('New Series 3', (8.9, 9.1, 1.2))

>>> chart.replace_data(chart_data)
```

Note that the dimensions of the replacement data can differ from that of the existing chart.

Chart - Tick Labels

The vertical and horizontal divisions of a chart axis may be labeled with *tick labels*, text that describes the division, most commonly its category name or value. A tick labels object is not a collection. There is no object that represents in individual tick label.

Tick label text for a category axis comes from the name of each category. The default tick label text for a category axis is the number that indicates the position of the category relative to the low end of this axis. To change the number of unlabeled tick marks between tick-mark labels, you must change the TickLabelSpacing property for the category axis.

Tick label text for the value axis is calculated based on the *major_unit*, *minimum_scale*, and *maximum_scale* properties of the value axis. To change the tick label text for the value axis, you must change the values of these properties.

Candidate protocol

```
>>> tick_labels = value_axis.tick_labels

>>> tick_labels.font
<pptx.text.Font object at 0xdeadbeef1>
```

```
>>> tick_labels.number_format
'General'
>>> tick_labels.number_format = '0"%"'
>>> tick_labels.number_format
'0"%"'

>>> tick_labels.number_format_is_linked
True
>>> tick_labels.number_format_is_linked = False
>>> tick_labels.number_format_is_linked
False

# offset property is only available on category axis
>>> tick_labels = category_axis.tick_labels
>>> tick_labels.offset
100
>>> tick_labels.offset = 250
>>> tick_labels.offset
250
```

Feature Summary

- **TickLabels.font** – Read/only Font object for tick labels.
- **TickLabels.number_format** – Read/write string.
- **TickLabels.number_format_is_linked** – Read/write boolean.
- **TickLabels.offset** – Read/write int between 0 and 1000, inclusive.

Microsoft API

Font Returns the font of the specified object. Read-only ChartFont.

NumberFormat Returns or sets the format code for the object. Read/write String.

NumberFormatLinked True if the number format is linked to the cells (so that the number format changes in the labels when it changes in the cells). Read/write Boolean.

Offset Returns or sets the distance between the levels of labels, and the distance between the first level and the axis line. Read/write Long.

XML specimens

Example category axis XML showing tick label-related elements:

```
<c:catAx>
  <c:axId val="-2097691448"/>
  <c:scaling>
    <c:orientation val="minMax"/>
  </c:scaling>
  <c:axPos val="b"/>
  <c:numFmt formatCode="General" sourceLinked="0"/>
  <c:tickLblPos val="nextTo"/>
  <c:txPr>
```

```
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr>
        <a:defRPr sz="1000"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:crossAx val="-2097683336"/>
  <c:crosses val="autoZero"/>
  <c:lblAlgn val="ctr"/>
  <c:lblOffset val="100"/>
  <c:noMultiLvlLbl val="0"/>
</c:catAx>
```

Related Schema Definitions

```
<xsd:complexType name="CT_CatAx">
  <xsd:sequence>
    <xsd:group    ref="EG_AxShared"/>
    <xsd:element name="auto"          type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="lblAlgn"       type="CT_LblAlgn"          minOccurs="0"/>
    <xsd:element name="lblOffset"     type="CT_LblOffset"        minOccurs="0"/>
    <xsd:element name="tickLblSkip"   type="CT_Skip"             minOccurs="0"/>
    <xsd:element name="tickMarkSkip"  type="CT_Skip"             minOccurs="0"/>
    <xsd:element name="noMultiLvlLbl" type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ValAx">
  <xsd:sequence>
    <xsd:group    ref="EG_AxShared"/>
    <xsd:element name="crossBetween"  type="CT_CrossBetween"     minOccurs="0"/>
    <xsd:element name="majorUnit"     type="CT_AxisUnit"         minOccurs="0"/>
    <xsd:element name="minorUnit"     type="CT_AxisUnit"         minOccurs="0"/>
    <xsd:element name="dispUnits"     type="CT_DispUnits"        minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DateAx">
  <xsd:sequence>
    <xsd:group    ref="EG_AxShared"/>
    <xsd:element name="auto"          type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="lblOffset"     type="CT_LblOffset"     minOccurs="0"/>
    <xsd:element name="baseTimeUnit"  type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="majorUnit"     type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="majorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="minorUnit"     type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="minorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="CT_SerAx">
  <xsd:sequence>
    <xsd:group    ref="EG_AxShared"/>
    <xsd:element name="tickLblSkip"  type="CT_Skip"          minOccurs="0"/>
    <xsd:element name="tickMarkSkip" type="CT_Skip"          minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_AxShared">
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="crosses"   type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_LblOffset">
  <xsd:attribute name="val" type="ST_LblOffset" default="100%"/>
</xsd:complexType>

<xsd:simpleType name="ST_LblOffset">
  <xsd:union memberTypes="ST_LblOffsetPercent ST_LblOffsetUShort"/>
</xsd:simpleType>

<xsd:simpleType name="ST_LblOffsetPercent">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="0*(([0-9])|([1-9][0-9])|([1-9][0-9][0-9])|1000)%"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_LblOffsetUShort">
  <xsd:restriction base="xsd:unsignedShort">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="1000"/>
  </xsd:restriction>
</xsd:simpleType>
```

Chart - Plot Data

The values and categories of each plot is specified by a data set. In the typical case, that data is composed of a sequence of categories and a sequence of series, where each series has a numeric value corresponding to each of the categories.

Candidate protocol

```
>>> plot = chart.plots[0]
>>> plot.categories
('Foo', 'Bar', 'Baz')
>>> len(plot.series)
3
>>> series = iter(plot.series)
>>> next(series).values
(1.2, 2.3, 3.4)
>>> next(series).values
(4.5, 5.6, 6.7)
>>> next(series).values
(7.8, 8.9, 9.0)
```

Feature Summary

- **Plot.categories** – Read/only for now. Returns tuple of string.
- **Series.values** – Read/only for now. Returns tuple of float.

Microsoft API

ChartGroup.CategoryCollection Returns all the visible categories in the chart group, or the specified visible category.

ChartGroup.SeriesCollection Returns all the series in the chart group.

Series.Values Returns or sets a collection of all the values in the series. Read/write Variant. Returns an array of float; accepts a spreadsheet formula or an array of numeric values.

Series.Formula Returns or sets the object's formula in A1-style notation. Read/write String.

XML specimens

Example series XML:

```
<c:ser>
  <!-- ... -->
  <c:cat>
    <c:strRef>
      <c:f>Sheet1!$A$2:$A$4</c:f>
      <c:strCache>
        <c:ptCount val="3"/>
        <c:pt idx="0">
          <c:v>Foo</c:v>
        </c:pt>
        <c:pt idx="1">
```

```
          <c:v>Bar</c:v>
        </c:pt>
        <c:pt idx="2">
          <c:v>Baz</c:v>
        </c:pt>
      </c:strCache>
    </c:strRef>
  </c:cat>
  <c:val>
    <c:numRef>
      <c:f>Sheet1!$B$2:$B$4</c:f>
      <c:numCache>
        <c:ptCount val="3"/>
        <c:pt idx="0">
          <c:v>1.2</c:v>
        </c:pt>
        <c:pt idx="1">
          <c:v>2.3</c:v>
        </c:pt>
        <c:pt idx="2">
          <c:v>3.4</c:v>
        </c:pt>
      </c:numCache>
    </c:numRef>
  </c:val>
</c:ser>
```

Related Schema Definitions

```
<xsd:complexType name="CT_BarSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"              type="CT_UnsignedInt"/>
    <xsd:element name="order"            type="CT_UnsignedInt"/>
    <xsd:element name="tx"               type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"             type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="pictureOptions"   type="CT_PictureOptions"    minOccurs="0"/>
    <xsd:element name="dPt"              type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"            type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"        type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"          type="CT_ErrBars"           minOccurs="0"/>
    <xsd:element name="cat"              type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"              type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="shape"            type="CT_Shape"             minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_AxDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="multiLvlStrRef" type="CT_MultiLvlStrRef"/>
      <xsd:element name="numRef"         type="CT_NumRef"/>
      <xsd:element name="numLit"         type="CT_NumData"/>
```

```
      <xsd:element name="strRef"         type="CT_StrRef"/>
      <xsd:element name="strLit"         type="CT_StrData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumDataSource">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="numRef" type="CT_NumRef"/>
      <xsd:element name="numLit" type="CT_NumData"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_StrRef">
  <xsd:sequence>
    <xsd:element name="f"        type="xsd:string"/>
    <xsd:element name="strCache" type="CT_StrData"       minOccurs="0"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_StrData">
  <xsd:sequence>
    <xsd:element name="ptCount" type="CT_UnsignedInt"   minOccurs="0"/>
    <xsd:element name="pt"      type="CT_StrVal"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="extLst"  type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_StrVal">
  <xsd:sequence>
    <xsd:element name="v" type="s:ST_Xstring"/>
  </xsd:sequence>
  <xsd:attribute name="idx" type="xsd:unsignedInt" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_NumRef">
  <xsd:sequence>
    <xsd:element name="f"        type="xsd:string"/>
    <xsd:element name="numCache" type="CT_NumData"       minOccurs="0"/>
    <xsd:element name="extLst"   type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumData">
  <xsd:sequence>
    <xsd:element name="formatCode" type="s:ST_Xstring"     minOccurs="0"/>
    <xsd:element name="ptCount"    type="CT_UnsignedInt"   minOccurs="0"/>
    <xsd:element name="pt"         type="CT_NumVal"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumVal">
```

```
  <xsd:sequence>
    <xsd:element name="v" type="s:ST_Xstring"/>
  </xsd:sequence>
  <xsd:attribute name="idx"        type="xsd:unsignedInt" use="required"/>
  <xsd:attribute name="formatCode" type="s:ST_Xstring"/>
</xsd:complexType>

<xsd:simpleType name="ST_Xstring">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

Chart - ValueAxis.major/minor_unit

The value axis has major and minor divisions, corresponding to where tick marks, tick labels, and gridlines appear, when present. How frequently these appear is determined by the major/minor units setting, specified as a floating point number of units to skip between divisions. These settings may be specified explictly, or PowerPoint can determine a sensible default based on the chart data. The latter setting is labeled 'Auto' in the UI. By default, major and minor unit are both set to 'Auto' on new charts.

Candidate protocol

The properties `ValueAxis.major_unit` and `ValueAxis.minor_unit` are used to access and change this setting.

`None` is used as an out-of-band value to signify *Auto* behavior. No separate boolean properties are required.

```
>>> value_axis = chart.value_axis
>>> value_axis.major_unit
None
>>> value_axis.major_unit = 10
>>> value_axis.major_unit
10.0
>>> value_axis.major_unit = -4.2
Traceback ...
ValueError: must be positive numeric value
>>> value_axis.major_unit = None
>>> value_axis.major_unit
None
```

Microsoft API

Axis.MajorUnit Returns or sets the major units for the value axis. Read/write Double.

Axis.MinorUnit Returns or sets the minor units on the value axis. Read/write Double.

Axis.MajorUnitIsAuto True if PowerPoint calculates the major units for the value axis. Read/write Boolean.

Axis.MinorUnitIsAuto True if PowerPoint calculates minor units for the value axis. Read/write Boolean.

PowerPoint behavior

Major and minor unit values are viewed and changed using the *Scale* pane of the *Format Axis* dialog. Checkboxes are used to set a value to *Auto*. Changing the floating point value causes the *Auto* checkbox to turn off.

XML Semantics

- Only a value axis axis or date axis can have a `<c:majorUnit>` or a `<c:minorUnit>` element.
- *Auto* behavior is signified by having no element for that unit.

XML specimens

Example value axis XML having an override for major unit:

```
<c:valAx>
  <c:axId val="-2101345848"/>
  <c:scaling>
    <c:orientation val="minMax"/>
  </c:scaling>
  <c:delete val="0"/>
  <c:axPos val="l"/>
  <c:majorGridlines/>
  <c:numFmt formatCode="General" sourceLinked="1"/>
  <c:majorTickMark val="out"/>
  <c:minorTickMark val="none"/>
  <c:tickLblPos val="nextTo"/>
  <c:crossAx val="-2030568888"/>
  <c:crosses val="autoZero"/>
  <c:crossBetween val="between"/>
  <c:majorUnit val="20.0"/>
</c:valAx>
```

Related Schema Definitions

```
<xsd:complexType name="CT_ValAx">
  <xsd:sequence>
    <xsd:group ref="EG_AxShared" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="crossBetween" type="CT_CrossBetween"  minOccurs="0"/>
    <xsd:element name="majorUnit"    type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="minorUnit"    type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="dispUnits"    type="CT_DispUnits"     minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DateAx">
  <xsd:sequence>
    <xsd:group    ref="EG_AxShared"/>
    <xsd:element name="auto"          type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="lblOffset"     type="CT_LblOffset"     minOccurs="0"/>
    <xsd:element name="baseTimeUnit"  type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="majorUnit"     type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="majorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="minorUnit"     type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="minorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:group name="EG_AxShared">
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="crosses"   type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_AxisUnit">
  <xsd:attribute name="val" type="ST_AxisUnit" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_AxisUnit">
  <xsd:restriction base="xsd:double">
    <xsd:minExclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>
```

Chart - Chart Type

There are 73 distinct chart types supported by PowerPoint.

These are implemented with 16 distinct elements (e.g. c:bar3DChart). Within an element tag name, they are further differentiated by child elements and attributes such as c:grouping and c:shape.

These differentiators are summaried below.

c:areaChart

- ./c:grouping{val=stacked} => AREA_STACKED
- ./c:grouping{val=percentStacked} => AREA_STACKED_100
- ./c:grouping{val=standard} => AREA
- . => AREA

c:area3DChart

- ./c:grouping{val=stacked} => THREE_D_AREA_STACKED

- ./c:grouping{val=percentStacked} => THREE_D_AREA_STACKED_100
- ./c:grouping{val=standard} => THREE_D_AREA
- . => THREE_D_AREA

c:barChart

- ./c:barDir{val=bar}
  - ./c:grouping{val=clustered} => BAR_CLUSTERED
  - ./c:grouping{val=stacked} => BAR_STACKED
  - ./c:grouping{val=percentStacked} => BAR_STACKED_100
- ./c:barDir{val=col}
  - ./c:grouping{val=clustered} => COLUMN_CLUSTERED
  - ./c:grouping{val=stacked} => COLUMN_STACKED
  - ./c:grouping{val=percentStacked} => COLUMN_STACKED_100

c:bar3DChart

- ./c:barDir{val=bar}
  - ./c:grouping{val=clustered}
    - ./c:shape{val=box} => THREE_D_BAR_CLUSTERED
    - ./c:shape{val=cone} => CONE_BAR_CLUSTERED
    - ./c:shape{val=cylinder} => CYLINDER_BAR_CLUSTERED
    - ./c:shape{val=pyramid} => PYRAMID_BAR_CLUSTERED
  - ./c:grouping{val=stacked}
    - ./c:shape{val=box} => THREE_D_BAR_STACKED
    - ./c:shape{val=cone} => CONE_BAR_STACKED
    - ./c:shape{val=cylinder} => CYLINDER_BAR_STACKED
    - ./c:shape{val=pyramid} => PYRAMID_BAR_STACKED
  - ./c:grouping{val=percentStacked}
    - ./c:shape{val=box} => THREE_D_BAR_STACKED_100
    - ./c:shape{val=cone} => CONE_BAR_STACKED_100
    - ./c:shape{val=cylinder} => CYLINDER_BAR_STACKED_100
    - ./c:shape{val=pyramid} => PYRAMID_BAR_STACKED_100
- ./c:barDir{val=col}
  - ./c:grouping{val=clustered}
    - ./c:shape{val=box} => THREE_D_COLUMN_CLUSTERED
    - ./c:shape{val=cone} => CONE_COL_CLUSTERED

        - ./c:shape{val=cylinder} => CYLINDER_COL_CLUSTERED
        - ./c:shape{val=pyramid} => PYRAMID_COL_CLUSTERED
    - ./c:grouping{val=stacked}
        - ./c:shape{val=box} => THREE_D_COLUMN_STACKED
        - ./c:shape{val=cone} => CONE_COL_STACKED
        - ./c:shape{val=cylinder} => CYLINDER_COL_STACKED
        - ./c:shape{val=pyramid} => PYRAMID_COL_STACKED
    - ./c:grouping{val=percentStacked}
        - ./c:shape{val=box} => THREE_D_COLUMN_STACKED_100
        - ./c:shape{val=cone} => CONE_COL_STACKED_100
        - ./c:shape{val=cylinder} => CYLINDER_COL_STACKED_100
        - ./c:shape{val=pyramid} => PYRAMID_COL_STACKED_100
    - ./c:grouping{val=standard}
        - ./c:shape{val=box} => THREE_D_COLUMN
        - ./c:shape{val=cone} => CONE_COL
        - ./c:shape{val=cylinder} => CYLINDER_COL
        - ./c:shape{val=pyramid} => PYRAMID_COL

c:bubbleChart

- ./c:bubble3D{val=0} => BUBBLE
- ./c:bubble3D{val=1} => BUBBLE_THREE_D_EFFECT

c:doughnutChart

- . => DOUGHNUT
- ./c:ser/c:explosion{val>0} => DOUGHNUT_EXPLODED

c:lineChart

- ./c:grouping{val=standard}
    - ./c:ser/c:marker/c:symbol{val=none} => LINE
    - ./c:marker{val=1} => LINE_MARKERS
- ./c:grouping{val=stacked}
    - ./c:marker{val=1} => LINE_MARKERS_STACKED
    - ./c:ser/c:marker/c:symbol{val=none} => LINE_STACKED
- ./c:grouping{val=percentStacked}
    - ./c:marker{val=1} => LINE_MARKERS_STACKED_100

  - ./c:ser/c:marker/c:symbol{val=none} => LINE_STACKED_100

c:line3DChart

- . => THREE_D_LINE

c:ofPieChart

- ./c:ofPieType{val=bar} => BAR_OF_PIE
- ./c:ofPieType{val=pie} => PIE_OF_PIE

c:pieChart

- . => PIE
- ./c:ser/c:explosion{val>0} => PIE_EXPLODED

c:pie3DChart

- . => THREE_D_PIE
- ./c:ser/c:explosion{val>0} => THREE_D_PIE_EXPLODED

c:radarChart

- ./c:radarStyle{val=standard} => RADAR
- ./c:radarStyle{val=filled} => RADAR_FILLED
- ./c:radarStyle{val=marker} => RADAR_MARKERS

c:scatterChart

- ./c:scatterStyle{val=lineMarker} => XY_SCATTER
- has to do with ./c:ser/c:spPr/a:ln/a:noFill
- ./c:scatterStyle{val=lineMarker} => XY_SCATTER_LINES
- ./c:scatterStyle{val=line} => XY_SCATTER_LINES_NO_MARKERS
- ./c:scatterStyle{val=smoothMarker} => XY_SCATTER_SMOOTH
- ./c:scatterStyle{val=smooth} => XY_SCATTER_SMOOTH_NO_MARKERS
- check all these to verify

c:stockChart

- ./? => STOCK_HLC
- ./? => STOCK_OHLC
- ./? => STOCK_VHLC
- ./? => STOCK_VOHLC
- possibly related to 3 vs. 4 series. VOHLC has a second plot and axis for volume

c:surface3DChart

- ./c:wireframe{val=0} => SURFACE_TOP_VIEW
- ./c:wireframe{val=1} => SURFACE_TOP_VIEW_WIREFRAME

c:surfaceChart

- ./c:wireframe{val=0} => SURFACE
- ./c:wireframe{val=1} => SURFACE_WIREFRAME

XML specimens

```
<c:barChart>
  <c:barDir val="col"/>
  <c:grouping val="clustered"/>
  <c:ser>
    <c:idx val="0"/>
    <c:order val="0"/>
    <c:cat>...</c:cat>
    <c:val>...</c:val>
  </c:ser>
  <c:axId val="-2068027336"/>
  <c:axId val="-2113994440"/>
</c:barChart>
```

Related Schema Definitions

```
<!-- implemented chart types -->

<xsd:complexType name="CT_AreaChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="grouping"   type="CT_Grouping"      minOccurs="0"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_AreaSer"       minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
```

```
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_BarChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="barDir"     type="CT_BarDir"/>
    <xsd:element name="grouping"   type="CT_BarGrouping"   minOccurs="0"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_BarSer"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="gapWidth"   type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="overlap"    type="CT_Overlap"       minOccurs="0"/>
    <xsd:element name="serLines"   type="CT_ChartLines"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_BubbleChart">
  <xsd:sequence>
    <xsd:element name="varyColors"     type="CT_Boolean"        minOccurs="0"/>
    <xsd:element name="ser"            type="CT_BubbleSer"      minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"          type="CT_DLbls"          minOccurs="0"/>
    <xsd:element name="bubble3D"       type="CT_Boolean"        minOccurs="0"/>
    <xsd:element name="bubbleScale"    type="CT_BubbleScale"    minOccurs="0"/>
    <xsd:element name="showNegBubbles" type="CT_Boolean"        minOccurs="0"/>
    <xsd:element name="sizeRepresents" type="CT_SizeRepresents" minOccurs="0"/>
    <xsd:element name="axId"           type="CT_UnsignedInt"    minOccurs="2"␣
↪maxOccurs="2"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"  minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DoughnutChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="varyColors"    type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"           type="CT_PieSer"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"         type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="firstSliceAng" type="CT_FirstSliceAng" minOccurs="0"/>
    <xsd:element name="holeSize"      type="CT_HoleSize"      minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_LineChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="grouping"   type="CT_Grouping"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_LineSer"       minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines"    minOccurs="0"/>
```

```
    <xsd:element name="hiLowLines" type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="upDownBars" type="CT_UpDownBars"    minOccurs="0"/>
    <xsd:element name="marker"     type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="smooth"     type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_PieChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="varyColors"    type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"           type="CT_PieSer"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"         type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="firstSliceAng" type="CT_FirstSliceAng" minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_RadarChart">
  <xsd:sequence>
    <xsd:element name="radarStyle" type="CT_RadarStyle"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_RadarSer"      minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ScatterChart">
  <xsd:sequence>
    <xsd:element name="scatterStyle" type="CT_ScatterStyle"/>
    <xsd:element name="varyColors"   type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"          type="CT_ScatterSer"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"        type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="axId"         type="CT_UnsignedInt"   minOccurs="2" maxOccurs=
↪"2"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>


<!-- not-yet-implemented chart types -->

<xsd:complexType name="CT_Area3DChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="grouping"   type="CT_Grouping"      minOccurs="0"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_AreaSer"       minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines"    minOccurs="0"/>
```

```
    <xsd:element name="gapDepth"   type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="3
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Bar3DChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="barDir"     type="CT_BarDir"/>
    <xsd:element name="grouping"   type="CT_BarGrouping"   minOccurs="0"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_BarSer"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="gapWidth"   type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="gapDepth"   type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="shape"      type="CT_Shape"         minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="3
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Line3DChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="grouping"   type="CT_Grouping"/>
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_LineSer"       minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="gapDepth"   type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="3" maxOccurs="3
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_OfPieChart">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="ofPieType"     type="CT_OfPieType"/>
    <xsd:element name="varyColors"    type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"           type="CT_PieSer"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"         type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="gapWidth"      type="CT_GapAmount"     minOccurs="0"/>
    <xsd:element name="splitType"     type="CT_SplitType"     minOccurs="0"/>
    <xsd:element name="splitPos"      type="CT_Double"        minOccurs="0"/>
    <xsd:element name="custSplit"     type="CT_CustSplit"     minOccurs="0"/>
    <xsd:element name="secondPieSize" type="CT_SecondPieSize" minOccurs="0"/>
    <xsd:element name="serLines"      type="CT_ChartLines"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>

<xsd:complexType name="CT_Pie3DChart">  <!-- denormalized -->
  <xsd:sequence>
```

```
    <xsd:element name="varyColors" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"        type="CT_PieSer"        minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_StockChart">
  <xsd:sequence>
    <xsd:element name="ser"        type="CT_LineSer"       minOccurs="3" maxOccurs="4
↪"/>
    <xsd:element name="dLbls"      type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="dropLines"  type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="hiLowLines" type="CT_ChartLines"    minOccurs="0"/>
    <xsd:element name="upDownBars" type="CT_UpDownBars"    minOccurs="0"/>
    <xsd:element name="axId"       type="CT_UnsignedInt"   minOccurs="2" maxOccurs="2
↪"/>
    <xsd:element name="extLst"     type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SurfaceChart">
  <xsd:sequence>
    <xsd:element name="wireframe" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"       type="CT_SurfaceSer"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="bandFmts"  type="CT_BandFmts"      minOccurs="0"/>
    <xsd:element name="axId"      type="CT_UnsignedInt"   minOccurs="2" maxOccurs="3"/
↪>
    <xsd:element name="extLst"    type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Surface3DChart">
  <xsd:sequence>
    <xsd:element name="wireframe" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="ser"       type="CT_SurfaceSer"    minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="bandFmts"  type="CT_BandFmts"      minOccurs="0"/>
    <xsd:element name="axId"      type="CT_UnsignedInt"   minOccurs="3" maxOccurs="3"/
↪>
    <xsd:element name="extLst"    type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Chart - InvertIfNegative

Can be set on a bar series or on a data point. It's not clear what its meaning is on a non-bar data point. `c:barSer/c:invertIfNegative` is the target element. Also valid on a bubble series.

Protocol

```
>>> assert isinstance(plot, pptx.chart.plot.BarPlot)
>>> series = plot.series[0]
>>> series
<pptx.chart.series.BarSeries instance at 0x1deadbeef>
>>> series.invert_if_negative
True
>>> series.invert_if_negative = False
>>> series.invert_if_negative
False
```

Semantics

- Defaults to `True` if the `<c:invertIfNegative>` element is not present.
- Defaults to `True` if the parent element is present but the *val* attribute is not.

PowerPoint® behavior

In my tests, the `<c:invertIfNegative>` element is always present in a new bar chart and always explicitly initialized to False.

XML specimens

A series element from a simple column chart (single series):

```
<c:ser>
  <c:idx val="0"/>
  <c:order val="0"/>
  <c:tx>
    <!-- ... -->
  </c:tx>
  <c:invertIfNegative val="0"/>
  <c:dLbls>
    <!-- ... -->
  </c:dLbls>
  <c:cat>
    <!-- ... -->
  </c:cat>
  <c:val>
    <!-- ... -->
  </c:val>
</c:ser>
```

Related Schema Definitions

```
<xsd:complexType name="CT_BarSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"              type="CT_UnsignedInt"/>
    <xsd:element name="order"            type="CT_UnsignedInt"/>
```

```
    <xsd:element name="tx"              type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="pictureOptions"  type="CT_PictureOptions"    minOccurs="0"/>
    <xsd:element name="dPt"             type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"           type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"       type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"         type="CT_ErrBars"           minOccurs="0"/>
    <xsd:element name="cat"             type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"             type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="shape"           type="CT_Shape"             minOccurs="0"/>
    <xsd:element name="extLst"          type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_BarSer">
  <xsd:sequence>
    <xsd:group   ref="EG_SerShared"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"        minOccurs="0"/>
    <xsd:element name="pictureOptions"   type="CT_PictureOptions" minOccurs="0"/>
    <xsd:element name="dPt"              type="CT_DPt"            minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"            type="CT_DLbls"          minOccurs="0"/>
    <xsd:element name="trendline"        type="CT_Trendline"      minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"          type="CT_ErrBars"        minOccurs="0"/>
    <xsd:element name="cat"              type="CT_AxDataSource"   minOccurs="0"/>
    <xsd:element name="val"              type="CT_NumDataSource"  minOccurs="0"/>
    <xsd:element name="shape"            type="CT_Shape"          minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList"  minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_BubbleSer">
  <xsd:sequence>
    <xsd:group ref="EG_SerShared"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="dPt"              type="CT_DPt"           minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"            type="CT_DLbls"         minOccurs="0"/>
    <xsd:element name="trendline"        type="CT_Trendline"     minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"          type="CT_ErrBars"       minOccurs="0"␣
↪maxOccurs="2"/>
    <xsd:element name="xVal"             type="CT_AxDataSource"  minOccurs="0"/>
    <xsd:element name="yVal"             type="CT_NumDataSource" minOccurs="0"/>
    <xsd:element name="bubbleSize"       type="CT_NumDataSource" minOccurs="0"/>
    <xsd:element name="bubble3D"         type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Chart - BaseAxis.has_gridlines

A chart axis can have gridlines that extend the axis tick marks across the body of the plot area. Gridlines for major and minor tick marks can be controlled separately. Both vertical and horizontal axes can have gridlines.

Candidate protocol

```
>>> value_axis = chart.value_axis
>>> value_axis.has_major_gridlines
False
>>> value_axis.has_major_gridlines = True
>>> value_axis.has_major_gridlines
True
>>> category_axis = chart.category_axis
>>> category_axis.has_minor_gridlines
False
>>> category_axis.has_minor_gridlines = True
>>> category_axis.has_minor_gridlines
True
```

Can be implemented in BaseAxis since the controlling element appears in `EG_AxShared`.

Microsoft API

Axis.HasMajorGridlines True if the axis has major gridlines. Read/write Boolean.

Axis.HasMinorGridlines True if the axis has minor gridlines. Read/write Boolean.

Axis.MajorGridlines Returns the major gridlines for the specified axis. Read-only Gridlines.

Axis.MinorGridlines Returns the minor gridlines for the specified axis. Read-only Gridlines.

Gridlines.Border

Gridlines.Format.Line

Gridlines.Format.Shadow

PowerPoint behavior

- Turning gridlines on and off is controlled from the ribbon (on the Mac at least)
- Each set of gridlines (horz/vert, major/minor) can be formatted individually.
- The formatting dialog has panes for Line, Shadow, and Glow & Soft Edges

XML Semantics

- Each axis can have a `<c:majorGridlines>` and a `<c:minorGridlines>` element.
- The corresponding gridlines appear if the element is present and not if it doesn't.
- Line formatting, like width and color, is specified in the child `<c:spPr>` element.

XML specimens

Example axis XML for a single-series line plot:

```
<c:catAx>
  <c:axId val="-2097691448"/>
  <c:scaling>
    <c:orientation val="minMax"/>
  </c:scaling>
  <c:delete val="0"/>
  <c:axPos val="b"/>
  <c:majorGridlines/>
  <c:minorGridlines/>
  <c:numFmt formatCode="General" sourceLinked="0"/>
  <c:majorTickMark val="out"/>
  <c:minorTickMark val="none"/>
  <c:tickLblPos val="nextTo"/>
  <c:txPr>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr>
        <a:defRPr sz="1000"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:crossAx val="-2097683336"/>
  <c:crosses val="autoZero"/>
  <c:auto val="1"/>
  <c:lblAlgn val="ctr"/>
  <c:lblOffset val="100"/>
  <c:noMultiLvlLbl val="0"/>
</c:catAx>
```

Related Schema Definitions

```
<xsd:complexType name="CT_CatAx">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice                                                    minOccurs="0">
      <xsd:element name="crosses"      type="CT_Crosses"/>
      <xsd:element name="crossesAt"    type="CT_Double"/>
```

```
    </xsd:choice>
    <xsd:element name="auto"          type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="lblAlgn"       type="CT_LblAlgn"          minOccurs="0"/>
    <xsd:element name="lblOffset"     type="CT_LblOffset"        minOccurs="0"/>
    <xsd:element name="tickLblSkip"   type="CT_Skip"             minOccurs="0"/>
    <xsd:element name="tickMarkSkip"  type="CT_Skip"             minOccurs="0"/>
    <xsd:element name="noMultiLvlLbl" type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ValAx">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="axId"          type="CT_UnsignedInt"/>
    <xsd:element name="scaling"       type="CT_Scaling"/>
    <xsd:element name="delete"        type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="axPos"         type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"      minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"      minOccurs="0"/>
    <xsd:element name="title"         type="CT_Title"            minOccurs="0"/>
    <xsd:element name="numFmt"        type="CT_NumFmt"           minOccurs="0"/>
    <xsd:element name="majorTickMark" type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="minorTickMark" type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="tickLblPos"    type="CT_TickLblPos"       minOccurs="0"/>
    <xsd:element name="spPr"          type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"          type="a:CT_TextBody"       minOccurs="0"/>
    <xsd:element name="crossAx"       type="CT_UnsignedInt"/>
    <xsd:choice                                                  minOccurs="0">
      <xsd:element name="crosses"   type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
    <xsd:element name="crossBetween"  type="CT_CrossBetween"     minOccurs="0"/>
    <xsd:element name="majorUnit"     type="CT_AxisUnit"         minOccurs="0"/>
    <xsd:element name="minorUnit"     type="CT_AxisUnit"         minOccurs="0"/>
    <xsd:element name="dispUnits"     type="CT_DispUnits"        minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DateAx">
  <xsd:sequence>
    <xsd:group    ref="EG_AxShared"/>
    <xsd:element name="auto"          type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="lblOffset"     type="CT_LblOffset"     minOccurs="0"/>
    <xsd:element name="baseTimeUnit"  type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="majorUnit"     type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="majorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="minorUnit"     type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="minorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SerAx">
  <xsd:sequence>
    <xsd:group    ref="EG_AxShared"/>
    <xsd:element name="tickLblSkip"  type="CT_Skip"          minOccurs="0"/>
    <xsd:element name="tickMarkSkip" type="CT_Skip"          minOccurs="0"/>
```

```
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_AxShared">
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="crosses"   type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_ChartLines">
  <xsd:sequence>
    <xsd:element name="spPr" type="a:CT_ShapeProperties" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Chart - Series

A series in a sequence of *points* that correspond to the categories of a plot. A chart may have more than one series, which gives rise, for example, to a *clustered column chart* or a line chart with multiple lines plotted.

A series belongs to a *plot*. When a chart has multiple plots, such as a bar chart with a superimposed line plot, each of the series in the chart belong to one plot or the other.

Although the Microsoft API has a method on Chart to access all the series in a chart, it also has the same method for accessing the series of a plot (ChartGroup in MS API).

There are eight distinct series types, corresponding to major chart types. They all share a common base of attributes and others appear on one or more of the types.

Series Access

Series are perhaps most naturally accessed from a plot, to which they belong:

```
>>> plot.series
<pptx.chart.series.SeriesCollection instance at x11091e750>
```

However, there is also a property on *Chart* which allows access to all the series in the chart:

```
>>> chart.series
<pptx.chart.series.SeriesCollection instance at x11091f970>
```

Each series in a chart has an explicit sequence indicator, the value of its required *c:order* child element. The series for a plot appear in order of this value. The series for a chart appear in plot order, then their order within that plot, such that all series for the first plot appear before those in the next plot, and so on.

Properties

Series.format

All series have an optional *c:spPr* element that control the drawing shape properties of the series such as fill and line, including transparency and shadow.

Series.points

Perhaps counterintuitively, a Point object does not provide access to all the attributes one might think. It only provides access to attributes of the visual representation of the point in that chart, such as the color, datum label, or marker. It does not provide access to the data point values, such as the Y value or the bubble size.

Surface charts do not have a distinct data point representation (rather just an inflection in the surface. So series of surface charts will not have the .points member. Since surface charts are not yet implemented, this will come into play sometime later.

Note that bubble and XY have a different way of organizing their data points so have a distinct implementation from that of category charts.

Implementation notes:

- Introduce _BaseCategorySeries and subclass all category series types from it. Add tests to test inheritance. No acceptance test since this is internals-only.
- It's possible the only requirement is to create CategoryPoints. The rest of the implementation might work all on its own. Better spike it and see.

Protocol

```
>>> assert isinstance(chart, pptx.chart.chart.Chart)
>>> plot = chart.plots[0]
>>> plot
<pptx.chart.plot.BarChart instance at 0x1deadbeef>
>>> series = plot.series[0]
>>> series
<pptx.chart.series.BarSeries instance at 0x...>
>>> fill = series.fill
>>> fill
<pptx.dml.fill.FillFormat instance at 0x...>
>>> fill.type
None
>>> fill.solid()
>>> fill.fore_color.rgb = RGB(0x3F, 0x2C, 0x36)
# also available are theme_color and brightness
```

XML specimens

A series element from a simple column chart (single series):

```
<c:ser>
  <c:idx val="0"/>
  <c:order val="0"/>
  <c:tx>
    <c:strRef>
      <c:f>Sheet1!$A$2</c:f>
      <c:strCache>
        <c:ptCount val="1"/>
        <c:pt idx="0">
          <c:v>Base</c:v>
        </c:pt>
      </c:strCache>
    </c:strRef>
  </c:tx>
  <c:invertIfNegative val="0"/>
  <c:dLbls>
    <c:numFmt formatCode="General" sourceLinked="0"/>
    <c:spPr>
      <a:noFill/>
      <a:ln>
        <a:noFill/>
      </a:ln>
      <a:effectLst/>
    </c:spPr>
    <c:txPr>
      <a:bodyPr/>
      <a:lstStyle/>
      <a:p>
        <a:pPr>
          <a:defRPr sz="1000"/>
        </a:pPr>
        <a:endParaRPr lang="en-US"/>
      </a:p>
    </c:txPr>
    <c:dLblPos val="outEnd"/>
    <c:showLegendKey val="0"/>
    <c:showVal val="1"/>
    <c:showCatName val="0"/>
    <c:showSerName val="0"/>
    <c:showPercent val="0"/>
    <c:showBubbleSize val="0"/>
    <c:showLeaderLines val="0"/>
    <c:extLst xmlns:c15="http://schemas.microsoft.com/office/drawing/2012/chart"
              xmlns:c14="http://schemas.microsoft.com/office/drawing/2007/8/2/chart"
              xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006">
      <c:ext xmlns:c15="http://schemas.microsoft.com/office/drawing/2012/chart"
             uri="{CE6537A1-D6FC-4f65-9D91-7224C49458BB}">
        <c15:layout/>
        <c15:showLeaderLines val="0"/>
      </c:ext>
    </c:extLst>
  </c:dLbls>
  <c:cat>
    <c:strRef>
```

```
      <c:f>Sheet1!$B$1:$F$1</c:f>
      <c:strCache>
        <c:ptCount val="5"/>
        <c:pt idx="0">
          <c:v>Très probable</c:v>
        </c:pt>
        <c:pt idx="1">
          <c:v>Plutôt probable</c:v>
        </c:pt>
        <c:pt idx="2">
          <c:v>Plutôt improbable</c:v>
        </c:pt>
        <c:pt idx="3">
          <c:v>Très improbable</c:v>
        </c:pt>
        <c:pt idx="4">
          <c:v>Je ne sais pas</c:v>
        </c:pt>
      </c:strCache>
    </c:strRef>
  </c:cat>
  <c:val>
    <c:numRef>
      <c:f>Sheet1!$B$2:$F$2</c:f>
      <c:numCache>
        <c:formatCode>0</c:formatCode>
        <c:ptCount val="5"/>
        <c:pt idx="0">
          <c:v>19.0</c:v>
        </c:pt>
        <c:pt idx="1">
          <c:v>13.0</c:v>
        </c:pt>
        <c:pt idx="2">
          <c:v>10.0</c:v>
        </c:pt>
        <c:pt idx="3">
          <c:v>46.0</c:v>
        </c:pt>
        <c:pt idx="4">
          <c:v>12.0</c:v>
        </c:pt>
      </c:numCache>
    </c:numRef>
  </c:val>
</c:ser>
```

Related Schema Definitions

```
<xsd:complexType name="CT_AreaSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"            type="CT_UnsignedInt"/>
    <xsd:element name="order"          type="CT_UnsignedInt"/>
    <xsd:element name="tx"             type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="pictureOptions" type="CT_PictureOptions"    minOccurs="0"/>
```

```
    <xsd:element name="dPt"            type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"          type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"      type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"        type="CT_ErrBars"           minOccurs="0"␣
↪maxOccurs="2"/>
    <xsd:element name="cat"            type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"            type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_BarSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"             type="CT_UnsignedInt"/>
    <xsd:element name="order"           type="CT_UnsignedInt"/>
    <xsd:element name="tx"              type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="pictureOptions"  type="CT_PictureOptions"    minOccurs="0"/>
    <xsd:element name="dPt"             type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"           type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"       type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"         type="CT_ErrBars"           minOccurs="0"/>
    <xsd:element name="cat"             type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"             type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="shape"           type="CT_Shape"             minOccurs="0"/>
    <xsd:element name="extLst"          type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_BubbleSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"             type="CT_UnsignedInt"/>
    <xsd:element name="order"           type="CT_UnsignedInt"/>
    <xsd:element name="tx"              type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"            type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="dPt"             type="CT_DPt"               minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="dLbls"           type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline"       type="CT_Trendline"         minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="errBars"         type="CT_ErrBars"           minOccurs="0"␣
↪maxOccurs="2"/>
    <xsd:element name="xVal"            type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="yVal"            type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="bubbleSize"      type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="bubble3D"        type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"          type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_LineSer">  <!-- denormalized -->
  <xsd:sequence>
```

```
    <xsd:element name="idx"       type="CT_UnsignedInt"/>
    <xsd:element name="order"     type="CT_UnsignedInt"/>
    <xsd:element name="tx"        type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"      type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker"    type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"       type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"     type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline" type="CT_Trendline"         minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="errBars"   type="CT_ErrBars"           minOccurs="0"/>
    <xsd:element name="cat"       type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"       type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="smooth"    type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_PieSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"       type="CT_UnsignedInt"/>
    <xsd:element name="order"     type="CT_UnsignedInt"/>
    <xsd:element name="tx"        type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"      type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="explosion" type="CT_UnsignedInt"       minOccurs="0"/>
    <xsd:element name="dPt"       type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"     type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="cat"       type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"       type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_RadarSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"    type="CT_UnsignedInt"/>
    <xsd:element name="order"  type="CT_UnsignedInt"/>
    <xsd:element name="tx"     type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker" type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"    type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="dLbls"  type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="cat"    type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"    type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ScatterSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"       type="CT_UnsignedInt"/>
    <xsd:element name="order"     type="CT_UnsignedInt"/>
    <xsd:element name="tx"        type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"      type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="marker"    type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="dPt"       type="CT_DPt"               minOccurs="0" maxOccurs=
↪"unbounded"/>
```

```
    <xsd:element name="dLbls"     type="CT_DLbls"             minOccurs="0"/>
    <xsd:element name="trendline" type="CT_Trendline"         minOccurs="0" maxOccurs=
↪"unbounded"/>
    <xsd:element name="errBars"   type="CT_ErrBars"           minOccurs="0" maxOccurs=
↪"2"/>
    <xsd:element name="xVal"      type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="yVal"      type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="smooth"    type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SurfaceSer">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="idx"    type="CT_UnsignedInt"/>
    <xsd:element name="order"  type="CT_UnsignedInt"/>
    <xsd:element name="tx"     type="CT_SerTx"             minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="cat"    type="CT_AxDataSource"      minOccurs="0"/>
    <xsd:element name="val"    type="CT_NumDataSource"     minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DPt">
  <xsd:sequence>
    <xsd:element name="idx"              type="CT_UnsignedInt"/>
    <xsd:element name="invertIfNegative" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="marker"           type="CT_Marker"            minOccurs="0"/>
    <xsd:element name="bubble3D"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="explosion"        type="CT_UnsignedInt"       minOccurs="0"/>
    <xsd:element name="spPr"             type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="pictureOptions"   type="CT_PictureOptions"    minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Chart - Embedded Worksheet

The data for a chart in PowerPoint is stored in an embedded Excel spreadsheet.

Spike approach

- [ ] driver with one blank slide, builder column chart single series.
- [ ] Can edit Excel data in resulting file
- [ ] resulting file has an embedded Excel package of expected name
- [ ] XML uses worksheet references, not just cached data

Proposed protocol

```
>>> graphic_frame = prs.slides[0].shapes[0]
>>> chart = graphic_frame.chart
>>> chart_data = chart.chart_data
>>> chart_data
<pptx.chart.chart.ChartData instance at 0xdeadbeef1>
>>> chart_data.update_from_xlsx_stream(xlsx_stream)
```

MS API Protocol

```
>>> chart = prs.Slides(1).Shapes(1).Chart
>>> chart_data = chart.ChartData
>>> workbook = chart_data.Workbook
>>> worksheet = Workbook.Worksheets(1)
```

ChartData objects

BreakLink Removes the link between the data for a chart and a Microsoft Excel workbook.

IsLinked True if the data for the chart is linked to an external Microsoft Excel workbook. Read-only Boolean.

Workbook Returns the workbook that contains the chart data associated with the chart. Read-only Object.

Code sketches

`ChartPart.xlsx_blob = blob`:

```
@xlsx_blob.setter
def xlsx_blob(self, blob):
    xlsx_part = self.xlsx_part
    if xlsx_part:
        xlsx_part.blob = blob
    else:
        xlsx_part = EmbeddedXlsxPart.new(blob, self.package)
        rId = self.relate_to(xlsx_part, RT.PACKAGE)
        externalData = self._element.get_or_add_externalData
        externalData.rId = rId
```

`@classmethod EmbeddedXlsxPart.new(cls, blob, package)`:

```
partname = cls.next_partname(package)
content_type = CT.SML_SHEET
xlsx_part = EmbeddedXlsxPart(partname, content_type, blob, package)
return xlsx_part
```

`ChartPart.add_or_replace_xlsx(xlsx_stream)`:

```
xlsx_part = self.get_or_add_xlsx_part()
xlsx_stream.seek(0)
xlsx_bytes = xlsx_stream.read()
xlsx_part.blob = xlsx_bytes
```

ChartPart.xlsx_part:

```
externalData = self._element.externalData
if externalData is None:
    raise ValueError("chart has no embedded worksheet")
rId = externalData.rId
xlsx_part = self.related_parts[rId]
return xlsx_part

# later ...

xlsx_stream = BytesIO(xlsx_part.blob)
xlsx_package = OpcPackage.open(xlsx_stream)
workbook_part = xlsx_package.main_document
```

- Maybe can implement just a few Excel parts, enough to access and manipulate the data necessary. Like Workbook (start part I think) and Worksheet.
- What about linked rather than embedded Worksheet?

