Documentation for package "reportlab.graphics"
Generated by: graphdocpy.py version 0.8
Date generated: 2006-05-23 16:08
Format: PDF

# reportlab.graphics

## eanbc

## Classes

### Ean13BarcodeWidget(PlotArea)

## Public Attributes

**background** Handle to background object.

**barFillColor** bar color

**barHeight** Height of bars.

**barStrokeColor** Color of bar borders.

**barStrokeWidth** Width of bar borders.

**barWidth** Width of bars.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**fontName** fontName

**fontSize** font size

**height** Height of the chart.

**humanReadable** if human readable

**lquiet** left quiet zone length

**quiet** if quiet zone to be used

**rquiet** right quiet zone length

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**textColor** human readable text color

**value** the number

**width** Width of the chart.

**x** x-coord

**y** y-coord

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

```
value = '123456789012'
```

## Ean8BarcodeWidget(Ean13BarcodeWidget)

## Public Attributes

**background** Handle to background object.

**barFillColor** bar color

**barHeight** Height of bars.

**barStrokeColor** Color of bar borders.

**barStrokeWidth** Width of bar borders.

**barWidth** Width of bars.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**fontName** fontName

**fontSize** font size

**height** Height of the chart.

**humanReadable** if human readable

**lquiet** left quiet zone length

**quiet** if quiet zone to be used

**rquiet** right quiet zone length

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**textColor** human readable text color

**value** the number

**width** Width of the chart.

**x** x-coord

**y** y-coord

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

```
value = '1234567'
```

### `areas`

This module defines a Area mixin classes

## Classes

### `PlotArea(Widget)`

Abstract base class representing a chart's plot area, pretty unusable by itself.

## Public Attributes

**background** Handle to background object.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**height** Height of the chart.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

```
background = None
debug = 0
fillColor = None
height = 85
strokeColor = None
strokeWidth = 1
width = 180
x = 20
y = 10
```

## axes

Collection of axes for charts.

The current collection comprises axes for charts using cartesian coordinate systems. All axes might have tick marks and labels. There are two dichotomies for axes: one of X and Y flavours and another of category and value flavours.

Category axes have an ordering but no metric. They are divided into a number of equal-sized buckets. Their tick marks or labels, if available, go BETWEEN the buckets, and the labels are placed below to/left of the X/Y-axis, respectively.

Value axes have an ordering AND metric. They correspond to a numeric quantity. Value axis have a real number quantity associated with it. The chart tells it where to go.
The most basic axis divides the number line into equal spaces and has tickmarks and labels associated with each; later we will add variants where you can specify the sampling interval.

The charts using axis tell them where the labels should be placed.

Axes of complementary X/Y flavours can be connected to each other in various ways, i.e. with a specific reference point, like an x/value axis to a y/value (or category) axis. In this case the connection can be either at the top or bottom of the former or at any absolute value (specified in points) or at some value of the former axes in its own coordinate system.

## Classes

### AdjYValueAxis(YValueAxis)

A Y-axis applying additional rules.

Depending on the data and some built-in rules, the axis may choose to adjust its range and origin.