XML specimens

simple column chart:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes'?>
<c:chartSpace
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <c:date1904 val="0"/>
  <c:lang val="en-US"/>
  <c:roundedCorners val="0"/>
  <mc:AlternateContent xmlns:mc="http://schemas.openxmlformats.org/markup-
↪compatibility/2006">
    <mc:Choice xmlns:c14="http://schemas.microsoft.com/office/drawing/2007/8/2/chart"
               Requires="c14">
      <c14:style val="102"/>
    </mc:Choice>
    <mc:Fallback>
      <c:style val="2"/>
    </mc:Fallback>
  </mc:AlternateContent>
  <c:chart>
    <!-- 179 rows elided -->
  </c:chart>
  <c:txPr>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr>
        <a:defRPr sz="1800"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:externalData r:id="rId1">
```

```
    <c:autoUpdate val="0"/>
  </c:externalData>
</c:chartSpace>
```

Related Schema Definitions

```
<xsd:complexType name="CT_ChartSpace">
  <xsd:sequence>
    <xsd:element name="date1904"       type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="lang"           type="CT_TextLanguageID"    minOccurs="0"/>
    <xsd:element name="roundedCorners" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="style"          type="CT_Style"             minOccurs="0"/>
    <xsd:element name="clrMapOvr"      type="a:CT_ColorMapping"    minOccurs="0"/>
    <xsd:element name="pivotSource"    type="CT_PivotSource"       minOccurs="0"/>
    <xsd:element name="protection"     type="CT_Protection"        minOccurs="0"/>
    <xsd:element name="chart"          type="CT_Chart"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="externalData"   type="CT_ExternalData"      minOccurs="0"/>
    <xsd:element name="printSettings"  type="CT_PrintSettings"     minOccurs="0"/>
    <xsd:element name="userShapes"     type="CT_RelId"             minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ExternalData">
  <xsd:sequence>
    <xsd:element name="autoUpdate" type="CT_Boolean" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute ref="r:id" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_Boolean">
  <xsd:attribute name="val" type="xsd:boolean" default="true"/>
</xsd:complexType>
```

Chart Axes

PowerPoint chart axes come in four varieties: category axis, value axis, date axis, and series axis. A series axis only appears on a 3D chart and is also known as its depth axis.

A chart may have two category axes and/or two value axes. The second axis, if there is one, is known as the *secondary category axis* or *secondary value axis*.

A category axis may appear as either the horizontal or vertical axis, depending upon the chart type. Likewise for a value axis.

PowerPoint behavior

Tick label position

Proposed python-pptx protocol:

```
>>> axis.tick_label_position
XL_TICK_LABEL_POSITION.NEXT_TO_AXIS
>>> axis.tick_label_position = XL_TICK_LABEL_POSITION.LOW
>>> axis.tick_label_position
XL_TICK_LABEL_POSITION.LOW
```

MS API protocol:

```
>>> axis.TickLabelPosition
xlTickLabelPositionNextToAxis
>>> axis.TickLabelPosition = xlTickLabelPositionLow

c:catAx/c:tickLblPos{val=nextTo}
```

Option "none" causes tick labels to be hidden.

Default when no `<c:tickLblPos>` element is present is nextTo. Same if element is present with no `val` attribute.

TickLabels.number_format

Proposed python-pptx protocol:

```
>>> tick_labels = axis.tick_labels
>>> tick_labels.number_format
'General'
>>> tick_labels.number_format_is_linked
True
>>> tick_labels.number_format = '#,##0.00'
>>> tick_labels.number_format_is_linked = False
```

Tick-mark label text for the category axis comes from the name of the associated category in the chart. The default tick-mark label text for the category axis is the number that indicates the position of the category relative to the left end of this axis. To change the number of unlabeled tick marks between tick-mark labels, you must change the TickLabelSpacing property for the category axis.

Tick-mark label text for the value axis is calculated based on the MajorUnit, MinimumScale, and MaximumScale properties of the value axis. To change the tick-mark label text for the value axis, you must change the values of these properties.

MS API protocol:

```
>>> tick_labels = axis.TickLabels
>>> tick_labels.NumberFormat
'General'
>>> tick_labels.NumberFormatLinked
True
>>> tick_labels.NumberFormatLinked = False
>>> tick_labels.NumberFormat = "#,##0.00"

# output of 1234.5678 is '1,234.57'
```

When `sourceLinked` attribute is True, UI shows "General" number format category regardless of contents of `formatCode` attribute.

The `sourceLinked` attribute defaults to True when the `<c:numFmt>` element is present but that attribute is omitted.

When the <c:numFmt> element is not present, the behavior is as though the element <c:numFmt formatCode="General" sourceLinked="0"/> was present.

The default PowerPoint chart contains this numFmt element:

```
``<c:numFmt formatCode="General" sourceLinked="1"/>``.
```

_BaseAxis.visible property

<c:delete val="0"/> element

- when delete element is absent, the default value is True
- when val attribute is absent, the default value is True

XML specimens

Example axis XML for a single-series line plot:

```
<c:catAx>
  <c:axId val="-2097691448"/>
  <c:scaling>
    <c:orientation val="minMax"/>
  </c:scaling>
  <c:delete val="0"/>
  <c:axPos val="b"/>
  <c:numFmt formatCode="General" sourceLinked="0"/>
  <c:majorTickMark val="out"/>
  <c:minorTickMark val="none"/>
  <c:tickLblPos val="nextTo"/>
  <c:txPr>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr>
        <a:defRPr sz="1000"/>
      </a:pPr>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </c:txPr>
  <c:crossAx val="-2097683336"/>
  <c:crosses val="autoZero"/>
  <c:auto val="1"/>
  <c:lblAlgn val="ctr"/>
  <c:lblOffset val="100"/>
  <c:noMultiLvlLbl val="0"/>
</c:catAx>
```

Related Schema Definitions

```
<xsd:complexType name="CT_PlotArea">
  <xsd:sequence>
    <!-- 17 others -->
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
```

```
      <xsd:element name="valAx"  type="CT_ValAx"/>
      <xsd:element name="catAx"  type="CT_CatAx"/>
      <xsd:element name="dateAx" type="CT_DateAx"/>
      <xsd:element name="serAx"  type="CT_SerAx"/>
    </xsd:choice>
    <xsd:element name="dTable" type="CT_DTable"            minOccurs="0"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_CatAx">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice                                                    minOccurs="0">
      <xsd:element name="crosses"      type="CT_Crosses"/>
      <xsd:element name="crossesAt"    type="CT_Double"/>
    </xsd:choice>
    <xsd:element name="auto"           type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="lblAlgn"        type="CT_LblAlgn"           minOccurs="0"/>
    <xsd:element name="lblOffset"      type="CT_LblOffset"         minOccurs="0"/>
    <xsd:element name="tickLblSkip"    type="CT_Skip"              minOccurs="0"/>
    <xsd:element name="tickMarkSkip"   type="CT_Skip"              minOccurs="0"/>
    <xsd:element name="noMultiLvlLbl"  type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ValAx">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="axId"           type="CT_UnsignedInt"/>
    <xsd:element name="scaling"        type="CT_Scaling"/>
    <xsd:element name="delete"         type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="axPos"          type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"        minOccurs="0"/>
    <xsd:element name="title"          type="CT_Title"             minOccurs="0"/>
    <xsd:element name="numFmt"         type="CT_NumFmt"            minOccurs="0"/>
    <xsd:element name="majorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="minorTickMark"  type="CT_TickMark"          minOccurs="0"/>
    <xsd:element name="tickLblPos"     type="CT_TickLblPos"        minOccurs="0"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="crossAx"        type="CT_UnsignedInt"/>
    <xsd:choice                                                    minOccurs="0">
```

```
      <xsd:element name="crosses"   type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
    <xsd:element name="crossBetween"   type="CT_CrossBetween"      minOccurs="0"/>
    <xsd:element name="majorUnit"      type="CT_AxisUnit"          minOccurs="0"/>
    <xsd:element name="minorUnit"      type="CT_AxisUnit"          minOccurs="0"/>
    <xsd:element name="dispUnits"      type="CT_DispUnits"         minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_DateAx">
  <xsd:sequence>
    <xsd:group    ref="EG_AxShared"/>
    <xsd:element name="auto"          type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="lblOffset"     type="CT_LblOffset"     minOccurs="0"/>
    <xsd:element name="baseTimeUnit"  type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="majorUnit"     type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="majorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="minorUnit"     type="CT_AxisUnit"      minOccurs="0"/>
    <xsd:element name="minorTimeUnit" type="CT_TimeUnit"      minOccurs="0"/>
    <xsd:element name="extLst"        type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SerAx">
  <xsd:sequence>
    <xsd:group    ref="EG_AxShared"/>
    <xsd:element name="tickLblSkip"  type="CT_Skip"          minOccurs="0"/>
    <xsd:element name="tickMarkSkip" type="CT_Skip"          minOccurs="0"/>
    <xsd:element name="extLst"       type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_AxShared">
  <xsd:sequence>
    <xsd:element name="axId"          type="CT_UnsignedInt"/>
    <xsd:element name="scaling"       type="CT_Scaling"/>
    <xsd:element name="delete"        type="CT_Boolean"          minOccurs="0"/>
    <xsd:element name="axPos"         type="CT_AxPos"/>
    <xsd:element name="majorGridlines" type="CT_ChartLines"      minOccurs="0"/>
    <xsd:element name="minorGridlines" type="CT_ChartLines"      minOccurs="0"/>
    <xsd:element name="title"         type="CT_Title"            minOccurs="0"/>
    <xsd:element name="numFmt"        type="CT_NumFmt"           minOccurs="0"/>
    <xsd:element name="majorTickMark" type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="minorTickMark" type="CT_TickMark"         minOccurs="0"/>
    <xsd:element name="tickLblPos"    type="CT_TickLblPos"       minOccurs="0"/>
    <xsd:element name="spPr"          type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"          type="a:CT_TextBody"       minOccurs="0"/>
    <xsd:element name="crossAx"       type="CT_UnsignedInt"/>
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="crosses"   type="CT_Crosses"/>
      <xsd:element name="crossesAt" type="CT_Double"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:group>

<xsd:complexType name="CT_ChartLines">
```

```
  <xsd:sequence>
    <xsd:element name="spPr" type="a:CT_ShapeProperties" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Crosses">
  <xsd:attribute name="val" type="ST_Crosses" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_Scaling">
  <xsd:sequence>
    <xsd:element name="logBase"     type="CT_LogBase"       minOccurs="0"/>
    <xsd:element name="orientation" type="CT_Orientation"   minOccurs="0"/>
    <xsd:element name="max"         type="CT_Double"        minOccurs="0"/>
    <xsd:element name="min"         type="CT_Double"        minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NumFmt">
  <xsd:attribute name="formatCode"   type="xsd:string"  use="required"/>
  <xsd:attribute name="sourceLinked" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_TickLblPos">
  <xsd:attribute name="val" type="ST_TickLblPos" default="nextTo"/>
</xsd:complexType>

<xsd:complexType name="CT_TickMark">
  <xsd:attribute name="val" type="ST_TickMark" default="cross"/>
</xsd:complexType>

<xsd:complexType name="CT_Boolean">
  <xsd:attribute name="val" type="xsd:boolean" default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_Double">
  <xsd:attribute name="val" type="xsd:double" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_Crosses">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="autoZero"/>
    <xsd:enumeration value="max"/>
    <xsd:enumeration value="min"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_TickLblPos">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="high"/>
    <xsd:enumeration value="low"/>
    <xsd:enumeration value="nextTo"/>
    <xsd:enumeration value="none"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_TickMark">
```

```
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="cross"/>
    <xsd:enumeration value="in"/>
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="out"/>
  </xsd:restriction>
</xsd:simpleType>
```

Chart Shape

A chart is not actually a shape. It is a graphical object held inside a graphics frame. The graphics frame is a shape and the chart must be retrieved from it.

Protocol

```
>>> shape = shapes.add_textbox(0, 0, 0, 0)
>>> shape.has_chart
False
>>> shape.chart
...
AttributeError: 'Shape' object has no attribute 'chart'
>>> shape = shapes.add_chart(style, type, x, y, cx, cy)
>>> type(shape)
<class 'pptx.shapes.graphfrm.GraphicFrame'>
>>> shape.has_chart
True
>>> shape.chart
<pptx.parts.chart.ChartPart object at 0x108c0e290>
```

Acceptance tests

```
Feature: Access chart object
  In order to operate on a chart in a presentation
  As a developer using python-pptx
  I need a way to find and access a chart

  Scenario Outline: Identify a shape containing a chart
    Given a <shape type>
     Then the shape <has?> a chart

    Examples: Shape types
      | shape type                        | has?          |
      | shape                             | does not have |
      | graphic frame containing a chart  | has           |
      | picture                           | does not have |
      | graphic frame containing a table  | does not have |
      | group shape                       | does not have |
      | connector                         | does not have |


  Scenario: Access chart object from graphic frame shape
    Given a graphic frame containing a chart
```

```
    When I get the chart from its graphic frame
    Then the chart is a ChartPart object
```

Charts - Main Chart Object

Related Schema Definitions

```
<!-- homonym <c:chart> element in graphicData element -->

<xsd:element name="chart" type="CT_RelId"/>

<xsd:complexType name="CT_RelId">
  <xsd:attribute ref="r:id" use="required"/>
</xsd:complexType>


<!-- elements in chartX.xml part -->

<xsd:element name="chartSpace" type="CT_ChartSpace"/>

<xsd:complexType name="CT_ChartSpace">
  <xsd:sequence>
    <xsd:element name="date1904"       type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="lang"           type="CT_TextLanguageID"    minOccurs="0"/>
    <xsd:element name="roundedCorners" type="CT_Boolean"           minOccurs="0"/>
    <xsd:element name="style"          type="CT_Style"             minOccurs="0"/>
    <xsd:element name="clrMapOvr"      type="a:CT_ColorMapping"    minOccurs="0"/>
    <xsd:element name="pivotSource"    type="CT_PivotSource"       minOccurs="0"/>
    <xsd:element name="protection"     type="CT_Protection"        minOccurs="0"/>
    <xsd:element name="chart"          type="CT_Chart"/>
    <xsd:element name="spPr"           type="a:CT_ShapeProperties" minOccurs="0"/>
    <xsd:element name="txPr"           type="a:CT_TextBody"        minOccurs="0"/>
    <xsd:element name="externalData"   type="CT_ExternalData"      minOccurs="0"/>
    <xsd:element name="printSettings"  type="CT_PrintSettings"     minOccurs="0"/>
    <xsd:element name="userShapes"     type="CT_RelId"             minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_ExtensionList"     minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Chart">
  <xsd:sequence>
    <xsd:element name="title"            type="CT_Title"         minOccurs="0"/>
    <xsd:element name="autoTitleDeleted" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="pivotFmts"        type="CT_PivotFmts"     minOccurs="0"/>
    <xsd:element name="view3D"           type="CT_View3D"        minOccurs="0"/>
    <xsd:element name="floor"            type="CT_Surface"       minOccurs="0"/>
    <xsd:element name="sideWall"         type="CT_Surface"       minOccurs="0"/>
    <xsd:element name="backWall"         type="CT_Surface"       minOccurs="0"/>
    <xsd:element name="plotArea"         type="CT_PlotArea"/>
    <xsd:element name="legend"           type="CT_Legend"        minOccurs="0"/>
    <xsd:element name="plotVisOnly"      type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="dispBlanksAs"     type="CT_DispBlanksAs"  minOccurs="0"/>
    <xsd:element name="showDLblsOverMax" type="CT_Boolean"       minOccurs="0"/>
    <xsd:element name="extLst"           type="CT_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
```

```
</xsd:complexType>

<xsd:complexType name="CT_Style">
  <xsd:attribute name="val" type="ST_Style" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_Style">
  <xsd:restriction base="xsd:unsignedByte">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="48"/>
  </xsd:restriction>
</xsd:simpleType>
```

Shape

Freeform Shapes

PowerPoint AutoShapes, such as rectangles, circles, and triangles, have a *preset* geometry. The user simply picks from a selection of some 160 pre-defined shapes to place one on the slide, at which point they can be scaled. Many shapes allow their geometry to be adjusted in certain ways using adjustment points (by dragging the small yellow diamonds).

PowerPoint also supports a much less commonly used but highly flexible sort of shape having *custom geometry*. Whereas a shape with preset geometry has that geometry defined internally to PowerPoint and simply referred to by a keyname (such as 'rect'), a shape with custom geometry spells out the custom geometry step-by-step in an XML subtree, using a sequence of moveTo, lineTo, and other drawing commands.

Vocabulary

- A freeform shape is composed of one or more *paths*. Each path may be configured to get a stroke, a fill, both, or neither. This allows drawing some shapes such as call-outs where, say, a reference line should be stroked but not filled, and the callout balloon contour should be filled but not stroked.

  Each path also has its own local coordinate system, which may come in handy when using guides. I haven't seen any clear examples of this yet but I haven't gone looking either.

  In any case, only a single path will be supported in the initial implementation.

- A path is composed of one or more *contours*. A contour can be a *closed contour* or and *open contour*. The closure state only affects the continuity of the contour outline (although it also adds a straight line between start and end if needed).

- A contour is specified by a starting point (*<a:moveTo>*) followed by a sequence of line segments, each of which can be a straight or curved. Curve segments will not be supported in the initial implementation.

  The term *contour* is not "official", but there are behaviors that depend on "continuous pen-down strokes", such as "cut-out" behavior. Consequently, a term is needed.

Scope

- Move to, line to, and close, initially.
- Line style (solid, dashed, etc.)
- Fill pattern (solid, dashed, etc.)

Limits of scope

- Curve segments will not be implemented initially.

  Curves wouldn't be that hard to add, but until anyone cares about the control points and we have some live example about how those would be calculated etc., we don't have a requirement to design to.

- Multiple-path freeforms are out of scope for now.

  It's not immediately apparent what they would be good for, and they don't seem to be required to fulfill present requirements.

Use Case 1 - Specify all points in slide coordinates

A very simple case can be had by specifying the verticies in slide coordinates:

```
>>> freeform_builder = shapes.build_freeform(Inches(1.5), Inches(1))
>>> freeform_builder.add_line_segments((
...     (Inches(2),   Inches(2)),
...     (Inches(1),   Inches(2)),
...     (Inches(1.5), Inches(1)),
... ))
>>> freeform_shape = freeform_builder.convert_to_shape()
>>> freeform_shape.left.inches, freeform_shape.top.inches
1.0, 1.0
>>> freeform_shape.width.inches, freeform_shape.height.inches
1.0, 1.0
```

This method may be used to place a freeform shape when you know exactly where you want the vertices to be positioned on the slide. The position and size of the shape (its bounding box) is worked out automatically from the minimum and maximum x and y positions of the points specified.

The shape is automatically closed if the last vertex is the same as the beginning vertex; otherwise the shape is left open.

The library automatically subtracts the top-left corner (origin) coordinates from each point to write the XML.

Use Case 2 - Specify points in custom coordinates

The next step up in complexity is where a custom coordinate system is to be used to specify vertices for a single shape.

In this alternative, all the verticies are specified in custom coordinates and a scaling factor is provided:

```
>>> freeform_builder = shapes.build_freeform(
...     1500, 1000, scale=1000/Inches(1)
... )
>>> freeform_builder.add_line_segments((
...     (2000, 2000),
...     (1000, 2000),
...     (1500, 1000),
... ))
>>> freeform_shape = freeform_builder.convert_to_shape()
>>> freeform_shape.left.inches, freeform_shape.top.inches
1.0, 1.0
>>> freeform_shape.width.inches, freeform_shape.height.inches
1.0, 1.0
```

This method creates a shape of exactly the same appearance as the prior example. It's probably not the most natural approach because all coordinates have to be translated to their position on the slide. However, it is an option and it's more compact. It could also be made to work with floating point values.

This could also work by creating the shape at (0, 0) on the slide and then simply moving the shape to where you wanted using its *.top* and *.left* properties.

Use Case 3 - Specify origin for custom coordinates

This is perhaps the most natural approach for doing something fancy, like creating individual shapes for each of the 50 US states within a particular area of the slide such that their boundaries all align with adjacent states to make a map of the United States.

Like in the prior use case, verticies are specified in custom coordinates and a scaling factor is provided. In addition, an origin in provided in slide coordinates (EMU). This would correspond to the origin of the entire map of the US in our example and represents where you want the top-left corner of the map to be on the slide.

It is up to the client code to adjust the vertices provided such that they represent their position relative to this origin in the custom coordinates provided. It is also up to the client to get the scaling factor right such that the entire map fits in the intended area of the slide.

In this scenario, *top* and *left* arguments are added to the *.build_freeform()* call. These arguments should be given the same values for each of the "interlocking" shapes. These two arguments each independently take on the value of zero when omitted, which produces the behavior of the prior use case:

```
>>> freeform_builder = shapes.build_freeform(
...     start_x=1500, start_y=1000, scale=1000/Inches(1),
...     top=Inches(1), left=Inches(2)
... )
>>> freeform_builder.add_line_segments((
...     (2000, 2000),
...     (1000, 2000),
...     (1500, 1000),
... ))
>>> freeform_shape = freeform_builder.convert_to_shape()
>>> freeform_shape.left.inches, freeform_shape.top.inches
3.0, 2.0
>>> freeform_shape.width.inches, freeform_shape.height.inches
1.0, 1.0
```

As we can see here, the size of the shape is exactly the same, but it has been translated to be relative to the slide coordinates provided in the *.build_freeform()* call.

Use Case 4 - Specify position on conversion

Another use case is where you want to specify a shape entirely in terms of its local coordinates, relative to its local origin (top-left corner). This might be used when you want a custom shape that is not one of the 168 provided preset geometry shapes, say an orthoganal projection cube or something like that, and then place that in a variety of locations around the slide.

For this you build the shape as in Use Case 2, allowing the origin to default to (0, 0), and then specify a position for the shape in the *.convert_to_shape()* call. You can even call *.convert_to_shape()* multiple times to "rubber stamp" the shape in various positions around the slide.

Can you call *.convert_to_shape()* multiple times? Maybe that would be good!

Usage Notes

Closing a contour. A contour is only *truly* closed if it ends with a *<a:close>* element. It can *appear* closed if its end point is the same as its start point, but close inspection of the line-ends will reveal there is no smooth transition around the starting corner. This is more apparent when a wide line style is used for the outline.

A *<a:close>* element will cause a straight line segment to be drawn between the last point point of the contour and the first. So it's not necessary to repeat the first point as the last one if a straight line segment suits. The advantage is small however, and might only be an optimization to consider when thousands of freeform shapes were being drawn in a single presentation.

Candidate Protocol

Obtain a FreeformBuilder object:

```
>>> freeform_builder = shapes.build_freeform(50, 0, scale=100/Inches(1))
>>> freeform_builder
<pptx.shapes.freeform.FreeformBuilder objext at 0x...>
```

The *x* and *y* arguments specify the starting pen location in local coordinates. The *scale* argument determines the rendered size of the shape relative to the values specified for the vertices. The *scale* argument can be an *(x_scale, y_scale)* pair to specify a shape coordinate system that is rectangular rather than square.

Add vertices to the freeform shape an arbitrary number of straight line segments can be added to the shape using the *add_line_segments()* method:

```
>>> vertices = ((100, 200), (200, 300), (300, 400))
>>> freeform_builder.add_line_segments(vertices, close=True)
```

The method also specifies whether the contour so generated should be closed (*True* by default). Closing a contour causes its end point to be connected to its starting point with a straight line segment. It also causes the contour to form a smooth perimeter outline. A contour that is not closed will have line end caps and may be assigned line ends like arrowheads.

Although PowerPoint supports a variety of curved line segments, those will not be implemented in this initial effort.

Add second and further contours A path can have multiple distinct contours. These contours can be separate and distinct or they can overlap. A contour that is completely enclosed by another contour forms a "cutout" in the enclosing contour. Where two contours partially overlap, the fill begins at the outermost boundary and stops when it reaches another boundary.

A new contour is begun by calling the *.move_to()* method:

```
>>> freeform_builder.move_to(x, y)
```

- Set freeform coordinate system (extents)

  There is no need to explicitly specify the extents of the coordinate system. The position and size of the shape can be calculated from the origin provided and the maxima and minima of the vertices provided.

- Specify the scaled size of the freeform shape

  This is also unnecessary to specify as it is implied by the scaling factor provided and the maxima and minima of the vertices specified.

Possible approach:

Optionally specify an alternate coordinate system in the initial call such that parameters are:

- *x, y* - X and Y coordinates of initial pen position. If no scaling factor is provided, these are interpreted as *Length* values from the top-left corner of the slide.
- *scale=1.0* (*optional*) - Determines the scaling of the x and y values used to specify the vertices. This is a proportion to a *Length* unit, which is an *Emu*, 914400 to the inch. This value is conveniently formed by division with a *Length* subclass, like *1000/Inches(1)* to give "1000 units per inch" in that case. This could just as easily be *2142/Cm(1)* to give "2142 units per centimeter". The default value of 1.0 means "1 unit per Emu".
- *left* (*optional*) - X-axis position of top-left corner of freeform shape bounding box, specified as a *Length* (not scaled) value relative to left edge of slide. If this value is not specified, it is calculated from the minimum x position of the points in the shape.
- *top* (*optional*) - Y-axis position of top-left corner of freeform shape bounding box, specified as a *Length* (not scaled) value relative to top edge of the slide. If this value is not specified, it is calculated from the minimum y position of the points in the shape.
- *y_scale=None* (*optional, not implemented*) - Y-axis scaling factor (defaults to same as X-axis scaling factor)

The shape size (extents) are calculated from max_X and max_Y in given coordinates, multiplied by the scaling factor.

Experimentation

17. How does PowerPoint behave if we leave out all the "guides" and connection points, i.e. if the *a:gdLst* and *a:cxnLst* elements are empty?

1. It likes it just fine. Might not be a bad idea to leave empty *a:gdLst* and *a:cxnLst* elements in there though, when creating a new freeform XML subtree, just to be a little safer compatibility-wise since PowerPoint seems to always put them in there.

---

17. What happens if there's no *a:moveTo* element at the start of a path?

1. The path does not appear; it is not drawn. Note this is contrary to the (unofficial) documentation indicating that (0, 0) is used as the starting pen location.

---

17. What happens if a second path overlaps the first one, i.e. partly in and partly out?

1. The "outside" portion of the overlapping contour becomes the outside of the shape and the region between that contour and the boundary of the first shape is shaded. This produces two (or more) "point" areas on the perimeter of the first shape where the width is zero.

---

17. What happens if the last point is not the same as the first point and the path is closed?

1. A straight-line segment is added between the last point and the starting point. The appearance is just as it is when the last point is the same as the starting point.

---

17. What happens if you do a shape with an *a:moveTo* in the middle (producing a gap in the outline) but then close the shape? Does it still get a fill or is it considered open then?

1. The *moveTo* operation essentially resets the "starting" point for closure purposes. The remainder of the path is closed, but the part before the *moveTo* remains an open contour.

---

17. What happens when the last point is the same as the first point but there is no *a:close* element?

1. The shape outline is discontinuous at the start/end point and does not form a smooth contour at that point. The visual appearance depends on the line ends and line end caps chosen.

- [ ] What happens when negative coordinate values are used?
- [ ] What happens when you close a contour right after an *a:moveTo*?
- **[ ] What is the point of having multiple paths? The only difference I can** see is that overlapping areas are not "subtracted" and you can have a different coordinate system. I'm inclined to think it's all about needing distinct coordinate systems for some reason, perhaps when variables (guides <*a:gd*>) are used.
- [ ] What is the purpose of the boolean *stroke* attribute on path?

PowerPoint UI behaviors

- A freeform shape can be created using the ribbon Shape > Lines and Connectors > Freeform command. There is also a "Curve" and "Scribble" option.
- A shape is then created by placing points with the mouse. In the freeform case, each vertex defines a line segment. In the case of a curve freeform, the vertices are control points.
- Clicking close to the starting point closes the shape. Double-clicking also ends the drawing, leaving the shape open.
- Once created, an "Edit Points" option appears on the context menu when right-clicking the shape. This allows the points to be fine-tuned.
- Hypothesis: PowerPoint uses an arbitrary scale of 10,000 (period, not per inch) as the coordinate space for a freeform shape added or edited using the UI. The rules are more complicated, not sure what they are, but it seems to start with a square of about that and move from there.

MS API

MS API protocol

Example:

```
Set myDocument = ActivePresentation.Slides(1)

With myDocument.Shapes.BuildFreeform(msoEditingCorner, 360, 200)
    .AddNodes msoSegmentCurve, msoEditingCorner, _
        380, 230, 400, 250, 450, 300
    .AddNodes msoSegmentCurve, msoEditingAuto, 480, 200
    .AddNodes msoSegmentLine, msoEditingAuto, 480, 40
    .AddNodes msoSegmentLine, msoEditingAuto, 360, 200
    .ConvertToShape
End With
```

- Specify pen starting point in initial *BuildFreeform* call.
- All points are specified in slide coordinates.
- Specify each vertex separately in an *.AddNodes()* method call. It seems like it should actually be named *.AddNode()*, but perhaps they consider control points to be nodes separate from the vertex point.

  A line segment can be specified by specifying the x, y vertex of the ending point.

  A curve segment can be specified by specifying additional control points (nodes). The vertex can be specified to be a corner (one control point on starting vertex), smooth (tangent at the vertex), or symmetric (we'd have to experiment to see what that does exactly, no ready available documentation).
- The path is closed (or not) when the *.ConvertToShape()* method is called on the *FreeformBuilder* object. It looks like it's closed if the last point is coincident with the first and open otherwise.
- There is no way to make a multi-path shape as far as I can tell.
- The MS API provides no way to "lift the pen" to make a discontinuous path as far as I can tell. (And so does not provide a way to make "cutouts" like a lake within a landmass shape.)

MS API Objects

- Shapes.BuildFreeform(x, y)
  https://msdn.microsoft.com/VBA/PowerPoint-VBA/articles/shapes-buildfreeform-method-powerpoint
- FreeformBuilder
  https://msdn.microsoft.com/VBA/PowerPoint-VBA/articles/freeformbuilder-object-powerpoint
- FreeformBuilder.AddNodes()
  https://msdn.microsoft.com/en-us/vba/powerpoint-vba/articles/freeformbuilder-addnodes-method-powerpoint
- FreeformBuilder.ConvertToShape()
  https://msdn.microsoft.com/VBA/PowerPoint-VBA/articles/freeformbuilder-converttoshape-method-powerpoint
  Cannot be called before adding at least one segment to the shape.
- Shape.Nodes
- MsoSegmentType enumeration
  https://msdn.microsoft.com/VBA/Office-Shared-VBA/articles/msosegmenttype-enumeration-office

Name	Value	Description
msoSegmentCurve	1	Curve.
msoSegmentLine	0	Line.

- MsoEditingType enumeration

https://msdn.microsoft.com/VBA/Office-Shared-VBA/articles/msoeditingtype-enumeration-office

Name	Value	Description
msoEditingAuto	0	Editing type is appropriate to the segments being connected.
msoEditingCorner	1	Corner node.
msoEditingSmooth	2	Smooth node.
msoEditingSymmetric	3	Symmetric node.

XML Semantics

- The *w* and *h* attributes of a *a:path* element specify size of the shape canvas in local coordinates.
  - The **origin of the shape canvas** is at its top left corner and has the value (0, 0) in local coordinates. The vertices must be translated by the X and Y minima to place the bounding box tightly around the shape.
- The *a:rect* element specifies the text-box extents for the shape.
- The *a:pathLst* element contains zero or more *a:path* elements, each of which specify a *path*.
- A path has a *private* and *arbitrary integral* coordinate system defined by the *w* and *h* attributes on the *a:path* element. This means that each path in an *a:custGeom/a:pathLst* has an independent scaling for its coordinate system.
- A path is composed of a sequence of the following possible elements:
  - *a:moveTo*
  - *a:lnTo*
  - *a:arcTo*
  - *a:quadBezTo*
  - *a:cubicBezTo*
  - *a:close*

  A path may begin with an *a:moveTo* element. This essentially locates the starting location of the "pen". Each subsequent drawing command extends the shape by adding a line segment. If the path does not begin with an *a:moveTo* element, the path does not appear (it is not drawn). Note this is contrary to the (unofficial) documentation indicating that (0, 0) is used as the default starting pen location.

  A path can be open or closed. If an *a:close* element is added, a straight line segment is drawn from the current pen location to the initial location of the drawing sequence and the shape appears with a fill. If the pen is already at the starting location, no additional line segment appears. If no *a:close* element is added, the shape remains "open" and only the path appears (no fill).

  A path can contain more than one drawing sequence, i.e. one sequence can be "closed" and another sequence started. If a subsequent drawing sequence is entirely enclosed within a prior sequence, it appears as a "cutout", or an interior boundary. This behavior does not occur when the two drawing sequences are in separate paths, even within the same shape.

  The pen can be "lifted" using an *a:moveTo* element, in which case no line segment is drawn between the prior location and the new location. This can be used to produce a discontinuous outline.

  A path has a boolean *stroke* attribute (default True) that specifies whether a line should appear on the path.
- The *a:pathLst* element can contain multiple *a:path* elements. In this case, each path is essentially a "sub-shape", such as a shape that depicts the islands of Hawaii.

  If a prior path is not closed, its end point path will be connected to the first point of the subsequent path.

The paths within a shape all have the same z-position, i.e. they appear on a single plane such that all outlines appear, even when they intersect. There is no cropping behavior such as occurs for individual shapes on a slide.

Fill behavior

Hypothesis: Any shape can have a fill, even an "open" shape. The question of whether a shape has a fill is not determined by whether it is closed, but by whether a fill is *applied* to the shape.

In the PowerPoint UI, a closed shape is automatically assigned a fill when it is drawn. Conversely, an open shape is not assigned a fill. This behavior is built into the drawing code and is not dependent solely on the "closed-ness" of the shape.

The XML template for a freeform shape (as of this writing) includes an element for a fill and effect, consistent with the PowerPoint defaults for a closed shape. The fill would need to be removed from an open shape if the user didn't want it.

Coordinate system

- Each path has its own local coordinate system, distinct both from the *shape* coordinate system and the coordinate systems of the other paths in the shape.
- The x and y extents of a path coordinate system are specified by the *w* and *h* attributes on the *a:path* element, respectively. The top, left corner of the path bounding box is (0, 0) and the bottom, right corner is at (*h*, *w*). Coordinates are positive integers in the range 0 to 27,273,042,316,900 (about 2^44.63).

Resources

- Office Open XML - Custom Geometry http://officeopenxml.com/drwSp-custGeom.php

XML Specimens

Hand-built, as-written by PowerPoint

This triangle is hand-built using the 'Freeform' shape tool in the PowerPoint UI. A couple things to notice:

- The shape includes a full complement of <*a:gd*> and <*a:cxn*> elements. The shape works fine without these and providing a way to specify these is out of scope for the current effort.

- The freeform shape itself is specified in *p:sp/p:spPr/a:custGeom*. That's a little interesting that the geometry is considered a shape property (spPr) rather than somehow more core to the shape.
- The *<a:pathLst>* element contains only a single path. I haven't yet discovered how to add a second path to a shape using the UI.

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="2" name="Freeform 1"/>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="700718" y="642282"/>
      <a:ext cx="897794" cy="773657"/>
    </a:xfrm>
    <a:custGeom>
      <a:avLst/>
      <a:gdLst>
        <a:gd name="connsiteX0" fmla="*/ 423350 w 897794"/>
        <a:gd name="connsiteY0" fmla="*/ 0 h 773657"/>
        <a:gd name="connsiteX1" fmla="*/ 897794 w 897794"/>
        <a:gd name="connsiteY1" fmla="*/ 773657 h 773657"/>
        <a:gd name="connsiteX2" fmla="*/ 0 w 897794"/>
        <a:gd name="connsiteY2" fmla="*/ 766359 h 773657"/>
        <a:gd name="connsiteX3" fmla="*/ 423350 w 897794"/>
        <a:gd name="connsiteY3" fmla="*/ 0 h 773657"/>
      </a:gdLst>
      <a:ahLst/>
      <a:cxnLst>
        <a:cxn ang="0">
          <a:pos x="connsiteX0" y="connsiteY0"/>
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX1" y="connsiteY1"/>
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX2" y="connsiteY2"/>
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX3" y="connsiteY3"/>
        </a:cxn>
      </a:cxnLst>
      <a:rect l="l" t="t" r="r" b="b"/>
      <a:pathLst>
        <a:path w="897794" h="773657">
          <a:moveTo>
            <a:pt x="423350" y="0"/>
          </a:moveTo>
          <a:lnTo>
            <a:pt x="897794" y="773657"/>
          </a:lnTo>
          <a:lnTo>
            <a:pt x="0" y="766359"/>
          </a:lnTo>
          <a:lnTo>
            <a:pt x="423350" y="0"/>
          </a:lnTo>
```

```
          <a:close/>
        </a:path>
      </a:pathLst>
    </a:custGeom>
    <a:ln w="57150" cmpd="sng">
      <a:solidFill>
        <a:schemeClr val="accent6"/>
      </a:solidFill>
    </a:ln>
  </p:spPr>
  <p:style>
    <a:lnRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="3">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="lt1"/>
    </a:fontRef>
  </p:style>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr"/>
    <a:lstStyle/>
    <a:p>
      <a:pPr algn="ctr"/>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </p:txBody>
</p:sp>
```

The rest of these shape subtrees have the *<a:gdLst>* and *<a:cxnLst>* elements removed for brevity and many focus simply on the *a:pathLst* element for compact presentation.

Effect of *<a:close>* element

This one is a little tricky to see, but if you look closely, you'll see that the outline at the apex of the triangle is not "closed". This behavior arises when there is no *<a:close/>* element at the end of the path, even when the end-point is

the same as the start-point:

```
<a:pathLst>
  <a:path w="100" h="100">
    <a:moveTo><a:pt x="50" y="0"/></a:moveTo>
    <a:lnTo><a:pt x="100" y="100"/></a:lnTo>
    <a:lnTo><a:pt x="0" y="100"/></a:lnTo>
    <a:lnTo><a:pt x="50" y="0"/></a:lnTo>
  </a:path>
</a:pathLst>
```

XML Schema excerpt

```
<xsd:complexType name="CT_Shape">  <!-- p:sp element -->
  <xsd:sequence>
    <xsd:element name="nvSpPr" type="CT_ShapeNonVisual"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties"/>
    <xsd:element name="style"  type="a:CT_ShapeStyle"        minOccurs="0"/>
    <xsd:element name="txBody" type="a:CT_TextBody"          minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="useBgFill" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_ShapeProperties">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="xfrm"               type="CT_Transform2D"
↪minOccurs="0"/>
    <xsd:choice minOccurs="0">  <!-- EG_Geometry -->
      <xsd:element name="custGeom" type="CT_CustomGeometry2D"/>
      <xsd:element name="prstGeom" type="CT_PresetGeometry2D"/>
    </xsd:choice>
    <xsd:choice minOccurs="0">  <!-- EG_FillProperties -->
      <xsd:element name="noFill" type="CT_NoFillProperties"/>
      <xsd:element name="solidFill" type="CT_SolidColorFillProperties"/>
      <xsd:element name="gradFill"  type="CT_GradientFillProperties"/>
      <xsd:element name="blipFill"  type="CT_BlipFillProperties"/>
      <xsd:element name="pattFill"  type="CT_PatternFillProperties"/>
      <xsd:element name="grpFill"   type="CT_GroupFillProperties"/>
    </xsd:choice>
    <xsd:element name="ln"                 type="CT_LineProperties"
↪minOccurs="0"/>
    <xsd:choice minOccurs="0">  <!-- EG_EffectProperties -->
      <xsd:element name="effectLst" type="CT_EffectList"/>
      <xsd:element name="effectDag" type="CT_EffectContainer"/>
    </xsd:choice>
    <xsd:element name="scene3d"            type="CT_Scene3D"
↪minOccurs="0"/>
    <xsd:element name="sp3d"               type="CT_Shape3D"
↪minOccurs="0"/>
    <xsd:element name="extLst"             type="CT_OfficeArtExtensionList"
↪minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</xsd:complexType>

<xsd:complexType name="CT_CustomGeometry2D">
```

```
  <xsd:sequence>
    <xsd:element name="avLst"   type="CT_GeomGuideList"      minOccurs="0"/>
    <xsd:element name="gdLst"   type="CT_GeomGuideList"      minOccurs="0"/>
    <xsd:element name="ahLst"   type="CT_AdjustHandleList"   minOccurs="0"/>
    <xsd:element name="cxnLst"  type="CT_ConnectionSiteList" minOccurs="0"/>
    <xsd:element name="rect"    type="CT_GeomRect"           minOccurs="0"/>
    <xsd:element name="pathLst" type="CT_Path2DList"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Path2DList">
  <xsd:sequence>
    <xsd:element name="path" type="CT_Path2D" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Path2D">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="close"      type="CT_Path2DClose"/>
    <xsd:element name="moveTo"     type="CT_Path2DMoveTo"/>
    <xsd:element name="lnTo"       type="CT_Path2DLineTo"/>
    <xsd:element name="arcTo"      type="CT_Path2DArcTo"/>
    <xsd:element name="quadBezTo"  type="CT_Path2DQuadBezierTo"/>
    <xsd:element name="cubicBezTo" type="CT_Path2DCubicBezierTo"/>
  </xsd:choice>
  <xsd:attribute name="w"           type="ST_PositiveCoordinate" default="0"/>
  <xsd:attribute name="h"           type="ST_PositiveCoordinate" default="0"/>
  <xsd:attribute name="fill"        type="ST_PathFillMode"       default="norm"/>
  <xsd:attribute name="stroke"      type="xsd:boolean"           default="true"/>
  <xsd:attribute name="extrusionOk" type="xsd:boolean"           default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_Path2DClose"/>

<xsd:complexType name="CT_Path2DLineTo">
  <xsd:sequence>
    <xsd:element name="pt" type="CT_AdjPoint2D"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Path2DMoveTo">
  <xsd:sequence>
    <xsd:element name="pt" type="CT_AdjPoint2D"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_AdjPoint2D">
  <xsd:attribute name="x" type="ST_AdjCoordinate" use="required"/>
  <xsd:attribute name="y" type="ST_AdjCoordinate" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_GeomGuideName">
  <xsd:restriction base="xsd:token"/>
</xsd:simpleType>

<xsd:simpleType name="ST_GeomGuideFormula">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

```
<xsd:complexType name="CT_GeomGuide">
  <xsd:attribute name="name" type="ST_GeomGuideName"    use="required"/>
  <xsd:attribute name="fmla" type="ST_GeomGuideFormula" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_GeomGuideList">
  <xsd:sequence>
    <xsd:element name="gd" type="CT_GeomGuide" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ST_AdjCoordinate">
  <xsd:union memberTypes="ST_Coordinate ST_GeomGuideName"/>
</xsd:simpleType>

<xsd:simpleType name="ST_AdjCoordinate">
  <xsd:union memberTypes="ST_Coordinate ST_GeomGuideName"/>
</xsd:simpleType>

<xsd:simpleType name="ST_PositiveCoordinate">
  <xsd:restriction base="xsd:long">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="27273042316900"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_Coordinate">
  <xsd:union memberTypes="ST_CoordinateUnqualified s:ST_UniversalMeasure"/>
</xsd:simpleType>

<xsd:simpleType name="ST_CoordinateUnqualified">
  <xsd:restriction base="xsd:long">
    <xsd:minInclusive value="-27273042329600"/>
    <xsd:maxInclusive value="27273042316900"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_UniversalMeasure">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="-?[0-9]+(\.[0-9]+)?(mm|cm|in|pt|pc|pi)"/>
  </xsd:restriction>
</xsd:simpleType>
```

Movie

PowerPoint allows a video to be inserted into a slide and run in slide show mode. In the PowerPoint UI, a shape of this type is called a *movie*.

A movie is implemented as a variant form of a *p:pic* element, which makes it closely related to a *Picture* object. This close relationship makes sense, because a movie acts very much like an image when it is not playing and the two shapes therefore share much of their behavior. When not playing, a movie shape displays a static image, called a *poster frame*.

Candidate Protocol

A video can be added to a slide by specifying a video file, position, size, and optional poster frame:

```
>>> movie = shapes.add_movie(
...     movie_file, left, top, width, height, poster_frame_image,
...     mime_type
... )

>>> movie
<pptx.shape.picture.Movie object @ 0x12345678>

>>> movie.shape_type
MSO.MEDIA

>>> movie.media_type
PP_MEDIA_TYPE.MOVIE

>>> movie.media_format
<pptx.shape.picture.MediaFormat object @ 0x123456ab>
```

Animation/timing resources

- Working with animation (Open XML SDK) https://msdn.microsoft.com/en-us/library/office/gg278329.aspx
- AnimationBehaviors (Collection) Members (PowerPoint) https://msdn.microsoft.com/en-us/library/office/ff746028.aspx
- AnimationBehavior Members (PowerPoint) https://msdn.microsoft.com/en-us/library/office/ff746141.aspx
- <p:par> stands for *parallel*, and is one of the available time containers (like "parallel" timing, or executing at the same time). https://folk.uio.no/annembek/inf3210/how_2_SMIL.html
- https://technet.microsoft.com/en-au/library/gg278329.aspx
- http://openxmldeveloper.org/discussions/formats/f/15/t/6838.aspx

Possibly interesting resources

- Apache POI discussion
- Apache POI discussion on inserting video

MS API

- Slide.Timeline https://msdn.microsoft.com/en-us/library/office/ff745382.aspx
- TimeLine Object https://msdn.microsoft.com/en-us/library/office/ff743965.aspx
- Shapes.AddMediaObject() (deprecated in PowerPoint 2013, see AddMediaObject2) https://msdn.microsoft.com/EN-US/library/office/ff745385.aspx
- Shapes.AddMediaObject2() https://msdn.microsoft.com/en-us/library/office/ff744080.aspx
- MediaFormat https://msdn.microsoft.com/en-us/library/office/ff745983.aspx

Enumerations

PP_MEDIA_TYPE - https://msdn.microsoft.com/en-us/library/office/ff746008.aspx

PP_MEDIA_TYPE.MOVIE	3	Movie
PP_MEDIA_TYPE.SOUND	2	Sound
PP_MEDIA_TYPE.OTHER	1	Others
PP_MEDIA_TYPE.MIXED	-2	Mixed

Click Behavior

The video's "play on click" behavior in slideshow mode is implemented by the use of a *<p:timing>* element in the *<p:sld>* element. No "play" button or slider control appears when this element is not present.

Poster Frame

A *poster frame* is the static image displayed in the video location when the video is not playing. Each image that appears on the YouTube home page representing a video is an example of a poster frame.

The poster frame is perhaps most frequently a frame from the video itself. In some contexts, the first frame is used by default. The poster frame can be undefined, or empty, and it can also be an unrelated image.

Some of the example videos for this feature get a poster frame upon insertion; however at least one does not. In that case, a media "speaker" icon (stretched to fit) is shown instead.

XML Semantics

- *id*= of p:cTn element just needs to be unique among *p:cTn* elements, apparently.

Example XML

Inserted MPEG-4 H.264 video:

```
<!--slide{n}.xml-->

<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="6" name="video-filename.mp4">
      <a:hlinkClick r:id="" action="ppaction://media"/>
    </p:cNvPr>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr>
      <a:videoFile r:link="rId2"/>
      <p:extLst>
        <p:ext uri="{DAA4B4D4-6D71-4841-9C94-3DE7FCFB9230}">
          <p14:media
              xmlns:p14="http://schemas.microsoft.com/office/powerpoint/2010/main"
              r:embed="rId1"/>
        </p:ext>
      </p:extLst>
    </p:nvPr>
```

```
  </p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId4"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
    <a:xfrm>
      <a:off x="5059279" y="876300"/>
      <a:ext cx="2390526" cy="5184274"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
  </p:spPr>
</p:pic>
```

Regular picture shape, for comparison:

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="6" name="Picture 5" descr="python-logo.gif"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
    <a:xfrm>
      <a:off x="5580112" y="1988840"/>
      <a:ext cx="2679700" cy="901700"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
  </p:spPr>
</p:pic>


<!--minimal (I think) p:video element-->

<p:sld
    xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
    xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    >
  <p:cSld>
    <!-- ... -->
  </p:cSld>
  <p:clrMapOvr>
    <a:masterClrMapping/>
```

```
  </p:clrMapOvr>
  <p:timing>
    <p:tnLst>
      <p:par>
        <p:cTn xmlns:p14="http://schemas.microsoft.com/office/powerpoint/2010/main"␣
↪id="1" dur="indefinite" restart="never" nodeType="tmRoot">
          <p:childTnLst>
            <p:video>
              <p:cMediaNode vol="80000">
                <p:cTn id="7" fill="hold" display="0">
                  <p:stCondLst>
                    <p:cond delay="indefinite"/>
                  </p:stCondLst>
                </p:cTn>
                <p:tgtEl>
                  <p:spTgt spid="3"/>
                </p:tgtEl>
              </p:cMediaNode>
            </p:video>
          </p:childTnLst>
        </p:cTn>
      </p:par>
    </p:tnLst>
  </p:timing>
</p:sld>


<!--p:timing element for two videos in slide, as added by PowerPoint-->

<p:timing>
  <p:tnLst>
    <p:par>
      <p:cTn xmlns:p14="http://schemas.microsoft.com/office/powerpoint/2010/main" id=
↪"1" dur="indefinite" restart="never" nodeType="tmRoot">
        <p:childTnLst>
          <p:seq concurrent="1" nextAc="seek">
            <p:cTn id="2" restart="whenNotActive" fill="hold" evtFilter="cancelBubble
↪" nodeType="interactiveSeq">
              <p:stCondLst>
                <p:cond evt="onClick" delay="0">
                  <p:tgtEl>
                    <p:spTgt spid="3"/>
                  </p:tgtEl>
                </p:cond>
              </p:stCondLst>
              <p:endSync evt="end" delay="0">
                <p:rtn val="all"/>
              </p:endSync>
              <p:childTnLst>
                <p:par>
                  <p:cTn id="3" fill="hold">
                    <p:stCondLst>
                      <p:cond delay="0"/>
                    </p:stCondLst>
                    <p:childTnLst>
                      <p:par>
                        <p:cTn id="4" fill="hold">
                          <p:stCondLst>
```

```
                          <p:cond delay="0"/>
                        </p:stCondLst>
                        <p:childTnLst>
                          <p:par>
                            <p:cTn id="5" presetID="2" presetClass="mediacall"␣
↪presetSubtype="0" fill="hold" nodeType="clickEffect">
                              <p:stCondLst>
                                <p:cond delay="0"/>
                              </p:stCondLst>
                              <p:childTnLst>
                                <p:cmd type="call" cmd="togglePause">
                                  <p:cBhvr>
                                    <p:cTn id="6" dur="1" fill="hold"/>
                                    <p:tgtEl>
                                      <p:spTgt spid="3"/>
                                    </p:tgtEl>
                                  </p:cBhvr>
                                </p:cmd>
                              </p:childTnLst>
                            </p:cTn>
                          </p:par>
                        </p:childTnLst>
                      </p:cTn>
                    </p:par>
                  </p:childTnLst>
                </p:cTn>
              </p:par>
            </p:childTnLst>
          </p:cTn>
          <p:nextCondLst>
            <p:cond evt="onClick" delay="0">
              <p:tgtEl>
                <p:spTgt spid="3"/>
              </p:tgtEl>
            </p:cond>
          </p:nextCondLst>
        </p:seq>
        <p:video>
          <p:cMediaNode vol="80000">
            <p:cTn id="7" fill="hold" display="0">
              <p:stCondLst>
                <p:cond delay="indefinite"/>
              </p:stCondLst>
            </p:cTn>
            <p:tgtEl>
              <p:spTgt spid="3"/>
            </p:tgtEl>
          </p:cMediaNode>
        </p:video>
        <p:seq concurrent="1" nextAc="seek">
          <p:cTn id="8" restart="whenNotActive" fill="hold" evtFilter="cancelBubble
↪" nodeType="interactiveSeq">
            <p:stCondLst>
              <p:cond evt="onClick" delay="0">
                <p:tgtEl>
                  <p:spTgt spid="4"/>
                </p:tgtEl>
              </p:cond>
```

```
              </p:stCondLst>
              <p:endSync evt="end" delay="0">
                <p:rtn val="all"/>
              </p:endSync>
              <p:childTnLst>
                <p:par>
                  <p:cTn id="9" fill="hold">
                    <p:stCondLst>
                      <p:cond delay="0"/>
                    </p:stCondLst>
                    <p:childTnLst>
                      <p:par>
                        <p:cTn id="10" fill="hold">
                          <p:stCondLst>
                            <p:cond delay="0"/>
                          </p:stCondLst>
                          <p:childTnLst>
                            <p:par>
                              <p:cTn id="11" presetID="2" presetClass="mediacall"
↪presetSubtype="0" fill="hold" nodeType="clickEffect">
                                <p:stCondLst>
                                  <p:cond delay="0"/>
                                </p:stCondLst>
                                <p:childTnLst>
                                  <p:cmd type="call" cmd="togglePause">
                                    <p:cBhvr>
                                      <p:cTn id="12" dur="1" fill="hold"/>
                                      <p:tgtEl>
                                        <p:spTgt spid="4"/>
                                      </p:tgtEl>
                                    </p:cBhvr>
                                  </p:cmd>
                                </p:childTnLst>
                              </p:cTn>
                            </p:par>
                          </p:childTnLst>
                        </p:cTn>
                      </p:par>
                    </p:childTnLst>
                  </p:cTn>
                </p:par>
              </p:childTnLst>
            </p:cTn>
            <p:nextCondLst>
              <p:cond evt="onClick" delay="0">
                <p:tgtEl>
                  <p:spTgt spid="4"/>
                </p:tgtEl>
              </p:cond>
            </p:nextCondLst>
          </p:seq>
          <p:video>
            <p:cMediaNode vol="80000">
              <p:cTn id="13" fill="hold" display="0">
                <p:stCondLst>
                  <p:cond delay="indefinite"/>
                </p:stCondLst>
              </p:cTn>
```

```
              <p:tgtEl>
                <p:spTgt spid="4"/>
              </p:tgtEl>
            </p:cMediaNode>
          </p:video>
        </p:childTnLst>
      </p:cTn>
    </p:par>
  </p:tnLst>
</p:timing>



<!--slide{n}.xml.rels-->

<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship Id="rId1"
      Type="http://schemas.microsoft.com/office/2007/relationships/media"
      Target="../media/media1.mp4"/>
  <Relationship Id="rId2"
      Type="http://sc.../officeDocument/2006/relationships/video"
      Target="../media/media1.mp4"/>
  <Relationship Id="rId3"
      Type="http://sc.../officeDocument/2006/relationships/slideLayout"
      Target="../slideLayouts/slideLayout1.xml"/>
  <!-- this one is the poster frame -->
  <Relationship Id="rId4"
      Type="http://sc.../officeDocument/2006/relationships/image"
      Target="../media/image1.png"/>
</Relationships>


<!--[Content_Types].xml-->

<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
  <!-- ... -->
  <Default Extension="mp4" ContentType="video/unknown"/>
  <Default Extension="png" ContentType="image/png"/>
  <Default Extension="jpeg" ContentType="image/jpeg"/>
  <!-- ... -->
</Types>
```

p:video element

Provides playback controls.

http://openxmldeveloper.org/discussions/formats/f/15/p/1124/2842.aspx#2842

XML Semantics

- Extension DAA4B4D4-6D71-4841-9C94-3DE7FCFB9230 is described as a media extension. It appears to allow:
  - "cropping" the video period (set start and stop time markers)
  - provide for "fade-in"

  - allow for setting bookmarks in the video for fast jumps to a particular location

- This and other extensions are documented in this PDF.

Related Schema Definitions

The root element of a picture shape is a *p:pic (CT_Picture)* element:

```
<xsd:complexType name="CT_Picture">
  <xsd:sequence>
    <xsd:element name="nvPicPr"  type="CT_PictureNonVisual"/>
    <xsd:element name="blipFill" type="a:CT_BlipFillProperties"/>
    <xsd:element name="spPr"     type="a:CT_ShapeProperties"/>
    <xsd:element name="style"    type="a:CT_ShapeStyle"        minOccurs="0"/>
    <xsd:element name="extLst"   type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_PictureNonVisual">
  <xsd:sequence>
    <xsd:element name="cNvPr"    type="a:CT_NonVisualDrawingProps"          minOccurs=
↪"1" maxOccurs="1"/>
    <xsd:element name="cNvPicPr" type="a:CT_NonVisualPictureProperties"     minOccurs=
↪"1" maxOccurs="1"/>
    <xsd:element name="nvPr"     type="CT_ApplicationNonVisualDrawingProps" minOccurs=
↪"1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="hlinkClick" type="CT_Hyperlink"             minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="hlinkHover" type="CT_Hyperlink"             minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"     type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="id"     type="ST_DrawingElementId" use="required"/>
  <xsd:attribute name="name"   type="xsd:string"          use="required"/>
  <xsd:attribute name="descr"  type="xsd:string"          use="optional" default=""/>
  <xsd:attribute name="hidden" type="xsd:boolean"         use="optional" default=
↪"false"/>
  <xsd:attribute name="title"  type="xsd:string"          use="optional" default=""/>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualPictureProperties">
  <xsd:sequence>
    <xsd:element name="picLocks" type="CT_PictureLocking"         minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"   type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="preferRelativeResize" type="xsd:boolean" use="optional"␣
↪default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_ApplicationNonVisualDrawingProps">
```

```
  <xsd:sequence>
    <xsd:element name="ph"          type="CT_Placeholder"      minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:group   ref="a:EG_Media"                              minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="isPhoto"   type="xsd:boolean" use="optional" default="false"/>
  <xsd:attribute name="userDrawn" type="xsd:boolean" use="optional" default="false"/>
</xsd:complexType>

<xsd:group name="EG_Media">
  <xsd:choice>
    <xsd:element name="audioCd"       type="CT_AudioCD"/>
    <xsd:element name="wavAudioFile"  type="CT_EmbeddedWAVAudioFile"/>
    <xsd:element name="audioFile"     type="CT_AudioFile"/>
    <xsd:element name="videoFile"     type="CT_VideoFile"/>
    <xsd:element name="quickTimeFile" type="CT_QuickTimeFile"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_BlipFillProperties">
  <xsd:sequence>
    <xsd:element name="blip"    type="CT_Blip"         minOccurs="0"/>
    <xsd:element name="srcRect" type="CT_RelativeRect" minOccurs="0"/>
    <xsd:group   ref="EG_FillModeProperties"           minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="dpi"          type="xsd:unsignedInt" use="optional"/>
  <xsd:attribute name="rotWithShape" type="xsd:boolean"     use="optional"/>
</xsd:complexType>

<xsd:complexType name="CT_Blip">
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="alphaBiLevel" type="CT_AlphaBiLevelEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaCeiling" type="CT_AlphaCeilingEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaFloor"   type="CT_AlphaFloorEffect"         minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaInv"     type="CT_AlphaInverseEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaMod"     type="CT_AlphaModulateEffect"      minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaModFix"  type="CT_AlphaModulateFixedEffect" minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaRepl"    type="CT_AlphaReplaceEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="biLevel"      type="CT_BiLevelEffect"            minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="blur"         type="CT_BlurEffect"               minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="clrChange"    type="CT_ColorChangeEffect"        minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="clrRepl"      type="CT_ColorReplaceEffect"       minOccurs="1
↪" maxOccurs="1"/>
```

```
      <xsd:element name="duotone"      type="CT_DuotoneEffect"           minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="fillOverlay"  type="CT_FillOverlayEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="grayscl"      type="CT_GrayscaleEffect"         minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="hsl"          type="CT_HSLEffect"               minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="lum"          type="CT_LuminanceEffect"         minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="tint"         type="CT_TintEffect"              minOccurs="1
↪" maxOccurs="1"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="AG_Blob"/>
  <xsd:attribute name="cstate" type="ST_BlipCompression" use="optional" default="none
↪"/>
</xsd:complexType>

<xsd:attributeGroup name="AG_Blob">
  <xsd:attribute ref="r:embed" use="optional" default=""/>
  <xsd:attribute ref="r:link"  use="optional" default=""/>
</xsd:attributeGroup>

<xsd:group name="EG_FillModeProperties">
  <xsd:choice>
    <xsd:element name="tile"    type="CT_TileInfoProperties"    minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="stretch" type="CT_StretchInfoProperties" minOccurs="1"␣
↪maxOccurs="1"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_StretchInfoProperties">
  <xsd:sequence>
    <xsd:element name="fillRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_RelativeRect">
  <xsd:attribute name="l" type="ST_Percentage" use="optional" default="0%"/>
  <xsd:attribute name="t" type="ST_Percentage" use="optional" default="0%"/>
  <xsd:attribute name="r" type="ST_Percentage" use="optional" default="0%"/>
  <xsd:attribute name="b" type="ST_Percentage" use="optional" default="0%"/>
</xsd:complexType>

<xsd:complexType name="CT_ShapeProperties">
  <xsd:sequence>
    <xsd:element name="xfrm"    type="CT_Transform2D"            minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:group ref="EG_Geometry"                                 minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:group ref="EG_FillProperties"                           minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="ln"      type="CT_LineProperties"         minOccurs="0"␣
↪maxOccurs="1"/>
```

```
    <xsd:group ref="EG_EffectProperties"                          minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="scene3d" type="CT_Scene3D"                 minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="sp3d"    type="CT_Shape3D"                 minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</xsd:complexType>

<xsd:complexType name="CT_Transform2D">
  <xsd:sequence>
    <xsd:element name="off" type="CT_Point2D" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="ext" type="CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="rot" type="ST_Angle" use="optional" default="0"/>
  <xsd:attribute name="flipH" type="xsd:boolean" use="optional" default="false"/>
  <xsd:attribute name="flipV" type="xsd:boolean" use="optional" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_Point2D">
  <xsd:attribute name="x" type="ST_Coordinate" use="required"/>
  <xsd:attribute name="y" type="ST_Coordinate" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_PositiveSize2D">
  <xsd:attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
  <xsd:attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</xsd:complexType>

<xsd:group name="EG_Geometry">
  <xsd:choice>
    <xsd:element name="custGeom" type="CT_CustomGeometry2D" minOccurs="1" maxOccurs="1
↪"/>
    <xsd:element name="prstGeom" type="CT_PresetGeometry2D" minOccurs="1" maxOccurs="1
↪"/>
  </xsd:choice>
</xsd:group>

<xsd:group name="EG_FillProperties">
  <xsd:choice>
    <xsd:element name="noFill"    type="CT_NoFillProperties"         minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="solidFill" type="CT_SolidColorFillProperties" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="gradFill"  type="CT_GradientFillProperties"   minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blipFill"  type="CT_BlipFillProperties"       minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="pattFill"  type="CT_PatternFillProperties"    minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="grpFill"   type="CT_GroupFillProperties"      minOccurs="1"␣
↪maxOccurs="1"/>
  </xsd:choice>
</xsd:group>
```

```
<xsd:group name="EG_EffectProperties">
  <xsd:choice>
    <xsd:element name="effectLst" type="CT_EffectList"      minOccurs="1" maxOccurs="1
↪"/>
    <xsd:element name="effectDag" type="CT_EffectContainer" minOccurs="1" maxOccurs="1
↪"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_ShapeStyle">
  <xsd:sequence>
    <xsd:element name="lnRef"     type="CT_StyleMatrixReference" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="fillRef"   type="CT_StyleMatrixReference" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="effectRef" type="CT_StyleMatrixReference" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="fontRef"   type="CT_FontReference"        minOccurs="1"␣
↪maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="notes" type="CT_NotesSlide"/>

<xsd:complexType name="CT_NotesSlide">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="cSld"      type="CT_CommonSlideData"/>
    <xsd:element name="clrMapOvr" type="a:CT_ColorMappingOverride" minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionListModify"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="showMasterSp"     type="xsd:boolean" default="true"/>
  <xsd:attribute name="showMasterPhAnim" type="xsd:boolean" default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_CommonSlideData">
  <xsd:sequence>
    <xsd:element name="bg"          type="CT_Background"       minOccurs="0"/>
    <xsd:element name="spTree"      type="CT_GroupShape"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element name="controls"    type="CT_ControlList"      minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="optional" default=""/>
</xsd:complexType>
```

p:timing related Schema Definitions

The *p:timing* element is a child of the *p:sld (CT_Slide)* element:

```
<xsd:complexType name="CT_Slide">  <!-- denormalized -->
  <xsd:sequence minOccurs="1" maxOccurs="1">
    <xsd:element name="cSld"       type="CT_CommonSlideData"/>
    <xsd:element name="clrMapOvr"  type="a:CT_ColorMappingOverride" minOccurs="0"/>
    <xsd:element name="transition" type="CT_SlideTransition"       minOccurs="0"/>
    <xsd:element name="timing"     type="CT_SlideTiming"           minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_ExtensionListModify"   minOccurs="0"/>
```

```
  </xsd:sequence>
  <xsd:attribute name="showMasterSp"     type="xsd:boolean" use="optional" default=
↪"true"/>
  <xsd:attribute name="showMasterPhAnim" type="xsd:boolean" use="optional" default=
↪"true"/>
  <xsd:attribute name="show"             type="xsd:boolean" use="optional" default=
↪"true"/>
</xsd:complexType>

<xsd:complexType name="CT_SlideTiming">
  <xsd:sequence>
    <xsd:element name="tnLst"  type="CT_TimeNodeList"        minOccurs="0"/>
    <xsd:element name="bldLst" type="CT_BuildList"           minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TimeNodeList">
  <xsd:choice minOccurs="1" maxOccurs="unbounded">
    <xsd:element name="par"        type="CT_TLTimeNodeParallel"/>
    <xsd:element name="seq"        type="CT_TLTimeNodeSequence"/>
    <xsd:element name="excl"       type="CT_TLTimeNodeExclusive"/>
    <xsd:element name="anim"       type="CT_TLAnimateBehavior"/>
    <xsd:element name="animClr"    type="CT_TLAnimateColorBehavior"/>
    <xsd:element name="animEffect" type="CT_TLAnimateEffectBehavior"/>
    <xsd:element name="animMotion" type="CT_TLAnimateMotionBehavior"/>
    <xsd:element name="animRot"    type="CT_TLAnimateRotationBehavior"/>
    <xsd:element name="animScale"  type="CT_TLAnimateScaleBehavior"/>
    <xsd:element name="cmd"        type="CT_TLCommandBehavior"/>
    <xsd:element name="set"        type="CT_TLSetBehavior"/>
    <xsd:element name="audio"      type="CT_TLMediaNodeAudio"/>
    <xsd:element name="video"      type="CT_TLMediaNodeVideo"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="CT_TLTimeNodeParallel">
  <xsd:sequence>
    <xsd:element name="cTn" type="CT_TLCommonTimeNodeData"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TLTimeNodeSequence">
  <xsd:sequence>
    <xsd:element name="cTn"         type="CT_TLCommonTimeNodeData"/>
    <xsd:element name="prevCondLst" type="CT_TLTimeConditionList" minOccurs="0"/>
    <xsd:element name="nextCondLst" type="CT_TLTimeConditionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="concurrent" type="xsd:boolean"/>
  <xsd:attribute name="prevAc"     type="ST_TLPreviousActionType"/>
  <xsd:attribute name="nextAc"     type="ST_TLNextActionType"/>
</xsd:complexType>

<xsd:complexType name="CT_TLCommonTimeNodeData">
  <xsd:sequence>
    <xsd:element name="stCondLst"  type="CT_TLTimeConditionList" minOccurs="0"/>
    <xsd:element name="endCondLst" type="CT_TLTimeConditionList" minOccurs="0"/>
    <xsd:element name="endSync"    type="CT_TLTimeCondition"     minOccurs="0"/>
    <xsd:element name="iterate"    type="CT_TLIterateData"       minOccurs="0"/>
```

```
    <xsd:element name="childTnLst" type="CT_TimeNodeList"        minOccurs="0"/>
    <xsd:element name="subTnLst"   type="CT_TimeNodeList"        minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id"            type="ST_TLTimeNodeID"/>
  <xsd:attribute name="presetID"      type="xsd:int"/>
  <xsd:attribute name="presetClass"   type="ST_TLTimeNodePresetClassType"/>
  <xsd:attribute name="presetSubtype" type="xsd:int"/>
  <xsd:attribute name="dur"           type="ST_TLTime"/>
  <xsd:attribute name="repeatCount"   type="ST_TLTime"                    default=
↪"1000"/>
  <xsd:attribute name="repeatDur"     type="ST_TLTime"/>
  <xsd:attribute name="spd"           type="a:ST_Percentage"              default=
↪"100%"/>
  <xsd:attribute name="accel"         type="a:ST_PositiveFixedPercentage"  default="0%
↪"/>
  <xsd:attribute name="decel"         type="a:ST_PositiveFixedPercentage"  default="0%
↪"/>
  <xsd:attribute name="autoRev"       type="xsd:boolean"                  default=
↪"false"/>
  <xsd:attribute name="restart"       type="ST_TLTimeNodeRestartType"/>
  <xsd:attribute name="fill"          type="ST_TLTimeNodeFillType"/>
  <xsd:attribute name="syncBehavior"  type="ST_TLTimeNodeSyncType"/>
  <xsd:attribute name="tmFilter"      type="xsd:string"/>
  <xsd:attribute name="evtFilter"     type="xsd:string"/>
  <xsd:attribute name="display"       type="xsd:boolean"/>
  <xsd:attribute name="masterRel"     type="ST_TLTimeNodeMasterRelation"/>
  <xsd:attribute name="bldLvl"        type="xsd:int"/>
  <xsd:attribute name="grpId"         type="xsd:unsignedInt"/>
  <xsd:attribute name="afterEffect"   type="xsd:boolean"/>
  <xsd:attribute name="nodeType"      type="ST_TLTimeNodeType"/>
  <xsd:attribute name="nodePh"        type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_TLMediaNodeVideo">
  <xsd:sequence>
    <xsd:element name="cMediaNode" type="CT_TLCommonMediaNodeData"/>
  </xsd:sequence>
  <xsd:attribute name="fullScrn" type="xsd:boolean" default="false"/>
</xsd:complexType>
```

Shapes - In General

Each visual element in a PowerPoint presentation is a *shape*. A shape appears on the "canvas" of a slide, which includes the various types of *master*. Within a slide, shapes appear in a *shape tree*, corresponding to an `<p:spTree>` element.

The following table summarizes the six shape types:

shape type	element
auto shape	`<p:sp>`
group shape	`<p:grpSp>`
graphicFrame	`<p:graphicFrame>`
connector	`<p:cxnSp>`
picture	`<p:pic>`
content part	`<p:contentPart>`

Some of these shape types have important sub-types. For example, a placeholder, a text box, and a preset geometry shape such as a circle, are all defined with an <p:sp> element.

<p:sp> shape elements

The <p:sp> element is used for three types of shape: placeholder, text box, and geometric shapes. A geometric shape with preset geometry is referred to as an *auto shape*. Placeholder shapes are documented on the *Placeholders* page. Auto shapes are documented on the *Auto Shape* page.

Geometric shapes are the familiar shapes that may be placed on a slide such as a rectangle or an ellipse. In the PowerPoint UI they are simply called shapes. There are two types of geometric shapes, preset geometry shapes and custom geometry shapes.

Shape.shape_id and Shape.name

Shape.shape_id is read-only and is assigned by python-pptx when necessary.

Proposed protocol:

```
>>> shape.shape_id
42
>>> shape.name
u'Picture 5'
>>> shape.name = 'T501 - Foo; B. Baz; 2014'
>>> shape.name
u'T501 - Foo; B. Baz; 2014'
```

Shape.rotation

Read/write float degrees of clockwise rotation. Negative values can be used for counter-clockwise rotation.

XML Semantics ST_Angle is an integer value, 60,000 to each degree. PowerPoint appears to only ever use positive values. Oddly, the UI uses positive values for counter-clockwise rotation while the XML uses positive increase for clockwise rotation.

PowerPoint behavior It appears graphic frame shapes can't be rotated. AutoShape, group, connector, and picture all rotate fine.

Proposed protocol:

```
>>> shape.rotation
0.0
>>> shape.rotation = 45.2
>>> shape.rotation
45.2
```

Math:

```
def rot_from_angle(value):
    """
    Return positive integer rotation in 60,000ths of a degree
    corresponding to *value* expressed as a float number of degrees.
    """
    DEGREE_INCREMENTS = 60000
    THREE_SIXTY = 360 * DEGREE_INCREMENTS
```

```
    # modulo normalizes negative and >360 degree values
    return int(round(value * DEGREE_INCREMENTS)) % THREE_SIXTY
```

Specimen XML

Geometric shape (rounded rectangle):

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="3" name="Rounded Rectangle 2"/>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="760096" y="562720"/>
      <a:ext cx="2520824" cy="914400"/>
    </a:xfrm>
    <a:prstGeom prst="roundRect">
      <a:avLst>
        <a:gd name="adj" fmla="val 30346"/>
      </a:avLst>
    </a:prstGeom>
  </p:spPr>
  <p:style>
    <a:lnRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="3">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="lt1"/>
    </a:fontRef>
  </p:style>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr"/>
    <a:lstStyle/>
    <a:p>
      <a:pPr algn="ctr"/>
      <a:r>
        <a:rPr lang="en-US" dirty="0" smtClean="0"/>
        <a:t>This is text inside a rounded rectangle</a:t>
      </a:r>
      <a:endParaRPr lang="en-US" dirty="0"/>
    </a:p>
  </p:txBody>
</p:sp>
```

Default textbox shape as inserted by PowerPoint:

```
<p:sp>
  <p:nvSpPr>
```

```
    <p:cNvPr id="2" name="TextBox 1"/>
    <p:cNvSpPr txBox="1"/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="1997289" y="2529664"/>
      <a:ext cx="2390398" cy="369332"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
    <a:noFill/>
  </p:spPr>
  <p:txBody>
    <a:bodyPr wrap="none" rtlCol="0">
      <a:spAutoFit/>
    </a:bodyPr>
    <a:lstStyle/>
    <a:p>
      <a:r>
        <a:rPr lang="en-US" dirty="0" smtClean="0"/>
        <a:t>This is text in a text box</a:t>
      </a:r>
      <a:endParaRPr lang="en-US" dirty="0"/>
    </a:p>
  </p:txBody>
</p:sp>
```

Group shape (some contents elided for size):

```
<p:grpSp>
  <p:nvGrpSpPr>
    <p:cNvPr id="4" name="Group 3"/>
    <p:cNvGrpSpPr/>
    <p:nvPr/>
  </p:nvGrpSpPr>
  <p:grpSpPr>
    <a:xfrm>
      <a:off x="2438400" y="2971800"/>
      <a:ext cx="4267200" cy="914400"/>
      <a:chOff x="2438400" y="2971800"/>
      <a:chExt cx="4267200" cy="914400"/>
    </a:xfrm>
  </p:grpSpPr>
  <p:sp>
    <p:nvSpPr>
      <p:cNvPr id="2" name="Rectangle 1"/>
      <p:cNvSpPr/>
      <p:nvPr/>
    </p:nvSpPr>
    <!-- some contents elided -->
  </p:sp>
  <p:sp>
    <p:nvSpPr>
      <p:cNvPr id="3" name="Oval 2"/>
      <p:cNvSpPr/>
      <p:nvPr/>
```

```
    </p:nvSpPr>
    <!-- some contents elided -->
  </p:sp>
</p:grpSp>
```

Graphical object (e.g. table, chart) in a graphic frame:

```
<p:graphicFrame>
  <p:nvGraphicFramePr>
    <p:cNvPr id="2" name="Table 1"/>
    <p:cNvGraphicFramePr>
      <a:graphicFrameLocks noGrp="1"/>
    </p:cNvGraphicFramePr>
    <p:nvPr/>
  </p:nvGraphicFramePr>
  <p:xfrm>
    <a:off x="1524000" y="1397000"/>
    <a:ext cx="6096000" cy="741680"/>
  </p:xfrm>
  <a:graphic>
    <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/table">
      <!-- graphical object XML or ref goes here -->
    </a:graphicData>
  </a:graphic>
</p:graphicFrame>
```

Connector shape:

```
<p:cxnSp>
  <p:nvCxnSpPr>
    <p:cNvPr id="6" name="Straight Connector 5"/>
    <p:cNvCxnSpPr/>
    <p:nvPr/>
  </p:nvCxnSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="3131840" y="3068960"/>
      <a:ext cx="2736304" cy="0"/>
    </a:xfrm>
    <a:prstGeom prst="line">
      <a:avLst/>
    </a:prstGeom>
  </p:spPr>
  <p:style>
    <a:lnRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="0">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="tx1"/>
    </a:fontRef>
  </p:style>
</p:cxnSp>
```

Picture shape:

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="6" name="Picture 5" descr="python-logo.gif"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
    <a:xfrm>
      <a:off x="5580112" y="1988840"/>
      <a:ext cx="2679700" cy="901700"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
    <a:ln>
      <a:solidFill>
        <a:schemeClr val="bg1">
          <a:lumMod val="85000"/>
        </a:schemeClr>
      </a:solidFill>
    </a:ln>
  </p:spPr>
</p:pic>
```

Resources

- DrawingML Shapes on officeopenxml.com
- Shape Object MSDN page
- MsoShapeType Enumeration

Schema excerpt

```
<xsd:complexType name="CT_Shape">
  <xsd:sequence>
    <xsd:element name="nvSpPr" type="CT_ShapeNonVisual"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties"/>
    <xsd:element name="style"  type="a:CT_ShapeStyle"        minOccurs="0"/>
    <xsd:element name="txBody" type="a:CT_TextBody"          minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="useBgFill" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_ShapeNonVisual">
```

```
  <xsd:sequence>
    <xsd:element name="cNvPr"   type="a:CT_NonVisualDrawingProps"/>
    <xsd:element name="cNvSpPr" type="a:CT_NonVisualDrawingShapeProps"/>
    <xsd:element name="nvPr"    type="CT_ApplicationNonVisualDrawingProps"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ShapeProperties">
  <xsd:sequence>
    <xsd:element name="xfrm"               type="CT_Transform2D"          ␣
↪minOccurs="0"/>
    <xsd:group   ref ="EG_Geometry"                                       ␣
↪minOccurs="0"/>
    <xsd:group   ref ="EG_FillProperties"                                 ␣
↪minOccurs="0"/>
    <xsd:element name="ln"                 type="CT_LineProperties"       ␣
↪minOccurs="0"/>
    <xsd:group   ref ="EG_EffectProperties"                               ␣
↪minOccurs="0"/>
    <xsd:element name="scene3d"            type="CT_Scene3D"              ␣
↪minOccurs="0"/>
    <xsd:element name="sp3d"               type="CT_Shape3D"              ␣
↪minOccurs="0"/>
    <xsd:element name="extLst"             type="CT_OfficeArtExtensionList"␣
↪minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode"/>
</xsd:complexType>

<xsd:complexType name="CT_ShapeStyle">
  <xsd:sequence>
    <xsd:element name="lnRef"     type="CT_StyleMatrixReference"/>
    <xsd:element name="fillRef"   type="CT_StyleMatrixReference"/>
    <xsd:element name="effectRef" type="CT_StyleMatrixReference"/>
    <xsd:element name="fontRef"   type="CT_FontReference"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextBody">
  <xsd:sequence>
    <xsd:element name="bodyPr"   type="CT_TextBodyProperties"/>
    <xsd:element name="lstStyle" type="CT_TextListStyle" minOccurs="0"/>
    <xsd:element name="p"        type="CT_TextParagraph" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ExtensionListModify">
  <xsd:sequence>
    <xsd:group ref="EG_ExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="mod" type="xsd:boolean" default="false"/>
</xsd:complexType>

<!-- Supporting elements -->

<xsd:complexType name="CT_NonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="hlinkClick" type="CT_Hyperlink"             minOccurs="0"/>
```

```
    <xsd:element name="hlinkHover" type="CT_Hyperlink"              minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id"     type="ST_DrawingElementId" use="required"/>
  <xsd:attribute name="name"   type="xsd:string"          use="required"/>
  <xsd:attribute name="descr"  type="xsd:string"          default=""/>
  <xsd:attribute name="hidden" type="xsd:boolean"         default="false"/>
  <xsd:attribute name="title"  type="xsd:string"          default=""/>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualDrawingShapeProps">
  <xsd:sequence>
    <xsd:element name="spLocks" type="CT_ShapeLocking"           minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="txBox" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_ApplicationNonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="ph"          type="CT_Placeholder"      minOccurs="0"/>
    <xsd:group   ref ="a:EG_Media"                             minOccurs="0"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="isPhoto"   type="xsd:boolean" default="false"/>
  <xsd:attribute name="userDrawn" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_Transform2D">
  <xsd:sequence>
    <xsd:element name="off" type="CT_Point2D"        minOccurs="0"/>
    <xsd:element name="ext" type="CT_PositiveSize2D" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="rot"   type="ST_Angle"    default="0"/>
  <xsd:attribute name="flipH" type="xsd:boolean" default="false"/>
  <xsd:attribute name="flipV" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:group name="EG_Geometry">
  <xsd:choice>
    <xsd:element name="custGeom" type="CT_CustomGeometry2D"/>
    <xsd:element name="prstGeom" type="CT_PresetGeometry2D"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_CustomGeometry2D">
  <xsd:sequence>
    <xsd:element name="avLst"   type="CT_GeomGuideList"      minOccurs="0"/>
    <xsd:element name="gdLst"   type="CT_GeomGuideList"      minOccurs="0"/>
    <xsd:element name="ahLst"   type="CT_AdjustHandleList"   minOccurs="0"/>
    <xsd:element name="cxnLst"  type="CT_ConnectionSiteList" minOccurs="0"/>
    <xsd:element name="rect"    type="CT_GeomRect"           minOccurs="0"/>
    <xsd:element name="pathLst" type="CT_Path2DList"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_PresetGeometry2D">
```

```
  <xsd:sequence>
    <xsd:element name="avLst" type="CT_GeomGuideList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="prst" type="ST_ShapeType" use="required"/>
</xsd:complexType>

<xsd:group name="EG_FillProperties">
  <xsd:choice>
    <xsd:element name="noFill"    type="CT_NoFillProperties"/>
    <xsd:element name="solidFill" type="CT_SolidColorFillProperties"/>
    <xsd:element name="gradFill"  type="CT_GradientFillProperties"/>
    <xsd:element name="blipFill"  type="CT_BlipFillProperties"/>
    <xsd:element name="pattFill"  type="CT_PatternFillProperties"/>
    <xsd:element name="grpFill"   type="CT_GroupFillProperties"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_LineProperties">
  <xsd:sequence>
    <xsd:group   ref="EG_LineFillProperties"                    minOccurs="0"/>
    <xsd:group   ref="EG_LineDashProperties"                    minOccurs="0"/>
    <xsd:group   ref="EG_LineJoinProperties"                    minOccurs="0"/>
    <xsd:element name="headEnd" type="CT_LineEndProperties"     minOccurs="0"/>
    <xsd:element name="tailEnd" type="CT_LineEndProperties"     minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="w"    type="ST_LineWidth"/>
  <xsd:attribute name="cap"  type="ST_LineCap"/>
  <xsd:attribute name="cmpd" type="ST_CompoundLine"/>
  <xsd:attribute name="algn" type="ST_PenAlignment"/>
</xsd:complexType>

<xsd:complexType name="CT_Point2D">
  <xsd:attribute name="x" type="ST_Coordinate" use="required"/>
  <xsd:attribute name="y" type="ST_Coordinate" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_PositiveSize2D">
  <xsd:attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
  <xsd:attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</xsd:complexType>

<xsd:group name="EG_EffectProperties">
  <xsd:choice>
    <xsd:element name="effectLst" type="CT_EffectList"/>
    <xsd:element name="effectDag" type="CT_EffectContainer"/>
  </xsd:choice>
</xsd:group>

<xsd:group name="EG_Media">
  <xsd:choice>
    <xsd:element name="audioCd"       type="CT_AudioCD"/>
    <xsd:element name="wavAudioFile"  type="CT_EmbeddedWAVAudioFile"/>
    <xsd:element name="audioFile"     type="CT_AudioFile"/>
    <xsd:element name="videoFile"     type="CT_VideoFile"/>
    <xsd:element name="quickTimeFile" type="CT_QuickTimeFile"/>
  </xsd:choice>
</xsd:group>
```

```
<xsd:simpleType name="ST_DrawingElementId">
  <xsd:restriction base="xsd:unsignedInt"/>
</xsd:simpleType>

<xsd:simpleType name="ST_Angle">
  <xsd:restriction base="xsd:int"/>
</xsd:simpleType>
```

SlideShapes

Shapes on a slide are specified in the shape tree, a hierarchical data structure rooted in a `<p:spTree>` element.

Schema excerpt

```
<xsd:element name="sld" type="CT_Slide"/>

<xsd:complexType name="CT_Slide">  <!-- denormalized -->
  <xsd:sequence minOccurs="1" maxOccurs="1">
    <xsd:element name="cSld"       type="CT_CommonSlideData"/>
    <xsd:element name="clrMapOvr"  type="a:CT_ColorMappingOverride" minOccurs="0"/>
    <xsd:element name="transition" type="CT_SlideTransition"        minOccurs="0"/>
    <xsd:element name="timing"     type="CT_SlideTiming"            minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_ExtensionListModify"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="showMasterSp"     type="xsd:boolean" default="true"/>
  <xsd:attribute name="showMasterPhAnim" type="xsd:boolean" default="true"/>
  <xsd:attribute name="show"             type="xsd:boolean" default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_CommonSlideData">
  <xsd:sequence>
    <xsd:element name="bg"          type="CT_Background"       minOccurs="0"/>
    <xsd:element name="spTree"      type="CT_GroupShape"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element name="controls"    type="CT_ControlList"      minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="optional" default=""/>
</xsd:complexType>

<xsd:complexType name="CT_GroupShape">
  <xsd:sequence>
    <xsd:element name="nvGrpSpPr" type="CT_GroupShapeNonVisual"/>
    <xsd:element name="grpSpPr"   type="a:CT_GroupShapeProperties"/>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="sp"           type="CT_Shape"/>
      <xsd:element name="grpSp"        type="CT_GroupShape"/>
      <xsd:element name="graphicFrame" type="CT_GraphicalObjectFrame"/>
      <xsd:element name="cxnSp"        type="CT_Connector"/>
      <xsd:element name="pic"          type="CT_Picture"/>
      <xsd:element name="contentPart"  type="CT_Rel"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0"/>
```

```
  </xsd:sequence>
</xsd:complexType>
```

Auto Shape

Summary

AutoShape is the name the MS Office API uses for a shape with preset geometry, those referred to in the PowerPoint UI simply as *shape*. Typical examples are a rectangle, a circle, or a star shape. An auto shape has a type (preset geometry), an outline line style, a fill, and an effect, in addition to a position and size. The outline, fill, and effect can each be None.

Each auto shape is contained in an `<p:sp>` element.

Auto shape type

There are 187 pre-defined auto shape types, each corresponding to a distinct preset geometry. Some types have one or more *adjustment value*, which corresponds to the position of an adjustment handle (small yellow diamond) on the shape in the UI that changes an aspect of the shape geometry, for example the corner radius of a rounded rectangle.

- MsoAutoShapeType Enumeration page on MSDN

Adjustment values

Many auto shapes may be adjusted to change their shape beyond just their width, height, and rotation. In the PowerPoint application, this is accomplished by dragging small yellow diamond-shaped handles that appear on the shape when it is selected.

In the XML, the position of these adjustment handles is reflected in the contents of the `<a:avLst>` element:

```
<a:prstGeom prst="roundRect">
  <a:avLst>
    <a:gd name="adj" fmla="val 30346"/>
  </a:avLst>
</a:prstGeom>
```

The `<a:avLst>` element can be empty, even if the shape has available adjustments.

guide a guide is essentially a variable within DrawingML. Its value is the result of a simple expression like `"*/ 100000 w ss"`. Its value may then be used in other expressions or directly to specify a point, by referring to it by name.

`<a:gd>` stands for *guide* ...

Where are the custom geometries defined? The geometries for autoshapes are defined in `presetShapeDefinitions.xml`, located in:

`pptx/ref/ISO-IEC-29500-1/schemas/dml-geometries/OfficeOpenXML-DrawingMLGeometries/`.

What is the range of the formula values? In general, the nominal adjustment value range for the preset shapes is 0 to 100000. However, the actual value stored can be outside that range, depending on the formulas involved. For example, in the case of the chevron shape, the range is from 0 to 100000 * width / short-side. When the shape is taller than it is wide, width is the short side and the range is 0 to 100000. However, when the shape is wider

than it is tall, the range can be much larger. Required values are probably best discovered via experimentation or calculation based on the preset formula.

The `<a:avLst>` element does not appear initially when the shape is inserted. However, if the adjustment handle is moved and then moved back to the default value, the `<a:avLst>` element is not removed. The default value becomes the effective value for the guide if the `<a:avLst>` element is not present.

The adjustment value is preserved over scaling of the object. Its effective value can change however, depending on the formula. In the case of the chevron shape, the effective value is a function of `ss`, the "short side", and the adjustment handle moves on screen when the chevron is made wider than it is tall.

A certain level of documentation for auto shapes could be generated from the XML shape definition file. The preset name and the number of adjustment values it has (guide definitions) and the adjustment value name can readily be determined by inspection of the XML.

What coordinate system are the formula values expressed in? There appear to be two coordinate systems at work. The 0 to 100000 bit the adjustment handles are expressed in appears to be the (arbitrary) shape coordinate system. The built-in values like `w`, `ss`, etc., appear to be in the slide coordinate system, the one the `xfrm` element traffics in.