## Public Attributes

**avoidBoundFrac** Fraction of interval to allow above and below.

**forceZero** Ensure zero in range if true.

**gridEnd** End of grid lines wrt axis origin

**gridStart** Start of grid lines wrt axis origin

**gridStrokeColor** Color of grid lines.

**gridStrokeDashArray** Dash array used for grid lines.

**gridStrokeWidth** Width of grid lines.

**joinAxis** Join both axes if true.

**joinAxisMode** Mode used for connecting axis ('left', 'right', 'value', 'points', None).

**joinAxisPos** Position at which to join with other axis.

**labelTextFormat** Formatting string or function used for axis labels.

**labelTextPostFormat** Extra Formatting string.

**labelTextScale** Scaling for label tick values.

**labels** Handle of the axis labels.

**leftAxisOrigShiftIPC** Lowest label shift interval ratio.

**leftAxisOrigShiftMin** Minimum amount to shift.

**leftAxisPercent** When true add percent sign to label values.

**leftAxisSkipLL0** Skip/Keep lowest tick label when true/false. Or skiplist

**maximumTicks** Maximum number of ticks.

**minimumTickSpacing** Minimum value for distance between ticks.

**rangeRound** How to round the axis limits

**requiredRange** Minimum required value range.

**strokeColor** Color of axis line and ticks.

**strokeDashArray** Dash array used for axis line.

**strokeWidth** Width of axis line and ticks.

**style** How values are plotted!

**tickLeft** Tick length left of the axis.

**tickRight** Tick length right of the axis.

**valueMax** Maximum value on axis.

**valueMin** Minimum value on axis.

**valueStep** Step size used between ticks.

**valueSteps** List of step sizes used between ticks.

**visible** Display entire object, if true.

**visibleAxis** Display axis line, if true.

**visibleGrid** Display axis grid, if true.

**visibleLabels** Display axis labels, if true.

**visibleTicks** Display axis ticks, if true.

**zrangePref** Zero range axis limit preference.


*Example*

```
def demo(self):
    data = [(10, 20, 30, 42)]
    self.setPosition(100, 10, 80)
    self.configure(data)

    drawing = Drawing(200, 100)
    drawing.add(self)
    return drawing
```

```
45%┤
40%┤
35%┤
30%┤
25%┤
20%┤
15%┤
10%┤
```

*Properties of Example Widget*

```
avoidBoundFrac = None
forceZero = 0
gridEnd = 0
gridStart = 0
gridStrokeColor = Color(0,0,0)
gridStrokeDashArray = None
gridStrokeWidth = 0.25
joinAxis = None
joinAxisMode = None
joinAxisPos = None
labelTextFormat = None
labelTextPostFormat = None
labelTextScale = None
labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x884778c>
leftAxisOrigShiftIPC = 0.14999999999999999
leftAxisOrigShiftMin = 12
leftAxisPercent = 1
leftAxisSkipLL0 = 0
maximumTicks = 7
minimumTickSpacing = 10
rangeRound = 'none'
requiredRange = 30
strokeColor = Color(0,0,0)
strokeDashArray = None
strokeWidth = 1
style = 'normal'
tickLeft = 5
tickRight = 0
valueMax = None
valueMin = None
valueStep = None
valueSteps = [10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 45.0]
visible = 1
visibleAxis = 1
visibleGrid = 0
visibleLabels = 1
visibleTicks = 1
zrangePref = 0
```

## `CategoryAxis(_AxisG)`

Abstract category axis, unusable in itself.

## Public Attributes

**categoryNames** List of category names.

**gridEnd** End of grid lines wrt axis origin

**gridStart** Start of grid lines wrt axis origin

**gridStrokeColor** Color of grid lines.

**gridStrokeDashArray** Dash array used for grid lines.

**gridStrokeWidth** Width of grid lines.

**joinAxis** Join both axes if true.

**joinAxisPos** Position at which to join with other axis.

**labelAxisMode** Like joinAxisMode, but for the axis labels

**labels** Handle of the axis labels.

**reverseDirection** If true reverse category direction.

**strokeColor** Color of axis line and ticks.

**strokeDashArray** Dash array used for axis line.

**strokeWidth** Width of axis line and ticks.

**style** How common category bars are plotted

**tickShift** Tick shift typically

**visible** Display entire object, if true.

**visibleAxis** Display axis line, if true.

**visibleGrid** Display axis grid, if true.

**visibleLabels** Display axis labels, if true.

**visibleTicks** Display axis ticks, if true.

## `NormalDateXValueAxis(XValueAxis)`

An X axis applying additional rules.

Depending on the data and some built-in rules, the axis
displays normalDate values as nicely formatted dates.

The client chart should have NormalDate X values.