Probably worth experimenting with some custom shapes to find out (edit and rezip with bash one-liner).

useful web resources What useful web resources are out there on DrawingML?

http://officeopenxml.com/drwSp-custGeom.php

XML produced by PowerPoint® client

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="3" name="Rounded Rectangle 2"/>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="760096" y="562720"/>
      <a:ext cx="2520824" cy="914400"/>
    </a:xfrm>
    <a:prstGeom prst="roundRect">
      <a:avLst>
        <a:gd name="adj" fmla="val 30346"/>
      </a:avLst>
    </a:prstGeom>
  </p:spPr>
  <p:style>
    <a:lnRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="3">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="lt1"/>
    </a:fontRef>
```

```
  </p:style>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr"/>
    <a:lstStyle/>
    <a:p>
      <a:pPr algn="ctr"/>
      <a:r>
        <a:rPr lang="en-US" dirty="0" smtClean="0"/>
        <a:t>This is text inside a rounded rectangle</a:t>
      </a:r>
      <a:endParaRPr lang="en-US" dirty="0"/>
    </a:p>
  </p:txBody>
</p:sp>
```

Resources

- DrawingML Shapes on officeopenxml.com
- Shape Object MSDN page
- MsoShapeType Enumeration

CT_PresetGeometry2D

Spec Name	Preset Geometry
Tag(s)	a:prstGeom
Namespace	drawingml (dml-main.xsd)
Spec Section	20.1.9.18

Spec text

> This element specifies when a preset geometric shape should be used instead of a custom geometric shape. The generating application should be able to render all preset geometries enumerated in the ST_ShapeType list.

Schema excerpt

```
<xsd:complexType name="CT_PresetGeometry2D">
  <xsd:sequence>
    <xsd:element name="avLst" type="CT_GeomGuideList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="prst" type="ST_ShapeType" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_GeomGuideList">
  <xsd:sequence>
    <xsd:element name="gd" type="CT_GeomGuide" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="CT_GeomGuide">
  <xsd:attribute name="name" type="ST_GeomGuideName"    use="required"/>
  <xsd:attribute name="fmla" type="ST_GeomGuideFormula" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_GeomGuideName">
  <xsd:restriction base="xsd:token"/>
</xsd:simpleType>

<xsd:simpleType name="ST_GeomGuideFormula">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="ST_ShapeType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="line"/>
    <xsd:enumeration value="lineInv"/>
    <xsd:enumeration value="triangle"/>
    <xsd:enumeration value="rtTriangle"/>
    <xsd:enumeration value="rect"/>
    <xsd:enumeration value="diamond"/>
    <xsd:enumeration value="parallelogram"/>
    <xsd:enumeration value="trapezoid"/>
    <xsd:enumeration value="nonIsoscelesTrapezoid"/>
    <xsd:enumeration value="pentagon"/>
    <xsd:enumeration value="hexagon"/>
    <xsd:enumeration value="heptagon"/>
    <xsd:enumeration value="octagon"/>
    <xsd:enumeration value="decagon"/>
    <xsd:enumeration value="dodecagon"/>
    <xsd:enumeration value="star4"/>
    <xsd:enumeration value="star5"/>
    <xsd:enumeration value="star6"/>
    <xsd:enumeration value="star7"/>
    <xsd:enumeration value="star8"/>
    <xsd:enumeration value="star10"/>
    <xsd:enumeration value="star12"/>
    <xsd:enumeration value="star16"/>
    <xsd:enumeration value="star24"/>
    <xsd:enumeration value="star32"/>
    <xsd:enumeration value="roundRect"/>
    <xsd:enumeration value="round1Rect"/>
    <xsd:enumeration value="round2SameRect"/>
    <xsd:enumeration value="round2DiagRect"/>
    <xsd:enumeration value="snipRoundRect"/>
    <xsd:enumeration value="snip1Rect"/>
    <xsd:enumeration value="snip2SameRect"/>
    <xsd:enumeration value="snip2DiagRect"/>
    <xsd:enumeration value="plaque"/>
    <xsd:enumeration value="ellipse"/>
    <xsd:enumeration value="teardrop"/>
    <xsd:enumeration value="homePlate"/>
    <xsd:enumeration value="chevron"/>
    <xsd:enumeration value="pieWedge"/>
    <xsd:enumeration value="pie"/>
    <xsd:enumeration value="blockArc"/>
    <xsd:enumeration value="donut"/>
    <xsd:enumeration value="noSmoking"/>
```

```
    <xsd:enumeration value="rightArrow"/>
    <xsd:enumeration value="leftArrow"/>
    <xsd:enumeration value="upArrow"/>
    <xsd:enumeration value="downArrow"/>
    <xsd:enumeration value="stripedRightArrow"/>
    <xsd:enumeration value="notchedRightArrow"/>
    <xsd:enumeration value="bentUpArrow"/>
    <xsd:enumeration value="leftRightArrow"/>
    <xsd:enumeration value="upDownArrow"/>
    <xsd:enumeration value="leftUpArrow"/>
    <xsd:enumeration value="leftRightUpArrow"/>
    <xsd:enumeration value="quadArrow"/>
    <xsd:enumeration value="leftArrowCallout"/>
    <xsd:enumeration value="rightArrowCallout"/>
    <xsd:enumeration value="upArrowCallout"/>
    <xsd:enumeration value="downArrowCallout"/>
    <xsd:enumeration value="leftRightArrowCallout"/>
    <xsd:enumeration value="upDownArrowCallout"/>
    <xsd:enumeration value="quadArrowCallout"/>
    <xsd:enumeration value="bentArrow"/>
    <xsd:enumeration value="uturnArrow"/>
    <xsd:enumeration value="circularArrow"/>
    <xsd:enumeration value="leftCircularArrow"/>
    <xsd:enumeration value="leftRightCircularArrow"/>
    <xsd:enumeration value="curvedRightArrow"/>
    <xsd:enumeration value="curvedLeftArrow"/>
    <xsd:enumeration value="curvedUpArrow"/>
    <xsd:enumeration value="curvedDownArrow"/>
    <xsd:enumeration value="swooshArrow"/>
    <xsd:enumeration value="cube"/>
    <xsd:enumeration value="can"/>
    <xsd:enumeration value="lightningBolt"/>
    <xsd:enumeration value="heart"/>
    <xsd:enumeration value="sun"/>
    <xsd:enumeration value="moon"/>
    <xsd:enumeration value="smileyFace"/>
    <xsd:enumeration value="irregularSeal1"/>
    <xsd:enumeration value="irregularSeal2"/>
    <xsd:enumeration value="foldedCorner"/>
    <xsd:enumeration value="bevel"/>
    <xsd:enumeration value="frame"/>
    <xsd:enumeration value="halfFrame"/>
    <xsd:enumeration value="corner"/>
    <xsd:enumeration value="diagStripe"/>
    <xsd:enumeration value="chord"/>
    <xsd:enumeration value="arc"/>
    <xsd:enumeration value="leftBracket"/>
    <xsd:enumeration value="rightBracket"/>
    <xsd:enumeration value="leftBrace"/>
    <xsd:enumeration value="rightBrace"/>
    <xsd:enumeration value="bracketPair"/>
    <xsd:enumeration value="bracePair"/>
    <xsd:enumeration value="straightConnector1"/>
    <xsd:enumeration value="bentConnector2"/>
    <xsd:enumeration value="bentConnector3"/>
    <xsd:enumeration value="bentConnector4"/>
    <xsd:enumeration value="bentConnector5"/>
    <xsd:enumeration value="curvedConnector2"/>
```

```
    <xsd:enumeration value="curvedConnector3"/>
    <xsd:enumeration value="curvedConnector4"/>
    <xsd:enumeration value="curvedConnector5"/>
    <xsd:enumeration value="callout1"/>
    <xsd:enumeration value="callout2"/>
    <xsd:enumeration value="callout3"/>
    <xsd:enumeration value="accentCallout1"/>
    <xsd:enumeration value="accentCallout2"/>
    <xsd:enumeration value="accentCallout3"/>
    <xsd:enumeration value="borderCallout1"/>
    <xsd:enumeration value="borderCallout2"/>
    <xsd:enumeration value="borderCallout3"/>
    <xsd:enumeration value="accentBorderCallout1"/>
    <xsd:enumeration value="accentBorderCallout2"/>
    <xsd:enumeration value="accentBorderCallout3"/>
    <xsd:enumeration value="wedgeRectCallout"/>
    <xsd:enumeration value="wedgeRoundRectCallout"/>
    <xsd:enumeration value="wedgeEllipseCallout"/>
    <xsd:enumeration value="cloudCallout"/>
    <xsd:enumeration value="cloud"/>
    <xsd:enumeration value="ribbon"/>
    <xsd:enumeration value="ribbon2"/>
    <xsd:enumeration value="ellipseRibbon"/>
    <xsd:enumeration value="ellipseRibbon2"/>
    <xsd:enumeration value="leftRightRibbon"/>
    <xsd:enumeration value="verticalScroll"/>
    <xsd:enumeration value="horizontalScroll"/>
    <xsd:enumeration value="wave"/>
    <xsd:enumeration value="doubleWave"/>
    <xsd:enumeration value="plus"/>
    <xsd:enumeration value="flowChartProcess"/>
    <xsd:enumeration value="flowChartDecision"/>
    <xsd:enumeration value="flowChartInputOutput"/>
    <xsd:enumeration value="flowChartPredefinedProcess"/>
    <xsd:enumeration value="flowChartInternalStorage"/>
    <xsd:enumeration value="flowChartDocument"/>
    <xsd:enumeration value="flowChartMultidocument"/>
    <xsd:enumeration value="flowChartTerminator"/>
    <xsd:enumeration value="flowChartPreparation"/>
    <xsd:enumeration value="flowChartManualInput"/>
    <xsd:enumeration value="flowChartManualOperation"/>
    <xsd:enumeration value="flowChartConnector"/>
    <xsd:enumeration value="flowChartPunchedCard"/>
    <xsd:enumeration value="flowChartPunchedTape"/>
    <xsd:enumeration value="flowChartSummingJunction"/>
    <xsd:enumeration value="flowChartOr"/>
    <xsd:enumeration value="flowChartCollate"/>
    <xsd:enumeration value="flowChartSort"/>
    <xsd:enumeration value="flowChartExtract"/>
    <xsd:enumeration value="flowChartMerge"/>
    <xsd:enumeration value="flowChartOfflineStorage"/>
    <xsd:enumeration value="flowChartOnlineStorage"/>
    <xsd:enumeration value="flowChartMagneticTape"/>
    <xsd:enumeration value="flowChartMagneticDisk"/>
    <xsd:enumeration value="flowChartMagneticDrum"/>
    <xsd:enumeration value="flowChartDisplay"/>
    <xsd:enumeration value="flowChartDelay"/>
    <xsd:enumeration value="flowChartAlternateProcess"/>
```

```
    <xsd:enumeration value="flowChartOffpageConnector"/>
    <xsd:enumeration value="actionButtonBlank"/>
    <xsd:enumeration value="actionButtonHome"/>
    <xsd:enumeration value="actionButtonHelp"/>
    <xsd:enumeration value="actionButtonInformation"/>
    <xsd:enumeration value="actionButtonForwardNext"/>
    <xsd:enumeration value="actionButtonBackPrevious"/>
    <xsd:enumeration value="actionButtonEnd"/>
    <xsd:enumeration value="actionButtonBeginning"/>
    <xsd:enumeration value="actionButtonReturn"/>
    <xsd:enumeration value="actionButtonDocument"/>
    <xsd:enumeration value="actionButtonSound"/>
    <xsd:enumeration value="actionButtonMovie"/>
    <xsd:enumeration value="gear6"/>
    <xsd:enumeration value="gear9"/>
    <xsd:enumeration value="funnel"/>
    <xsd:enumeration value="mathPlus"/>
    <xsd:enumeration value="mathMinus"/>
    <xsd:enumeration value="mathMultiply"/>
    <xsd:enumeration value="mathDivide"/>
    <xsd:enumeration value="mathEqual"/>
    <xsd:enumeration value="mathNotEqual"/>
    <xsd:enumeration value="cornerTabs"/>
    <xsd:enumeration value="squareTabs"/>
    <xsd:enumeration value="plaqueTabs"/>
    <xsd:enumeration value="chartX"/>
    <xsd:enumeration value="chartStar"/>
    <xsd:enumeration value="chartPlus"/>
  </xsd:restriction>
```

Shape hyperlink

In addition to hyperlinks in text, PowerPoint supports hyperlinks on shapes. However, this feature is just a subset of the more general *click action* functionality on shapes.

A run or shape can actually have two distinct mouse event behaviors, one for (left) clicking and another for rolling over with the mouse. These are independent; a run or shape can have one, the other, both, or neither. These are the two hyperlink types reported by the Hyperlink.type attribute and using enumeration values from MsoHyperlinkType.

A "true" hyperlink in PowerPoint is indicated by having no *action* attribute on the *a:hlinkClick* element. The general hyperlink mechanism of storing a URL in a mapped relationship is used by other verbs such as *hlinkfile* and *program*, but these are considered distinct from hyperlinks for the sake of this analysis. That distinction is reflected in the API.

See also: _Run.hyperlink, pptx.text._Hyperlink

Glossary

click action The behavior to be executed when the user clicks on a shape. There are 14 possible behaviors, such as navigate to a different slide, open a web page, and run a macro. The possible behaviors are described by the members of the *PP_ACTION_TYPE* enumeration.

hover action All click action behaviors can also be triggered by a mouse-over (hover) event.

hyperlink A hyperlink is a particular class of click action that roughly corresponds to opening a web page, although it can also be used to send an email. While a similar mechanism is used to specify other actions, such as open a file, this term here is reserved for the action of navigating to a web URL (including *mailto://*).

action The click or hover action is specified in the XML using a URL on the *ppaction://* protocol contained in the *action* attribute of the *a:hlinkClick* (or *a:hlinkHover*) element. A hyperlink action is implied when no *action* attribute is present.

action verb The specific action to be performed is contained in the *host* field of the *ppaction://* URL. For instance, *customshow* appears in *ppaction://customshow?id=0&return=true* to indicate a *PP_ACTION.NAMED_SLIDE_SHOW* action.

OLE verb The term *verb* also appears in this context to indicate an OLE verb such as *Open* or *Edit*. This is not to be confused with an *action verb*.

Candidate Protocol

- Shape
  - *.click_action* - unconditionally returns an *ActionSetting* object, regardless of whether a click action is defined. Returns an *ActionSetting* object even when the shape type does not support a click action (such as a table).
- ActionSetting
  - *.action* - returns a member of the *PP_ACTION_TYPE* (*PP_ACTION*) enumeration
  - *.action_url* - returns the *ppaction://* URL as a string, in its entirety, or None if no action attribute is present. Maybe this should do XML character entity decoding.
  - *.action_verb* - returns the verb in the *ppaction://* URL, or None if no action URL is present. e.g. *'hlinksld-jump'*
  - *.action_fields* - returns a dictionary containing the fields in the query string of the action URL.
  - *.hyperlink* - returns a Hyperlink object that represents the hyperlink defined for the shape. A Hyperlink object is always returned.
  - *.target_slide* - returns a Slide object when the action is a jump to another slide in the same presentation. This is the case when action is *FIRST_SLIDE*, *LAST_SLIDE*, *PREVIOUS_SLIDE*, *NEXT_SLIDE*, or *NAMED_SLIDE*.
- Hyperlink
  - *.address* - returns the URL contained in the relationship for this hyperlink.
  - *.screen_tip* - tool-tip text displayed on mouse rollover is slideshow mode. Put in the XML hooks for this but API call is second priority

Detect that a shape has a hyperlink:

```
for shape in shapes:
    click_action = shape.click_action
    if click_action.action == PP_ACTION.HYPERLINK:
        print(click_action.hyperlink)
```

Add a hyperlink:

```
p = shape.text_frame.paragraphs[0]
r = p.add_run()
r.text = 'link to python-pptx @ GitHub'
hlink = r.hyperlink
hlink.address = 'https://github.com/scanny/python-pptx'
```

Delete a hyperlink:

```
r.hyperlink = None

# or -----------

r.hyperlink.address = None  # empty string '' will do it too
```

A Hyperlink instance is lazy-created on first reference. The object persists until garbage collected once created. The link XML is not written until .address is specified. Setting `hlink.address` to None or '' causes the hlink entry to be removed if present.

PP_ACTION_TYPE mapping logic

```
# _ClickAction.action property

hlinkClick = shape_elm.hlinkClick

if hlinkClick is None:
    return PP_ACTION.NONE

action_verb = hlinkClick.action_verb

if action_verb == 'hlinkshowjump':
    relative_target = hlinkClick.action_fields['jump']
    return {
        'firstslide':      PP_ACTION.FIRST_SLIDE,
        'lastslide':       PP_ACTION.LAST_SLIDE,
        'lastslideviewed': PP_ACTION.LAST_SLIDE_VIEWED,
        'nextslide':       PP_ACTION.NEXT_SLIDE,
        'previousslide':   PP_ACTION.PREVIOUS_SLIDE,
        'endshow':         PP_ACTION.END_SHOW,
    }.relative_target

return {
    None:           PP_ACTION.HYPERLINK,
    'hlinksldjump': PP_ACTION.NAMED_SLIDE,
    'hlinkpres':    PP_ACTION.PLAY,
    'hlinkfile':    PP_ACTION.OPEN_FILE,
    'customshow':   PP_ACTION.NAMED_SLIDE_SHOW,
    'ole':          PP_ACTION.OLE_VERB,
    'macro':        PP_ACTION.RUN_MACRO,
    'program':      PP_ACTION.RUN_PROGRAM,
}.action_verb
```

PowerPoint® application behavior

The general domain here is mouse event behaviors, with respect to a shape. So far, the only two mouse events are (left) click and hover (mouse over). These can trigger a variety of actions. I'm not sure if all actions can be triggered by either event, but the XML appears to support it.

Action inventory

The following behaviors can be triggered by a click:

- Jump to a relative slide in same presentation (first, last, next, previous, etc.).
- Jump to specific slide in same presentation (by slide index, perhaps title as fallback)
- Jump to a slide in different presentation (by slide index)
- End the slide show
- Jump to bookmark in Microsoft Word document
- Open an arbitrary file on the same computer
- Web link - Open a browser and navigate to a specified web page
- Run a macro
- Run an arbitrary program
- Execute an OLE action

In addition to performing one of these actions, zero, one, or both of two auxiliarly actions can be triggered by clicking:

- Play a sound
- Highlight the shape with a dashed line for a short time

Hyperlinkable shapes

These shape types can have hyperlinks:

- Autoshapes
- Textbox
- Picture
- Connector (Line)
- Chart

These shape types cannot:

- Table
- Group shape

UI procedures

Hyperlink autoshape to other slide by title

- Right-click > Hyperlink... (Cmd-K)
- Select Document panel
- Anchor: > Locate... > Slide Titles
- select slide by number and title, e.g. “2

Add Anchor point in a document (or perhaps a slide)

- A hyperlink can link to a bookmark in a Word document
- It appears that maximum granularity in PowerPoint is to an entire slide (not to a range of text in a shape, for example)

MS API

Shape.ActionSettings(ppMouseClick | ppMouseOver)

The Shape object has an ActionSettings property, which is a collection of two ActionSetting objects, one for click and the other for hover. https://msdn.microsoft.com/EN-US/library/office/ff745656.aspx

ActionSetting

- Shape.ActionSettings(ppMouseClick | ppMouseOver) => ActionSetting
- ActionSetting.Action
  - one of: ppActionHyperlink, ppActionFirstSlide, ppActionPlay, or several others: https://msdn.microsoft.com/EN-US/library/office/ff744511.aspx
- ActionSetting.Hyperlink => Hyperlink
- Hyperlink members:
  - Address
  - SubAddress
  - TextToDisplay
  - ScreenTip
  - EmailSubject
  - Type (read-only, one of msoHyperlinkRange (run) or msoHyperlinkShape)

XML specimens

These are representative samples of shape XML showing the hyperlinks associated the shape (as opposed to text contained by the shape).

- The *a:hlinkClick* element can be present or absent.
- Its parent, *p:cNvPr* is always present (is a required element).
- All of its attributes are optional, but an *a:hlinkClick* having no attributes has no meaning (or may trigger an error).
- Its *r:id* element is always present on click actions created by PowerPoint. Its value is an empty string when the action is first, last, next, previous, macro, and perhaps others.
- Adding a *highlightClick* attribute set True causes the shape to get a dashed line border for a short time when it is clicked.
- There are some more obscure attributes like "stop playing sound before navigating" that are available on *CT_Hyperlink*, perhaps meant for kiosk-style applications.

Summary

The action to perform on a mouse click is specified by the *action* attribute of the *a:hlinkClick* element. Its value is a URL having the *ppaction://* protocol, a verb, and an optional query string.

Some actions reference a relationship that specifies the target of the action.

verb	rId	behavior
none	external	Open a browser and navigate to URL in relationship
hlinkshowjump	none	Jump to a relative slide in the same presentation
hlinksldjump	internal	Jump to a specified slide in the same presentation
hlinkpres	external	Jump to a specified slide in another presentation
hlinkfile	external	Open an arbitrary file on the same computer
customshow	none	Start a custom slide show, option to return after
ole	none	Execute an OLE action (open, edit)
macro	none	Run an embedded VBA macro
program	external	Execute an arbitrary program on same computer

Jump to relative slide within presentation

hlinkshowjump action

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="7" name="Rounded Rectangle 6">
      <!-- this element does the needful -->
      <a:hlinkClick r:id="" action="ppaction://hlinkshowjump?jump=firstslide"/>
    </p:cNvPr>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="1020781" y="1684235"/>
      <a:ext cx="1495562" cy="1775031"/>
    </a:xfrm>
    <a:prstGeom prst="roundRect">
      <a:avLst/>
    </a:prstGeom>
  </p:spPr>
  <p:txBody>
    <a:p>
      <a:pPr algn="ctr"/>
      <a:r>
        <a:rPr lang="en-US" dirty="0" smtClean="0"/>
        <a:t>Click to go to Foobar Slide</a:t>
      </a:r>
      <a:endParaRPr lang="en-US" dirty="0" smtClean="0"/>
    </a:p>
  </p:txBody>
</p:sp>
```

- *jump* key can have value *firstslide*, *lastslide*, *previousslide*, *nextslide*, *lastslideviewed*, *endshow*.
- Note that *r:id* attribute is empty string; no relationship is required to determine target slide.

Jump to specific slide within presentation

hlinksldjump action

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="7" name="Rounded Rectangle 6">
      <a:hlinkClick r:id="rId2" action="ppaction://hlinksldjump"/>
    </p:cNvPr>
    ...
</p:sp>
```

The corresponding *Relationship* element must be of type *slide*, be internal, and point to the target slide in the package:

```
<Relationship
  Id="rId2"
  Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/slide"
  Target="slide1.xml"/>
```

Jump to slide in another presentation

hlinkpres action

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="7" name="Rounded Rectangle 6">
      <a:hlinkClick r:id="rId3" action="ppaction://hlinkpres?slideindex=3&amp;
↪slidetitle=Key Questions"/>
    </p:cNvPr>
    ...
</p:sp>
```

The corresponding *Relationship* element must be of type *hyperlink*, be *external*, and point to the target presentation with a URL (using the *file://* protocol for a local file). The slide number and slide title are provided in the *ppaction://* URL in the *a:hlinkClick* element:

```
<Relationship
  Id="rId3"
  Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink"
  Target="file://localhost/Users/scanny/Documents/checksec-prelim-analysis.pptx"
  TargetMode="External"/>
```

Web link (hyperlink)

Note: The *action* attribute of *a:hlinkClick* has no value in this case.

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="4" name="Rounded Rectangle 3">
      <a:hlinkClick r:id="rId3"/>
    ...
</p:sp>
```

The corresponding *Relationship* element must be of type *hyperlink*, be *external*, and point to the target URL (using a web protocol).

The target is often a web URL, such as https://github/scanny/python-pptx, including an optional anchor (e.g. #sub-heading suffix to jump mid-page). The target can also be an email address, launching the local email client. A mailto: URI is used in this case, with subject specifiable using a '?subject=xyz' suffix.

An optional ScreenTip, a roll-over tool-tip sort of message, can also be specified for a hyperlink. The XML schema does not limit its use to hyperlinks, although the PowerPoint UI may not provide access to this field in non-hyperlink cases.:

```
<Relationship
  Id="rId3"
  Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink"
  Target="https://www.google.com/"
  TargetMode="External"/>
```

Open an arbitrary file on the same computer

hlinkfile action

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="7" name="Rounded Rectangle 6">
      <a:hlinkClick r:id="rId2" action="ppaction://hlinkfile"/>
      ...
</p:sp>
```

- PowerPoint opens the file (after a warning dialog) using the default application for the file.

The corresponding *Relationship* element must be of type *hyperlink*, be *external*, and point to the target file with a *file://* protocol URL:

```
<Relationship
  Id="rId2"
  Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink"
  Target="file:///C:\Install.log"
  TargetMode="External"/>
```

Run Custom SlideShow

customshow action

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="4" name="Rounded Rectangle 3">
      <a:hlinkClick r:id="" action="ppaction://customshow?id=0&amp;return=true"/>
      ...
</p:sp>
```

- The *return* query field determines whether focus returns to the current show after running the linked show. This field can be omitted, and defaults to *false*.

Execute an OLE action

ole action

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="9" name="Object 8">
      <a:hlinkClick r:id="" action="ppaction://ole?verb=0"/>
    </p:cNvPr>
  ...
</p:sp>
```

This option is only available on an embedded (OLE) object. The verb field is '0' for Edit and '1' for Open.

Run macro

macro action

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="4" name="Rounded Rectangle 3">
      <a:hlinkClick r:id="" action="ppaction://macro?name=Hello"/>
    </p:cNvPr>
  ...
</p:sp>
```

Run a program

program action

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="4" name="Rounded Rectangle 3">
      <a:hlinkClick r:id="rId2" action="ppaction://program"/>
    ...
</p:sp>
```

The corresponding *Relationship* element must be of type *hyperlink*, be *external*, and point to the target application with a *file://* protocol URL.

```
<Relationship
  Id="rId2"
  Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink"
  Target="file:///C:\Program%20Files%20(x86)\Vim\vim74\gvim.exe"
  TargetMode="External"/>
```

Play a sound

Playing a sound is not a distinct action; rather, like highlighting, it is an optional additional action to be performed on a click or hover event.

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="5" name="Rounded Rectangle 4">
      <a:hlinkClick r:id="" action="ppaction://..any..">
        <a:snd r:embed="rId3" name="applause.wav"/>
      </a:hlinkClick>
    ...
</p:sp>
```

The corresponding *Relationship* element must be of type *audio*, be internal, and point to a sound file embedded in the presentation:

```
<Relationship
  Id="rId3"
  Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/audio"
  Target="../media/audio1.wav"/>
```

Related Schema Definitions

```
<xsd:complexType name="CT_Shape">
  <xsd:sequence>
    <xsd:element name="nvSpPr" type="CT_ShapeNonVisual"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties"/>
    <xsd:element name="style"  type="a:CT_ShapeStyle"        minOccurs="0"/>
    <xsd:element name="txBody" type="a:CT_TextBody"          minOccurs="0"/>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="useBgFill" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_ShapeNonVisual">
  <xsd:sequence>
    <xsd:element name="cNvPr"   type="a:CT_NonVisualDrawingProps"/>
    <xsd:element name="cNvSpPr" type="a:CT_NonVisualDrawingShapeProps"/>
    <xsd:element name="nvPr"    type="CT_ApplicationNonVisualDrawingProps"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="hlinkClick" type="CT_Hyperlink"             minOccurs="0"/>
    <xsd:element name="hlinkHover" type="CT_Hyperlink"             minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id"     type="ST_DrawingElementId" use="required"/>
  <xsd:attribute name="name"   type="xsd:string"          use="required"/>
  <xsd:attribute name="descr"  type="xsd:string"          default=""/>
  <xsd:attribute name="hidden" type="xsd:boolean"         default="false"/>
  <xsd:attribute name="title"  type="xsd:string"          default=""/>
</xsd:complexType>

<xsd:complexType name="CT_Hyperlink">
  <xsd:sequence>
    <xsd:element name="snd"    type="CT_EmbeddedWAVAudioFile"   minOccurs="0"/>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
```

```
  <xsd:attribute ref="r:id"/>
  <xsd:attribute name="invalidUrl"     type="xsd:string"  default=""/>
  <xsd:attribute name="action"         type="xsd:string"  default=""/>
  <xsd:attribute name="tgtFrame"       type="xsd:string"  default=""/>
  <xsd:attribute name="tooltip"        type="xsd:string"  default=""/>
  <xsd:attribute name="history"        type="xsd:boolean" default="true"/>
  <xsd:attribute name="highlightClick" type="xsd:boolean" default="false"/>
  <xsd:attribute name="endSnd"         type="xsd:boolean" default="false"/>
</xsd:complexType>
```

Graphic Frame

A *graphic frame* is a shape that contains a graphical object such as a table, chart, or SmartArt object. While often referred to as a shape in common parlance, a table or chart is not a shape. The graphic frame container is the shape and the table, chart, or SmartArt object is a DrawingML (DML) object within it.

A chart, for example, is a shared DML object. It can appear in a Word, Excel, or PowerPoint file, virtually unchanged except for the container in which it appears.

The graphical content is contained in the `p:graphicFrame/a:graphic/a:graphicData` grandchild element. The type of graphical object contained is specified by an XML namespace contained in the `uri` attribute of the `<a:graphicData>` element. The graphical content may appear directly in the `<p:graphicFrame>` element or may be in a separate part related by an rId. XML for a table is embedded inline. Chart XML is stored in a related Chart part.

Protocol

```
>>> shape = shapes.add_chart(style, type, x, y, cx, cy)
>>> type(shape)
<class 'pptx.shapes.graphfrm.GraphicFrame'>
>>> shape.has_chart
True
>>> shape.has_table
False
>>> shape.chart
<pptx.parts.chart.ChartPart object at 0x108c0e290>

>>> shape = shapes.add_table(rows=2, cols=2, x, y, cx, cy)
>>> type(shape)
<class 'pptx.shapes.graphfrm.GraphicFrame'>
>>> shape.has_chart
False
>>> shape.has_table
True
>>> shape.table
<pptx.shapes.table.Table object at 0x108c0e310>
```

Specimen XML

Table in a graphic frame:

```
<p:graphicFrame>
  <p:nvGraphicFramePr>
    <p:cNvPr id="2" name="Table 1"/>
    <p:cNvGraphicFramePr>
      <a:graphicFrameLocks noGrp="1"/>
    </p:cNvGraphicFramePr>
    <p:nvPr/>
  </p:nvGraphicFramePr>
  <p:xfrm>
    <a:off x="1524000" y="1397000"/>
    <a:ext cx="6096000" cy="741680"/>
  </p:xfrm>
  <a:graphic>
    <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/table">
      <a:tbl>
        ... remaining table XML ...
      </a:tbl>
    </a:graphicData>
  </a:graphic>
</p:graphicFrame>
```

Chart in a graphic frame:

```
<p:graphicFrame>
  <p:nvGraphicFramePr>
    <p:cNvPr id="2" name="Chart 1"/>
    <p:cNvGraphicFramePr/>
    <p:nvPr>
      <p:extLst>
        <p:ext uri="{D42A27DB-BD31-4B8C-83A1-F6EECF244321}">
          <p14:modId xmlns:p14="http://schemas.microsoft.com/office/powerpoint/2010/
↪main"
                     val="11227583"/>
        </p:ext>
      </p:extLst>
    </p:nvPr>
  </p:nvGraphicFramePr>
  <p:xfrm>
    <a:off x="1524000" y="1397000"/>
    <a:ext cx="6096000" cy="4064000"/>
  </p:xfrm>
  <a:graphic>
    <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/chart">
      <c:chart xmlns:c="http://schemas.openxmlformats.org/drawingml/2006/chart"
               xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/
↪relationships"
               r:id="rId2"/>
    </a:graphicData>
  </a:graphic>
</p:graphicFrame>
```

Related Schema Definitions

A `<p:graphicFrame>` element appears in a `CT_GroupShape` element, typically a `<p:spTree>` (shape tree) element:

```
<xsd:complexType name="CT_GroupShape">
  <xsd:sequence>
    <xsd:element name="nvGrpSpPr" type="CT_GroupShapeNonVisual"/>
    <xsd:element name="grpSpPr"   type="a:CT_GroupShapeProperties"/>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="sp"           type="CT_Shape"/>
      <xsd:element name="grpSp"        type="CT_GroupShape"/>
      <xsd:element name="graphicFrame" type="CT_GraphicalObjectFrame"/>
      <xsd:element name="cxnSp"        type="CT_Connector"/>
      <xsd:element name="pic"          type="CT_Picture"/>
      <xsd:element name="contentPart"  type="CT_Rel"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Graphic frame-related elements:

```
<xsd:complexType name="CT_GraphicalObjectFrame">
  <xsd:sequence>
    <xsd:element name="nvGraphicFramePr" type="CT_GraphicalObjectFrameNonVisual"/>
    <xsd:element name="xfrm"             type="a:CT_Transform2D"/>
    <xsd:element ref="a:graphic"/>  <!-- type="CT_GraphicalObject" -->
    <xsd:element name="extLst"           type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="a:ST_BlackWhiteMode"/>
</xsd:complexType>

<xsd:complexType name="CT_GraphicalObjectFrameNonVisual">
  <xsd:sequence>
    <xsd:element name="cNvPr"            type="a:CT_NonVisualDrawingProps"/>
    <xsd:element name="cNvGraphicFramePr" type="a:CT_NonVisualGraphicFrameProperties"/
↪>
    <xsd:element name="nvPr"             type="CT_ApplicationNonVisualDrawingProps"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_GraphicalObject">
  <xsd:sequence>
    <xsd:element name="graphicData" type="CT_GraphicalObjectData"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_GraphicalObjectData">
  <xsd:sequence>
    <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="strict"/>
  </xsd:sequence>
  <xsd:attribute name="uri" type="xsd:token" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="hlinkClick" type="CT_Hyperlink"             minOccurs="0"/>
    <xsd:element name="hlinkHover" type="CT_Hyperlink"             minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id"     type="ST_DrawingElementId" use="required"/>
  <xsd:attribute name="name"   type="xsd:string"          use="required"/>
```

```
  <xsd:attribute name="descr"  type="xsd:string"          default=""/>
  <xsd:attribute name="hidden" type="xsd:boolean"         default="false"/>
  <xsd:attribute name="title"  type="xsd:string"          default=""/>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualGraphicFrameProperties">
  <xsd:sequence>
    <xsd:element name="graphicFrameLocks" type="CT_GraphicalObjectFrameLocking"␣
↪minOccurs="0"/>
    <xsd:element name="extLst"            type="CT_OfficeArtExtensionList"     ␣
↪minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_GraphicalObjectFrameLocking">
  <xsd:sequence>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="noGrp"          type="xsd:boolean" default="false"/>
  <xsd:attribute name="noDrilldown"    type="xsd:boolean" default="false"/>
  <xsd:attribute name="noSelect"       type="xsd:boolean" default="false"/>
  <xsd:attribute name="noChangeAspect" type="xsd:boolean" default="false"/>
  <xsd:attribute name="noMove"         type="xsd:boolean" default="false"/>
  <xsd:attribute name="noResize"       type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_ApplicationNonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="ph"          type="CT_Placeholder"      minOccurs="0"/>
    <xsd:group   ref="a:EG_Media"                              minOccurs="0"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="isPhoto"   type="xsd:boolean" default="false"/>
  <xsd:attribute name="userDrawn" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:group name="EG_Media">
  <xsd:choice>
    <xsd:element name="audioCd"       type="CT_AudioCD"/>
    <xsd:element name="wavAudioFile"  type="CT_EmbeddedWAVAudioFile"/>
    <xsd:element name="audioFile"     type="CT_AudioFile"/>
    <xsd:element name="videoFile"     type="CT_VideoFile"/>
    <xsd:element name="quickTimeFile" type="CT_QuickTimeFile"/>
  </xsd:choice>
</xsd:group>
```

CT_Picture

Schema Name	CT_Picture
Spec Name	Picture
Tag(s)	p:pic
Namespace	presentationml (pml.xsd)
Schema Line	1245
Spec Section	19.3.1.37

Example

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="6" name="Picture 5" descr="python-logo.gif"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
    <a:xfrm>
      <a:off x="5580112" y="1988840"/>
      <a:ext cx="2679700" cy="901700"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
    <a:ln>
      <a:solidFill>
        <a:schemeClr val="bg1">
          <a:lumMod val="85000"/>
        </a:schemeClr>
      </a:solidFill>
    </a:ln>
  </p:spPr>
</p:pic>
```

Minimal `pic` shape

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="9" name="Picture 8" descr="python-logo.gif"/>
    <p:cNvPicPr/>
    <p:nvPr/>
  </p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId9"/>
  </p:blipFill>
  <p:spPr/>
</p:pic>
```

Analysis

...

attributes

None.

child elements

name	#	type	line
nvPicPr	1	CT_PictureNonVisual	1236
blipFill	1	a:CT_BlipFillProperties	1489 dml
spPr	1	a:CT_ShapeProperties	2210 dml
style	?	a:CT_ShapeStyle	2245 dml
extLst	?	CT_ExtensionListModify	

Spec text

This element specifies the existence of a picture object within the document.

Schema excerpt

```
<xsd:complexType name="CT_Picture">
  <xsd:sequence>
    <xsd:element name="nvPicPr"  type="CT_PictureNonVisual"     minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blipFill" type="a:CT_BlipFillProperties" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="spPr"     type="a:CT_ShapeProperties"    minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="style"    type="a:CT_ShapeStyle"         minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"   type="CT_ExtensionListModify"  minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_PictureNonVisual">
  <xsd:sequence>
    <xsd:element name="cNvPr"    type="a:CT_NonVisualDrawingProps"          minOccurs=
↪"1" maxOccurs="1"/>
    <xsd:element name="cNvPicPr" type="a:CT_NonVisualPictureProperties"     minOccurs=
↪"1" maxOccurs="1"/>
    <xsd:element name="nvPr"     type="CT_ApplicationNonVisualDrawingProps" minOccurs=
↪"1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="hlinkClick" type="CT_Hyperlink"              minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="hlinkHover" type="CT_Hyperlink"              minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"     type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪maxOccurs="1"/>
```

```
  </xsd:sequence>
  <xsd:attribute name="id"     type="ST_DrawingElementId" use="required"/>
  <xsd:attribute name="name"   type="xsd:string"          use="required"/>
  <xsd:attribute name="descr"  type="xsd:string"          use="optional" default=""/>
  <xsd:attribute name="hidden" type="xsd:boolean"         use="optional" default=
↪"false"/>
  <xsd:attribute name="title"  type="xsd:string"          use="optional" default=""/>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualPictureProperties">
  <xsd:sequence>
    <xsd:element name="picLocks" type="CT_PictureLocking"         minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"   type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="preferRelativeResize" type="xsd:boolean" use="optional"␣
↪default="true"/>
</xsd:complexType>

<xsd:complexType name="CT_ApplicationNonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element name="ph"          type="CT_Placeholder"      minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:group   ref="a:EG_Media"                              minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="custDataLst" type="CT_CustomerDataList" minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"      type="CT_ExtensionList"    minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="isPhoto"   type="xsd:boolean" use="optional" default="false"/>
  <xsd:attribute name="userDrawn" type="xsd:boolean" use="optional" default="false"/>
</xsd:complexType>

<xsd:group name="EG_Media">
  <xsd:choice>
    <xsd:element name="audioCd"       type="CT_AudioCD"/>
    <xsd:element name="wavAudioFile"  type="CT_EmbeddedWAVAudioFile"/>
    <xsd:element name="audioFile"     type="CT_AudioFile"/>
    <xsd:element name="videoFile"     type="CT_VideoFile"/>
    <xsd:element name="quickTimeFile" type="CT_QuickTimeFile"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_BlipFillProperties">
  <xsd:sequence>
    <xsd:element name="blip"    type="CT_Blip"         minOccurs="0" maxOccurs="1"/>
    <xsd:element name="srcRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
    <xsd:group   ref="EG_FillModeProperties"           minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="dpi"          type="xsd:unsignedInt" use="optional"/>
  <xsd:attribute name="rotWithShape" type="xsd:boolean"     use="optional"/>
</xsd:complexType>

<xsd:complexType name="CT_Blip">
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
```

```
      <xsd:element name="alphaBiLevel" type="CT_AlphaBiLevelEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaCeiling" type="CT_AlphaCeilingEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaFloor"   type="CT_AlphaFloorEffect"         minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaInv"     type="CT_AlphaInverseEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaMod"     type="CT_AlphaModulateEffect"      minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaModFix"  type="CT_AlphaModulateFixedEffect" minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="alphaRepl"    type="CT_AlphaReplaceEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="biLevel"      type="CT_BiLevelEffect"            minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="blur"         type="CT_BlurEffect"               minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="clrChange"    type="CT_ColorChangeEffect"        minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="clrRepl"      type="CT_ColorReplaceEffect"       minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="duotone"      type="CT_DuotoneEffect"            minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="fillOverlay"  type="CT_FillOverlayEffect"        minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="grayscl"      type="CT_GrayscaleEffect"          minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="hsl"          type="CT_HSLEffect"                minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="lum"          type="CT_LuminanceEffect"          minOccurs="1
↪" maxOccurs="1"/>
      <xsd:element name="tint"         type="CT_TintEffect"               minOccurs="1
↪" maxOccurs="1"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="AG_Blob"/>
  <xsd:attribute name="cstate" type="ST_BlipCompression" use="optional" default="none
↪"/>
</xsd:complexType>

<xsd:attributeGroup name="AG_Blob">
  <xsd:attribute ref="r:embed" use="optional" default=""/>
  <xsd:attribute ref="r:link"  use="optional" default=""/>
</xsd:attributeGroup>

<xsd:group name="EG_FillModeProperties">
  <xsd:choice>
    <xsd:element name="tile"    type="CT_TileInfoProperties"    minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="stretch" type="CT_StretchInfoProperties" minOccurs="1"␣
↪maxOccurs="1"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_StretchInfoProperties">
```

```
  <xsd:sequence>
    <xsd:element name="fillRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_RelativeRect">
  <xsd:attribute name="l" type="ST_Percentage" use="optional" default="0%"/>
  <xsd:attribute name="t" type="ST_Percentage" use="optional" default="0%"/>
  <xsd:attribute name="r" type="ST_Percentage" use="optional" default="0%"/>
  <xsd:attribute name="b" type="ST_Percentage" use="optional" default="0%"/>
</xsd:complexType>

<xsd:complexType name="CT_ShapeProperties">
  <xsd:sequence>
    <xsd:element name="xfrm"    type="CT_Transform2D"           minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:group ref="EG_Geometry"                                minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:group ref="EG_FillProperties"                          minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="ln"      type="CT_LineProperties"        minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:group ref="EG_EffectProperties"                        minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="scene3d" type="CT_Scene3D"               minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="sp3d"    type="CT_Shape3D"               minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</xsd:complexType>

<xsd:complexType name="CT_Transform2D">
  <xsd:sequence>
    <xsd:element name="off" type="CT_Point2D" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="ext" type="CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="rot" type="ST_Angle" use="optional" default="0"/>
  <xsd:attribute name="flipH" type="xsd:boolean" use="optional" default="false"/>
  <xsd:attribute name="flipV" type="xsd:boolean" use="optional" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_Point2D">
  <xsd:attribute name="x" type="ST_Coordinate" use="required"/>
  <xsd:attribute name="y" type="ST_Coordinate" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_PositiveSize2D">
  <xsd:attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
  <xsd:attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</xsd:complexType>

<xsd:group name="EG_Geometry">
  <xsd:choice>
    <xsd:element name="custGeom" type="CT_CustomGeometry2D" minOccurs="1" maxOccurs="1
↪"/>
```

```
    <xsd:element name="prstGeom" type="CT_PresetGeometry2D" minOccurs="1" maxOccurs="1
↪"/>
  </xsd:choice>
</xsd:group>

<xsd:group name="EG_FillProperties">
  <xsd:choice>
    <xsd:element name="noFill"    type="CT_NoFillProperties"         minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="solidFill" type="CT_SolidColorFillProperties" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="gradFill"  type="CT_GradientFillProperties"   minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blipFill"  type="CT_BlipFillProperties"       minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="pattFill"  type="CT_PatternFillProperties"    minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="grpFill"   type="CT_GroupFillProperties"      minOccurs="1"␣
↪maxOccurs="1"/>
  </xsd:choice>
</xsd:group>

<xsd:group name="EG_EffectProperties">
  <xsd:choice>
    <xsd:element name="effectLst" type="CT_EffectList"      minOccurs="1" maxOccurs="1
↪"/>
    <xsd:element name="effectDag" type="CT_EffectContainer" minOccurs="1" maxOccurs="1
↪"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_ShapeStyle">
  <xsd:sequence>
    <xsd:element name="lnRef"     type="CT_StyleMatrixReference" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="fillRef"   type="CT_StyleMatrixReference" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="effectRef" type="CT_StyleMatrixReference" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="fontRef"   type="CT_FontReference"        minOccurs="1"␣
↪maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

Line/Connector Shape

Lines are a sub-category of auto shape that differ in certain properties and behaviors. In particular, they have a start point and end point in addition to extents (left, top, width, height).

Connectors are based on the `<p:cxnSp>` element and have one of a handful of different preset geometry values, such as `line`. Freeform connectors, despite the name, are not connectors, and are a custom geometry shape based on the p:sp element.

Connectors can be "connected" to an auto shape such that a connected end point remains connected when the auto shape is moved. This relies on a concept of "connection points" on the auto shape. These connections points are preset features of the auto shape, similar to how adjustment points are pre-defined. Connection points are identified by index.

Connector shapes do not have a text frame and cannot have text that is anchored to the shape.

Design Issues

- See how PowerPoint interprets the remaining connector types like bentConnector2

Unimplemented Scope

- Connector.shape_type = MSO_LINE
- BaseShape.is_connector + ConnectorShape.is_connector (not yet implemented)
- [ ] see how a connector returns for *connector.AutoShapeType* in MS API.

PowerPoint behaviors

- Once connected, moving the target shape moves the connector the same amount, however, the relative position of the connector end-point is not adjusted to align to the connection point; rather, the relative offset is maintained.

  Also, when the connector is selected, the connected point is highlighted in red, even though it is not coincident with the connection point.
- Arrow heads are not a direct property of a connector, despite perhaps appearing so in the UI. Line endings are a property of the line style of the connector (and other shapes).

MS API

Shapes Object Members

- AddConnector(Type, BeginX, BeginY, EndX, EndY)

MsoConnectorType Enumeration

msoConnectorCurve	3	Curved connector.
msoConnectorElbow	2	Elbow connector.
msoConnectorStraight	1	Straight line connector.
msoConnectorTypeMixed	-2	Return only; indicates combination of states.

Protocols

Properties:

```
>>> connector.is_connector
True
>>> connector.type
MSO_SHAPE_TYPE.LINE
>>> connector.start_x
914400
>>> connector.end_y
914400
>>> connector.adjustments
<pptx.shapes.shared.Adjustments instance at 0x123456789>
```

Creation protocol:

```
>>> line = shapes.add_connector(
...     MSO_CONNECTOR.STRAIGHT, start_x, start_y, end_x, end_y
... )
```

MS API Protocol

Behaviors

UI display

Connector shapes do not display a bounding box when selected in the UI. Rather their start and end points are highlighted with a small gray circle that can be moved. If there is an adjustment, such as a mid-point, it is indicated in the normal way with a small yellow diamond. These endpoints and adjustment diamond cannot be individually selected.

Naming

What determines the name automatically applied to a line or connector shape? Can it change after shape creation, for example by applying an arrowhead?

Enumerations

MsoConnectorType

http://msdn.microsoft.com/en-us/library/office/ff860918(v=office.15).aspx

Name	Value	Description
msoConnectorCurve	3	Curved connector.
msoConnectorElbow	2	Elbow connector.
msoConnectorStraight	1	Straight line connector.
msoConnectorTypeMixed	-2	Return value only; indicates a combination of the other states.

Specimen XML

Default connector shapes inserted from PowerPoint UI.

Straight line (Connector):

```
<p:cxnSp>
  <p:nvCxnSpPr>
    <p:cNvPr id="3" name="Straight Connector 2"/>
    <p:cNvCxnSpPr/>
    <p:nvPr/>
  </p:nvCxnSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="611560" y="620688"/>
      <a:ext cx="914400" cy="914400"/>
    </a:xfrm>
    <a:prstGeom prst="line">
      <a:avLst/>
```

```
    </a:prstGeom>
  </p:spPr>
  <p:style>
    <a:lnRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="0">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="tx1"/>
    </a:fontRef>
  </p:style>
</p:cxnSp>
```

Straight arrow Connector:

```
<p:cxnSp>
  <p:nvCxnSpPr>
    <p:cNvPr id="7" name="Straight Arrow Connector 6"/>
    <p:cNvCxnSpPr/>
    <p:nvPr/>
  </p:nvCxnSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="950964" y="1673307"/>
      <a:ext cx="1257921" cy="0"/>
    </a:xfrm>
    <a:prstGeom prst="straightConnector1">
      <a:avLst/>
    </a:prstGeom>
    <a:ln>
      <a:tailEnd type="arrow"/>
    </a:ln>
  </p:spPr>
  <p:style>
    <a:lnRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="0">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="tx1"/>
    </a:fontRef>
  </p:style>
</p:cxnSp>
```

Straight segment jointed connector:

```
<p:cxnSp>
  <p:nvCxnSpPr>
```

```
    <p:cNvPr id="9" name="Elbow Connector 8"/>
    <p:cNvCxnSpPr/>
    <p:nvPr/>
  </p:nvCxnSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="950964" y="2124739"/>
      <a:ext cx="1257921" cy="415317"/>
    </a:xfrm>
    <a:prstGeom prst="bentConnector3">
      <a:avLst/>
    </a:prstGeom>
  </p:spPr>
  <p:style>
    <a:lnRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="0">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="tx1"/>
    </a:fontRef>
  </p:style>
</p:cxnSp>
```

Curved (S-like) connector:

```
<p:cxnSp>
  <p:nvCxnSpPr>
    <p:cNvPr id="11" name="Curved Connector 10"/>
    <p:cNvCxnSpPr/>
    <p:nvPr/>
  </p:nvCxnSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="950964" y="2925277"/>
      <a:ext cx="1257921" cy="619967"/>
    </a:xfrm>
    <a:prstGeom prst="curvedConnector3">
      <a:avLst/>
    </a:prstGeom>
  </p:spPr>
  <p:style>
    <a:lnRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="0">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="tx1"/>
```

```
    </a:fontRef>
  </p:style>
</p:cxnSp>
```

Freeform connector:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="12" name="Freeform 11"/>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="981058" y="4086962"/>
      <a:ext cx="1372277" cy="686176"/>
    </a:xfrm>
    <a:custGeom>
      <a:avLst/>
      <a:gdLst>
        <a:gd name="connsiteX0" fmla="*/ 0 w 1372277"/>
        <a:gd name="connsiteY0" fmla="*/ 0 h 686176"/>
        <a:gd name="connsiteX1" fmla="*/ 379182 w 1372277"/>
        <a:gd name="connsiteY1" fmla="*/ 306973 h 686176"/>
        <a:gd name="connsiteX2" fmla="*/ 944945 w 1372277"/>
        <a:gd name="connsiteY2" fmla="*/ 48152 h 686176"/>
      </a:gdLst>
      <a:ahLst/>
      <a:cxnLst>
        <a:cxn ang="0">
          <a:pos x="connsiteX0" y="connsiteY0"/>
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX1" y="connsiteY1"/>
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX2" y="connsiteY2"/>
        </a:cxn>
      </a:cxnLst>
      <a:rect l="l" t="t" r="r" b="b"/>
      <a:pathLst>
        <a:path w="1372277" h="686176">
          <a:moveTo>
            <a:pt x="0" y="0"/>
          </a:moveTo>
          <a:cubicBezTo>
            <a:pt x="110845" y="149474"/>
            <a:pt x="221691" y="298948"/>
            <a:pt x="379182" y="306973"/>
          </a:cubicBezTo>
          <a:cubicBezTo>
            <a:pt x="536673" y="314998"/>
            <a:pt x="811529" y="4012"/>
            <a:pt x="944945" y="48152"/>
          </a:cubicBezTo>
        </a:path>
      </a:pathLst>
    </a:custGeom>
```

```
  </p:spPr>
  <p:style>
    <a:lnRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="0">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="tx1"/>
    </a:fontRef>
  </p:style>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr"/>
    <a:lstStyle/>
    <a:p>
      <a:pPr algn="ctr"/>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </p:txBody>
</p:sp>
```

Completely free-form line:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="13" name="Freeform 12"/>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="1005133" y="5483390"/>
      <a:ext cx="1360239" cy="379203"/>
    </a:xfrm>
    <a:custGeom>
      <a:avLst/>
      <a:gdLst>
        <a:gd name="connsiteX0" fmla="*/ 0 w 1360239"/>
        <a:gd name="connsiteY0" fmla="*/ 0 h 379203"/>
        <a:gd name="connsiteX1" fmla="*/ 0 w 1360239"/>
        <a:gd name="connsiteY1" fmla="*/ 0 h 379203"/>
        <a:gd name="connsiteX2" fmla="*/ 96300 w 1360239"/>
        <a:gd name="connsiteY2" fmla="*/ 6020 h 379203"/>
        <a:gd name="connsiteX3" fmla="*/ 138431 w 1360239"/>
        <a:gd name="connsiteY3" fmla="*/ 18058 h 379203"/>
        <a:gd name="connsiteX4" fmla="*/ 222694 w 1360239"/>
        <a:gd name="connsiteY4" fmla="*/ 24077 h 379203"/>
        <a:gd name="connsiteX5" fmla="*/ 511594 w 1360239"/>
        <a:gd name="connsiteY5" fmla="*/ 24077 h 379203"/>
      </a:gdLst>
      <a:ahLst/>
      <a:cxnLst>
        <a:cxn ang="0">
          <a:pos x="connsiteX0" y="connsiteY0"/>
```

```
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX1" y="connsiteY1"/>
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX2" y="connsiteY2"/>
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX3" y="connsiteY3"/>
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX4" y="connsiteY4"/>
        </a:cxn>
        <a:cxn ang="0">
          <a:pos x="connsiteX5" y="connsiteY5"/>
        </a:cxn>
      </a:cxnLst>
      <a:rect l="l" t="t" r="r" b="b"/>
      <a:pathLst>
        <a:path w="1360239" h="379203">
          <a:moveTo>
            <a:pt x="0" y="0"/>
          </a:moveTo>
          <a:lnTo>
            <a:pt x="0" y="0"/>
          </a:lnTo>
          <a:cubicBezTo>
            <a:pt x="32100" y="2007"/>
            <a:pt x="64408" y="1860"/>
            <a:pt x="96300" y="6020"/>
          </a:cubicBezTo>
          <a:cubicBezTo>
            <a:pt x="110783" y="7909"/>
            <a:pt x="123972" y="15992"/>
            <a:pt x="138431" y="18058"/>
          </a:cubicBezTo>
          <a:cubicBezTo>
            <a:pt x="166307" y="22040"/>
            <a:pt x="194606" y="22071"/>
            <a:pt x="222694" y="24077"/>
          </a:cubicBezTo>
          <a:cubicBezTo>
            <a:pt x="333136" y="60893"/>
            <a:pt x="138800" y="-1634"/>
            <a:pt x="511594" y="24077"/>
          </a:cubicBezTo>
          <a:lnTo>
            <a:pt x="1360239" y="343089"/>
          </a:lnTo>
        </a:path>
      </a:pathLst>
    </a:custGeom>
  </p:spPr>
  <p:style>
    <a:lnRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="0">
```

```
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="tx1"/>
    </a:fontRef>
  </p:style>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr"/>
    <a:lstStyle/>
    <a:p>
      <a:pPr algn="ctr"/>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </p:txBody>
</p:sp>
```

Analysis

- The p:style element represents indirection of the connector visual attributes to the theme part.
- What's up with the p:style element? Does that have to be there? What happens if we just leave that out? Is the *accent1* default universal enough to pop in there without consideration?
- Seems like the most common lines are connectors
- Connectors are a distinct shape type. They are very similar to regular <p:sp>-based auto shapes, but lack a text frame.
- Hypothesis: There are really two types, connectors and free-form.
  - Connectors are based on the <p:cxnSp> element and have a preset geometry (<a:prstGeom> child of <p:spPr>).
  - Free-form lines are based on the <p:sp> element and have a custom geometry (<a:custGeom> child of <p:spPr>).
- Connectors don't have a fill. Free-form shapes do. Fill of free-form shapes extends between the line and a line connecting the end points, whether present or not. Since the lines can cross, this produces some possibly surprising fill behaviors; there is no clear concept of inside and outside for such a shape.

Related Schema Definitions

```
<xsd:complexType name="CT_GroupShape">
  <xsd:sequence>
    <xsd:element name="nvGrpSpPr" type="CT_GroupShapeNonVisual"/>
    <xsd:element name="grpSpPr"   type="a:CT_GroupShapeProperties"/>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="sp"           type="CT_Shape"/>
      <xsd:element name="grpSp"        type="CT_GroupShape"/>
      <xsd:element name="graphicFrame" type="CT_GraphicalObjectFrame"/>
      <xsd:element name="cxnSp"        type="CT_Connector"/>
      <xsd:element name="pic"          type="CT_Picture"/>
      <xsd:element name="contentPart"  type="CT_Rel"/>
```

```
    </xsd:choice>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Connector">
  <xsd:sequence>
    <xsd:element name="nvCxnSpPr" type="CT_ConnectorNonVisual"/>
    <xsd:element name="spPr"      type="a:CT_ShapeProperties"/>
    <xsd:element name="style"     type="a:CT_ShapeStyle"        minOccurs="0"/>
    <xsd:element name="extLst"    type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ConnectorNonVisual">
  <xsd:sequence>
    <xsd:element name="cNvPr"      type="a:CT_NonVisualDrawingProps"/>
    <xsd:element name="cNvCxnSpPr" type="a:CT_NonVisualConnectorProperties"/>
    <xsd:element name="nvPr"       type="CT_ApplicationNonVisualDrawingProps"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_Connection">
  <xsd:attribute name="id"  type="ST_DrawingElementId" use="required"/>
  <xsd:attribute name="idx" type="xsd:unsignedInt"     use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_NonVisualConnectorProperties">
  <xsd:sequence>
    <xsd:element name="cxnSpLocks" type="CT_ConnectorLocking"       minOccurs="0"/>
    <xsd:element name="stCxn"      type="CT_Connection"             minOccurs="0"/>
    <xsd:element name="endCxn"     type="CT_Connection"             minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ShapeProperties">
  <xsd:sequence>
    <xsd:element name="xfrm"                type="CT_Transform2D"                ␣
↪minOccurs="0"/>
    <xsd:group    ref="EG_Geometry"                                              ␣
↪minOccurs="0"/>
    <xsd:group    ref="EG_FillProperties"                                        ␣
↪minOccurs="0"/>
    <xsd:element name="ln"                  type="CT_LineProperties"             ␣
↪minOccurs="0"/>
    <xsd:group    ref="EG_EffectProperties"                                      ␣
↪minOccurs="0"/>
    <xsd:element name="scene3d"             type="CT_Scene3D"                    ␣
↪minOccurs="0"/>
    <xsd:element name="sp3d"                type="CT_Shape3D"                    ␣
↪minOccurs="0"/>
    <xsd:element name="extLst"              type="CT_OfficeArtExtensionList"␣
↪minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</xsd:complexType>
```

```
<xsd:complexType name="CT_ShapeStyle">
  <xsd:sequence>
    <xsd:element name="lnRef"     type="CT_StyleMatrixReference"/>
    <xsd:element name="fillRef"   type="CT_StyleMatrixReference"/>
    <xsd:element name="effectRef" type="CT_StyleMatrixReference"/>
    <xsd:element name="fontRef"   type="CT_FontReference"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_StyleMatrixReference">
  <xsd:choice minOccurs="0">
    <xsd:element name="scrgbClr"  type="CT_ScRgbColor"/>
    <xsd:element name="srgbClr"   type="CT_SRgbColor"/>
    <xsd:element name="hslClr"    type="CT_HslColor"/>
    <xsd:element name="sysClr"    type="CT_SystemColor"/>
    <xsd:element name="schemeClr" type="CT_SchemeColor"/>
    <xsd:element name="prstClr"   type="CT_PresetColor"/>
  </xsd:choice>
  <xsd:attribute name="idx" type="ST_StyleMatrixColumnIndex" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_FontReference">
  <xsd:choice minOccurs="0">
    <xsd:element name="scrgbClr"  type="CT_ScRgbColor"/>
    <xsd:element name="srgbClr"   type="CT_SRgbColor"/>
    <xsd:element name="hslClr"    type="CT_HslColor"/>
    <xsd:element name="sysClr"    type="CT_SystemColor"/>
    <xsd:element name="schemeClr" type="CT_SchemeColor"/>
    <xsd:element name="prstClr"   type="CT_PresetColor"/>
  </xsd:choice>
  <xsd:attribute name="idx" type="ST_FontCollectionIndex" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_DrawingElementId">
  <xsd:restriction base="xsd:unsignedInt"/>
</xsd:simpleType>

<xsd:simpleType name="ST_ShapeType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="line"/>
    <xsd:enumeration value="straightConnector1"/>
    <xsd:enumeration value="bentConnector2"/>
    <xsd:enumeration value="bentConnector3"/>
    <xsd:enumeration value="bentConnector4"/>
    <xsd:enumeration value="bentConnector5"/>
    <xsd:enumeration value="curvedConnector2"/>
    <xsd:enumeration value="curvedConnector3"/>
    <xsd:enumeration value="curvedConnector4"/>
    <xsd:enumeration value="curvedConnector5"/>
    ... other shape types removed ...
  </xsd:restriction>
</xsd:simpleType>
```

Autofit setting

Overview

PowerPoint provides the *autofit* property to specify how to handle text that is too big to fit within its shape. The three possible settings are:

- resize shape to fit text
- resize text to fit shape
- resize neither the shape nor the text, allowing the text to extend beyond the bounding box

Auto size is closely related to word wrap.

There are certain constraining settings for resizing the text, not sure if those are supported in PowerPoint or not.

Protocol

```
>>> text_frame = shape.text_frame
>>> text_frame.auto_size
None
>>> text_frame.auto_size = MSO_AUTO_SIZE.SHAPE_TO_FIT_TEXT
>>> text_frame.auto_size
1
>>> text_frame.auto_size = MSO_AUTO_SIZE.TEXT_TO_FIT_SHAPE
>>> text_frame.auto_size
2
>>> str(text_frame.auto_size)
'MSO_AUTO_SIZE.TEXT_TO_FIT_SHAPE'
>>> text_frame.auto_size = MSO_AUTO_SIZE.NONE
>>> text_frame.auto_size
0
>>> text_frame.auto_size = None
>>> text_frame.auto_size
None
```

Scenarios

Default new textbox. When adding a new texbox from the PowerPoint toolbar, the new textbox sizes itself to fit the entered text. This corresponds to the setting MSO_AUTO_SIZE.SHAPE_TO_FIT_TEXT and `word_wrap = None`.

`MSO_AUTO_SIZE.NONE`, `word_wrap = False`. Text is free form, with PowerPoint exhibiting no formatting behavior. Text appears just as entered and lines are broken only where hard breaks are inserted.

`MSO_AUTO_SIZE.NONE`, `word_wrap = True`. Text is wrapped into a column the width of the shape, but is not constrained vertically.

`MSO_AUTO_SIZE.SHAPE_TO_FIT_TEXT`, `word_wrap = False`. The width of the shape expands as new text is entered. Line breaks occur only where hard breaks are entered. The height of the shape grows to accommodate the number of lines of entered text. Width and height shrink as extents of the text are reduced by deleting text or reducing font size.

`MSO_AUTO_SIZE.SHAPE_TO_FIT_TEXT`, `word_wrap = True`. Text is wrapped into a column the width of the shape, with the height of the shape growing to accommodate the resulting number of lines.

`MSO_AUTO_SIZE.TEXT_TO_FIT_SHAPE`, `word_wrap = False`. Experiment ...

`MSO_AUTO_SIZE.TEXT_TO_FIT_SHAPE`, `word_wrap = True`. Experiment ...

XML specimens

Default new textbox:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="4" name="Foobar 3"/>
    <p:cNvSpPr txBox="1"/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="914400" y="914400"/>
      <a:ext cx="914400" cy="914400"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
    <a:noFill/>
  </p:spPr>
  <p:txBody>
    <a:bodyPr wrap="none">
      <a:spAutoFit/>
    </a:bodyPr>
    <a:lstStyle/>
    <a:p/>
  </p:txBody>
</p:sp>
```

Related Schema Definitions

```
<xsd:complexType name="CT_TextBody">
  <xsd:sequence>
    <xsd:element name="bodyPr"   type="CT_TextBodyProperties"/>
    <xsd:element name="lstStyle" type="CT_TextListStyle" minOccurs="0"/>
    <xsd:element name="p"        type="CT_TextParagraph" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextBodyProperties">
  <xsd:sequence>
    <xsd:element name="prstTxWarp" type="CT_PresetTextShape"        minOccurs="0"/>
    <xsd:group   ref="EG_TextAutofit"                               minOccurs="0"/>
    <xsd:element name="scene3d"    type="CT_Scene3D"                minOccurs="0"/>
    <xsd:group   ref="EG_Text3D"                                    minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>

<xsd:group name="EG_TextAutofit">
  <xsd:choice>
    <xsd:element name="noAutofit"   type="CT_TextNoAutofit"/>
    <xsd:element name="normAutofit" type="CT_TextNormalAutofit"/>
    <xsd:element name="spAutoFit"   type="CT_TextShapeAutofit"/>
  </xsd:choice>
</xsd:group>
```

```
<xsd:complexType name="CT_TextNormalAutofit">
  <xsd:attribute name="fontScale" type="ST_TextFontScalePercentOrPercentString"
                 use="optional" default="100%"/>
  <xsd:attribute name="lnSpcReduction" type="ST_TextSpacingPercentOrPercentString"
                 use="optional" default="0%"/>
</xsd:complexType>

<xsd:complexType name="CT_TextShapeAutofit"/>

<xsd:complexType name="CT_TextNoAutofit"/>

<xsd:group name="EG_Text3D">
  <xsd:choice>
    <xsd:element name="sp3d"   type="CT_Shape3D"/>
    <xsd:element name="flatTx" type="CT_FlatText"/>
  </xsd:choice>
</xsd:group>
```

Shape position and size

Shapes of the following types can appear in the shape tree of a slide (`<p:spTree>`) and each will require support for querying size and position.

- **sp** – Auto Shape *(completed)*
- **pic** – Picture *(completed)*
- **graphicFrame** – container for table and chart shapes
- **grpSp** – Group Shape
- **cxnSp** – Connector (line)
- **contentPart** – has no position, but should return None instead of raising an exception

Protocol

```
>>> assert isinstance(shape, pptx.shapes.autoshape.Shape)
>>> shape.left
914400
>>> shape.left = Inches(0.5)
>>> shape.left
457200
```

XML specimens

Here is a representative sample of shape XML showing the placement of the position and size elements (in the <a:xfrm> element).

Auto shape (rounded rectangle in this case):

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="3" name="Rounded Rectangle 2"/>
    <p:cNvSpPr/>
```

```
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="760096" y="562720"/>
      <a:ext cx="2520824" cy="914400"/>
    </a:xfrm>
    <a:prstGeom prst="roundRect">
      <a:avLst>
        <a:gd name="adj" fmla="val 30346"/>
      </a:avLst>
    </a:prstGeom>
  </p:spPr>
  <p:style>
    <a:lnRef idx="1">
      <a:schemeClr val="accent1"/>
    </a:lnRef>
    <a:fillRef idx="3">
      <a:schemeClr val="accent1"/>
    </a:fillRef>
    <a:effectRef idx="2">
      <a:schemeClr val="accent1"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="lt1"/>
    </a:fontRef>
  </p:style>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr"/>
    <a:lstStyle/>
    <a:p>
      <a:pPr algn="ctr"/>
      <a:r>
        <a:rPr lang="en-US" dirty="0" smtClean="0"/>
        <a:t>This is text inside a rounded rectangle</a:t>
      </a:r>
      <a:endParaRPr lang="en-US" dirty="0"/>
    </a:p>
  </p:txBody>
</p:sp>
```

Example picture shape:

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="6" name="Picture 5" descr="python-logo.gif"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
```

```
    <a:xfrm>
      <a:off x="5580112" y="1988840"/>
      <a:ext cx="2679700" cy="901700"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
    <a:ln>
      <a:solidFill>
        <a:schemeClr val="bg1">
          <a:lumMod val="85000"/>
        </a:schemeClr>
      </a:solidFill>
    </a:ln>
  </p:spPr>
</p:pic>
```

Related Schema Definitions

```
<xsd:complexType name="CT_Shape">
  <xsd:sequence>
    <xsd:element name="nvSpPr" type="CT_ShapeNonVisual"      minOccurs="1" maxOccurs=
↪"1"/>
    <xsd:element name="spPr"   type="a:CT_ShapeProperties"   minOccurs="1" maxOccurs=
↪"1"/>
    <xsd:element name="style"  type="a:CT_ShapeStyle"        minOccurs="0" maxOccurs=
↪"1"/>
    <xsd:element name="txBody" type="a:CT_TextBody"          minOccurs="0" maxOccurs=
↪"1"/>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs=
↪"1"/>
  </xsd:sequence>
  <xsd:attribute name="useBgFill" type="xsd:boolean" use="optional" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_Picture">
  <xsd:sequence>
    <xsd:element name="nvPicPr"  type="CT_PictureNonVisual"     minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blipFill" type="a:CT_BlipFillProperties" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="spPr"     type="a:CT_ShapeProperties"    minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="style"    type="a:CT_ShapeStyle"         minOccurs="0"␣
↪maxOccurs="1"/>
    <xsd:element name="extLst"   type="CT_ExtensionListModify"  minOccurs="0"␣
↪maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_ShapeProperties">
  <xsd:sequence>
    <xsd:element name="xfrm" type="CT_Transform2D" minOccurs="0" maxOccurs="1"/>
    ...
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
```

```
</xsd:complexType>

<xsd:complexType name="CT_Transform2D">
  <xsd:sequence>
    <xsd:element name="off" type="CT_Point2D" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="ext" type="CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="rot" type="ST_Angle" use="optional" default="0"/>
  <xsd:attribute name="flipH" type="xsd:boolean" use="optional" default="false"/>
  <xsd:attribute name="flipV" type="xsd:boolean" use="optional" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_Point2D">
  <xsd:attribute name="x" type="ST_Coordinate" use="required"/>
  <xsd:attribute name="y" type="ST_Coordinate" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_PositiveSize2D">
  <xsd:attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
  <xsd:attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_PositiveCoordinate">
  <xsd:restriction base="xsd:long">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="27273042316900"/>
  </xsd:restriction>
</xsd:simpleType>
```

Placeholders

Slide Placeholders

Picture placeholder

A picture placeholder is an unpopulated placeholder into which an image can be inserted. This comprises three concrete placeholder types, a picture placeholder, a clip art placeholder, and a content placeholder. Each of these, and indeed any unpopulated placeholder, take the form of a *p:sp* element in the XML. Once populated with an image, these placeholders become a *placeholder picture* and take the form of a *p:pic* element.

Candidate protocol

Placeholder access:

```
>>> picture_placeholder = slide.placeholders[1]  # keyed by idx, not offset
>>> picture_placeholder
<pptx.shapes.placeholder.PicturePlaceholder object at 0x100830510>
>>> picture_placeholder.shape_type
MSO_SHAPE_TYPE.PLACEHOLDER (14)
```

PicturePlaceholder.insert_picture():

```
>>> placeholder_picture = picture_placeholder.insert_picture('image.png')
>>> placeholder_picture
<pptx.shapes.placeholder.PlaceholderPicture object at 0x10083087a>
>>> placeholder_picture.shape_type
MSO_SHAPE_TYPE.PLACEHOLDER (14)
```

Example XML

A picture-only layout placholder:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="3" name="Picture Placeholder 2"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph type="pic" idx="1"/>
    </p:nvPr>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="1792288" y="612775"/>
      <a:ext cx="5486400" cy="4114800"/>
    </a:xfrm>
  </p:spPr>
  <p:txBody>
    <a:bodyPr/>
    <a:lstStyle>
      <!-- a:lvl{n}pPr for n in 1:9 -->
    </a:lstStyle>
    <a:p>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </p:txBody>
</p:sp>
```

An unpopulated picture-only placeholder on a slide:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="7" name="Picture Placeholder 6"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph type="pic" idx="1"/>
    </p:nvPr>
  </p:nvSpPr>
  <p:spPr/>
</p:sp>
```

A picture-only placeholder populated with an image:

```
<p:pic>
  <p:nvPicPr>
```

```
    <p:cNvPr id="8" name="Picture Placeholder 7"
             descr="aphrodite.brinsmead.jpg"/>
    <p:cNvPicPr>
      <a:picLocks noGrp="1" noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr>
      <p:ph type="pic" idx="1"/>
    </p:nvPr>
  </p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId2">
      <a:extLst>
        <a:ext uri="{28A0092B-C50C-407E-A947-70E740481C1C}">
          <a14:useLocalDpi xmlns:a14="http://../drawing/2010/main" val="0"/>
        </a:ext>
      </a:extLst>
    </a:blip>
    <a:srcRect t="20000" b="20000"/>  <!-- 20% crop, top and bottom -->
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr/>
</p:pic>
```

Table placeholder

A table placeholder is an unpopulated placeholder into which a table can be inserted. This comprises two concrete placeholder types, a table placeholder and a content placeholder. Both take the form of a *p:sp* element in the XML. Once populated with a table, these placeholders become a *placeholder graphic frame* and take the form of a *p:graphicFrame* element.

Candidate protocol

Placeholder access:

```
>>> table_placeholder = slide.placeholders[10]  # keyed by idx, not offset
>>> table_placeholder
<pptx.shapes.placeholder.TablePlaceholder object at 0x100830510>
>>> table_placeholder.shape_type
MSO_SHAPE_TYPE.PLACEHOLDER (14)
```

TablePlaceholder.insert_table():

```
>>> ph_graphic_frame = table_placeholder.insert_table(rows=2, cols=2)
>>> ph_graphic_frame
<pptx.shapes.placeholder.PlaceholderGraphicFrame object at 0x10083087a>
>>> ph_graphic_frame.shape_type
MSO_SHAPE_TYPE.PLACEHOLDER (14)
>>> ph_graphic_frame.has_table
True
>>> table = ph_graphic_frame.table
>>> len(table.rows), len(table.columns)
(2, 2)
```

Example XML

A table-only layout placholder:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="3" name="Table Placeholder 2"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph type="tbl" sz="quarter" idx="10"/>
    </p:nvPr>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="2743200" y="2057400"/>
      <a:ext cx="3657600" cy="2743200"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
  </p:spPr>
  <p:txBody>
    <a:bodyPr vert="horz"/>
    <a:lstStyle/>
    <a:p>
      <a:endParaRPr lang="en-US"/>
    </a:p>
  </p:txBody>
</p:sp>
```

An unpopulated table-only placeholder on a slide:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="2" name="Table Placeholder 1"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph type="tbl" sz="quarter" idx="10"/>
    </p:nvPr>
  </p:nvSpPr>
  <p:spPr/>
</p:sp>
```

A table-only placeholder populated with a table:

```
<p:graphicFrame>
  <p:nvGraphicFramePr>
    <p:cNvPr id="3" name="Table Placeholder 2"/>
    <p:cNvGraphicFramePr>
      <a:graphicFrameLocks noGrp="1"/>
    </p:cNvGraphicFramePr>
    <p:nvPr>
      <p:ph type="tbl" sz="quarter" idx="10"/>
      <p:extLst>
```

```
        <p:ext uri="{D42A27DB-BD31-4B8C-83A1-F6EECF244321}">
          <p14:modId xmlns:p14="http://../..nt/2010/main" val="933747684"/>
        </p:ext>
      </p:extLst>
    </p:nvPr>
  </p:nvGraphicFramePr>
  <p:xfrm>
    <a:off x="2743200" y="2057400"/>
    <a:ext cx="3657600" cy="3337560"/>
  </p:xfrm>
  <a:graphic>
    <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/table">
      <a:tbl>
        <a:tblPr firstRow="1" bandRow="1">
          <a:tableStyleId>{5C22544A-7EE6-4342-B048-85BDC9FD1C3A}</a:tableStyleId>
        </a:tblPr>
        <a:tblGrid>
          <a:gridCol w="457200"/>
        </a:tblGrid>
        <a:tr h="370840">
          <a:tc>
            <a:txBody>
              <a:bodyPr/>
              <a:lstStyle/>
              <a:p>
                <a:endParaRPr lang="en-US"/>
              </a:p>
            </a:txBody>
            <a:tcPr/>
          </a:tc>
        </a:tr>
      </a:tbl>
    </a:graphicData>
  </a:graphic>
</p:graphicFrame>
```

Placeholders in new slide

Topic of inquiry

What placeholder shapes are added to a new slide when it is added to a presentation?

Abstract

Slides were created based on a variety of slide layouts so their placeholder elements could be inspected to reveal slide construction behaviors that depend on the slide layout chosen. A placeholder shape appeared in each slide corresponding to each of the placeholder shapes in the slide layout, excepting those of footer, date, and slide number type. Placeholder shapes were `<p:sp>` elements for all placeholder types rather than matching the element type of the shape they hold a place for. Each slide-side placeholder had a `<p:ph>` element that exactly matched the corresponding element in the slide layout, with the exception that a *hasCustomPrompt* attribute, if present in the slide layout, did not appear in the slide. Placeholders in the slide were named and numbered differently. All placeholders in the slide layout had a `<p:txBody>` element, but only certain types had one in the slide. All placeholders were locked against grouping.

Procedure

This procedure was run using PowerPoint for Mac 2011 running on an Intel-based Mac Pro running Mountain Lion 10.8.2.

1. Create a new presentation based on the "White" theme.
2. Add a new slide layout and add placeholders of the following types. The new slide layout is (automatically) named *Custom Layout*.
   - Chart
   - Table
   - SmartArt Graphic
   - Media
   - Clip Art
3. Add a new slide for each of the following layouts. Do not enter any content into the placeholders.
   - Title Slide (added automatically on File > New)
   - Title and Content
   - Two Content
   - Comparison
   - Content with Caption
   - Picture with Caption
   - Vertical Title and Text
   - Custom Layout
4. Save and unpack the presentation. This command works on OS X: `unzip white.pptx -d white`
5. Inspect the slide.xml files and compare placeholders with those present in their corresponding slideLayout.xml file.

Observations

Typical XML for a placeholder:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="2" name="Title 1"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph type="ctrTitle"/>
    </p:nvPr>
  </p:nvSpPr>
  <p:spPr/>
  <p:txBody>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:endParaRPr lang="en-US"/>
```

```
    </a:p>
  </p:txBody>
</p:sp>
```

elements

- Placeholder shapes **WERE** created corresponding to layout placeholders of the following types in the slide layout:
  - `'title'`
  - `'body'`
  - `'ctrTitle'`
  - `'subTitle'`
  - `'obj'`
  - `'chart'`
  - `'tbl'`
  - `'clipArt'`
  - `'dgm'` (SmartArt Graphic, perhaps abbreviation of 'diagram')
  - `'media'`
  - `'pic'`
  - `None` (type attribute default is `'obj'`)
- Placeholder shapes **WERE NOT** created corresponding to layout placeholders of these types in the slide layout:
  - `'dt'` (date)
  - `'sldNum'` (slide number)
  - `'ftr'` (footer)
- The following placeholder types do not apply to slides (they apply to notes pages and/or handouts).
  - `'hdr'`
  - `'sldImg'`
- **<p:txBody> element**. Placeholder shapes of the following types had a `<p:txBody>` element. Placeholders of all other types did not have a `<p:txBody>` element.
  - `'title'`
  - `'body'`
  - `'ctrTitle'`
  - `'subTitle'`
  - `'obj'` (and those without a `type` attribute)
- **<p:cNvPr> element**. Each placeholder shape in each slide had a `<p:cNvPr>` element (it is a required element) populated with *id* and *name* attributes.
  - The value of the *id* attribute of `p:cNvPr` does not correspond to its value in the slide layout. Shape ids were renumbered starting from 2 and proceeding in sequence (e.g. 2, 3, 4, 5, ...). Note that id="1" belongs to the spTree element, at `/p:sld/p:cSld/p:spTree/p:nvGrpSpPr/p:cNvPr@id`.

  - The value of the *name* attribute is of the form '%s %d' % (type_name, num), where type_name is determined by the placeholder type (e.g. 'Media Placeholder') and num appears to default to `id - 1`. The assigned name in each case was similar to its counterpart in the slide layout (same *type_name*), but its number was generally different.

- **<p:cNvSpPr>** element. Placeholder shapes of all types had the element `<a:spLocks noGrp="1"/>` specifying the placeholder could not be grouped with any other shapes.
- **<p:spPr>** element. All placeholder shapes had an empty `<p:spPr>` element.

attributes

- **type**. The `type` attribute of the `<p:ph>` element was copied without change. Where no `type` attribute appeared (defaults to "obj", indicating a "content" placeholder), no `type` attribute was added to the `<p:ph>` element.
- **orient**. The `orient` attribute of the `<p:ph>` element was copied without change. Where no `orient` attribute appeared (defaults to "horz"), no `orient` attribute was added to the `<p:ph>` element.
- **sz**. The `sz` (size) attribute of the `<p:ph>` element was copied without change. Where no `sz` attribute appeared (defaults to "full"), no `sz` attribute was added to the `<p:ph>` element.
- **idx**. The `idx` attribute of the `<p:ph>` element was copied without change. Where no `idx` attribute appeared (defaults to "0", indicating the singleton title placeholder), no `idx` attribute was added to the `<p:ph>` element.
- **hasCustomPrompt**. The `hasCustomPrompt` attribute of the `<p:ph>` element was NOT copied from the slide layout to the slide.

Other observations

Placeholders created from the slide layout were always `<p:sp>` elements, even when they were a placeholder for a shape of another type, e.g. `<p:graphicFrame>`. When the on-screen placeholder for a table was clicked to add a table, the `<p:sp` element was replaced by a `<p:graphicFrame>` element containing the table. The `<p:ph>` element was retained in the new shape. Reapplying the layout to the slide caused the table to move to the original location of the placeholder (the idx may have been used for matching). Deleting the table caused the original placeholder to reappear in its original position.

Conclusions

- Placeholder shapes are added to new slides based on the placeholders present in the slide layout specified on slide creation.
- Placeholders for footer, slide number, and date are excluded.
- Each new placeholder is assigned an id starting from 2 and proceeding in increments of 1 (e.g. 2, 3, 4, ...).
- Each new placeholder is named based on its type and the id assigned.
- Placeholders are locked against grouping.
- Placeholders of selected types get a minimal `<p:txBody>` element.
- Slide placeholders receive an exact copy of the `<p:ph>` element in the corresponding slide layout placeholder, except that the optional `hasCustomPrompt` attribute, if present, is not copied.

Open Questions

None.

Candidate protocol

```
>>> slide = prs.slides[0]
>>> slide.shapes
<pptx.shapes.shapetree.SlideShapes object at 0x104e60000>
>>> slide.shapes[0]
<pptx.shapes.placeholder.ShapePlaceholder object at 0x104e60020>
>>> slide_placeholders = slide.placeholders
>>> slide_placeholders
<pptx.shapes.shapetree.SlidePlaceholders object at 0x104e60040>
>>> len(slide_placeholders)
2
>>> slide_placeholders[1]
<pptx.shapes.placeholder.ContentPlaceholder object at 0x104e60060>
>>> slide_placeholders[1].type
'body'
>>> slide_placeholders.get(idx=1)
AttributeError ...
>>> slide_placeholders[1]._sp.spPr.xfrm
None
>>> slide_placeholders[1].width  # should inherit from layout placeholder
8229600
>>> table = slide_placeholders[1].insert_table(rows=2, cols=2)
>>> len(slide_placeholders)
2
```

Inheritance behaviors

A placeholder shape on a slide is initially little more than a reference to its "parent" placeholder shape on the slide layout. If it is a placeholder shape that can accept text, it contains a *<p:txBody>* element. Position, size, and even geometry are inherited from the layout placeholder, which may in turn inherit one or more of those properties from a master placeholder.

Substitution behaviors

Content may be placed into a placeholder shape two ways, by *insertion* and by *substitution*. Insertion is simply placing the text insertion point in the placeholder and typing or pasting in text. Substitution occurs when an object such as a table or picture is inserted into a placeholder by clicking on a placeholder button.

An empty placeholder is always a *<p:sp>* (autoshape) element. When an object such as a table is inserted into the placehoder by clicking on a placeholder button, the *<p:sp>* element is replaced with the appropriate new shape element, a table element in this case. The *<p:ph>* element is retained in the new shape element and preserves the linkage to the layout placeholder such that the 'empty' placeholder shape can be restored if the inserted object is deleted.

Operations

- clone on slide create

- query inherited property values
- substitution

Example XML

Baseline textbox shape:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="2" name="TextBox 1"/>
    <p:cNvSpPr txBox="1"/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="3016188" y="3273093"/>
      <a:ext cx="1133644" cy="369332"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
    <a:noFill/>
  </p:spPr>
  <p:txBody>
    <a:bodyPr wrap="none" rtlCol="0">
      <a:spAutoFit/>
    </a:bodyPr>
    <a:lstStyle/>
    <a:p>
      <a:r>
        <a:rPr lang="en-US" dirty="0" smtClean="0"/>
        <a:t>Some text</a:t>
      </a:r>
      <a:endParaRPr lang="en-US" dirty="0"/>
    </a:p>
  </p:txBody>
</p:sp>
```

Content placeholder:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="5" name="Content Placeholder 4"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph idx="1"/>
    </p:nvPr>
  </p:nvSpPr>
  <p:spPr/>
  <p:txBody>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:endParaRPr lang="en-US"/>
    </a:p>
```

```
  </p:txBody>
</p:sp>
```

Notable differences:

- placeholder has *<a:spLocks>* element
- placeholder has *<p:ph>* element
- placeholder has no *<p:spPr>* child elements, implying both that:
  - all shape properties are initially inherited from the layout placeholder, including position, size, and geometry
  - any specific shape property value may be overridden by specifying it on the inheriting shape

Matching slide layout placeholder:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="3" name="Content Placeholder 2"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph idx="1"/>
    </p:nvPr>
  </p:nvSpPr>
  <p:spPr/>
  <p:txBody>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr lvl="0"/>
      <a:r>
        <a:rPr lang="en-US" smtClean="0"/>
        <a:t>Click to edit Master text styles</a:t>
      </a:r>
    </a:p>
    <a:p>
      ... and others through lvl="4", five total
    </a:p>
  </p:txBody>
</p:sp>
```

Behaviors

- A placeholder is accessed through a slide-type object's shape collection, e.g. `slide.shapes.placeholders`. The contents of the placeholders collection is a subset of the shapes in the shape collection.
- The title placeholder, if it exists, always appears first in the placeholders collection.
- Placeholders can only be top-level shapes, they cannot be nested in a group shape.
- A slide placeholder may be either an *<p:sp>* (autoshape) element or a *<p:pic>* or *<p:graphicFrame>* element. In either case, its relationship to its layout placeholder is preserved.
- Slide inherits from layout strictly on idx value.

- The placeholder with idx="0" is the title placeholder. "0" is the default for the *idx* attribute on <p:ph>, so one with no idx attribute is the title placeholder.
- The document order of placeholders signifies their z-order and has no bearing on their index order. `_ShapeCollection.placeholders` contains the placeholders in order of *idx* value, which means the title placeholder always appears first in the sequence, if it is present.

Automatic naming

- Most placeholders are automatically named '{root_name} Placeholder {num}' where *root_name* is something like `Chart` and num is a positive integer. A typical example is `Table Placeholder 3`.
- A placeholder with vertical orientation, i.e. `<p:ph orient="vert">`, is prefixed with `'Vertical '`, e.g. `Vertical Text Placeholder 2`.
- The word `'Placeholder'` is omitted when the type is 'title', 'ctrTitle', or 'subTitle'.

On slide creation

When a new slide is added to a presentation, empty placeholders are added to it based on the placeholders present in the specified slide layout.

Only content placeholders are created automatically. Special placeholders must be created individually with additional calls. Special placeholders include header, footer, page numbers, date, and slide number; and perhaps others. In this respect, it seems like special placeholders on a slide layout are simply default settings for those placeholders in case they are added later.

During slide life-cycle

- Placeholders can be deleted from slides and can be restored by calling methods like AddTitle on the shape collection. AddTitle raises if there's already a title placeholder, so need to call .hasTitle() beforehand.

Experimental findings – Applying layouts

- Switching the layout of an empty title slide to the blank layout resulted in the placeholder shapes (title, subtitle) being removed.
- The same switch when the shapes had content (text), resulted in the shapes being preserved, complete with their *<p:ph>* element. Position and dimension values were added that preserve the height, width, top position but set the left position to zero.
- Restoring the original layout caused those position and dimension values to be removed (and "re-inherited").
- Applying a new (or the same) style to a slide appears to reset selected properties such that they are re-inherited from the new layout. Size and position are both reset. Background color and font, at least, are not reset.
- The "Reset Layout to Default Settings" option appears to reset all shape properties to inherited, without exception.
- Content of a placeholder shape is retained and displayed, even when the slide layout is changed to one without a matching layout placeholder.
- The behavior when placeholders are added to a slide layout (from the slide master) may also be worth characterizing.

- ... show master placeholder ...
- ... add (arbitrary) placeholder ...

Layout Placeholders

Candidate protocol

```
>>> slide_layout = prs.slide_layouts[0]
>>> slide_layout
<pptx.parts.slidelayout.SlideLayout object at 0x10a5d42d0>
>>> slide_layout.shapes
<pptx.shapes.shapetree.LayoutShapes object at 0x104e60000>
>>> slide_layout.shapes[0]
<pptx.shapes.placeholder.LayoutPlaceholder object at 0x104e60020>
>>> layout_placeholders = slide_layout.placeholders
>>> layout_placeholders
<pptx.shapes.shapetree.LayoutPlaceholders object at 0x104e60040>
>>> len(layout_placeholders)
5
>>> layout_placeholders[1].type
'body'
>>> layout_placeholders[2].idx
10
>>> layout_placeholders.get(idx=0)
<pptx.shapes.placeholder.MasterPlaceholder object at 0x104e60020>
>>> layout_placeholders.get(idx=666)
None
>>> layout_placeholders[1]._sp.spPr.xfrm
None
>>> layout_placeholders[1].width  # should inherit from master placehldr
8229600
```

Example XML

Slide layout placeholder:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="3" name="Content Placeholder 2"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph idx="1"/>
    </p:nvPr>
  </p:nvSpPr>
  <p:spPr/>
  <p:txBody>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr lvl="0"/>
      <a:r>
        <a:rPr lang="en-US" smtClean="0"/>
```

```
        <a:t>Click to edit Master text styles</a:t>
      </a:r>
    </a:p>
    <a:p>
      ... and others through lvl="4", five total
    </a:p>
  </p:txBody>
</p:sp>
```

Behaviors

- Hypothesis:
  - Layout inherits properties from master based only on type (e.g. body, dt), with no bearing on idx value.
  - Slide inherits from layout strictly on idx value.
  - Need to experiment to see if position and size are inherited on body placeholder by type or if idx has any effect
    - expanded/repackaged, change body ph idx on layout and see if still inherits position.
    - result: slide layout still appears with the proper position and size, however the slide placeholder appears at (0, 0) and apparently with size of (0, 0).
  - What behavior is exhibited when a slide layout contains two body placeholders, neither of which has a matching idx value and neither of which has a specified size or position?
    - result: both placeholders are present, directly superimposed upon each other. They both inherit position and fill.

Layout placeholder inheritance

Objective: determine layout placeholder inheritee for each ph type

Observations:

Layout placeholder with *lyt-ph-type* inherits color from master placeholder with *mst-ph-type*, noting idx value match.

lyt-ph-type	mst-ph-type	notes
ctrTitle	title	title layout - idx value matches (None, => 0)
subTitle	body	title layout - idx value matches (1)
dt	dt	title layout - idx 10 != 2
ftr	ftr	title layout - idx 11 != 3
sldNum	sldNum	title layout - idx 12 != 4
title	title	bullet layout - idx value matches (None, => 0)
None (obj)	body	bullet layout - idx value matches (1)
body	body	sect hdr - idx value matches (1)
None (obj)	body	two content - idx 2 != 1
body	body	comparison - idx 3 != 1
pic	body	picture - idx value matches (1)
chart	body	manual - idx 9 != 1
clipArt	body	manual - idx 9 != 1
dgm	body	manual - idx 9 != 1
media	body	manual - idx 9 != 1
tbl	body	manual - idx 9 != 1
hdr	repair err	valid only in Notes and Handout Slide
sldImg	repair err	valid only in Notes and Handout Slide

Master Placeholders

Candidate protocol

```
>>> slide_master = prs.slide_master
>>> slide_master.shapes
<pptx.shapes.slidemaster.MasterShapes object at 0x10a4df150>

>>> slide_master.shapes[0]
<pptx.shapes.placeholder.MasterPlaceholder object at 0x104e60290>
>>> master_placeholders = slide_master.placeholders
>>> master_placeholders
<pptx.shapes.shapetree.MasterPlaceholders object at 0x104371290>
>>> len(master_placeholders)
5
>>> master_placeholders[0].type
'title'
>>> master_placeholders[0].idx
0
>>> master_placeholders.get(type='body')
<pptx.shapes.placeholder.MasterPlaceholder object at 0x104e60290>
>>> master_placeholders.get(idx=666)
None
```

Example XML

Slide master placeholder:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="3" name="Text Placeholder 2"/>
    <p:cNvSpPr>
```

```
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
    <p:nvPr>
      <p:ph type="body" idx="1"/>
    </p:nvPr>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="457200" y="1600200"/>
      <a:ext cx="8229600" cy="4525963"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
  </p:spPr>
  <p:txBody>
    <a:bodyPr vert="horz" lIns="91440" tIns="45720" rIns="91440"
              bIns="45720" rtlCol="0">
      <a:normAutofit/>
    </a:bodyPr>
    <a:lstStyle/>
    <a:p>
      <a:pPr lvl="0"/>
      <a:r>
        <a:rPr lang="en-US" smtClean="0"/>
        <a:t>Click to edit Master text styles</a:t>
      </a:r>
    </a:p>
    <a:p>
      ... and others through lvl="4", five total
    </a:p>
  </p:txBody>
</p:sp>
```

Behaviors

- Master placeholder specifies size, position, geometry, and text body properties, such as margins (inset) and autofit
- It appears a slide master permits at most five placeholders, at most one each of title, body, date, footer, and slide number
- A master placeholder is always a *<p:sp>* (autoshape) element. It does not allow the insertion of content, although text entered in the shape somehow becomes the prompt text for inheriting placeholders.
- Hypothesis:
  - Layout inherits properties from master based only on type (e.g. body, dt), with no bearing on idx value.
  - Slide inherits from layout strictly on idx value.
  - Need to experiment to see if position and size are inherited on body placeholder by type or if idx has any effect
    - expanded/repackaged, change body ph idx on layout and see if still inherits position.
    - result: slide layout still appears with the proper position and size, however the slide placeholder appears at (0, 0) and apparently with size of (0, 0).

  - What behavior is exhibited when a slide layout contains two body placeholders, neither of which has a matching idx value and neither of which has a specified size or position?
    * result: both placeholders are present, directly superimposed upon each other. They both inherit position and fill.
- GUI doesn't seem to provide a way to add additional "footer" placeholders
- Does the MS API include populated object placeholders in the Placeholders collection?

From a user perspective, a placeholder is a container into which content can be inserted. The position, size, and formatting of content inserted into the placeholder is determined by the placeholder, allowing those key presentation design characteristics to be determined at design time and kept consistent across presentations created from a particular template.

The placeholder appearing on a slide is only part of the overall placeholder mechanism however. Placeholder behavior requires three different categories of placeholder shape; those that exist on a slide master, those on a slide layout, and those that ultimately appear on a slide in a presentation.

These three categories of placeholder participate in a property inheritance hierarchy, either as an inheritor, an inheritee, or both. Placeholder shapes on masters are inheritees only. Conversely placeholder shapes on slides are inheritors only. Placeholders on slide layouts are both, a possible inheritor from a slide master placeholder and an inheritee to placeholders on slides linked to that layout.

A layout inherits from its master differently than a slide inherits from its layout. A layout placeholder inherits from the master placeholder sharing the same type. A slide placeholder inherits from the layout placeholder having the same *idx* value.

Glossary

placeholder shape A shape on a slide that inherits from a layout placeholder.

layout placeholder a shorthand name for the placeholder shape on the slide layout from which a particular placeholder on a slide inherits shape properties

master placeholder the placeholder shape on the slide master which a layout placeholder inherits from, if any.

Schema excerpt

```
<xsd:complexType name="CT_ApplicationNonVisualDrawingProps">
  <xsd:sequence>
    <xsd:element   name="ph"           type="CT_Placeholder"      minOccurs="0"/>
    <xsd:choice minOccurs="0">  <!-- a:EG_Media -->
      <xsd:element name="audioCd"       type="CT_AudioCD"/>
      <xsd:element name="wavAudioFile"  type="CT_EmbeddedWAVAudioFile"/>
      <xsd:element name="audioFile"     type="CT_AudioFile"/>
      <xsd:element name="videoFile"     type="CT_VideoFile"/>
      <xsd:element name="quickTimeFile" type="CT_QuickTimeFile"/>
    </xsd:choice>
    <xsd:element   name="custDataLst"   type="CT_CustomerDataList" minOccurs="0"/>
    <xsd:element   name="extLst"        type="CT_ExtensionList"    minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="isPhoto"   type="xsd:boolean" default="false"/>
  <xsd:attribute name="userDrawn" type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_Placeholder">
```

```
  <xsd:sequence>
    <xsd:element name="extLst" type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="type"            type="ST_PlaceholderType" default="obj"/>
  <xsd:attribute name="orient"          type="ST_Direction"       default="horz"/>
  <xsd:attribute name="sz"              type="ST_PlaceholderSize" default="full"/>
  <xsd:attribute name="idx"             type="xsd:unsignedInt"    default="0"/>
  <xsd:attribute name="hasCustomPrompt" type="xsd:boolean"        default="false"/>
</xsd:complexType>

<xsd:simpleType name="ST_PlaceholderType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="title"/>
    <xsd:enumeration value="body"/>
    <xsd:enumeration value="ctrTitle"/>
    <xsd:enumeration value="subTitle"/>
    <xsd:enumeration value="dt"/>
    <xsd:enumeration value="sldNum"/>
    <xsd:enumeration value="ftr"/>
    <xsd:enumeration value="hdr"/>
    <xsd:enumeration value="obj"/>
    <xsd:enumeration value="chart"/>
    <xsd:enumeration value="tbl"/>
    <xsd:enumeration value="clipArt"/>
    <xsd:enumeration value="dgm"/>
    <xsd:enumeration value="media"/>
    <xsd:enumeration value="sldImg"/>
    <xsd:enumeration value="pic"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_PlaceholderSize">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="full"/>
    <xsd:enumeration value="half"/>
    <xsd:enumeration value="quarter"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_Direction">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="horz"/>
    <xsd:enumeration value="vert"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_PlaceholderSize">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="full"/>
    <xsd:enumeration value="half"/>
    <xsd:enumeration value="quarter"/>
  </xsd:restriction>
</xsd:simpleType>
```

Notes on footer item placeholders

Recommendation and Argumentation

Place page number (and perhaps copyright, date) on master in plain text boxes. Use "Hide Background Graphics" option on title slide to prevent them showing there if desired.

- In the slide placeholder method, an individually selectable footer item placeholder is present on the slide:
  - It can be moved accidentally, losing the placement consistency originally imposed by the master. Recovering exact placement requires reapplying the layout which may make other, unwanted changes.
  - It can be deleted, which could be a plus or minus depending.
  - It can be formatted inconsistently.
  - In can be accidentally selected and generally pollutes the shape space of the slide.
- I'm not sure python-pptx does the right thing when adding slides from layouts containing footer items, at least not the same thing as PowerPoint does in automatically working out which footer items to copy over.

  Also, there's no method to copy over footer items from the layout.
- Alternatives:
  1. A page number can be placed on the master as a text box. This causes it to appear on every layout and every slide. No refreshing is necessary as this item is inherited, not copied at slide or layout create time.

     It can be excluded from an individual layout (such as a title slide layout) by setting the *Hide Background Graphics* flag on a layout-by-layout basis. Unfortunately, this also hides other items such as a logo, copyright text, and slide background, but these can be reapplied individually directly to that slide layout. Since the title slide layout is generally the only one folks want to hide a slide number from, this is perhaps not a great burden. Others, like perhaps closing slides, might have their own background anyway.

Hypothesis:

- There is a class of placeholders known as "footer elements", comprising date and time, footer text, and slide number.
- These placeholders can only be directly added to a master. They are not available as options to add directly to a layout or to a slide.
- In every case, these placeholders are copied, although some attributes may be inherited at run time (not sure). Footer items are copied from the master to a new layout. Likewise, items are copied from a layout to a slide, but only when explicitly asked for using Insert > Header and Footer. PowerPoint sometimes adds these automatically when they're already present on other slides (don't know the rules it uses).

Intended procedure

This is how slide footer items are designed to be used:

- Add footer placeholders to master with intended standard position and formatting.
- Available by default on each layout. Can be disabled for individual layouts, for example, the title slide layout using the Slide Master Ribbon > Edit Layout > Allow Footers checkbox. Also can be deleted and formatting changed individually on a layout-by-layout basis.
- Hidden by default on new slides. Use Insert > Header and Footer dialog to add which items to show. Can be applied to all slides. Under certain circumstances, PowerPoint adds selected footer items to new slides automatically, based on them being present on other slides already in the presentation.

PowerPoint Behaviors

- Layout: Slide Master (ribbon) > Edit Layout > Allow Footers (checkbox)
  - Adding or removing a footer item from a layout, does not automatically reflect that change on existing slides based on that layout. The change is also not reflected when you reapply the layout, at least for deleting case.
    - It does not appear when you reapply the layout ('Reset Layout to Default Settings' on Mac)
- Position appears to be inherited as long as you haven't moved the placeholder on the slide. This applies at master -> layout as well as layout -> slide level.
  - Refreshing the layout after the page number item has been deleted leaves the page number item, but repositions it to a default (right justified to right slide edge) position. Apparently it attempts to inherit the position from the layout but not presence (or not).
- The *Allow Footers* checkbox is inactive (not greyed out but can't be set on) when there are no footer placeholder items on the master.
- Allow footers seems to be something of a one-way street. You can't clear the checkbox once it's set. However, if you delete the footer placeholder(s), the checkbox clears all by itself.
- The placeholder copied from master to layout can be repositioned and formatted (font etc.).
- Menu > Insert > Header and Footer ... Is available on individual slides and can be applied to all slides. It also works for removing footer items. This applies to *slides*, not to slide layouts.

  This dialog is also available from the Page Setup dialog.
- Hide background graphics flag does not control footer items. It does however control visibility of all non-placeholder items on the layout (or slide). It's available at both the layout and slide level. Not sure how that's reflected in the XML.
- When creating a new slide layout, the footer placeholders from the master are all automatically shown. This is to say that the "Allow Footers" ribbon option defaults to True when adding a new slide layout.

  Clicking Allow Footers off hides all the footer items. They can also be deleted or formatted individually.
- In order to appear (at create time anyway), footer items need to be:
  1. Present on the master
  2. Present on the layout ('Allow Footers' turned on, ph not deleted)
  3. Insert Header and Footer turned on for each ph element to be shown. This is applied on a slide-by-slide basis.

Notes

- Header is a misnomer as far as slides are concerned. There are no header items for a slide. A notes page can have a header though, which I expect is why it appears, since the same dialog is used for slides and notes pages (and handouts).
- A master is intended only to contain certain items:
  - Title
  - Body
  - Footer placeholders (date/time, footer text, slide number)
  - Background and theme items, perhaps logos and static text

- http://www.powerpointninja.com/templates/powerpoint-templates-beware-of-the-footers/
- https://github.com/scanny/python-pptx/issues/64

Group Shape

Group of shapes that appear as a single shape ...

Related Schema Definitions

```
<xsd:complexType name="CT_GroupShape">
  <xsd:sequence>
    <xsd:element name="nvGrpSpPr"      type="CT_GroupShapeNonVisual"/>
    <xsd:element name="grpSpPr"        type="a:CT_GroupShapeProperties"/>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="sp"           type="CT_Shape"/>
      <xsd:element name="grpSp"        type="CT_GroupShape"/>
      <xsd:element name="graphicFrame" type="CT_GraphicalObjectFrame"/>
      <xsd:element name="cxnSp"        type="CT_Connector"/>
      <xsd:element name="pic"          type="CT_Picture"/>
      <xsd:element name="contentPart"  type="CT_Rel"/>
    </xsd:choice>
    <xsd:element name="extLst"         type="CT_ExtensionListModify" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_GroupShapeNonVisual">
  <xsd:sequence>
    <xsd:element name="cNvPr"      type="a:CT_NonVisualDrawingProps"/>
    <xsd:element name="cNvGrpSpPr" type="a:CT_NonVisualGroupDrawingShapeProps"/>
    <xsd:element name="nvPr"       type="CT_ApplicationNonVisualDrawingProps"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_GroupShapeProperties">
  <xsd:sequence>
    <xsd:element name="xfrm"    type="CT_GroupTransform2D"       minOccurs="0"/>
    <xsd:group   ref="EG_FillProperties"                         minOccurs="0"/>
    <xsd:group   ref="EG_EffectProperties"                       minOccurs="0"/>
    <xsd:element name="scene3d" type="CT_Scene3D"                minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode"/>
</xsd:complexType>

<xsd:group name="EG_FillProperties">
  <xsd:choice>
    <xsd:element name="noFill"    type="CT_NoFillProperties"/>
    <xsd:element name="solidFill" type="CT_SolidColorFillProperties"/>
    <xsd:element name="gradFill"  type="CT_GradientFillProperties"/>
    <xsd:element name="blipFill"  type="CT_BlipFillProperties"/>
    <xsd:element name="pattFill"  type="CT_PatternFillProperties"/>
    <xsd:element name="grpFill"   type="CT_GroupFillProperties"/>
  </xsd:choice>
</xsd:group>
```

Table

One of the shapes available for placing on a PowerPoint slide is the *table*. As shapes go, it is one of the more complex. In addition to having standard shape properties such as position and size, it contains what amount to sub-shapes that each have properties of their own. Prominent among these sub-element types are row, column, and cell.

Open questions

- not sure of the semantics of the `<a:tableStyleId>` element. Assuming the GUID it contains somehow maps to a style in the tableStyles.xml, or perhaps some contant values defined elsewhere.
- What would be the Pythonic way to delete a row or column from a table? The MS API has a `delete()` method on the row or column itself. I'm inclined to believe it would be more Pythonic to use the `del` keyword on the indexed list, e.g. `del rows[2]`.

Table Properties

`first_row` read/write boolean property which, when true, indicates the first row should be formatted differently, as for a heading row at the top of the table.

`first_col` read/write boolean property which, when true, indicates the first column should be formatted differently, as for a side-heading column at the far left of the table.

`last_row` read/write boolean property which, when true, indicates the last row should be formatted differently, as for a totals row at the bottom of the table.

`last_col` read/write boolean property which, when true, indicates the last column should be formatted differently, as for a side totals column at the far right of the table.

`horz_banding` read/write boolean property indicating whether alternate color "banding" should be applied to the body rows.

`vert_banding` read/write boolean property indicating whether alternate color "banding" should be applied to the table columns.

Attribute Names

property name	attribute	optionality
first_row	firstRow	optional
first_col	firstCol	optional
last_row	lastRow	optional
last_col	lastCol	optional
horz_banding	bandRow	optional
vert_banding	bandCol	optional

Characteristics

value type	boolean, None
mode	read/write

XML location

```
<a:tbl>
  <a:tblPr firstRow="1" lastCol="1" bandRow="1">
```

`<a:tblPr>` is an optional element. If it appears, it is the first child of `<a:tbl>`

Observed behavior

The MS PowerPoint® client exhibits the following behavior related to table properties:

upon insertion of a default, empty table `<a:tblPr firstRow="1" bandRow="1">` A tblPr element is present with a `firstRow` and `bandRow` attribute, each set to True.

after setting `firstRow` property off `<a:tblPr bandRow="1">` The `firstRow` attribute is removed, not set to `0` or `false`.

The `<a:tblPr>` element is always present, even when it contains no attributes, because it contains an `<a:tableStyleId>` element, even when the table style is set to none using the UI.

API requirements

`Table` class

Properties and methods required for a `Table` shape.

- `apply_style(style_id)` – change the style of the table. Not sure what the domain of `style_id` is.
- `cell(row, col)` – method to access an individual `_Cell` object.
- `columns` – collection of `_Column` objects, each corresponding to a column in the table, in left-to-right order.
- `first_col` – read/write boolean property which, when true, indicates the first column should be formatted differently, as for a side-heading column at the far left of the table.
- `first_row` – read/write boolean property which, when true, indicates the first row should be formatted differently, as for a heading row at the top of the table.
- `horz_banding` – read/write boolean property indicating whether alternate color "banding" should be applied to the body rows.
- `last_col` – read/write boolean property which, when true, indicates the last column should be formatted differently, as for a side totals column at the far right of the table.
- `last_row` – read/write boolean property which, when true, indicates the last row should be formatted differently, as for a totals row at the bottom of the table.
- `rows` – collection of `_Row` objects, each corresponding to a row in the table, in top-to-bottom order.
- `vert_banding` – read/write boolean property indicating whether alternate color "banding" should be applied to the table columns.

`_Cell` class

- `text_frame` – container for text in the cell.
- borders, something like LineProperties on each side

- inset (margins)
- anchor and anchor_center
- horzOverflow, not sure what this is exactly, maybe wrap or auto-resize to fit.

`_Column` class

Provide the properties and methods appropriate to a table column.

- `width` – read/write integer width of the column in English Metric Units
- perhaps `delete()` method

`_ColumnCollection` class

- `add(before)` – add a new column to the left of the column having index *before*, returning a reference to the new column. *before* defaults to `-1`, which adds the column as the last column in the table.

`_Row` class

- `height` – read/write integer height of the row in English Metric Units (EMU).

`_RowCollection` class

- `add(before)` – add a new row before the row having index *before*, returning a reference to the new row. *before* defaults to `-1`, which adds the row as the last row in the table.

Behavior

Table width and column widths

A table is created by specifying a row and column count, a position, and an overall size. Initial column widths are set by dividing the overall width by the number of columns, resolving any rounding errors in the last column. Conversely, when a column's width is specified, the table width is adjusted to the sum of the widths of all columns. Initial row heights are set similarly and overall table height adjusted to the sum of row heights when a row's height is specified.

Discovery protocol

- (/) Review MS API documentation
- (/) Inspect minimal XML produced by PowerPoint® client
- (.) Review and document relevant schema elements

MS API Analysis

MS API method to add a table is:

```
Shapes.AddTable(NumRows, NumColumns, Left, Top, Width, Height)
```

There is a HasTable property on Shape to indicate the shape "has" a table. Seems like "is" a table would be more apt, but I'm still looking :)

From the Table Members page on MSDN.

Most interesting `Table` members:

- `Cell(row, col)` method to access individual cells.
- `Columns` collection reference, with `Add` method (`Delete` method is on `Column` object)
- `Rows` collection reference
- FirstCol and FirstRow boolean properties that indicate whether to apply special formatting from theme or whatever to first column/row.
- LastCol, LastRow, and HorizBanding, all also boolean with similar behaviors
- TableStyle read-only to table style in theme. Table.ApplyStyle() method is used to set table style.

Columns collection and Rows collection both have an Add() method

Column Members page on MSDN.

- Delete()
- Width property

Cell Members page on MSDN.

- Merge() and Split() methods
- Borders reference to Borders collection of LineFormat objects
- Shape reference to shape object that cell is or has.

LineFormat Members page on MSDN.

- ForeColor
- Weight

XML produced by PowerPoint® application

Inspection Notes

A `tableStyles.xml` part is fleshed out substantially; looks like it's populated from built-in defaults "Medium Style 2 - Accent 1". It appears to specify colors indirectly by reference to theme-specified values.

XML produced by PowerPoint® client

```
<p:graphicFrame>
  <p:nvGraphicFramePr>
    <p:cNvPr id="2" name="Table 1"/>
    <p:cNvGraphicFramePr>
```

```
      <a:graphicFrameLocks noGrp="1"/>
    </p:cNvGraphicFramePr>
    <p:nvPr/>
  </p:nvGraphicFramePr>
  <p:xfrm>
    <a:off x="1524000" y="1397000"/>
    <a:ext cx="6096000" cy="741680"/>
  </p:xfrm>
  <a:graphic>
    <a:graphicData uri="http://schemas.openxmlformats.org/drawingml/2006/table">
      <a:tbl>
        <a:tblPr firstRow="1" bandRow="1">
          <a:tableStyleId>{5C22544A-7EE6-4342-B048-85BDC9FD1C3A}</a:tableStyleId>
        </a:tblPr>
        <a:tblGrid>
          <a:gridCol w="3048000"/>
          <a:gridCol w="3048000"/>
        </a:tblGrid>
        <a:tr h="370840">
          <a:tc>
            <a:txBody>
              <a:bodyPr/>
              <a:lstStyle/>
              <a:p>
                <a:endParaRPr lang="en-US"/>
              </a:p>
            </a:txBody>
            <a:tcPr/>
          </a:tc>
          <a:tc>
            <a:txBody>
              <a:bodyPr/>
              <a:lstStyle/>
              <a:p>
                <a:endParaRPr lang="en-US"/>
              </a:p>
            </a:txBody>
            <a:tcPr/>
          </a:tc>
        </a:tr>
        <a:tr h="370840">
          <a:tc>
            <a:txBody>
              <a:bodyPr/>
              <a:lstStyle/>
              <a:p>
                <a:endParaRPr lang="en-US"/>
              </a:p>
            </a:txBody>
            <a:tcPr/>
          </a:tc>
          <a:tc>
            <a:txBody>
              <a:bodyPr/>
              <a:lstStyle/>
              <a:p>
                <a:endParaRPr lang="en-US"/>
              </a:p>
            </a:txBody>
            <a:tcPr/>
          </a:tc>
```

```
            </a:txBody>
            <a:tcPr/>
          </a:tc>
        </a:tr>
      </a:tbl>
    </a:graphicData>
  </a:graphic>
</p:graphicFrame>
```

Resources

Table.FirstCol Property page on MSDN

- ISO-IEC-29500-1, Section 21.1.3 (DrawingML) Tables, pp3331
- ISO-IEC-29500-1, Section 21.1.3.13 tbl (Table), pp3344

Schema excerpt

```
<xsd:element name="tbl" type="CT_Table"/>

<xsd:complexType name="CT_Table">
  <xsd:sequence>
    <xsd:element name="tblPr"   type="CT_TableProperties" minOccurs="0"/>
    <xsd:element name="tblGrid" type="CT_TableGrid"/>
    <xsd:element name="tr"      type="CT_TableRow"        minOccurs="0" maxOccurs=
↪"unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TableProperties">
  <xsd:sequence>
    <xsd:group   ref="EG_FillProperties"   minOccurs="0"/>
    <xsd:group   ref="EG_EffectProperties" minOccurs="0"/>
    <xsd:choice minOccurs="0">
      <xsd:element name="tableStyle"   type="CT_TableStyle"/>
      <xsd:element name="tableStyleId" type="s:ST_Guid"/>
    </xsd:choice>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="rtl"      type="xsd:boolean" default="false"/>
  <xsd:attribute name="firstRow" type="xsd:boolean" default="false"/>
  <xsd:attribute name="firstCol" type="xsd:boolean" default="false"/>
  <xsd:attribute name="lastRow"  type="xsd:boolean" default="false"/>
  <xsd:attribute name="lastCol"  type="xsd:boolean" default="false"/>
  <xsd:attribute name="bandRow"  type="xsd:boolean" default="false"/>
  <xsd:attribute name="bandCol"  type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:complexType name="CT_TableGrid">
  <xsd:sequence>
    <xsd:element name="gridCol" type="CT_TableCol" minOccurs="0" maxOccurs="unbounded
↪"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="CT_TableCol">
  <xsd:sequence>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="w" type="ST_Coordinate" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_TableRow">
  <xsd:sequence>
    <xsd:element name="tc"     type="CT_TableCell"             minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="h" type="ST_Coordinate" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_TableCell">
  <xsd:sequence>
    <xsd:element name="txBody" type="CT_TextBody"               minOccurs="0"/>
    <xsd:element name="tcPr"   type="CT_TableCellProperties"    minOccurs="0"/>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="rowSpan"  type="xsd:int"     default="1"/>
  <xsd:attribute name="gridSpan" type="xsd:int"     default="1"/>
  <xsd:attribute name="hMerge"   type="xsd:boolean" default="false"/>
  <xsd:attribute name="vMerge"   type="xsd:boolean" default="false"/>
  <xsd:attribute name="id"       type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="CT_TextBody">
  <xsd:sequence>
    <xsd:element name="bodyPr"   type="CT_TextBodyProperties"/>
    <xsd:element name="lstStyle" type="CT_TextListStyle" minOccurs="0"/>
    <xsd:element name="p"        type="CT_TextParagraph" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TableCellProperties">
  <xsd:sequence>
    <xsd:element name="lnL"      type="CT_LineProperties"         minOccurs="0"/>
    <xsd:element name="lnR"      type="CT_LineProperties"         minOccurs="0"/>
    <xsd:element name="lnT"      type="CT_LineProperties"         minOccurs="0"/>
    <xsd:element name="lnB"      type="CT_LineProperties"         minOccurs="0"/>
    <xsd:element name="lnTlToBr" type="CT_LineProperties"         minOccurs="0"/>
    <xsd:element name="lnBlToTr" type="CT_LineProperties"         minOccurs="0"/>
    <xsd:element name="cell3D"   type="CT_Cell3D"                 minOccurs="0"/>
    <xsd:group   ref="EG_FillProperties"                          minOccurs="0"/>
    <xsd:element name="headers"  type="CT_Headers"                minOccurs="0"/>
    <xsd:element name="extLst"   type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="marL"         type="ST_Coordinate32"         default="91440"/>
  <xsd:attribute name="marR"         type="ST_Coordinate32"         default="91440"/>
  <xsd:attribute name="marT"         type="ST_Coordinate32"         default="45720"/>
  <xsd:attribute name="marB"         type="ST_Coordinate32"         default="45720"/>
  <xsd:attribute name="vert"         type="ST_TextVerticalType"     default="horz"/>
  <xsd:attribute name="anchor"       type="ST_TextAnchoringType"    default="t"/>
  <xsd:attribute name="anchorCtr"    type="xsd:boolean"             default="false"/>
```

```
  <xsd:attribute name="horzOverflow" type="ST_TextHorzOverflowType" default="clip"/>
</xsd:complexType>

<xsd:simpleType name="ST_Coordinate">
  <xsd:union memberTypes="ST_CoordinateUnqualified s:ST_UniversalMeasure"/>
</xsd:simpleType>

<xsd:simpleType name="ST_CoordinateUnqualified">
  <xsd:restriction base="xsd:long">
    <xsd:minInclusive value="-27273042329600"/>
    <xsd:maxInclusive value="27273042316900"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_UniversalMeasure">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="-?[0-9]+(\.[0-9]+)?(mm|cm|in|pt|pc|pi)"/>
  </xsd:restriction>
</xsd:simpleType>
```

Text

Paragraph Spacing

A paragraph has three spacing properties. It can have spacing that separates it from the prior paragraph, spacing that separates it from the following paragraph, and line spacing less than or greater than the default single line spacing.

Candidate protocol

Get and set space before/after:

```
>>> paragraph = TextFrame.add_paragraph()
>>> paragraph.space_before
0
>>> paragraph.space_before = Pt(6)
>>> paragraph.space_before
76200
>>> paragraph.space_before.pt
6.0
```

An existing paragraph that somehow has space before or after set in terms of lines (`<a:spcPct>`) will return a value of Length(0) for that property.

Get and set line spacing:

```
>>> paragraph = TextFrame.add_paragraph()
>>> paragraph.line_spacing
1.0
>>> isinstance(paragraph.line_spacing, float)
True
>>> paragraph.line_spacing = Pt(18)
>>> paragraph.line_spacing
228600
>>> isinstance(paragraph.line_spacing, Length)
True
```

```
>>> isinstance(paragraph.line_spacing, int)
True
>>> paragraph.line_spacing.pt
18.0
```

- The default value for `Paragraph.line_spacing` is 1.0 (lines).
- The units of the return value can be distinguished by its type. In practice, units can also be distinguished by magnitude. Lines are returned as a small-ish float. Point values are returned as a *Length* value.

  If type is the most convenient discriminator, the expression *isinstance(line_spacing, float)* will serve as an effective *is_lines* predicate.

  In practice, line values will rarely be greater than 3.0 and it would be hard to imagine a useful line spacing less than 1 pt (12700). If magnitude is the most convenient discriminator, 256 can be assumed to be a safe threshold value.

MS API

LineRuleAfter Determines whether line spacing after the last line in each paragraph is set to a specific number of points or lines. Read/write.

LineRuleBefore Determines whether line spacing before the first line in each paragraph is set to a specific number of points or lines. Read/write.

LineRuleWithin Determines whether line spacing between base lines is set to a specific number of points or lines. Read/write.

SpaceAfter Returns or sets the amount of space after the last line in each paragraph of the specified text, in points or lines. Read/write.

SpaceBefore Returns or sets the amount of space before the first line in each paragraph of the specified text, in points or lines. Read/write.

SpaceWithin Returns or sets the amount of space between base lines in the specified text, in points or lines. Read/write.

PowerPoint behavior

- The UI doesn't appear to allow line spacing to be set in units of lines, although the XML and MS API allow it to be specified so.

XML specimens

Default paragraph spacing:

```
<a:p>
  <a:r>
    <a:t>Paragraph with default spacing</a:t>
  </a:r>
</a:p>
```

Spacing before = 6 pt:

```
<a:p>
  <a:pPr>
    <a:spcBef>
      <a:spcPts val="600"/>
    </a:spcBef>
  </a:pPr>
  <a:r>
    <a:t>Paragraph spacing before = 6pt</a:t>
  </a:r>
</a:p>
```

Spacing after = 12 pt:

```
<a:p>
  <a:pPr>
    <a:spcAft>
      <a:spcPts val="1200"/>
    </a:spcAft>
  </a:pPr>
  <a:r>
    <a:t>Paragraph spacing after = 12pt</a:t>
  </a:r>
</a:p>
```

Line spacing = 24 pt:

```
<a:p>
  <a:pPr>
    <a:lnSpc>
      <a:spcPts val="2400"/>
    </a:lnSpc>
  </a:pPr>
  <a:r>
    <a:t>Paragraph line spacing = 24pt</a:t>
  </a:r>
  <a:br/>
  <a:r>
    <a:t>second line</a:t>
  </a:r>
</a:p>
```

Line spacing = 2 lines:

```
<a:p>
  <a:pPr>
    <a:lnSpc>
      <a:spcPct val="200000"/>
    </a:lnSpc>
  </a:pPr>
  <a:r>
    <a:t>Paragraph line spacing = 2 line</a:t>
  </a:r>
  <a:br/>
  <a:r>
    <a:t>second line</a:t>
  </a:r>
</a:p>
```

Schema excerpt

```
<xsd:complexType name="CT_TextParagraph">
  <xsd:sequence>
    <xsd:element name="pPr"        type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:group    ref="EG_TextRun" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="endParaRPr" type="CT_TextCharacterProperties" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextParagraphProperties">
  <xsd:sequence>
    <xsd:element name="lnSpc"       type="CT_TextSpacing"             minOccurs="0"/>
    <xsd:element name="spcBef"      type="CT_TextSpacing"             minOccurs="0"/>
    <xsd:element name="spcAft"      type="CT_TextSpacing"             minOccurs="0"/>
    <xsd:choice minOccurs="0">       <!-- EG_TextBulletColor -->
      <xsd:element name="buClrTx"   type="CT_TextBulletColorFollowText"/>
      <xsd:element name="buClr"     type="CT_Color"/>
    </xsd:choice>
    <xsd:choice minOccurs="0">       <!-- EG_TextBulletSize -->
      <xsd:element name="buSzTx"    type="CT_TextBulletSizeFollowText"/>
      <xsd:element name="buSzPct"   type="CT_TextBulletSizePercent"/>
      <xsd:element name="buSzPts"   type="CT_TextBulletSizePoint"/>
    </xsd:choice>
    <xsd:choice minOccurs="0">       <!-- EG_TextBulletTypeface -->
      <xsd:element name="buFontTx"  type="CT_TextBulletTypefaceFollowText"/>
      <xsd:element name="buFont"    type="CT_TextFont"/>
    </xsd:choice>
    <xsd:choice minOccurs="0">       <!-- EG_TextBullet -->
      <xsd:element name="buNone"    type="CT_TextNoBullet"/>
      <xsd:element name="buAutoNum" type="CT_TextAutonumberBullet"/>
      <xsd:element name="buChar"    type="CT_TextCharBullet"/>
      <xsd:element name="buBlip"    type="CT_TextBlipBullet"/>
    </xsd:choice>
    <xsd:element name="tabLst"      type="CT_TextTabStopList"         minOccurs="0"/>
    <xsd:element name="defRPr"      type="CT_TextCharacterProperties" minOccurs="0"/>
    <xsd:element name="extLst"      type="CT_OfficeArtExtensionList"  minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="marL"         type="ST_TextMargin"/>
  <xsd:attribute name="marR"         type="ST_TextMargin"/>
  <xsd:attribute name="lvl"          type="ST_TextIndentLevelType"/>
  <xsd:attribute name="indent"       type="ST_TextIndent"/>
  <xsd:attribute name="algn"         type="ST_TextAlignType"/>
  <xsd:attribute name="defTabSz"     type="ST_Coordinate32"/>
  <xsd:attribute name="rtl"          type="xsd:boolean"/>
  <xsd:attribute name="eaLnBrk"      type="xsd:boolean"/>
  <xsd:attribute name="fontAlgn"     type="ST_TextFontAlignType"/>
  <xsd:attribute name="latinLnBrk"   type="xsd:boolean"/>
  <xsd:attribute name="hangingPunct" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_TextSpacing">
  <xsd:choice>
    <xsd:element name="spcPct" type="CT_TextSpacingPercent"/>
    <xsd:element name="spcPts" type="CT_TextSpacingPoint"/>
  </xsd:choice>
</xsd:complexType>
```

```
<xsd:complexType name="CT_TextSpacingPercent">
  <xsd:attribute name="val" type="ST_TextSpacingPercentOrPercentString" use="required
↪"/>
</xsd:complexType>

<xsd:simpleType name="ST_TextSpacingPercentOrPercentString">
  <xsd:union memberTypes="ST_TextSpacingPercent s:ST_Percentage"/>
</xsd:simpleType>

<xsd:simpleType name="ST_TextSpacingPercent">
  <xsd:restriction base="ST_PercentageDecimal">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="13200000"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_Percentage">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="-?[0-9]+(\.[0-9]+)?%"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="CT_TextSpacingPoint">
  <xsd:attribute name="val" type="ST_TextSpacingPoint" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_TextSpacingPoint">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="158400"/>
  </xsd:restriction>
</xsd:simpleType>
```

Text - Fit text to shape

An AutoShape has a text frame, referred to in the PowerPoint UI as the shape's *Text Box*. One of the settings provided is *Autofit*, which can be one of "Do not autofit", "Resize text to fit shape", or "Resize shape to fit text". The scope of this analysis is how best to provide an alternative to the "Resize text to fit shape" behavior that simply resizes the text to the largest point size that will fit entirely within the shape extents.

This produces a similar visual effect, but the "auto-size" behavior does not persist to later changes to the text or shape. It is just as though the user had reduced the font size just until all the text fit within the shape.

Candidate Protocol

Shape size and text are set before calling *TextFrame.fit_text()*:

```
>>> shape.width, shape.height = cx, cy
>>> shape.text = 'Lorem ipsum .. post facto.'
>>> text_frame = shape.text_frame
>>> text_frame.auto_size, text_frame.word_wrap
(None, False)
```

Calling `TextFrame.fit_text()` sets auto-size to MSO_AUTO_SIZE.NONE, turns word-wrap on, and sets the font size of all text in the shape to the maximum size that will fit entirely within the shape, not to exceed the optional

max_size. The default *max_size* is 18pt. The font size is applied directly on each run-level element in the shape. The path to the matching font file must be specified:

```
>>> text_frame.fit_text('calibriz.ttf', max_size=18)
>>> text_frame.auto_size, text_frame.word_wrap
(MSO_AUTO_SIZE.NONE (0), True)
>>> text_frame.paragraphs[0].runs[0].font.size.pt
10
```

Current constraints

`TextFrame.fit_text()`

- Path to font file must be provided manually.
- Bold and italic variant is selected unconditionally.
- Calibri typeface name is selected unconditionally.
- User must manually set the font typeface of all shape text to match the provided font file.
- Font typeface and variant is assumed to be uniform.
- Font size is made uniform, without regard to existing differences in font size in the shape text.
- Line breaks are replaced with a single space.

Incremental enhancements

- **Allow font file to be specified.** This allows *`TextFrame.fit_text()`* to be run from *pip* installed version. OS-specific file locations and names can be determined by client.
- **Allow bold and italic to be specified.** This allows the client to determine whether to apply the bold and/or italic typeface variant to the text.
- **Allow typeface name to be specified.** This allows the client to determine which base typeface is applied to the text.
- **Auto-locate font file based on typeface and variants.** Relax requirement for client to specify font file, looking it up in current system based on system type, typeface name, and variants.

PowerPoint behavior

- PowerPoint shrinks text in whole-number font sizes.
- When assigning a font size to a shape, PowerPoint applies that font size at the run level, adding a *sz* attribute to the *<a:rPr>* element for every content child of every *<a:p>* element in the shape. The *<a:endParaRPr>* element in each paragraph also gets a *sz* attribute set to that size.

XML specimens

`<p:txBody>` for default new textbox:

```
<p:txBody>
  <a:bodyPr wrap="none">
    <a:spAutoFit/>  <!-- fit shape to text -->
  </a:bodyPr>
  <a:lstStyle/>
  <a:p/>
</p:txBody>
```

8" x 0.5" text box, default margins, defaulting to 18pt "full-size" text, auto-reduced to 12pt. `<a:t>` element text wrapped for compact display:

```
<p:txBody>
  <a:bodyPr wrap="square" rtlCol="0">
    <a:noAutofit/>
  </a:bodyPr>
  <a:lstStyle/>
  <a:p>
    <a:r>
      <a:rPr lang="en-US" sz=1200 dirty="0" smtClean="0"/>
      <a:t>The art and craft of designing typefaces is called type design.
           Designers of typefaces are called type designers and are often
           employed by type foundries. In digital typography, type
           designers are sometimes also called font developers or font
           designers.</a:t>
    </a:r>
    <a:endParaRPr lang="en-US" sz=1200 dirty="0"/>
  </a:p>
</p:txBody>
```

Related Schema Definitions

```
<xsd:complexType name="CT_TextBody">
  <xsd:sequence>
    <xsd:element name="bodyPr"   type="CT_TextBodyProperties"/>
    <xsd:element name="lstStyle" type="CT_TextListStyle" minOccurs="0"/>
    <xsd:element name="p"        type="CT_TextParagraph" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextBodyProperties">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="prstTxWarp"  type="CT_PresetTextShape"        minOccurs="0"/>
    <xsd:choice minOccurs="0">      <!-- EG_TextAutofit -->
      <xsd:element name="noAutofit"   type="CT_TextNoAutofit"/>
      <xsd:element name="normAutofit" type="CT_TextNormalAutofit"/>
      <xsd:element name="spAutoFit"   type="CT_TextShapeAutofit"/>
    </xsd:choice>
    <xsd:element name="scene3d"     type="CT_Scene3D"                minOccurs="0"/>
    <xsd:choice minOccurs="0">      <!-- EG_Text3D -->
      <xsd:element name="sp3d"        type="CT_Shape3D"/>
      <xsd:element name="flatTx"      type="CT_FlatText"/>
    </xsd:choice>
    <xsd:element name="extLst"      type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="rot"              type="ST_Angle"/>
  <xsd:attribute name="spcFirstLastPara" type="xsd:boolean"/>
```

```
  <xsd:attribute name="vertOverflow"     type="ST_TextVertOverflowType"/>
  <xsd:attribute name="horzOverflow"     type="ST_TextHorzOverflowType"/>
  <xsd:attribute name="vert"             type="ST_TextVerticalType"/>
  <xsd:attribute name="wrap"             type="ST_TextWrappingType"/>
  <xsd:attribute name="lIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="tIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="rIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="bIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="numCol"           type="ST_TextColumnCount"/>
  <xsd:attribute name="spcCol"           type="ST_PositiveCoordinate32"/>
  <xsd:attribute name="rtlCol"           type="xsd:boolean"/>
  <xsd:attribute name="fromWordArt"      type="xsd:boolean"/>
  <xsd:attribute name="anchor"           type="ST_TextAnchoringType"/>
  <xsd:attribute name="anchorCtr"        type="xsd:boolean"/>
  <xsd:attribute name="forceAA"          type="xsd:boolean"/>
  <xsd:attribute name="upright"          type="xsd:boolean" default="false"/>
  <xsd:attribute name="compatLnSpc"      type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_TextNoAutofit"/>

<xsd:complexType name="CT_TextParagraph">
  <xsd:sequence>
    <xsd:element name="pPr"        type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:choice minOccurs="0" maxOccurs="unbounded"/>  <!-- EG_TextRun -->
      <xsd:element name="r"        type="CT_RegularTextRun"/>
      <xsd:element name="br"       type="CT_TextLineBreak"/>
      <xsd:element name="fld"      type="CT_TextField"/>
    </xsd:choice>
    <xsd:element name="endParaRPr" type="CT_TextCharacterProperties" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_RegularTextRun">
  <xsd:sequence>
    <xsd:element name="rPr" type="CT_TextCharacterProperties" minOccurs="0"/>
    <xsd:element name="t"   type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextLineBreak">
  <xsd:sequence>
    <xsd:element name="rPr" type="CT_TextCharacterProperties" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextField">
  <xsd:sequence>
    <xsd:element name="rPr" type="CT_TextCharacterProperties" minOccurs="0"/>
    <xsd:element name="pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="t"   type="xsd:string"                 minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id"   type="s:ST_Guid"  use="required"/>
  <xsd:attribute name="type" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="CT_TextCharacterProperties">
  <xsd:sequence>
```

```
    <xsd:element name="ln"                  type="CT_LineProperties"
↪minOccurs="0"/>
    <xsd:group    ref="EG_FillProperties"
↪minOccurs="0"/>
    <xsd:group    ref="EG_EffectProperties"
↪minOccurs="0"/>
    <xsd:element name="highlight"           type="CT_Color"
↪minOccurs="0"/>
    <xsd:group    ref="EG_TextUnderlineLine"
↪minOccurs="0"/>
    <xsd:group    ref="EG_TextUnderlineFill"
↪minOccurs="0"/>
    <xsd:element name="latin"               type="CT_TextFont"
↪minOccurs="0"/>
    <xsd:element name="ea"                  type="CT_TextFont"
↪minOccurs="0"/>
    <xsd:element name="cs"                  type="CT_TextFont"
↪minOccurs="0"/>
    <xsd:element name="sym"                 type="CT_TextFont"
↪minOccurs="0"/>
    <xsd:element name="hlinkClick"          type="CT_Hyperlink"
↪minOccurs="0"/>
    <xsd:element name="hlinkMouseOver"      type="CT_Hyperlink"
↪minOccurs="0"/>
    <xsd:element name="rtl"                 type="CT_Boolean"
↪minOccurs="0"/>
    <xsd:element name="extLst"              type="CT_OfficeArtExtensionList"
↪minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="kumimoji"   type="xsd:boolean"/>
  <xsd:attribute name="lang"       type="s:ST_Lang"/>
  <xsd:attribute name="altLang"    type="s:ST_Lang"/>
  <xsd:attribute name="sz"         type="ST_TextFontSize"/>
  <xsd:attribute name="b"          type="xsd:boolean"/>
  <xsd:attribute name="i"          type="xsd:boolean"/>
  <xsd:attribute name="u"          type="ST_TextUnderlineType"/>
  <xsd:attribute name="strike"     type="ST_TextStrikeType"/>
  <xsd:attribute name="kern"       type="ST_TextNonNegativePoint"/>
  <xsd:attribute name="cap"        type="ST_TextCapsType"/>
  <xsd:attribute name="spc"        type="ST_TextPoint"/>
  <xsd:attribute name="normalizeH" type="xsd:boolean"/>
  <xsd:attribute name="baseline"   type="ST_Percentage"/>
  <xsd:attribute name="noProof"    type="xsd:boolean"/>
  <xsd:attribute name="dirty"      type="xsd:boolean"     default="true"/>
  <xsd:attribute name="err"        type="xsd:boolean"     default="false"/>
  <xsd:attribute name="smtClean"   type="xsd:boolean"     default="true"/>
  <xsd:attribute name="smtId"      type="xsd:unsignedInt" default="0"/>
  <xsd:attribute name="bmk"        type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="CT_TextFont">
  <xsd:attribute name="typeface"    type="ST_TextTypeface" use="required"/>
  <xsd:attribute name="panose"      type="s:ST_Panose"/>
  <xsd:attribute name="pitchFamily" type="ST_PitchFamily"  default="0"/>
  <xsd:attribute name="charset"     type="xsd:byte"        default="1"/>
</xsd:complexType>

<xsd:complexType name="CT_Hyperlink">
```

```
  <xsd:sequence>
    <xsd:element name="snd"    type="CT_EmbeddedWAVAudioFile"   minOccurs="0"/>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute ref="r:id"/>
    <xsd:attribute name="invalidUrl"     type="xsd:string"  default=""/>
    <xsd:attribute name="action"         type="xsd:string"  default=""/>
    <xsd:attribute name="tgtFrame"       type="xsd:string"  default=""/>
    <xsd:attribute name="tooltip"        type="xsd:string"  default=""/>
    <xsd:attribute name="history"        type="xsd:boolean" default="true"/>
    <xsd:attribute name="highlightClick" type="xsd:boolean" default="false"/>
    <xsd:attribute name="endSnd"         type="xsd:boolean" default="false"/>
</xsd:complexType>

<xsd:simpleType name="ST_TextCapsType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="small"/>
    <xsd:enumeration value="all"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_TextFontSize">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value="100"/>
    <xsd:maxInclusive value="400000"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_TextTypeface">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

Text - Auto-fit text to shape

An AutoShape has a text frame, referred to in the PowerPoint UI as the shape's *Text Box*. One of the settings provided is *Autofit*, which can be one of "Do not autofit", "Resize text to fit shape", or "Resize shape to fit text". The scope of this analysis is how best to implement the "Resize text to fit shape" behavior.

A robust implementation would be complex and would lead the project outside the currently intended scope. In particular, because the shape size, text content, the "full" point size of the text, and the autofit and wrap settings of the text frame all interact to determine the proper "fit" of adjusted text, all events that could change the state of any of these five factors would need to be coupled to an "update" method. There would also need to be at least two "fitting" algorithms, one for when wrap was turned on and another for when it was off.

The initial solution we'll pursue is a special-purpose method on TextFrame that reduces the permutations of these variables to one and places responsibility on the developer to call it at the appropriate time. The method will calculate based on the current text and shape size, set wrap on, and set auto_size to fit text to the shape. If any of the variables change, the developer will be responsible for re-calling the method.

Current constraints

TextFrame.autofit_text()

- User must manually set all shape text to a uniform 12pt default full-size font point size.
  - This is intended to be done before the call, but might work okay if done after too, as long as it matches the default 12pt.
- Only 12pt "full-size" is supported. There is no mechanism to specify other sizes.
- Unconditionally sets autofit and wrap.
- Path to font file must be provided manually.
- User must manually set the font typeface of all shape text to match the provided font file.

Incremental enhancements

- **.fit_text() or .autofit_text().** Two related methods are used to fit text in a shape using different approaches. `TextFrame.autofit_text()` uses the *TEXT_TO_FIT_SHAPE* autofit setting to shrink a full-size paragraph of text. Later edits to that text using PowerPoint will re-fit the text automatically, up to the original (full-size) font size.

  `TextFrame.fit_text()` takes the approach of simply setting the font size for all text in the shape to the best-fit size. No automatic resizing occurs on later edits in PowerPoint, although the user can switch on auto-fit for that text box, perhaps after setting the full-size point size to their preferred size.
- **Specified full point size.** Allow the point size maximum to be specified; defaults to 18pt. All text in the shape is set to this size before calculating the best-fit based on that size.
- **Specified font.** In the process, this specifies the font to use, although it may require a tuple specifying the type family name as well as the bold and italic states, plus a file path.
- **Auto-locate installed font file.** Search for and use the appropriate locally-installed font file corresponding to the selected typeface. On Windows, the font directory can be located using a registry key, and is perhaps often *C:\Windows\Fonts*. However the font filename is not the same as the UI typeface name, so some mapping would be required, including detecting whether bold and/or italic were specified.
- **Accommodate line breaks.** Either from <*a:br*> elements or multiple paragraphs. Would involve ending lines at a break, other than the last paragraph.
- **Add line space reduction.** PowerPoint always reduces line spacing by as much as 20% to maximize the font size used. Add this factor into the calculation to improve the exact match likelihood for font scale and lnSpcReduction and thereby reduce occurence of "jumping" of text to a new size on edit.

Candidate Protocol

Shape size and text are set before calling `TextFrame.autofit_text()` or *`TextFrame.fit_text()`*:

```
>>> shape.width, shape.height = cx, cy
>>> shape.text = 'Lorem ipsum .. post facto.'
>>> text_frame = shape.text_frame
>>> text_frame.auto_size, text_frame.word_wrap, text_frame._font_scale
(None, False, None)
```

Calling `TextFrame.autofit_text()` turns on auto fit (text to shape), switches on word wrap, and calculates the best-fit font scaling factor:

```
>>> text_frame.autofit_text()
>>> text_frame.auto_size, text_frame.word_wrap, text_frame._font_scale
(TEXT_TO_FIT_SHAPE (2), True, 55.0)
```

Calling `TextFrame.fit_text()` produces the same sized text, but autofit is not turned on. Rather the actual font size for text in the shape is set to the calculated "best fit" point size:

```
>>> text_frame.fit_text()
>>> text_frame.auto_size, text_frame.word_wrap, text_frame._font_scale
(None, True, None)
>>> text_frame.paragraphs[0].font.size.pt
10
```

Proposed *python-pptx* behavior

The `TextFrame.fit_text()` method produces the following side effects:

- `TextFrame.auto_size` is set to `MSO_AUTO_SIZE.TEXT_TO_FIT_SHAPE`
- `TextFrame.word_wrap` is set to `True`.
- A suitable value is calculated for *<a:normAutofit fontScale="?"/>*. The *fontScale* attribute is set to this value and the *lnSpcReduction* attribute is removed, if present.

The operation can be undone by assigning `None`, `MSO_AUTO_SIZE.NONE`, or `MSO_AUTO_SIZE.SHAPE_TO_FIT_TEXT` to *TextFrame.auto_size*.

PowerPoint behavior

- PowerPoint shrinks text in whole-number font sizes.
- The behavior interacts with *Wrap text in shape*. The behavior we want here is only when wrap is turned on. When wrap is off, only height and manual line breaks are taken into account. Long lines simply extend outside the box.
- When assigning a font size to a shape, PowerPoint applies that font size at the run level, adding a *sz* attribute to the *<a:rPr>* element for every content child of every *<a:p>* element in the shape. The sentinel *<a:endParaRPr>* element also gets a *sz* attribute set to that size, but only in the last paragraph, it appears.

XML specimens

`<p:txBody>` for default new textbox:

```
<p:txBody>
  <a:bodyPr wrap="none">
    <a:spAutoFit/>  <!-- fit shape to text -->
  </a:bodyPr>
  <a:lstStyle/>
  <a:p/>
</p:txBody>
```

8" x 0.5" text box, default margins, defaulting to 18pt "full-size" text, auto-reduced to 10pt. `<a:t>` element text wrapped for compact display:

```
<p:txBody>
  <a:bodyPr wrap="square" rtlCol="0">
    <a:normAutofit fontScale="55000" lnSpcReduction="20000"/>
  </a:bodyPr>
  <a:lstStyle/>
  <a:p>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0"/>
      <a:t>The art and craft of designing typefaces is called type design.
           Designers of typefaces are called type designers and are often
           employed by type foundries. In digital typography, type
           designers are sometimes also called font developers or font
           designers.</a:t>
    </a:r>
    <a:endParaRPr lang="en-US" dirty="0"/>
  </a:p>
</p:txBody>
```

Related Schema Definitions

```
<xsd:complexType name="CT_TextBody">
  <xsd:sequence>
    <xsd:element name="bodyPr"   type="CT_TextBodyProperties"/>
    <xsd:element name="lstStyle" type="CT_TextListStyle" minOccurs="0"/>
    <xsd:element name="p"        type="CT_TextParagraph" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextBodyProperties">  <!-- denormalized -->
  <xsd:sequence>
    <xsd:element name="prstTxWarp"  type="CT_PresetTextShape"        minOccurs="0"/>
    <xsd:choice minOccurs="0">      <!-- EG_TextAutofit -->
      <xsd:element name="noAutofit"   type="CT_TextNoAutofit"/>
      <xsd:element name="normAutofit" type="CT_TextNormalAutofit"/>
      <xsd:element name="spAutoFit"   type="CT_TextShapeAutofit"/>
    </xsd:choice>
    <xsd:element name="scene3d"     type="CT_Scene3D"                minOccurs="0"/>
    <xsd:choice minOccurs="0">      <!-- EG_Text3D -->
      <xsd:element name="sp3d"        type="CT_Shape3D"/>
      <xsd:element name="flatTx"      type="CT_FlatText"/>
    </xsd:choice>
    <xsd:element name="extLst"      type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="rot"              type="ST_Angle"/>
  <xsd:attribute name="spcFirstLastPara" type="xsd:boolean"/>
  <xsd:attribute name="vertOverflow"     type="ST_TextVertOverflowType"/>
  <xsd:attribute name="horzOverflow"     type="ST_TextHorzOverflowType"/>
  <xsd:attribute name="vert"             type="ST_TextVerticalType"/>
  <xsd:attribute name="wrap"             type="ST_TextWrappingType"/>
  <xsd:attribute name="lIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="tIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="rIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="bIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="numCol"           type="ST_TextColumnCount"/>
  <xsd:attribute name="spcCol"           type="ST_PositiveCoordinate32"/>
  <xsd:attribute name="rtlCol"           type="xsd:boolean"/>
```

```
  <xsd:attribute name="fromWordArt"      type="xsd:boolean"/>
  <xsd:attribute name="anchor"           type="ST_TextAnchoringType"/>
  <xsd:attribute name="anchorCtr"        type="xsd:boolean"/>
  <xsd:attribute name="forceAA"          type="xsd:boolean"/>
  <xsd:attribute name="upright"          type="xsd:boolean" default="false"/>
  <xsd:attribute name="compatLnSpc"      type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_TextNormalAutofit">
  <xsd:attribute name="fontScale" type="ST_TextFontScalePercentOrPercentString"
                 use="optional" default="100%"/>
  <xsd:attribute name="lnSpcReduction" type="ST_TextSpacingPercentOrPercentString"
                 use="optional" default="0%"/>
</xsd:complexType>

<xsd:complexType name="CT_TextShapeAutofit"/>

<xsd:complexType name="CT_TextNoAutofit"/>

<xsd:simpleType name="ST_TextFontScalePercentOrPercentString">
  <xsd:union memberTypes="ST_TextFontScalePercent s:ST_Percentage"/>
</xsd:simpleType>

<xsd:simpleType name="ST_TextFontScalePercent">
  <xsd:restriction base="ST_PercentageDecimal">
    <xsd:minInclusive value="1000"/>
    <xsd:maxInclusive value="100000"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_Percentage">  <!-- s:ST_Percentage -->
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="-?[0-9]+(\.[0-9]+)?%"/>
  </xsd:restriction>
</xsd:simpleType>
```

Text

All text in PowerPoint appears in a shape. Of the six shape types, only an AutoShape can directly contain text. This includes text placeholders, text boxes, and geometric auto shapes. These correspond to shapes based on the `<p:sp>` XML element.

Each auto shape has a text frame, represented by a *`TextFrame`* object. A text frame contains one or more paragraphs, and each paragraph contains a sequence of zero or more run, line break, or field elements (`<a:r>`, `<a:br>`, and `<a:fld>`, respectively).

Notes

- include a:fld/a:t contents in Paragraph.text

XML specimens

Default text box with text, a line break, and a field:

```
<p:txBody>
  <a:bodyPr wrap="none" rtlCol="0">
    <a:spAutoFit/>
  </a:bodyPr>
  <a:lstStyle/>
  <a:p>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0"/>
      <a:t>This text box has a break here -&gt;</a:t>
    </a:r>
    <a:br>
      <a:rPr lang="en-US" dirty="0" smtClean="0"/>
    </a:br>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0"/>
      <a:t>and a Field here -&gt; </a:t>
    </a:r>
    <a:fld id="{466D331B-39A1-D247-9EB6-2F41F44AA032}" type="datetime2">
      <a:rPr lang="en-US" smtClean="0"/>
      <a:t>Monday, September 15, 14</a:t>
    </a:fld>
    <a:endParaRPr lang="en-US" dirty="0"/>
  </a:p>
</p:txBody>
```

Schema excerpt

```
<xsd:complexType name="CT_TextBody">
  <xsd:sequence>
    <xsd:element name="bodyPr"   type="CT_TextBodyProperties"/>
    <xsd:element name="lstStyle" type="CT_TextListStyle"      minOccurs="0"/>
    <xsd:element name="p"        type="CT_TextParagraph"      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextBodyProperties">
  <xsd:sequence>
    <xsd:element name="prstTxWarp" type="CT_PresetTextShape"        minOccurs="0"/>
    <xsd:group ref="EG_TextAutofit"                                 minOccurs="0"/>
    <xsd:element name="scene3d"    type="CT_Scene3D"                minOccurs="0"/>
    <xsd:group ref="EG_Text3D"                                      minOccurs="0"/>
    <xsd:element name="extLst"     type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="rot"              type="ST_Angle"/>
  <xsd:attribute name="spcFirstLastPara" type="xsd:boolean"/>
  <xsd:attribute name="vertOverflow"     type="ST_TextVertOverflowType"/>
  <xsd:attribute name="horzOverflow"     type="ST_TextHorzOverflowType"/>
  <xsd:attribute name="vert"             type="ST_TextVerticalType"/>
  <xsd:attribute name="wrap"             type="ST_TextWrappingType"/>
  <xsd:attribute name="lIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="tIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="rIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="bIns"             type="ST_Coordinate32"/>
  <xsd:attribute name="numCol"           type="ST_TextColumnCount"/>
  <xsd:attribute name="spcCol"           type="ST_PositiveCoordinate32"/>
  <xsd:attribute name="rtlCol"           type="xsd:boolean"/>
```

```
  <xsd:attribute name="fromWordArt"      type="xsd:boolean"/>
  <xsd:attribute name="anchor"           type="ST_TextAnchoringType"/>
  <xsd:attribute name="anchorCtr"        type="xsd:boolean"/>
  <xsd:attribute name="forceAA"          type="xsd:boolean"/>
  <xsd:attribute name="upright"          type="xsd:boolean" default="false"/>
  <xsd:attribute name="compatLnSpc"      type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_TextListStyle">
  <xsd:sequence>
    <xsd:element name="defPPr"  type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl1pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl2pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl3pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl4pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl5pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl6pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl7pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl8pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="lvl9pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList"  minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextParagraph">
  <xsd:sequence>
    <xsd:element name="pPr"        type="CT_TextParagraphProperties" minOccurs="0"/>
    <xsd:group    ref="EG_TextRun"                                    minOccurs="0"␣
↪maxOccurs="unbounded"/>
    <xsd:element name="endParaRPr" type="CT_TextCharacterProperties" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_TextRun">
  <xsd:choice>
    <xsd:element name="r"   type="CT_RegularTextRun"/>
    <xsd:element name="br"  type="CT_TextLineBreak"/>
    <xsd:element name="fld" type="CT_TextField"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_RegularTextRun">
  <xsd:sequence>
    <xsd:element name="rPr" type="CT_TextCharacterProperties" minOccurs="0"/>
    <xsd:element name="t"   type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextLineBreak">
  <xsd:sequence>
    <xsd:element name="rPr" type="CT_TextCharacterProperties" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_TextField">
  <xsd:sequence>
    <xsd:element name="rPr" type="CT_TextCharacterProperties" minOccurs="0"/>
    <xsd:element name="pPr" type="CT_TextParagraphProperties" minOccurs="0"/>
```

```
    <xsd:element name="t"   type="xsd:string"                minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="id"   type="s:ST_Guid"  use="required"/>
  <xsd:attribute name="type" type="xsd:string"/>
</xsd:complexType>
```

Font - Underline

Text in PowerPoint shapes can be formatted with a rich choice of underlining styles. Two choices control the underline appearance of a font: the underline style and the underline color. There are 18 available underline styles, including such choices as single line, wavy, and dot-dash.

Candidate Protocol

```
>>> font.underline
False
>>> font.underline = True
>>> font.underline
SINGLE_LINE (2)
>>> font.underline = MSO_UNDERLINE.WAVY_DOUBLE_LINE
>>> font.underline
WAVY_DOUBLE_LINE (17)
>>> font.underline = False
>>> font.underline
False
```

XML specimens

Run with MSO_UNDERLINE.DOTTED_HEAVY_LINE:

```
<a:p>
  <a:r>
    <a:rPr lang="en-US" u="dottedHeavy" dirty="0" smtClean="0"/>
    <a:t>foobar</a:t>
  </a:r>
  <a:endParaRPr lang="en-US" dirty="0"/>
</a:p>
```

Related Schema Definitions

```
<xsd:complexType name="CT_TextCharacterProperties">
  <xsd:sequence>
    <xsd:element name="ln"            type="CT_LineProperties" minOccurs="0"/>
    <xsd:group   ref="EG_FillProperties"                       minOccurs="0"/>
    <xsd:group   ref="EG_EffectProperties"                     minOccurs="0"/>
    <xsd:element name="highlight"     type="CT_Color"          minOccurs="0"/>
    <xsd:group   ref="EG_TextUnderlineLine"                    minOccurs="0"/>
    <xsd:group   ref="EG_TextUnderlineFill"                    minOccurs="0"/>
    <xsd:element name="latin"         type="CT_TextFont"       minOccurs="0"/>
    <xsd:element name="ea"            type="CT_TextFont"       minOccurs="0"/>
```

```
    <xsd:element name="cs"             type="CT_TextFont"       minOccurs="0"/>
    <xsd:element name="sym"            type="CT_TextFont"       minOccurs="0"/>
    <xsd:element name="hlinkClick"     type="CT_Hyperlink"      minOccurs="0"/>
    <xsd:element name="hlinkMouseOver" type="CT_Hyperlink"      minOccurs="0"/>
    <xsd:element name="rtl"            type="CT_Boolean"        minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪/>
  </xsd:sequence>
  <xsd:attribute name="kumimoji"   type="xsd:boolean"/>
  <xsd:attribute name="lang"       type="s:ST_Lang"/>
  <xsd:attribute name="altLang"    type="s:ST_Lang"/>
  <xsd:attribute name="sz"         type="ST_TextFontSize"/>
  <xsd:attribute name="b"          type="xsd:boolean"/>
  <xsd:attribute name="i"          type="xsd:boolean"/>
  <xsd:attribute name="u"          type="ST_TextUnderlineType"/>
  <xsd:attribute name="strike"     type="ST_TextStrikeType"/>
  <xsd:attribute name="kern"       type="ST_TextNonNegativePoint"/>
  <xsd:attribute name="cap"        type="ST_TextCapsType"/>
  <xsd:attribute name="spc"        type="ST_TextPoint"/>
  <xsd:attribute name="normalizeH" type="xsd:boolean"/>
  <xsd:attribute name="baseline"   type="ST_Percentage"/>
  <xsd:attribute name="noProof"    type="xsd:boolean"/>
  <xsd:attribute name="dirty"      type="xsd:boolean"     default="true"/>
  <xsd:attribute name="err"        type="xsd:boolean"     default="false"/>
  <xsd:attribute name="smtClean"   type="xsd:boolean"     default="true"/>
  <xsd:attribute name="smtId"      type="xsd:unsignedInt" default="0"/>
  <xsd:attribute name="bmk"        type="xsd:string"/>
</xsd:complexType>

<xsd:simpleType name="ST_TextUnderlineType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="none"/>
    <xsd:enumeration value="words"/>
    <xsd:enumeration value="sng"/>
    <xsd:enumeration value="dbl"/>
    <xsd:enumeration value="heavy"/>
    <xsd:enumeration value="dotted"/>
    <xsd:enumeration value="dottedHeavy"/>
    <xsd:enumeration value="dash"/>
    <xsd:enumeration value="dashHeavy"/>
    <xsd:enumeration value="dashLong"/>
    <xsd:enumeration value="dashLongHeavy"/>
    <xsd:enumeration value="dotDash"/>
    <xsd:enumeration value="dotDashHeavy"/>
    <xsd:enumeration value="dotDotDash"/>
    <xsd:enumeration value="dotDotDashHeavy"/>
    <xsd:enumeration value="wavy"/>
    <xsd:enumeration value="wavyHeavy"/>
    <xsd:enumeration value="wavyDbl"/>
  </xsd:restriction>

<xsd:group name="EG_TextUnderlineLine">
  <xsd:choice>
    <xsd:element name="uLnTx" type="CT_TextUnderlineLineFollowText"/>
    <xsd:element name="uLn"   type="CT_LineProperties" minOccurs="0"/>
  </xsd:choice>
</xsd:group>
```

```
<xsd:group name="EG_TextUnderlineFill">
  <xsd:choice>
    <xsd:element name="uFillTx" type="CT_TextUnderlineFillFollowText"/>
    <xsd:element name="uFill"   type="CT_TextUnderlineFillGroupWrapper"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_TextUnderlineLineFollowText"/>

<xsd:complexType name="CT_TextUnderlineFillFollowText"/>

<xsd:complexType name="CT_TextUnderlineFillGroupWrapper">
  <xsd:group ref="EG_FillProperties"/>
</xsd:complexType>
```

Font Color

Overview

Font color is a particular case of the broader topic of *fill*, the case of *solid fill*, in which a drawing element is filled with a single color. Other possible fill types are *gradient*, *picture*, *pattern*, and *background* (transparent).

Any drawing element that can have a fill can have a solid fill, or color. Elements that can have a fill include shape, line, text, table, table cell, and slide background.

Scope

This analysis focuses on *font color*, although much of it is general to solid fills. The focus is simply to enable getting to feature implementation as quickly as possible, just not before its clear I understand how it works in general.

Candidate API

New members:

- `Font.color`
- `ColorFormat(fillable_elm)`
- `ColorFormat.type`
- `ColorFormat.rgb`
- `ColorFormat.theme_color`
- `ColorFormat.brightness`
- `RGBColor(r, g, b)`
- `RGBColor.__str__`
- `RGBColor.from_str(rgb_str)`

Enumerations:

- MSO_COLOR_TYPE_INDEX
- MSO_THEME_COLOR_INDEX

Protocol:

```
>>> assert isinstance(font.color, ColorFormat)

>>> font.color = 'anything'
AttributeError: can't set attribute

>>> assert font.color.type in MSO_COLOR_TYPE_INDEX
wouldn't actually work, but conceptually true

>>> font.color.rgb = RGB(0x3F, 0x2c, 0x36)

>>> font.color.theme_color = MSO_THEME_COLOR.ACCENT_1

>>> font.color.brightness = -0.25
```

XML specimens

Here is a representative sample of the various font color cases as they appear in the XML, as produced by PowerPoint. Some inoperative attributes have been removed for clarity.

Baseline run:

```
<a:r>
  <a:rPr/>
  <a:t>test text</a:t>
</a:r>
```

set to a specific RGB color, using color wheel; HSB sliders yield the same:

```
<a:rPr>
  <a:solidFill>
    <a:srgbClr val="748E1D"/>
  </a:solidFill>
</a:rPr>
```

set to theme color, Text 2:

```
<a:rPr>
  <a:solidFill>
    <a:schemeClr val="tx2"/>
  </a:solidFill>
</a:rPr>
```

set to theme color, Accent 2, 80% Lighter:

```
<a:rPr>
  <a:solidFill>
    <a:schemeClr val="accent2">
      <a:lumMod val="20000"/>
      <a:lumOff val="80000"/>
    </a:schemeClr>
  </a:solidFill>
</a:rPr>
```

PowerPoint API

Changing the color of text in the PowerPoint API is accomplished with something like this:

```
Set font = textbox.TextFrame.TextRange.Font

'set font to a specific RGB color
font.Color.RGB = RGB(12, 34, 56)

Debug.Print font.Color.RGB
> 3678732

'set font to a theme color; Accent 1, 25% Darker in this example
font.Color.ObjectThemeColor = msoThemeColorAccent1
font.Color.Brightness = -0.25
```

The ColorFormat object is the first interesting object here, the type of object returned by `TextRange.Font.Color`. It includes the following properties that will likely have counterparts in python-pptx:

Brightness Returns or sets the brightness of the specified object. The value for this property must be a number from -1.0 (darker) to 1.0 (lighter), with 0 corresponding to no brightness adjustment. Read/write Single. Note: this corresponds to selecting an adjusted theme color from the PowerPoint ribbon color picker, like 'Accent 1, 40% Lighter'.

ObjectThemeColor Returns or sets the theme color of the specified ColorFormat object. Read/Write. Accepts and returns values from the enumeration MsoThemeColorIndex.

RGB Returns or sets the red-green-blue (RGB) value of the specified color. Read/write.

Type Represents the type of color. Read-only.

Legacy properties

These two properties will probably not need to be implemented in python-pptx.

SchemeColor Returns or sets the color in the applied color scheme that's associated with the specified object. Accepts and returns values from PpColorSchemeIndex. Read/write. Appears to be a legacy method to accomodate code prior to PowerPoint 2007.

TintAndShade Sets or returns the lightening or darkening of the the color of a specified shape. Read/write.

Resources

- MSDN TextFrame2 Members
- MSDN TextRange Members
- MSDN Font Members
- MSDN ColorFormat Members
- MSDN MsoThemeColorIndex Enumeration

Behaviors

The API method `Brightness` corresponds to the UI action of selecting an auto-generated tint or shade of a theme color from the PowerPoint ribbon color picker:

Setting font color to Accent 1 from the UI produces:

```
<a:rPr>
  <a:solidFill>
    <a:schemeClr val="accent1"/>
  </a:solidFill>
</a:rPr>
```

The following code does the same from the API:

```
fnt.Color.ObjectThemeColor = msoThemeColorAccent1
```

Setting font color to Accent 1, Lighter 40% (40% tint) from the PowerPoint UI produces this XML:

```
<a:rPr>
  <a:solidFill>
    <a:schemeClr val="accent1">
      <a:lumMod val="60000"/>
      <a:lumOff val="40000"/>
    </a:schemeClr>
  </a:solidFill>
</a:rPr>
```

Setting `Brightness` to +0.4 has the same effect:

```
fnt.Color.Brightness = 0.4
```

Setting font color to Accent 1, Darker 25% (25% shade) from the UI results in the following XML. Note that no `<a:lumOff>` element is used.:

```
<a:rPr>
  <a:solidFill>
    <a:schemeClr val="accent1">
      <a:lumMod val="75000"/>
    </a:schemeClr>
  </a:solidFill>
</a:rPr>
```

Setting `Brightness` to -0.25 has the same effect:

```
fnt.Color.Brightness = -0.25
```

Calling `TintAndShade` with a positive value (between 0 and 1) causes a tint element to be inserted, but I'm not at all sure why and when one would want to use it rather than the `Brightness` property.:

```
fnt.Color.TintAndShade = 0.75

<a:rPr>
  <a:solidFill>
    <a:schemeClr val="accent1">
      <a:tint val="25000"/>
    </a:schemeClr>
  </a:solidFill>
</a:rPr>
```

Related Schema Definitions

```
<xsd:complexType name="CT_SolidColorFillProperties">
  <xsd:sequence>
    <xsd:group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_ColorChoice">
  <xsd:choice>
    <xsd:element name="scrgbClr"  type="CT_ScRgbColor"  minOccurs="1" maxOccurs="1"/>
    <xsd:element name="srgbClr"   type="CT_SRgbColor"   minOccurs="1" maxOccurs="1"/>
    <xsd:element name="hslClr"    type="CT_HslColor"    minOccurs="1" maxOccurs="1"/>
    <xsd:element name="sysClr"    type="CT_SystemColor" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="schemeClr" type="CT_SchemeColor" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="prstClr"   type="CT_PresetColor" minOccurs="1" maxOccurs="1"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_SRgbColor">
  <xsd:sequence>
    <xsd:group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="val" type="s:ST_HexColorRGB" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_SchemeColor">
  <xsd:sequence>
    <xsd:group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="val" type="ST_SchemeColorVal" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_PresetColor">
  <xsd:sequence>
    <xsd:group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="val" type="ST_PresetColorVal" use="required"/>
</xsd:complexType>

<xsd:group name="EG_ColorTransform">
  <xsd:choice>
    <xsd:element name="tint"     type="CT_PositiveFixedPercentage" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="shade"    type="CT_PositiveFixedPercentage" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="comp"     type="CT_ComplementTransform"     minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="inv"      type="CT_InverseTransform"        minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="gray"     type="CT_GrayscaleTransform"      minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="alpha"    type="CT_PositiveFixedPercentage" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="alphaOff" type="CT_FixedPercentage"         minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="alphaMod" type="CT_PositivePercentage"      minOccurs="1"␣
↪maxOccurs="1"/>
```

```
    <xsd:element name="hue"      type="CT_PositiveFixedAngle"      minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="hueOff"   type="CT_Angle"                   minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="hueMod"   type="CT_PositivePercentage"      minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="sat"      type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="satOff"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="satMod"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="lum"      type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="lumOff"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="lumMod"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="red"      type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="redOff"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="redMod"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="green"    type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="greenOff" type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="greenMod" type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blue"     type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blueOff"  type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blueMod"  type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="gamma"    type="CT_GammaTransform"          minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="invGamma" type="CT_InverseGammaTransform"   minOccurs="1"␣
↪maxOccurs="1"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_Percentage">
  <xsd:attribute name="val" type="ST_Percentage" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_Percentage">
  <xsd:union memberTypes="s:ST_Percentage"/>
</xsd:simpleType>

<xsd:simpleType name="ST_Percentage">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="-?[0-9]+(\.[0-9]+)?%"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_HexColorRGB">
```

```
  <xsd:restriction base="xsd:hexBinary">
    <xsd:length value="3" fixed="true"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_SchemeColorVal">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="bg1"/>
    <xsd:enumeration value="tx1"/>
    <xsd:enumeration value="bg2"/>
    <xsd:enumeration value="tx2"/>
    <xsd:enumeration value="accent1"/>
    <xsd:enumeration value="accent2"/>
    <xsd:enumeration value="accent3"/>
    <xsd:enumeration value="accent4"/>
    <xsd:enumeration value="accent5"/>
    <xsd:enumeration value="accent6"/>
    <xsd:enumeration value="hlink"/>
    <xsd:enumeration value="folHlink"/>
    <xsd:enumeration value="phClr"/>
    <xsd:enumeration value="dk1"/>
    <xsd:enumeration value="lt1"/>
    <xsd:enumeration value="dk2"/>
    <xsd:enumeration value="lt2"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:group name="EG_FillProperties">
  <xsd:choice>
    <xsd:element name="noFill"    type="CT_NoFillProperties"         minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="solidFill" type="CT_SolidColorFillProperties" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="gradFill"  type="CT_GradientFillProperties"   minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blipFill"  type="CT_BlipFillProperties"       minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="pattFill"  type="CT_PatternFillProperties"    minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="grpFill"   type="CT_GroupFillProperties"      minOccurs="1"␣
↪maxOccurs="1"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_NoFillProperties"/>

<xsd:complexType name="CT_GradientFillProperties">
  <xsd:sequence>
    <xsd:element name="gsLst"    type="CT_GradientStopList" minOccurs="0" maxOccurs="1
↪"/>
    <xsd:group   ref="EG_ShadeProperties"                   minOccurs="0" maxOccurs="1
↪"/>
    <xsd:element name="tileRect" type="CT_RelativeRect"     minOccurs="0" maxOccurs="1
↪"/>
  </xsd:sequence>
  <xsd:attribute name="flip"         type="ST_TileFlipMode" use="optional"/>
  <xsd:attribute name="rotWithShape" type="xsd:boolean"     use="optional"/>
</xsd:complexType>
```

Enumerations

MsoColorType

http://msdn.microsoft.com/en-us/library/office/aa432491(v=office.12).aspx

msoColorTypeRGB 1 - Color is determined by values of red, green, and blue.

msoColorTypeScheme 2 - Color is defined by an application-specific scheme.

msoColorTypeCMYK 3 - Color is determined by values of cyan, magenta, yellow, and black.

msoColorTypeCMS 4 - Color Management System color type.

msoColorTypeInk 5 - Not supported.

msoColorTypeMixed -2 - Not supported.

MsoThemeColorIndex

http://msdn.microsoft.com/en-us/library/office/aa432702(v=office.12).aspx

msoNotThemeColor 0 - Specifies no theme color.

msoThemeColorDark1 1 - Specifies the Dark 1 theme color.

msoThemeColorLight1 2 - Specifies the Light 1 theme color.

msoThemeColorDark2 3 - Specifies the Dark 2 theme color.

msoThemeColorLight2 4 - Specifies the Light 2 theme color.

msoThemeColorAccent1 5 - Specifies the Accent 1 theme color.

msoThemeColorAccent2 6 - Specifies the Accent 2 theme color.

msoThemeColorAccent3 7 - Specifies the Accent 3 theme color.

msoThemeColorAccent4 8 - Specifies the Accent 4 theme color.

msoThemeColorAccent5 9 - Specifies the Accent 5 theme color.

msoThemeColorAccent6 10 - Specifies the Accent 6 theme color.

msoThemeColorHyperlink 11 - Specifies the theme color for a hyperlink.

msoThemeColorFollowedHyperlink 12 - Specifies the theme color for a clicked hyperlink.

msoThemeColorText1 13 - Specifies the Text 1 theme color.

msoThemeColorBackground1 14 - Specifies the Background 1 theme color.

msoThemeColorText2 15 - Specifies the Text 2 theme color.

msoThemeColorBackground2 16 - Specifies the Background 2 theme color.

msoThemeColorMixed -2 - Specifies a mixed color theme.

Value Objects

RGB RGBColor would be an immutable value object that could be reused as often as needed and not tied to any part of the underlying XML tree.

Other possible bits

- acceptance test sketch
- test data requirements; files, builder(s)
- enumerations and mappings
- value types required
- test criteria

Example test criteria:

```
# XML
<a:ln>
  <a:solidFill>
    <a:srgbClr val="123456"/>
  </a:solidFill>
</a:ln>

assert font.color.type == MSO_COLOR_TYPE.RGB
assert font.color.rgb == RGB(0x12, 0x34, 0x56)
assert font.color.schemeClr == MSO_THEME_COLOR.NONE
assert font.color.brightness == 0.0
```

Font typeface

Overview

PowerPoint allows the font of text elements to be changed from, for example, Verdana to Arial. This aspect of a font is its *typeface*, as opposed to its size (e.g. 18 points) or style (e.g. bold, italic).

Minimum viable feature

```
>>> assert isinstance(font, pptx.text.Font)
>>> font.name
None
>>> font.name = 'Verdana'
>>> font.name
'Verdana'
```

Related MS API

- Font.Name
- Font.NameAscii
- Font.NameComplexScript
- Font.NameFarEast
- Font.NameOther

Protocol

```
>>> assert isinstance(font, pptx.text.Font)
>>> font.name
None
>>> font.name = 'Verdana'
>>> font.name
'Verdana'
```

XML specimens

Here is a representative sample of textbox XML showing the effect of applying a non-default typeface.

Baseline default textbox:

```
<p:txBody>
  <a:bodyPr wrap="none" rtlCol="0">
    <a:spAutoFit/>
  </a:bodyPr>
  <a:lstStyle/>
  <a:p>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0"/>
      <a:t>Baseline default textbox, no adjustments of any kind</a:t>
    </a:r>
    <a:endParaRPr lang="en-US" dirty="0"/>
  </a:p>
</p:txBody>
```

textbox with typeface applied at shape level:

```
<p:txBody>
  <a:bodyPr wrap="none" rtlCol="0">
    <a:spAutoFit/>
  </a:bodyPr>
  <a:lstStyle/>
  <a:p>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0">
        <a:latin typeface="Verdana"/>
        <a:cs typeface="Verdana"/>
      </a:rPr>
      <a:t>A textbox with Verdana typeface applied to shape, not text selection</a:t>
    </a:r>
    <a:endParaRPr lang="en-US" dirty="0">
      <a:latin typeface="Verdana"/>
      <a:cs typeface="Verdana"/>
    </a:endParaRPr>
  </a:p>
</p:txBody>
```

textbox with multiple runs, typeface applied at shape level:

```
<p:txBody>
  <a:bodyPr wrap="none" rtlCol="0">
    <a:spAutoFit/>
```

```
  </a:bodyPr>
  <a:lstStyle/>
  <a:p>
    <a:pPr algn="ctr"/>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0">
        <a:latin typeface="Arial Black"/>
        <a:cs typeface="Arial Black"/>
      </a:rPr>
      <a:t>textbox with multiple runs having typeface</a:t>
    </a:r>
    <a:br>
      <a:rPr lang="en-US" dirty="0" smtClean="0">
        <a:latin typeface="Arial Black"/>
        <a:cs typeface="Arial Black"/>
      </a:rPr>
    </a:br>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0">
        <a:latin typeface="Arial Black"/>
        <a:cs typeface="Arial Black"/>
      </a:rPr>
      <a:t>customized at shape level</a:t>
    </a:r>
    <a:endParaRPr lang="en-US" dirty="0">
      <a:latin typeface="Arial Black"/>
      <a:cs typeface="Arial Black"/>
    </a:endParaRPr>
  </a:p>
</p:txBody>
```

Asian characters, or possibly Asian font being applied:

```
<p:txBody>
  <a:bodyPr wrap="none" rtlCol="0">
    <a:spAutoFit/>
  </a:bodyPr>
  <a:lstStyle/>
  <a:p>
    <a:pPr algn="ctr"/>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0">
        <a:latin typeface="Hiragino Sans GB W3"/>
        <a:ea typeface="Hiragino Sans GB W3"/>
        <a:cs typeface="Hiragino Sans GB W3"/>
      </a:rPr>
      <a:t></a:t>
    </a:r>
    <a:endParaRPr lang="en-US" dirty="0">
      <a:latin typeface="Hiragino Sans GB W3"/>
      <a:ea typeface="Hiragino Sans GB W3"/>
      <a:cs typeface="Hiragino Sans GB W3"/>
    </a:endParaRPr>
  </a:p>
</p:txBody>
```

then applying Arial from font pull-down:

```
<p:txBody>
  <a:bodyPr wrap="none" rtlCol="0">
    <a:spAutoFit/>
  </a:bodyPr>
  <a:lstStyle/>
  <a:p>
    <a:pPr algn="ctr"/>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0">
        <a:latin typeface="Arial"/>
        <a:ea typeface="Hiragino Sans GB W3"/>
        <a:cs typeface="Arial"/>
      </a:rPr>
      <a:t></a:t>
    </a:r>
    <a:endParaRPr lang="en-US" dirty="0">
      <a:latin typeface="Arial"/>
      <a:ea typeface="Hiragino Sans GB W3"/>
      <a:cs typeface="Arial"/>
    </a:endParaRPr>
  </a:p>
</p:txBody>
```

Observations

- PowerPoint UI always applies typeface customization at run level rather than paragraph (defRPr) level, even when applied to shape rather than a specific selection of text.
- PowerPoint applies the same typeface to the `<a:latin>` and `<a:cs>` tag when a typeface is selected from the font pull-down in the UI.

Related Schema Definitions

```
<xsd:complexType name="CT_TextCharacterProperties">
  <xsd:sequence>
    <xsd:element name="ln"        type="CT_LineProperties" minOccurs="0" maxOccurs="1
↪"/>
    <xsd:group   ref="EG_FillProperties"                   minOccurs="0" maxOccurs="1
↪"/>
    <xsd:group   ref="EG_EffectProperties"                 minOccurs="0" maxOccurs="1
↪"/>
    <xsd:element name="highlight"      type="CT_Color"     minOccurs="0" maxOccurs="1
↪"/>
    <xsd:group   ref="EG_TextUnderlineLine"                minOccurs="0" maxOccurs="1
↪"/>
    <xsd:group   ref="EG_TextUnderlineFill"                minOccurs="0" maxOccurs="1
↪"/>
    <xsd:element name="latin"          type="CT_TextFont"  minOccurs="0" maxOccurs="1
↪"/>
    <xsd:element name="ea"             type="CT_TextFont"  minOccurs="0" maxOccurs="1
↪"/>
    <xsd:element name="cs"             type="CT_TextFont"  minOccurs="0" maxOccurs="1
↪"/>
    <xsd:element name="sym"            type="CT_TextFont"  minOccurs="0" maxOccurs="1
↪"/>
```

```
    <xsd:element name="hlinkClick"     type="CT_Hyperlink" minOccurs="0" maxOccurs="1
↪"/>
    <xsd:element name="hlinkMouseOver" type="CT_Hyperlink" minOccurs="0" maxOccurs="1
↪"/>
    <xsd:element name="rtl"            type="CT_Boolean"   minOccurs="0"/>
    <xsd:element name="extLst"         type="CT_OfficeArtExtensionList" minOccurs="0"␣
↪ maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="kumimoji"   type="xsd:boolean"             use="optional"/>
  <xsd:attribute name="lang"       type="s:ST_Lang"               use="optional"/>
  <xsd:attribute name="altLang"    type="s:ST_Lang"               use="optional"/>
  <xsd:attribute name="sz"         type="ST_TextFontSize"         use="optional"/>
  <xsd:attribute name="b"          type="xsd:boolean"             use="optional"/>
  <xsd:attribute name="i"          type="xsd:boolean"             use="optional"/>
  <xsd:attribute name="u"          type="ST_TextUnderlineType"    use="optional"/>
  <xsd:attribute name="strike"     type="ST_TextStrikeType"       use="optional"/>
  <xsd:attribute name="kern"       type="ST_TextNonNegativePoint" use="optional"/>
  <xsd:attribute name="cap"        type="ST_TextCapsType"         use="optional"/>
  <xsd:attribute name="spc"        type="ST_TextPoint"            use="optional"/>
  <xsd:attribute name="normalizeH" type="xsd:boolean"             use="optional"/>
  <xsd:attribute name="baseline"   type="ST_Percentage"           use="optional"/>
  <xsd:attribute name="noProof"    type="xsd:boolean"             use="optional"/>
  <xsd:attribute name="dirty"      type="xsd:boolean"             use="optional"␣
↪default="true"/>
  <xsd:attribute name="err"        type="xsd:boolean"             use="optional"␣
↪default="false"/>
  <xsd:attribute name="smtClean"   type="xsd:boolean"             use="optional"␣
↪default="true"/>
  <xsd:attribute name="smtId"      type="xsd:unsignedInt"         use="optional"␣
↪default="0"/>
  <xsd:attribute name="bmk"        type="xsd:string"              use="optional"/>
</xsd:complexType>

<xsd:complexType name="CT_TextFont">
  <xsd:attribute name="typeface"    type="ST_TextTypeface" use="required"/>
  <xsd:attribute name="panose"      type="s:ST_Panose"     use="optional"/>
  <xsd:attribute name="pitchFamily" type="ST_PitchFamily"  use="optional" default="0"/
↪>
  <xsd:attribute name="charset"     type="xsd:byte"        use="optional" default="1"/
↪>
</xsd:complexType>

<xsd:simpleType name="ST_TextTypeface">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

Hyperlinks

Overview

PowerPoint supports hyperlinks at two distinct levels:

- a run of text can be a hyperlink
- an entire shape can be a hyperlink

Hyperlinks in PowerPoint can have four types of targets:

- a Internet address, such as https://github/scanny/python-pptx, including an optional anchor (e.g. #sub-heading suffix to jump mid-page). This can also be an email address, launching the local email client. A mailto: URI is used, with subject specifiable using a '?subject=xyz' suffix.
- another file, e.g. another PowerPoint file, including an optional anchor to, for example, a specific slide. A file:// URI is used to specify the file path.
- another part in the same presentation. This uses an internal relationship (in the .rels item) to the target part.

An optional ScreenTip, a roll-over tool-tip sort of message, can also be specified for any of these link types.

There are some more obscure attributes like "stop playing sound before navigating" that are available, perhaps meant for kiosk-style applications.

A run or shape can actually have two distinct link behaviors, one for clicking and another for rolling over with the mouse. These are independent; a run or shape can have one, the other, both, or neither. These are the two hyperlink types reported by the Hyperlink.type attribute and using enumeration values from MsoHyperlinkType.

Minimum viable feature

Start with Run since that's the current use case and doesn't require working out exactly which shapes can be hyperlinked.

```
r = p.add_run()
r.text = 'link to python-pptx @ GitHub'
hlink = r.hyperlink
hlink.address = 'https://github.com/scanny/python-pptx'
```

Roadmap items

- add .hyperlink attribute to Shape and Picture

Resources

- Hyperlink Object (PowerPoint) on MSDN

Candidate Protocol

Add a hyperlink:

```
p = shape.text_frame.paragraphs[0]
r = p.add_run()
r.text = 'link to python-pptx @ GitHub'
hlink = r.hyperlink
hlink.address = 'https://github.com/scanny/python-pptx'
```

Delete a hyperlink:

```
r.hyperlink = None

# or -----------

r.hyperlink.address = None  # empty string '' will do it too
```

A Hyperlink instance is lazy-created on first reference. The object persists until garbage collected once created. The link XML is not written until .address is specified. Setting `hlink.address` to None or '' causes the hlink entry to be removed if present.

Candidate API

_Run.hyperlink

Shape.hyperlink

Hyperlink

.address - target URL

.screen_tip - tool-tip text displayed on mouse rollover is slideshow mode

.type - MsoHyperlinkType (shape or run)

.show_and_return ...

_Run.rolloverlink would be an analogous property corresponding to the <a:hlinkMouseOver> element

Enumerations

- MsoHyperlinkType: msoHyperlinkRange, msoHyperlinkShape

Open questions

- What is the precise scope of shape types that may have a hyperlink applied?
- does leading and trailing space around a hyperlink work as expected?
- not sure what PowerPoint does if you select multiple runs and then insert a hyperlink, like including a stretch of bold text surrounded by plain text in the selection.

XML specimens

Link on overall shape:

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="2" name="Rectangle 1">
      <a:hlinkClick r:id="rId2"/>
    </p:cNvPr>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  ...
<p:sp>
```

Link on a run within a paragraph:

```
<a:p>
  <a:r>
    <a:rPr lang="en-US" dirty="0" smtClean="0"/>
    <a:t>Code is available at </a:t>
```

```
  </a:r>
  <a:r>
    <a:rPr lang="en-US" dirty="0" smtClean="0">
      <a:hlinkClick r:id="rId2"/>
    </a:rPr>
    <a:t>the python-pptx repository on GitHub</a:t>
  </a:r>
  <a:endParaRPr lang="en-US" dirty="0"/>
</a:p>
```

Related Schema Definitions

```
<xsd:complexType name="CT_TextCharacterProperties">
  <xsd:sequence>
    <xsd:element name="ln"             type="CT_LineProperties"         minOccurs="0"/
↪>
    <xsd:group    ref="EG_FillProperties"                               minOccurs="0"/
↪>
    <xsd:group    ref="EG_EffectProperties"                             minOccurs="0"/
↪>
    <xsd:element name="highlight"      type="CT_Color"                  minOccurs="0"/
↪>
    <xsd:group    ref="EG_TextUnderlineLine"                            minOccurs="0"/
↪>
    <xsd:group    ref="EG_TextUnderlineFill"                            minOccurs="0"/
↪>
    <xsd:element name="latin"          type="CT_TextFont"               minOccurs="0"/
↪>
    <xsd:element name="ea"             type="CT_TextFont"               minOccurs="0"/
↪>
    <xsd:element name="cs"             type="CT_TextFont"               minOccurs="0"/
↪>
    <xsd:element name="sym"            type="CT_TextFont"               minOccurs="0"/
↪>
    <xsd:element name="hlinkClick"     type="CT_Hyperlink"              minOccurs="0"/
↪>
    <xsd:element name="hlinkMouseOver" type="CT_Hyperlink"              minOccurs="0"/
↪>
    <xsd:element name="rtl"            type="CT_Boolean"                minOccurs="0"/
↪>
    <xsd:element name="extLst"         type="CT_OfficeArtExtensionList" minOccurs="0"/
↪>
  </xsd:sequence>
  ... 19 attributes ...
</xsd:complexType>

<xsd:complexType name="CT_Hyperlink">
  <xsd:sequence>
    <xsd:element name="snd"    type="CT_EmbeddedWAVAudioFile"   minOccurs="0"/>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute ref="r:id"/>     <!-- type="ST_RelationshipId" -->
  <xsd:attribute name="invalidUrl"     type="xsd:string"        default=""/>
  <xsd:attribute name="action"         type="xsd:string"        default=""/>
  <xsd:attribute name="tgtFrame"       type="xsd:string"        default=""/>
  <xsd:attribute name="tooltip"        type="xsd:string"        default=""/>
```

```
  <xsd:attribute name="history"        type="xsd:boolean"       default="true"/>
  <xsd:attribute name="highlightClick" type="xsd:boolean"       default="false"/>
  <xsd:attribute name="endSnd"         type="xsd:boolean"       default="false"/>
</xsd:complexType>

<xsd:simpleType name="ST_RelationshipId">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
```

Presentation

Presentation properties

Overview

The presentation object has a few interesting properties. Right now I'm interested in slide size.

Protocol

```
>>> assert isinstance(prs, pptx.parts.presentation.PresentationPart)
>>> prs.slide_width
9144000
>>> prs.slide_height
6858000
>>> prs.slide_width = 11887200  # 13 inches
>>> prs.slide_height = 6686550  # 7.3125 inches
>>> prs.slide_width, prs.slide_height
(11887200, 6686550)
```

XML specimens

Example presentation.xml contents:

```
<p:presentation>
  <p:sldMasterIdLst>
    <p:sldMasterId id="2147483897" r:id="rId1"/>
  </p:sldMasterIdLst>
  <p:notesMasterIdLst>
    <p:notesMasterId r:id="rId153"/>
  </p:notesMasterIdLst>
  <p:sldIdLst>
    <p:sldId id="772" r:id="rId3"/>
    <p:sldId id="1244" r:id="rId4"/>
  </p:sldIdLst>
  <p:sldSz cx="9144000" cy="6858000" type="screen4x3"/>
  <p:notesSz cx="6858000" cy="9296400"/>
  <p:defaultTextStyle>
    <a:defPPr>
      <a:defRPr lang="en-US"/>
    </a:defPPr>
    <a:lvl1pPr algn="l" rtl="0" fontAlgn="base">
      <a:spcBef>
```

```
        <a:spcPct val="0"/>
      </a:spcBef>
      <a:spcAft>
        <a:spcPct val="0"/>
      </a:spcAft>
      <a:defRPr kern="1200">
        <a:solidFill>
          <a:schemeClr val="tx1"/>
        </a:solidFill>
        <a:latin typeface="Arial" pitchFamily="34" charset="0"/>
        <a:ea typeface=" " pitchFamily="34" charset="-128"/>
        <a:cs typeface="+mn-cs"/>
      </a:defRPr>
    </a:lvl1pPr>
  </p:defaultTextStyle>
</p:presentation>
```

Related Schema Definitions

```
<xsd:element name="presentation" type="CT_Presentation"/>

<xsd:complexType name="CT_Presentation">
  <xsd:sequence>
    <xsd:element name="sldMasterIdLst"     type="CT_SlideMasterIdList"   minOccurs="0
↪"/>
    <xsd:element name="notesMasterIdLst"   type="CT_NotesMasterIdList"   minOccurs="0
↪"/>
    <xsd:element name="handoutMasterIdLst" type="CT_HandoutMasterIdList" minOccurs="0
↪"/>
    <xsd:element name="sldIdLst"           type="CT_SlideIdList"         minOccurs="0
↪"/>
    <xsd:element name="sldSz"              type="CT_SlideSize"           minOccurs="0
↪"/>
    <xsd:element name="notesSz"            type="a:CT_PositiveSize2D"/>
    <xsd:element name="smartTags"          type="CT_SmartTags"           minOccurs="0
↪"/>
    <xsd:element name="embeddedFontLst"    type="CT_EmbeddedFontList"    minOccurs="0
↪"/>
    <xsd:element name="custShowLst"        type="CT_CustomShowList"      minOccurs="0
↪"/>
    <xsd:element name="photoAlbum"         type="CT_PhotoAlbum"          minOccurs="0
↪"/>
    <xsd:element name="custDataLst"        type="CT_CustomerDataList"    minOccurs="0
↪"/>
    <xsd:element name="kinsoku"            type="CT_Kinsoku"             minOccurs="0
↪"/>
    <xsd:element name="defaultTextStyle"   type="a:CT_TextListStyle"     minOccurs="0
↪"/>
    <xsd:element name="modifyVerifier"     type="CT_ModifyVerifier"      minOccurs="0
↪"/>
    <xsd:element name="extLst"             type="CT_ExtensionList"       minOccurs="0
↪"/>
  </xsd:sequence>
  <xsd:attribute name="serverZoom"              type="a:ST_Percentage"       default=
↪"50%"/>
  <xsd:attribute name="firstSlideNum"           type="xsd:int"               default=
↪"1"/>
```

```
  <xsd:attribute name="showSpecialPlsOnTitleSld" type="xsd:boolean"           default=
↪"true"/>
  <xsd:attribute name="rtl"                      type="xsd:boolean"           default=
↪"false"/>
  <xsd:attribute name="removePersonalInfoOnSave" type="xsd:boolean"           default=
↪"false"/>
  <xsd:attribute name="compatMode"               type="xsd:boolean"           default=
↪"false"/>
  <xsd:attribute name="strictFirstAndLastChars"  type="xsd:boolean"           default=
↪"true"/>
  <xsd:attribute name="embedTrueTypeFonts"       type="xsd:boolean"           default=
↪"false"/>
  <xsd:attribute name="saveSubsetFonts"          type="xsd:boolean"           default=
↪"false"/>
  <xsd:attribute name="autoCompressPictures"     type="xsd:boolean"           default=
↪"true"/>
  <xsd:attribute name="bookmarkIdSeed"           type="ST_BookmarkIdSeed"     default=
↪"1"/>
  <xsd:attribute name="conformance"              type="s:ST_ConformanceClass"/>
</xsd:complexType>

<xsd:complexType name="CT_SlideSize">
  <xsd:attribute name="cx"   type="ST_SlideSizeCoordinate" use="required"/>
  <xsd:attribute name="cy"   type="ST_SlideSizeCoordinate" use="required"/>
  <xsd:attribute name="type" type="ST_SlideSizeType" default="custom"/>
</xsd:complexType>

<xsd:simpleType name="ST_SlideSizeType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="screen4x3"/>
    <xsd:enumeration value="letter"/>
    <xsd:enumeration value="A4"/>
    <xsd:enumeration value="35mm"/>
    <xsd:enumeration value="overhead"/>
    <xsd:enumeration value="banner"/>
    <xsd:enumeration value="custom"/>
    <xsd:enumeration value="ledger"/>
    <xsd:enumeration value="A3"/>
    <xsd:enumeration value="B4ISO"/>
    <xsd:enumeration value="B5ISO"/>
    <xsd:enumeration value="B4JIS"/>
    <xsd:enumeration value="B5JIS"/>
    <xsd:enumeration value="hagakiCard"/>
    <xsd:enumeration value="screen16x9"/>
    <xsd:enumeration value="screen16x10"/>
  </xsd:restriction>
</xsd:simpleType>
```

DrawingML

Color

ColorFormat object ...

Related Schema Definitions

```
<xsd:group name="EG_ColorChoice">
  <xsd:choice>
    <xsd:element name="scrgbClr"  type="CT_ScRgbColor"  minOccurs="1" maxOccurs="1"/>
    <xsd:element name="srgbClr"   type="CT_SRgbColor"   minOccurs="1" maxOccurs="1"/>
    <xsd:element name="hslClr"    type="CT_HslColor"    minOccurs="1" maxOccurs="1"/>
    <xsd:element name="sysClr"    type="CT_SystemColor" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="schemeClr" type="CT_SchemeColor" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="prstClr"   type="CT_PresetColor" minOccurs="1" maxOccurs="1"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_ScRgbColor">
  <xsd:sequence>
    <xsd:group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="r" type="ST_Percentage" use="required"/>
  <xsd:attribute name="g" type="ST_Percentage" use="required"/>
  <xsd:attribute name="b" type="ST_Percentage" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_SRgbColor">
  <xsd:sequence>
    <xsd:group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="val" type="s:ST_HexColorRGB" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_HslColor">
  <xsd:sequence>
    <xsd:group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="hue" type="ST_PositiveFixedAngle" use="required"/>
  <xsd:attribute name="sat" type="ST_Percentage"         use="required"/>
  <xsd:attribute name="lum" type="ST_Percentage"         use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_SystemColor">
  <xsd:sequence>
    <xsd:group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="val"     type="ST_SystemColorVal" use="required"/>
  <xsd:attribute name="lastClr" type="s:ST_HexColorRGB"  use="optional"/>
</xsd:complexType>

<xsd:complexType name="CT_SchemeColor">
  <xsd:sequence>
    <xsd:group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="val" type="ST_SchemeColorVal" use="required"/>
</xsd:complexType>

<xsd:complexType name="CT_PresetColor">
  <xsd:sequence>
    <xsd:group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="val" type="ST_PresetColorVal" use="required"/>
```

```
</xsd:complexType>

<xsd:group name="EG_ColorTransform">
  <xsd:choice>
    <xsd:element name="tint"     type="CT_PositiveFixedPercentage" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="shade"    type="CT_PositiveFixedPercentage" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="comp"     type="CT_ComplementTransform"     minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="inv"      type="CT_InverseTransform"        minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="gray"     type="CT_GrayscaleTransform"      minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="alpha"    type="CT_PositiveFixedPercentage" minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="alphaOff" type="CT_FixedPercentage"         minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="alphaMod" type="CT_PositivePercentage"      minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="hue"      type="CT_PositiveFixedAngle"      minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="hueOff"   type="CT_Angle"                   minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="hueMod"   type="CT_PositivePercentage"      minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="sat"      type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="satOff"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="satMod"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="lum"      type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="lumOff"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="lumMod"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="red"      type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="redOff"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="redMod"   type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="green"    type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="greenOff" type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="greenMod" type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blue"     type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blueOff"  type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="blueMod"  type="CT_Percentage"              minOccurs="1"␣
↪maxOccurs="1"/>
    <xsd:element name="gamma"    type="CT_GammaTransform"          minOccurs="1"␣
↪maxOccurs="1"/>
```

```
    <xsd:element name="invGamma" type="CT_InverseGammaTransform"   minOccurs="1"␣
↪maxOccurs="1"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_Percentage">
  <xsd:attribute name="val" type="ST_Percentage" use="required"/>
</xsd:complexType>

<xsd:simpleType name="ST_Percentage">
  <xsd:union memberTypes="ST_PercentageDecimal s:ST_Percentage"/>
</xsd:simpleType>

<xsd:simpleType name="ST_PercentageDecimal">
  <xsd:restriction base="xsd:int"/>
</xsd:simpleType>

<xsd:simpleType name="s:ST_Percentage">  <!-- denormalized -->
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="-?[0-9]+(\.[0-9]+)?%"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_HexColorRGB">
  <xsd:restriction base="xsd:hexBinary">
    <xsd:length value="3" fixed="true"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_SchemeColorVal">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="bg1"/>
    <xsd:enumeration value="tx1"/>
    <xsd:enumeration value="bg2"/>
    <xsd:enumeration value="tx2"/>
    <xsd:enumeration value="accent1"/>
    <xsd:enumeration value="accent2"/>
    <xsd:enumeration value="accent3"/>
    <xsd:enumeration value="accent4"/>
    <xsd:enumeration value="accent5"/>
    <xsd:enumeration value="accent6"/>
    <xsd:enumeration value="hlink"/>
    <xsd:enumeration value="folHlink"/>
    <xsd:enumeration value="phClr"/>
    <xsd:enumeration value="dk1"/>
    <xsd:enumeration value="lt1"/>
    <xsd:enumeration value="dk2"/>
    <xsd:enumeration value="lt2"/>
  </xsd:restriction>
</xsd:simpleType>
```

Fill (for shapes)

Overview

In simple terms, *fill* is the color of a shape. The term *fill* is used to distinguish from *line*, the border around a shape, which can have a different color. It turns out that a line has a fill all its own, but that's another topic.

While having a solid color fill is perhaps most common, it is just one of several possible fill types. A shape can have the following types of fill:

- solid (color)
- gradient – a smooth transition between multiple colors
- picture – in which an image "shows through" and is cropped by the boundaries of the shape
- pattern – in which a small square of pixels (tile) is repeated to fill the shape
- background (no fill) – the body of the shape is transparent, allowing whatever is behind it to show through.

Elements that can have a fill include autoshape, line, text (treating each glyph as a shape), table, table cell, and slide background.

Scope

This analysis focuses on solid fill for an autoshape (including text box). This relatively narrow initial focus is to enable getting to feature implementation as quickly as possible, while understanding enough to design the top-level of the API in a way that will allow other fill types to be added incrementally.

Minimum viable feature

Start with solid fill since that's the most frequently asked for and it reuses the work completed on ColorFormat for font color feature:

```
fill = shape.fill
fill.solid()
fill.fore_color.rgb = RGBColor(0x01, 0x23, 0x45)
fill.fore_color.theme_color = MSO_THEME_COLOR.ACCENT_1
fill.fore_color.brightness = 0.25
fill.transparency = 0.25
shape.fill = None
fill.background()  # is almost free once the rest is in place
```

Protocol

Accessing fill. A shape object unconditionally has a *FillFormat* object on *.fill*. The *.fill* property is idempotent; it always returns the same *FillFormat* object for a given shape object:

```
>>> fill = shape.fill
>>> assert(isinstance(fill, FillFormat)
```

Fill type. A fill has a type, which may be `None`. The fill type partially determines the valid calls on the fill:

```
>>> fill.type
None
>>> fill.solid()
>>> fill.type
```

```
1  # MSO_FILL.SOLID

>>> fill.fore_color = 'anything'
AttributeError: can't set attribute  # .fore_color is read-only
>>> fore_color = fill.fore_color
>>> assert(isinstance(fore_color, ColorFormat))

>>> fore_color.rgb = RGBColor(0x3F, 0x2c, 0x36)
>>> fore_color.theme_color = MSO_THEME_COLOR.ACCENT_1
>>> fore_color.brightness = -0.25

>>> fill.transparency
0.0
>>> fill.transparency = 0.25  # sets opacity to 75%

>>> sp.fill = None  # removes any fill, fill is inherited from theme
```

Pattern Fill.

```
>>> fill = shape.fill
>>> fill.type
None
>>> fill.patterned()
>>> fill.type
2  # MSO_FILL.PATTERNED
>>> fill.fore_color.rgb = RGBColor(79, 129, 189)
>>> fill.back_color.rgb = RGBColor(239, 169, 6)
```

fill type get/set

- `FillFormat.type` – `MSO_FILL.SOLID` or `None` for a start
- `FillFormat.solid()` – changes fill type to <a:solidFill>
- `FillFormat.fore_color` – changes fill type to <a:solidFill>
- `FillFormat.transparency` – adds/changes <a:alpha> or something
- MSDN FillFormat Object

Enumerations

MsoFillType

http://msdn.microsoft.com/EN-US/library/office/ff861408.aspx

msoFillBackground 5 – Fill is the same as the background.

msoFillGradient 3 – Gradient fill.

msoFillMixed -2 – Mixed fill.

msoFillPatterned 2 – Patterned fill.

msoFillPicture 6 – Picture fill.

msoFillSolid 1 – Solid fill.

msoFillTextured 4 – Textured fill.

XML specimens

Inherited fill on autoshape:

```
<p:spPr>
   ...
  <a:prstGeom prst="roundRect">
    <a:avLst/>
  </a:prstGeom>
</p:spPr>
```

Solid RGB color on autoshape:

```
<p:spPr>
   ...
  <a:prstGeom prst="roundRect">
    <a:avLst/>
  </a:prstGeom>
  <a:solidFill>
    <a:srgbClr val="2CB731"/>
  </a:solidFill>
</p:spPr>
```

Patterned fill:

```
<a:pattFill prst="ltDnDiag">
  <a:fgClr>
    <a:schemeClr val="accent1"/>
  </a:fgClr>
  <a:bgClr>
    <a:schemeClr val="accent6"/>
  </a:bgClr>
</a:pattFill>
```

XML semantics

- **No 'prst' attribute.** When an *a:pattFill* element contains no *prst* attribute, the pattern defaults to 5% (dotted). This is the first one in the pattern gallery in the PowerPoint UI.
- **No 'fgClr' or 'bgClr' elements.** When an *a:pattFill* element contains no *fgClr* or *bgClr* chile elements, the colors default to black and white respectively.

Related Schema Definitions

```
<xsd:complexType name="CT_ShapeProperties">
  <xsd:sequence>
    <xsd:element name="xfrm"    type="CT_Transform2D"            minOccurs="0"/>
    <xsd:group   ref="EG_Geometry"                               minOccurs="0"/>
    <xsd:group   ref="EG_FillProperties"                         minOccurs="0"/>
    <xsd:element name="ln"      type="CT_LineProperties"         minOccurs="0"/>
    <xsd:group   ref="EG_EffectProperties"                       minOccurs="0"/>
    <xsd:element name="scene3d" type="CT_Scene3D"                minOccurs="0"/>
    <xsd:element name="sp3d"    type="CT_Shape3D"                minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
```

```
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode"/>
</xsd:complexType>

<xsd:group name="EG_Geometry">
  <xsd:choice>
    <xsd:element name="custGeom" type="CT_CustomGeometry2D"/>
    <xsd:element name="prstGeom" type="CT_PresetGeometry2D"/>
  </xsd:choice>
</xsd:group>

<xsd:group name="EG_FillProperties">
  <xsd:choice>
    <xsd:element name="noFill"    type="CT_NoFillProperties"/>
    <xsd:element name="solidFill" type="CT_SolidColorFillProperties"/>
    <xsd:element name="gradFill"  type="CT_GradientFillProperties"/>
    <xsd:element name="blipFill"  type="CT_BlipFillProperties"/>
    <xsd:element name="pattFill"  type="CT_PatternFillProperties"/>
    <xsd:element name="grpFill"   type="CT_GroupFillProperties"/>
  </xsd:choice>
</xsd:group>

<xsd:group name="EG_EffectProperties">
  <xsd:choice>
    <xsd:element name="effectLst" type="CT_EffectList"/>
    <xsd:element name="effectDag" type="CT_EffectContainer"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_BlipFillProperties">
  <xsd:sequence>
    <xsd:element name="blip"    type="CT_Blip"         minOccurs="0"/>
    <xsd:element name="srcRect" type="CT_RelativeRect" minOccurs="0"/>
    <xsd:choice minOccurs="0">  <!-- EG_FillModeProperties -->
      <xsd:element name="tile"    type="CT_TileInfoProperties"/>
      <xsd:element name="stretch" type="CT_StretchInfoProperties"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="dpi"          type="xsd:unsignedInt"/>
  <xsd:attribute name="rotWithShape" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_GradientFillProperties">
  <xsd:sequence>
    <xsd:element name="gsLst" type="CT_GradientStopList" minOccurs="0"/>
    <xsd:choice minOccurs="0">  <!-- EG_ShadeProperties -->
      <xsd:element name="lin"  type="CT_LinearShadeProperties"/>
      <xsd:element name="path" type="CT_PathShadeProperties"/>
    </xsd:choice>
    <xsd:element name="tileRect" type="CT_RelativeRect" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="flip"         type="ST_TileFlipMode"/>
  <xsd:attribute name="rotWithShape" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="CT_GroupFillProperties"/>

<xsd:complexType name="CT_NoFillProperties"/>
```

```
<xsd:complexType name="CT_PatternFillProperties">
  <xsd:sequence>
    <xsd:element name="fgClr" type="CT_Color" minOccurs="0"/>
    <xsd:element name="bgClr" type="CT_Color" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="prst" type="ST_PresetPatternVal"/>
</xsd:complexType>

<xsd:complexType name="CT_Color">
  <xsd:sequence>
    <xsd:group ref="EG_ColorChoice"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CT_SolidColorFillProperties">
  <xsd:sequence>
    <xsd:group ref="EG_ColorChoice" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:group name="EG_ColorChoice">
  <xsd:choice>
    <xsd:element name="scrgbClr"  type="CT_ScRgbColor"/>
    <xsd:element name="srgbClr"   type="CT_SRgbColor"/>
    <xsd:element name="hslClr"    type="CT_HslColor"/>
    <xsd:element name="sysClr"    type="CT_SystemColor"/>
    <xsd:element name="schemeClr" type="CT_SchemeColor"/>
    <xsd:element name="prstClr"   type="CT_PresetColor"/>
  </xsd:choice>
</xsd:group>

<xsd:simpleType name="ST_PresetPatternVal">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="pct5"/>
    <xsd:enumeration value="pct10"/>
    <xsd:enumeration value="pct20"/>
    <xsd:enumeration value="pct25"/>
    <xsd:enumeration value="pct30"/>
    <xsd:enumeration value="pct40"/>
    <xsd:enumeration value="pct50"/>
    <xsd:enumeration value="pct60"/>
    <xsd:enumeration value="pct70"/>
    <xsd:enumeration value="pct75"/>
    <xsd:enumeration value="pct80"/>
    <xsd:enumeration value="pct90"/>
    <xsd:enumeration value="horz"/>
    <xsd:enumeration value="vert"/>
    <xsd:enumeration value="ltHorz"/>
    <xsd:enumeration value="ltVert"/>
    <xsd:enumeration value="dkHorz"/>
    <xsd:enumeration value="dkVert"/>
    <xsd:enumeration value="narHorz"/>
    <xsd:enumeration value="narVert"/>
    <xsd:enumeration value="dashHorz"/>
    <xsd:enumeration value="dashVert"/>
    <xsd:enumeration value="cross"/>
    <xsd:enumeration value="dnDiag"/>
    <xsd:enumeration value="upDiag"/>
```

```
    <xsd:enumeration value="ltDnDiag"/>
    <xsd:enumeration value="ltUpDiag"/>
    <xsd:enumeration value="dkDnDiag"/>
    <xsd:enumeration value="dkUpDiag"/>
    <xsd:enumeration value="wdDnDiag"/>
    <xsd:enumeration value="wdUpDiag"/>
    <xsd:enumeration value="dashDnDiag"/>
    <xsd:enumeration value="dashUpDiag"/>
    <xsd:enumeration value="diagCross"/>
    <xsd:enumeration value="smCheck"/>
    <xsd:enumeration value="lgCheck"/>
    <xsd:enumeration value="smGrid"/>
    <xsd:enumeration value="lgGrid"/>
    <xsd:enumeration value="dotGrid"/>
    <xsd:enumeration value="smConfetti"/>
    <xsd:enumeration value="lgConfetti"/>
    <xsd:enumeration value="horzBrick"/>
    <xsd:enumeration value="diagBrick"/>
    <xsd:enumeration value="solidDmnd"/>
    <xsd:enumeration value="openDmnd"/>
    <xsd:enumeration value="dotDmnd"/>
    <xsd:enumeration value="plaid"/>
    <xsd:enumeration value="sphere"/>
    <xsd:enumeration value="weave"/>
    <xsd:enumeration value="divot"/>
    <xsd:enumeration value="shingle"/>
    <xsd:enumeration value="wave"/>
    <xsd:enumeration value="trellis"/>
    <xsd:enumeration value="zigZag"/>
  </xsd:restriction>
</xsd:simpleType>
```

Line format

Intial scope

- Shape.line, Connector.line to follow, then Cell.border(s).line, possibly text underline and font.line (character outline)

Protocol

```
>>> shape.line
<pptx.dml.line.LineFormat instance at x123456789>
>>> shape.fore_color
<pptx.dml.line.LineFormat instance at x123456789>
```

Notes

definitely spike on this one.

start with line.color, first an object to hang members off of

Issue: How to accommodate these competing requirements:

- Polymorphism in parent shape. More than one type of shape can have a line and possibly several (table cell border, Run/text, and text underline use the same CT_LineProperties element, and perhaps others).
- The line object cannot hold onto a *<a:ln>* element, even if that was a good idea, because it is an optional child; not having an *<a:ln>* element is a legitimate and common situation, indicating line formatting should be inherited from the theme or perhaps a layout placeholder.
- Needing to accommodate XML changing might not be important, could make that operation immutable, such that changing the shape XML returns a new shape, not changing the existing one in-place.
- maybe having the following 'line_format_owner_interface', delegating create, read, and delete of the *<a:ln>* element to the parent, and allowing LineFormat to take responsibility for update.
  - line.parent has the shape having the line format
  - parent.ln has the *<a:ln>* element or None, delegating access to the parent
  - ln = parent.add_ln()
  - parent.remove_ln()

MS API

- LineFormat
  https://msdn.microsoft.com/en-us/vba/powerpoint-vba/articles/lineformat-object-powerpoint
- LineFormat.DashStyle
  https://msdn.microsoft.com/en-us/vba/powerpoint-vba/articles/lineformat-dashstyle-property-powerpoint
- MsoLineDashStyle Enumeration
  https://msdn.microsoft.com/en-us/library/aa432639(v=office.12).aspx

Name	Value	Description
msoLineDash	4	Line consists of dashes only.
msoLineDashDot	5	Line is a dash-dot pattern.
msoLineDashDotDot	6	Line is a dash-dot-dot pattern.
msoLineDashStyleMixed	-2	Not supported.
msoLineLongDash	7	Line consists of long dashes.
msoLineLongDashDot	8	Line is a long dash-dot pattern.
msoLineRoundDot	3	Line is made up of round dots.
msoLineSolid	1	Line is solid.
msoLineSquareDot	2	Line is made up of square dots.

Specimen XML

solid line color:

```
<p:spPr>
  <a:xfrm>
    <a:off x="950964" y="2925277"/>
    <a:ext cx="1257921" cy="619967"/>
  </a:xfrm>
  <a:prstGeom prst="curvedConnector3">
    <a:avLst/>
  </a:prstGeom>
  <a:ln>
    <a:solidFill>
```

```
      <a:schemeClr val="accent2"/>
    </a:solidFill>
  </a:ln>
</p:spPr>
```

little bit of everything:

```
<p:spPr>
  <a:xfrm>
    <a:off x="950964" y="1101493"/>
    <a:ext cx="1257921" cy="0"/>
  </a:xfrm>
  <a:prstGeom prst="line">
    <a:avLst/>
  </a:prstGeom>
  <a:ln w="57150" cap="rnd" cmpd="thickThin">
    <a:gradFill flip="none" rotWithShape="1">
      <a:gsLst>
        <a:gs pos="0">
          <a:schemeClr val="accent1"/>
        </a:gs>
        <a:gs pos="100000">
          <a:srgbClr val="FFFFFF"/>
        </a:gs>
      </a:gsLst>
      <a:lin ang="0" scaled="1"/>
      <a:tileRect/>
    </a:gradFill>
    <a:prstDash val="sysDash"/>
    <a:bevel/>
    <a:headEnd type="oval"/>
    <a:tailEnd type="diamond"/>
  </a:ln>
</p:spPr>
```

Related Schema Definitions

```
<xsd:complexType name="CT_ShapeProperties">
  <xsd:sequence>
    <xsd:element name="xfrm"    type="CT_Transform2D"           minOccurs="0"/>
    <xsd:group   ref="EG_Geometry"                              minOccurs="0"/>
    <xsd:group   ref="EG_FillProperties"                        minOccurs="0"/>
    <xsd:element name="ln"      type="CT_LineProperties"        minOccurs="0"/>
    <xsd:group   ref="EG_EffectProperties"                      minOccurs="0"/>
    <xsd:element name="scene3d" type="CT_Scene3D"               minOccurs="0"/>
    <xsd:element name="sp3d"    type="CT_Shape3D"               minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</xsd:complexType>

<xsd:complexType name="CT_LineProperties">
  <xsd:sequence>
    <xsd:group   ref="EG_LineFillProperties"                    minOccurs="0"/>
    <xsd:group   ref="EG_LineDashProperties"                    minOccurs="0"/>
    <xsd:group   ref="EG_LineJoinProperties"                    minOccurs="0"/>
```

```
    <xsd:element name="headEnd" type="CT_LineEndProperties"      minOccurs="0"/>
    <xsd:element name="tailEnd" type="CT_LineEndProperties"      minOccurs="0"/>
    <xsd:element name="extLst"  type="CT_OfficeArtExtensionList" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="w"    type="ST_LineWidth"/>
  <xsd:attribute name="cap"  type="ST_LineCap"/>
  <xsd:attribute name="cmpd" type="ST_CompoundLine"/>
  <xsd:attribute name="algn" type="ST_PenAlignment"/>
</xsd:complexType>

<xsd:group name="EG_LineFillProperties">
  <xsd:choice>
    <xsd:element name="noFill"    type="CT_NoFillProperties"/>
    <xsd:element name="solidFill" type="CT_SolidColorFillProperties"/>
    <xsd:element name="gradFill"  type="CT_GradientFillProperties"/>
    <xsd:element name="pattFill"  type="CT_PatternFillProperties"/>
  </xsd:choice>
</xsd:group>

<xsd:group name="EG_LineDashProperties">
  <xsd:choice>
    <xsd:element name="prstDash" type="CT_PresetLineDashProperties"/>
    <xsd:element name="custDash" type="CT_DashStopList"/>
  </xsd:choice>
</xsd:group>

<xsd:complexType name="CT_PresetLineDashProperties">
  <xsd:attribute name="val" type="ST_PresetLineDashVal" use="optional"/>
</xsd:complexType>

<xsd:group name="EG_LineJoinProperties">
  <xsd:choice>
    <xsd:element name="round" type="CT_LineJoinRound"/>
    <xsd:element name="bevel" type="CT_LineJoinBevel"/>
    <xsd:element name="miter" type="CT_LineJoinMiterProperties"/>
  </xsd:choice>
</xsd:group>

<xsd:group name="EG_EffectProperties">
  <xsd:choice>
    <xsd:element name="effectLst" type="CT_EffectList"/>
    <xsd:element name="effectDag" type="CT_EffectContainer"/>
  </xsd:choice>
</xsd:group>

<xsd:simpleType name="ST_LineWidth">
  <xsd:restriction base="ST_Coordinate32Unqualified">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="20116800"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ST_Coordinate32Unqualified">
  <xsd:restriction base="xsd:int"/>
</xsd:simpleType>

<xsd:simpleType name="ST_PresetLineDashVal">
  <xsd:restriction base="xsd:token">
```

```
    <xsd:enumeration value="solid"/>
    <xsd:enumeration value="dot"/>
    <xsd:enumeration value="dash"/>
    <xsd:enumeration value="lgDash"/>
    <xsd:enumeration value="dashDot"/>
    <xsd:enumeration value="lgDashDot"/>
    <xsd:enumeration value="lgDashDotDot"/>
    <xsd:enumeration value="sysDash"/>
    <xsd:enumeration value="sysDot"/>
    <xsd:enumeration value="sysDashDot"/>
    <xsd:enumeration value="sysDashDotDot"/>
  </xsd:restriction>
</xsd:simpleType>
```

Package

Core Document Properties

Updated 2013-06-22

Author Steve Canny

Status **WORKING DRAFT**

Introduction

The 'Core' in core document properties refers to Dublin Core, a metadata standard that defines a core set of elements to describe resources.

ISO/IEC 29500-2 Section 11

API Sketch

Presentation.core_properties

_CoreProperties():

All string values are limited to 255 chars (unicode chars, not bytes)

- title
- subject
- author
- keywords
- comments
- last_modified_by
- revision (integer)
- created (date in format: '2013-04-06T06:03:36Z')
- modified (date in same format)
- category

Status and company (and many others) are custom properties, held in `app.xml`.

XML produced by PowerPoint® client

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cp:coreProperties xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/
↪core-properties" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcterms="http://
↪purl.org/dc/terms/" xmlns:dcmitype="http://purl.org/dc/dcmitype/" xmlns:xsi="http://
↪www.w3.org/2001/XMLSchema-instance">
  <dc:title>Core Document Properties Exploration</dc:title>
  <dc:subject>PowerPoint core document properties</dc:subject>
  <dc:creator>Steve Canny</dc:creator>
  <cp:keywords>powerpoint; open xml; dublin core; microsoft office</cp:keywords>
  <dc:description>One thing I'd like to discover is just how line wrapping is handled␣
↪in the comments. This paragraph is all on a single line._x000d__x000d_This is a␣
↪second paragraph separated from the first by two line feeds.</dc:description>
  <cp:lastModifiedBy>Steve Canny</cp:lastModifiedBy>
  <cp:revision>2</cp:revision>
  <dcterms:created xsi:type="dcterms:W3CDTF">2013-04-06T06:03:36Z</dcterms:created>
  <dcterms:modified xsi:type="dcterms:W3CDTF">2013-06-15T06:09:18Z</dcterms:modified>
  <cp:category>analysis</cp:category>
</cp:coreProperties>
```

Schema

```
<xs:schema
  targetNamespace="http://schemas.openxmlformats.org/package/2006/metadata/core-
↪properties"
  xmlns="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  elementFormDefault="qualified"
  blockDefault="#all">

  <xs:import
    namespace="http://purl.org/dc/elements/1.1/"
    schemaLocation="http://dublincore.org/schemas/xmls/qdc/2003/04/02/dc.xsd"/>
  <xs:import
    namespace="http://purl.org/dc/terms/"
    schemaLocation="http://dublincore.org/schemas/xmls/qdc/2003/04/02/dcterms.xsd"/>
  <xs:import
    id="xml"
    namespace="http://www.w3.org/XML/1998/namespace"/>

  <xs:element name="coreProperties" type="CT_CoreProperties"/>

  <xs:complexType name="CT_CoreProperties">
    <xs:all>
      <xs:element name="category"        type="xs:string"   minOccurs="0"/>
      <xs:element name="contentStatus"   type="xs:string"   minOccurs="0"/>
      <xs:element ref="dcterms:created"                     minOccurs="0"/>
      <xs:element ref="dc:creator"                          minOccurs="0"/>
      <xs:element ref="dc:description"                      minOccurs="0"/>
      <xs:element ref="dc:identifier"                       minOccurs="0"/>
```

```
      <xs:element name="keywords"        type="CT_Keywords" minOccurs="0"/>
      <xs:element ref="dc:language"                         minOccurs="0"/>
      <xs:element name="lastModifiedBy"  type="xs:string"   minOccurs="0"/>
      <xs:element name="lastPrinted"     type="xs:dateTime" minOccurs="0"/>
      <xs:element ref="dcterms:modified"                    minOccurs="0"/>
      <xs:element name="revision"        type="xs:string"   minOccurs="0"/>
      <xs:element ref="dc:subject"                          minOccurs="0"/>
      <xs:element ref="dc:title"                            minOccurs="0"/>
      <xs:element name="version"         type="xs:string"   minOccurs="0"/>
    </xs:all>
  </xs:complexType>

  <xs:complexType name="CT_Keywords" mixed="true">
    <xs:sequence>
      <xs:element name="value" minOccurs="0" maxOccurs="unbounded" type="CT_Keyword"/>
    </xs:sequence>
    <xs:attribute ref="xml:lang" use="optional"/>
  </xs:complexType>

  <xs:complexType name="CT_Keyword">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="xml:lang" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

</xs:schema>
```

Enumerations

Overview

A large number of settings in PowerPoint are a selection between a small, finite set of discrete choices. Often, in the XML, these are represented by a simple type defined as an enumeration of a set of short string values. This page describes the strategy for providing access to these sets of values in the various parts of the library that require them.

Notes

- The general strategy is to provide a set of classes, one for each enumeration, that serves as the single source for all reference and behavior for that enumeration.
- These enumerations correspond one-to-one with Microsoft API enumerations.
- The naming is based on the Microsoft enumeration. .. all-caps snake case from the original mixed-case.

Definitions

member What constitutes a member of an enumeration? What attributes do all members have in common?

out-of-band member Is there such a thing? Or is there just an out-of-band value?

get-value ... value returned to indicate the current state of a property

set-value ... value passed to indicate the desired state of a property

Feature requirements

Design principle: All required values and mappings are defined in the enumeration class. All required names, data structures, and methods are generated by the metaclass. They don't have to be generated manually.

- [X] feature: dotted name access

  enumeration values can be accessed from an enumeration class with dotted notation, e.g. `MSO_FOO.BAR_BAZ`

- [X] feature: enumeration value behaves as an int

- [X] feature: __str__ value of enum member is the member name and int value

  In order to provide the developer with a directly readable name for an enumeration value (rather than a raw int), each enum values shall have a __str__ method that returns the member name. E.g.:

```
>>> print("text_frame.auto_size is '%s'" % text_frame.auto_size)
text_frame.auto_size is 'SHAPE_TO_FIT_TEXT (2)'
```

- [X] feature: out-of-band values – like None, True, and False

  From time to time it is desireable to allow return and setting values that are not named members of the enumeration to provide a more Pythonic protocol for a property that uses the enumeration. For example:

```
>>> run.underline
None
>>> run.underline = True
>>> str(run.underline)
'SINGLE'
>>> run.underline = False
>>> str(run.underline)
'NONE'
>>> run.underline = WD_UNDERLINE.DOUBLE
>>> str(run.underline)
'DOUBLE'
```

- [X] feature: alias decorator

  one or more convenience aliases can be declared for an Enumeration subclass by use of an @alias decorator

- [X] feature: setting validation

  In order to allow invalid input values to raise ValueError exceptions at the API method level, the enumeration shall provide an is_valid_setting() class method that returns False if the value passed as an argument is not a member of the enumeration. E.g.:

```
assert MSO_AUTO_SIZE.is_member('foobar') is False
```

- [X] feature: to_XML translation
  - **[X] issue: Not all enumerations have mappings to XML attribute values.** Consider creating an XmlEnumeration subclass that provides this extra feature.
  - **[X] issue: There exist 'return-only' enumeration values, out-of-band**

values used to indicate none of the other values apply, for example MIXED = -2 to indicate the call was made on a range that contains items not all having the same value.

To provide a single authoritative mapping from enumeration member values and the XML attribute values they correspond to, the Enumeration class shall provide a to_xml(member_value) method that returns the XML value corresponding to the passed member value.

The method shall raise ValueError if the passed value is not a member of the enumeration.

The special XML value `None` indicates the attribute should be removed altogether and should be returned where that is the appropriate behavior.

Where the Python value is an out-of-band value such as `None`, `True`, or `False`, there must be a valid XML value or `None` as the mapping.

- [X] feature: docstrings

  To provide built-in documentation for different options and to provide a source for option descriptions for a generated documentation page, each enumeration member shall have as its docstring a description of the member and the behavior that setting specifies in its context.

- [X] feature: auto-generate documentation
  - [X] add __ms_name__ var => 'MsoAutoSize'
  - [X] add __url__ var => 'http:// ...'

    In order to provide single-source for both code and documentation, the Enumeration class shall provide a documentation page generation method that generates Sphinx RST source for a documentation page for the enumeration. This page source shall be available as either the docstring (__doc__) value of the class or its __str__ value. The supplied docstring shall be pre-pended to the generated portion after the top matter such as page title and reference link are generated.

- [X] convert all existing enums to new type to make sure they all work
  - [X] MSO_AUTO_SIZE
  - [X] MSO_VERTICAL_ANCHOR (__init__ -> enum.text)
  - [X] MSO_FILL_TYPE (__init__ -> enum.dml)
  - [X] MSO_COLOR_TYPE (__init__ -> enum.dml)
  - [X] others ...
  - [>] PP_PARAGRAPH_ALIGNMENT (constants -> enum.text)
  - [X] TEXT_ALIGN_TYPE (constants -> delete)
  - [X] TEXT_ANCHORING_TYPE (constants -> delete)
  - [X] MSO_SHAPE_TYPE (constants.MSO -> enum.shapes.MSO_SHAPE_TYPE)
  - [X] MSO.ANCHOR_* (constants -> MSO_VERTICAL_ANCHOR)
  - [X] PP_PLACEHOLDER_TYPE: PH_TYPE_* in spec.py PpPlaceholderType
  - [X] ST_Direction: PH_ORIENT_*
  - [X] ST_PlaceholderSize: PH_SZ_*
  - [X] MSO_AUTO_SHAPE_TYPE (constants -> enum.shapes)
  - **[X] What is pml_parttypes doing in spec? I thought that went a long time** ago? must be dead code now.
  - [X] fix all enum references in documentation

  - [X] adjust all examples that use enum or constants
  - [X] spec: pretty sure default content types belong in opc somewhere

- **[ ] ... starting to think that combining to_xml() and from_xml() into enums** is not the best idea:
  - **[ ] There are two separate concerns: (1) provide a symbolic reference for** use in the API to specify an enumerated option. (2) provide one of potentially several mappings between that value and values in different domains.
  - **[ ] could create custom Enumeration subclasses to accommodate special** cases where there are mappings beyond to and from an XML attribute value.
  - **[ ] could create value objects that perhaps encapsulate those mappings.** I suppose that's what Enumeration is now.
- **[X] issue: does to/from XML translation replace an ST_* class in oxml?** (Probably can't fully replace, need validation and other bits as well. Might be best to wait until a few examples have accumulated.)

  In order to support DRY translation between enumeration values and XML simple type (ST_*) attribute values, the Enumeration class shall provide both to_xml(enum_value) and from_xml(attr_value) methods where enumeration values map to XML attribute values.
- **[ ] issue: how handle case where more than one XML to name mapping is** possible? Like both True and SINGLE set underline to 'single'? Where if anywhere does that cause a conflict?

  The other direction of this is when more than one XML value should map to None.
- **[ ] issue: how add clsdict._xml_to_member dict unconditionally to** subclasses of XmlEnumeration? use __init__ instead of __new__ for that part:

```
def __init__(cls, clsname, bases, clsdict):
    cls._collect_valid_settings = []
```

  - init time might be too late, no handy ref to metaclass?
  - could catch __new__ on separate XmlMappedEnumMember metaclass and add the attribute(s) then.
- [ ] deprecate prior versions:
  - [ ] MSO_VERTICAL_ANCHOR from enum/__init__.py
- **[ ] consider adding a feature whereby aliases can be defined for an enum** member name
- **[ ] consider what requirements an ST_* enumeration might have, aside from** or in addition to an API enumeration
- **[ ] need a page or whatever in the user guide to talk about how fills are** applied and give example code
- **[ ] how could enumeration documentation be generated automatically from** enumerations as part of the Sphinx run?

Code Sketches

enabling an @alias decorator to define aliases for an enumeration class:

```
#!/usr/bin/env python

from __future__ import absolute_import, print_function


def alias(*aliases):
```

```
    """
    Decorating a class with @alias('FOO', 'BAR', ..) allows the class to
    be referenced by each of the names provided as arguments.
    """
    def decorator(cls):
        for alias in aliases:
            globals()[alias] = cls
        return cls
    return decorator


@alias('BAR', 'BAZ')
class FOO(object):
    BAR = 'Foobarish'


print("FOO.BAR => '%s'" % FOO.BAR)
print("BAR.BAR => '%s'" % BAR.BAR)  # noqa
print("BAZ.BAR => '%s'" % BAZ.BAR)  # noqa
```

Major discovery sources

- ISO/IEC 29500 Open XML specification
- XSD schemas for Open XML
- MSDN PowerPoint® API documentation
- Inspection of XML generated by PowerPoint® client
- Inspection of PowerPoint® client GUI

Resources

... just collected bits and pieces here for now ...

Python packaging

Process Steps

1. Develop **setup.py** and other *distribution-related files*
2. Test distribution; should pass all *distribution tests*
3. Register project with the Python Package Index (PyPI)
4. Upload distribution(s) to PyPI

Distribution tests

- `python setup.py sdist` does not raise an exception
- all expected files are included in the distribution tarball
- `python setup.py test` works in install environment

- acceptance tests pass in install environment
- `python setup.py install` produces expected footprint in `site-packages`
- easy_install works
- `pip install python-pptx` works

Test can install with all popular methods

- manual
- easy_install
- pip

Distribution-related files

- setup.py
- MANIFEST.in
- setup.cfg

Distribution user stories

... some notions about who uses these and for what ...

Roles

- naive end-user

Use Cases

Test build before distribution

"Just-works" installation

```
In order to enable a new capability in my computing environment
As a naive end-user
I would like installation to "just work" and not scare me with error
messages that don't indicate a real problem.
```

Install as a dependency

Verify installation

```
In order to verify a new installation
As a python developer
I want to be able to easily run the test suite without having to invest in
any additional discovery or configuration.
```

Resources

- The Hitchhiker's Guide to Packaging
- Writing a Package in Python by Tarek Ziadé is an extract from his PACKT book *Expert Python Programming* and while being somewhat dated, contains some useful tidbits.
- Ian Bicking's blog post Python's Makefile discusses how to write extensions to setup.py, for perhaps a command like `coverage` that would automatically run `nosetests --with-coverage`.
- tox documentation
- virtualenv documentation
- How To Package Your Python Code
- Python Packaging: Hate, hate, hate everywhere
- Building and Distributing Packages with setuptools
- A guide to Python packaging
- Python Packaging by Tarek Ziade

About Open XML Packaging

Recent Notes

- The content type for **XML** parts (only) is strictly determined by the relationship type. Binary media parts may have multiple allowable content types.
- Each part type has one and only one relationship type.

About Packages

Open XML PowerPoint files are stored in files with the extension `.pptx`. These `.pptx` files are zip archives containing a separate file for each main component of the presentation along with other files which contain metadata about the overall presentation and relationships between its components.

The overall collection of files is known as a **package**. Each file within the package represents a **package item**, commonly although sometimes ambiguously referred to simply as an **item**. Package items that represent high-level presentation components, such as slides, slide masters, and themes, are referred to as **parts**. All parts are package items, but not all package items are parts.

Package items that are not parts are primarily **relationship items** that express relationships between one part and another. Examples of relationships include a slide having a relationship to the slide layout it is based on and a slide master's relationship to an image such as a logo it displays. There is one special relationship item, the **package relationship item** (`/_rels/.rels`) which contains relationships between the package and certain parts rather than relationships between parts.

The only other package item is the **content types item** (`/[Content_Types].xml`) which contains mappings of the parts to their content types (roughly MIME types).

Package Loading Strategies

Strategy 1

The content type item ([Content_Types].xml actually contains references to all the main parts of the package. So one approach might be:

1. Have passed in a mapping of content types to target classes that each know how to load themselves from the xml and a list of relationships. These classes are what OpenXML4J calls an *unmarshaller*. These would each look something like:

```
{ 'application/vnd...slide+xml'       : Slide
, 'application/vnd...slideLayout+xml' : SlideLayout
, ...
}
```

2. Have a ContentType class that can load from a content type item and allow lookups by partname. Have it load the content type item from the package. Lookups on it first look for an explicit override, but then fall back to defaults. Not sure yet whether it should try to be smart about whether a package part that looks like one that should have an override will get fed to the xml default or not.
3. Walk the package directory tree, and as each file is encountered:

OR

3. Walk the relationships tree, starting from /_rels/.rels
   - look up its content type in the content type manager
   - look up the unmarshaller for that content type
   - dispatch the part to the specified unmarshaller for loading
   - add the resulting part to the package parts collection
   - if it's a rels file, parse it and associate the relationships with the appropriate source part.

   If a content type or unmarshaller is not found, throw an exception and exit. Skip the content type item (already processed), and for now skip the package relationship item. Infer the corresponding part name from part rels item names. I think the walk can be configured so rels items are encountered only after their part has been processed.
4. Resolve all part relationship targets to in-memory references.

Principles upheld:

- If there are any stray items in the package (items not referenced in the content type part), they are identified and can be dealt with appropriately by throwing an exception if it looks like a package part or just writing it to the log if it's an extra file. Can set debug during development to throw an exception either way just to give a sense of what might typically be found in a package or to give notice that a hand-crafted package has an internal inconsistency.
- Conversely, if there are any parts referenced in the content type part that are not found in the zip archive, that throws an exception too.

Random thoughts

I'm starting to think that the packaging module could be a useful general-purpose capability that could be applied to .docx and .xlsx files in addition to .pptx ones.

Also I'm thinking that a generalized loading strategy that walks the zip archive directory tree and loads files based on combining what it discovers there with what it can look up in it's spec tables might be an interesting approach.

I can think of the following possible ways to identify the type of package, not sure which one is most reliable or definitive:

- Check the file extension. Kind of thinking this should do the trick 99.99% of the time.
- Might not hurt to confirm that by finding an expected directory of /ppt, /word, or /xl.
- A little further on that would be to find /ppt/presentation.xml, /word/document.xml, or /xl/workbook.xml.
- Even further would be to find a relationship in the package relationship item to one of those three and not to any of the others.
- Another confirmation might be finding a content type in [Content_Types].xml of:
  - .../presentationml.presentation.main+xml for /ppt/presentation.xml
  - .../spreadsheetml.sheet.main+xml for /xl/workbook.xml (Note that macro-enabled workbooks use a different content type 'application/vnd.ms-excel.sheet.macroEnabled.main+xml', I believe there are variants for PresentationML as well, used for templates and slide shows.
  - .../wordprocessingml.document.main+xml for /word/document.xml

It's probably worth consulting ECMA-376 Part 2 to see if there are any hard rules that might help determine what a definitive test would be.

About Part Relationships

Understanding Part Relationships

An Open XML package part can have one or more relationships to other parts in the package. For example, a slide layout has a relationship to the slide master it inherits properties from. And if the slide layout includes an image, such as a company logo, it will also have a relationship to the image part corresponding to that logo's image file.

Conceptually, a part relationship might be stated:

```
Part x is related to part y of type z.
```

While these relationships are reciprocal in a way of thinking (e.g. part y is also related to part x), in Open XML they are generally defined in one direction only. For example, a slide may have a relationship to an image it contains, but an image never has a stated relationships to a slide. One exception to this is slide masters and slide layouts. A slide layout has a relationship to the slide master it inherits from at the same time the slide master has a relationship to each of the slide layouts that inherit from it.

In the example above, part x can be referred to as the *source part* or *from-part*, part y as the *target part* or *to-part*, and type z as the *relationship type*. The relationship type essentially ends up being the part type of the target part (e.g. image, slide master, etc.).

Inbound and outbound relationships

Each relationship is *outbound* for its source part and *inbound* for its target part. The *outbound* relationships of a part are those found in its *part relationship item*, commonly referred to as its "rels file". Only outbound relationships are recorded in the package. Inbound relationships are purely abstract, although they can be inferred from outbound relationships if an application had a need for them.

Not all parts can have outbound relationships. For example, image and presentation properties parts never have outbound relationships. All parts, however, participate in at least one inbound relationship. Any part with no inbound

relationships could be removed from the package without consequence, and probably should be. For example, if upon saving it was noticed that a particular image had no inbound relationships, that image would be better not written to the package. *(Makes me wonder whether loading a package by walking its relationship graph might be a good idea in some instances.)*

Direct and indirect relationship references

Each relationship is recorded as a relationship element in the rels file belonging to a specific part. (The package relationship item `/_rels/.rels` is the only exception, it belongs to the package, not to any part). Each relationship entry specifies a part-local relationship id (rId) for the relationship, the relationship type (essentially the part type of the target, e.g. slide, image), and a path the target part file. The source part is not explicitly stated in the relationship entry, it is implicitly the part the .rels file belongs to.

These can be thought of as *external relationship references* in that the rels file is a separate package item, external to the part itself. However, in certain cases, a relationship may be referenced within the XML of the from-part. These can be thought of as *internal relationship references*.

As an example of where an internal relationship reference is required, consider a slide containing images of three different products. Each picture has a corresponding <p:pic> element in the slide's shape tree, each of which must specify the particular image file it displays, distinguishing it from the other two image files related to the slide.

The picture elements specify which of the related images to display using the part-local relationship id (rId) matching the required image, 'rId2' in the example below:

```
<p:blipFill>
  <a:blip r:embed="rId2">
    ...
  </a:blip>
</p:blipFill>
```

Which is an indirect reference to `image1.png` specified as the target of 'rId2' in the slide's rels file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://.../relationships">
  <Relationship Id="rId1" Type="http://.../slideLayout" Target=".../slideLayout2.xml"/
↪>
  <Relationship Id="rId2" Type="http://.../image" Target=".../image1.png"/>
</Relationships>
```

This indirection makes sense as a way to limit the coupling of presentation parts to the mechanics of package composition. For example, when the XML for the slide part in the example above is being composed, the slide can determine the reference to the image it's displaying without reference outside its own natural scope. In contrast, determining the eventual location and filename of that image in any particular package that was saved would require the slide code to have visibility into the packaging process, which would prevent packaging being delegated to a separate, black-box module.

Implicit and explicit relationships

There is also a distinction between implicit and explicit relationships which is described in the spec (ECMA-376-1) in section 9.2. I haven't encountered those yet in the context of PresentationML (the spec uses an example of footnotes from WordprocessingML), so I do not discuss them here.

Relationship Mechanics

Relationship life-cycle

A representation of a relationship must operate effectively in two distinct situations, in-package and in-memory. They must also support lifecycle transitions from in-package to in-memory and from in-memory back to in-package.

Abstract model

Each relationship has the following abstract attributes

source
: The "from-part" of the relationship.

id
: Source-local unique identifier for this relationship. Each source part's relationship ids should be a sequence of consecutive integers starting from 1.

target_type
: The content type of the relationship's to-part.

target
: A direct reference to the relationship's to-part.

Maintaining relationships by dynamic parts (e.g. Slides)

How will dynamic parts (like Slide) interact with its relationship list?

? Should it just add items to the relationship list when it creates new things?

? Does it need some sort of lookup capability in order to delete? Or just have a delete relationship method on RelationshipCollection or something like that.

Need to come up with a plausible set of use cases to think about a design. Right now the only use case is loading a template into a presentation and saving a presentation.

- Add an image to a slide.
- Change a slide's slide layout
- comment, notesSlide, tag, image, and slideLayout are the only outbound relationship types for a slide, although I expect there are some other DrawingML bits I haven't accounted for yet.

On reflection I'm thinking there's not too much urgency on noodling this out too far, the present construction should work fine for now and be able to be extended without disrupting other code too much.

SCRAP

Rationale for Relationship to be an association class

When loaded into memory, each relationship target must be a reference to an active part object (or at least a part key that can be resolved to a reference, but why do this lookup multiple times?). This is both because those relationships can change and also because the package path, while it can be calculated at runtime, is not guaranteed to be stable (e.g. a new slide can be inserted between two existing ones) and is not finally resolved until the presentation is saved.

General description of package relationships items

- Relationships items specify the relationships between parts of the package, although they are not themselves a part.
- All relationship items are XML documents having a filename with the extension '.rels' located in a directory named '_rels' located in the same directory as the part.
- The package relationship item has the URI '/_rels/.rels'.
- Part relationship items have the same filename as the part whose relationships they describe, with the '.rels' extension appended as a suffix. For example, the relationship item for a part named `/ppt/slides/slide1.xml` would have the URI `/ppt/slides/_rels/slide1.xml.rels`.

Web Resources

... just collected bits and pieces here for now ...

ECMA Spec

...

Microsoft PowerPoint API documentation

- PowerPoint 2010 Interop Namespace
- PowerPoint 2013 Object Model Reference
- PowerPoint 2010 Object Model Reference
- Open XML Presentation Namespace on MSDN
- Office 2013 Open XML Presentation Namespace
- Open XML Content Types as an XML Document

Sphinx Resources

- Alternate source for Sphinx documentation
- An example PyPi project

Documentation Guides

- Nice GitHub Project Style Guide (esp. Commit Style Guide) textmate / restructuredtext.tmbundle
- Python Documentation guide Documenting Python Guide
- Google Python Style Guide
- Read The Docs (readthedocs.org)
- OpenComparison documetation on readthedocs.org
- The Hitchhiker's Guide to Packaging

Other Resources

- Python Magic Methods
- lxml.etree Tutorial
- lxml API Reference
- The factory pattern in Python with __new__

Technical Topics

Some technical topics that have become relevant to this project ...

Python Metaclasses

- Metaclass programming in Python
- Python metaclasses by example

AppleScript Resources

In the fullness of time, it might make sense to incorporate some AppleScript steps into the acceptance tests, to make sure that PowerPoint isn't barfing after changes. I've collected some resources here that I discovered during some preliminary resource discovery.

Resources

- Outlook archive message shortcut
- AppleScript Essentials
- AppleScript Essentials: Introduction to Scripting Microsoft PowerPoint
- Applescript Microsoft 2011 PowerPoint
- Converting VBA Macros to AppleScript in Microsoft Office
- How can I use AppleScript to check if PowerPoint is playing a presentation?
- AppleScript on Wikipedia

Odds and Ends

Bits and pieces here without a more specific home

Class Hierarchy

```
pptx
|
+--.packaging
   |
   +--_PartCollection
   |  |
```

```
   |  +--ImageParts
   |  +--SlideLayoutParts
   |  +--SlideMasterParts
   |  +--SlideParts
   |  +--ThemeParts
   |
   +--Part
      |
      +--CollectionPart
      |  |
      |  +--ImagePart
      |  +--SlideLayoutPart
      |  +--SlideMasterPart
      |  +--ThemePart
      |
      +--PresentationPart
      +--PresPropsPart
      +--SlidePart
      +--TableStylesPart
      +--ViewPropsPart
```

Also try something like this:

```
Exception hierarchy
-------------------

The class hierarchy for built-in exceptions is:

.. literalinclude:: ../../Lib/test/exception_hierarchy.txt
```

Feature adding process:

- Analysis
  - Use case
  - Aspects of the spec and PowerPoint conventions that bear on the feature
- Implementation strategy
- Test Cases
- Documentation
- Implementation
- Commit

Odds and Ends

- DOC: Reserved Terms, e.g. package, part, item, element, etc. These have special meaning because of the terminology of Open XML packages and ElementTree XML conventions. Singleton, collection, perhaps tuple.
- DOC: Variable naming conventions. Variables containing XML elements are named the same as the tag name of the element they hold. For example: sldMaster = etree.parse('sldMaster1.xml')
- DOC: Suitable for server-side document generation, database publishing, taking tedium out of building intricate slides that have a distinct pattern, etc.

- DOC: Content Types mapping is discussed in ECMA-376-2 10.1.2 *Mapping Content Types*
- DOC: Explicit vs. Implicit relationships are discussed in section 9.2 of ECMA-376-1. Basically relationships to whole parts are explicit and relationships to one of the elements within another part (like a footnote) are implicit.
- DOC: [Open XML SDK How To's (How do I...)| http://207.46.22.237/en-us/library/cc850828.aspx]
- DOC: [Good hints on where formatting comes from in template|http://openxmldeveloper.org/discussions/formats/f/15/t/1301.aspx]
- Set work template slide layouts to standard type names for better layout re-mapping on theme/template change [http://msdn.microsoft.com/en-us/library/office/cc850846.aspx]
- spTree in cSld is a GroupShape (ShapeGroup?) perhaps Shape and ShapeGroup are ... ? not sure who should inherit from the other. GroupShapes can include other GroupShapes.

Python Module Index

p

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

T

U

V

W

X

Y