## Public Attributes

**avoidBoundFrac** Fraction of interval to allow above and below.

**bottomAxisLabelSlack** Fractional amount used to adjust label spacing

**dailyFreq** True if we are to assume daily data to be ticked at end of month.

**dayOfWeekName** Weekday names.

**forceEndDate** Flag for enforced displaying of last date value.

**forceFirstDate** Flag for enforced displaying of first date value.

**forceZero** Ensure zero in range if true.

**gridEnd** End of grid lines wrt axis origin

**gridStart** Start of grid lines wrt axis origin

**gridStrokeColor** Color of grid lines.

**gridStrokeDashArray** Dash array used for grid lines.

**gridStrokeWidth** Width of grid lines.

**joinAxis** Join both axes if true.

**joinAxisMode** Mode used for connecting axis ('bottom', 'top', 'value', 'points', None).

**joinAxisPos** Position at which to join with other axis.

**labelTextFormat** Formatting string or function used for axis labels.

**labelTextPostFormat** Extra Formatting string.

**labelTextScale** Scaling for label tick values.

**labels** Handle of the axis labels.

**maximumTicks** Maximum number of ticks.

**minimumTickSpacing** Minimum value for distance between ticks.

**monthName** Month names.

**niceMonth** Flag for displaying months 'nicely'.

**rangeRound** How to round the axis limits

**strokeColor** Color of axis line and ticks.

**strokeDashArray** Dash array used for axis line.

**strokeWidth** Width of axis line and ticks.

**style** How values are plotted!

**tickDown** Tick length down the axis.

**tickUp** Tick length up the axis.

**valueMax** Maximum value on axis.

**valueMin** Minimum value on axis.

**valueStep** Step size used between ticks.

**valueSteps** List of step sizes used between ticks.

**visible** Display entire object, if true.

**visibleAxis** Display axis line, if true.

**visibleGrid** Display axis grid, if true.

**visibleLabels** Display axis labels, if true.

**visibleTicks** Display axis ticks, if true.

**xLabelFormat** Label format string (e.g. '{mm}/{yy}') or function.

**zrangePref** Zero range axis limit preference.


*Example*

```
def demo(self):
```

```
        self.setPosition(20, 50, 150)
        self.configure([(10,20,30,40,50)])

        d = Drawing(200, 100)
        d.add(self)
        return d
```

*Properties of Example Widget*

```
avoidBoundFrac = None
bottomAxisLabelSlack = 0.10000000000000001
dailyFreq = 0
dayOfWeekName = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
forceEndDate = 0
forceFirstDate = 0
forceZero = 0
gridEnd = 0
gridStart = 0
gridStrokeColor = Color(0,0,0)
gridStrokeDashArray = None
gridStrokeWidth = 0.25
joinAxis = None
joinAxisMode = None
joinAxisPos = None
labelTextFormat = None
labelTextPostFormat = None
labelTextScale = None
labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x887216c>
maximumTicks = 7
minimumTickSpacing = 10
monthName = ['January',
             'February',
             'March',
             'April',
             'May',
             'June',
             'July',
             'August',
             'September',
             'October',
             'November',
             'December']
niceMonth = 1
rangeRound = 'none'
strokeColor = Color(0,0,0)
strokeDashArray = None
strokeWidth = 1
style = 'normal'
tickDown = 5
tickUp = 0
valueMax = None
valueMin = None
valueStep = None
valueSteps = None
visible = 1
visibleAxis = 1
visibleGrid = 0
visibleLabels = 1
visibleTicks = 1
xLabelFormat = '{mm}/{yy}'
zrangePref = 0
```

## `ValueAxis(_AxisG)`

Abstract value axis, unusable in itself.

## Public Attributes

**avoidBoundFrac** Fraction of interval to allow above and below.

**forceZero** Ensure zero in range if true.

**gridEnd** End of grid lines wrt axis origin

**gridStart** Start of grid lines wrt axis origin

**gridStrokeColor** Color of grid lines.

**gridStrokeDashArray** Dash array used for grid lines.

**gridStrokeWidth** Width of grid lines.

**labelTextFormat** Formatting string or function used for axis labels.

**labelTextPostFormat** Extra Formatting string.

**labelTextScale** Scaling for label tick values.

**labels** Handle of the axis labels.

**maximumTicks** Maximum number of ticks.

**minimumTickSpacing** Minimum value for distance between ticks.

**rangeRound** How to round the axis limits

**strokeColor** Color of axis line and ticks.

**strokeDashArray** Dash array used for axis line.

**strokeWidth** Width of axis line and ticks.

**style** How values are plotted!

**valueMax** Maximum value on axis.

**valueMin** Minimum value on axis.

**valueStep** Step size used between ticks.

**valueSteps** List of step sizes used between ticks.

**visible** Display entire object, if true.

**visibleAxis** Display axis line, if true.

**visibleGrid** Display axis grid, if true.
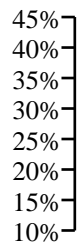
**visibleLabels** Display axis labels, if true.

**visibleTicks** Display axis ticks, if true.

**zrangePref** Zero range axis limit preference.

## `XCategoryAxis(CategoryAxis)`

X/category axis

## Public Attributes

**categoryNames** List of category names.

**gridEnd** End of grid lines wrt axis origin

**gridStart** Start of grid lines wrt axis origin

**gridStrokeColor** Color of grid lines.

**gridStrokeDashArray** Dash array used for grid lines.

**gridStrokeWidth** Width of grid lines.

**joinAxis** Join both axes if true.

**joinAxisMode** Mode used for connecting axis ('bottom', 'top', 'value', 'points', None).

**joinAxisPos** Position at which to join with other axis.

**labelAxisMode** Like joinAxisMode, but for the axis labels

**labels** Handle of the axis labels.

**reverseDirection** If true reverse category direction.

**strokeColor** Color of axis line and ticks.

**strokeDashArray** Dash array used for axis line.

**strokeWidth** Width of axis line and ticks.

**style** How common category bars are plotted

**tickDown** Tick length down the axis.

**tickShift** Tick shift typically

**tickUp** Tick length up the axis.

**visible** Display entire object, if true.

**visibleAxis** Display axis line, if true.

**visibleGrid** Display axis grid, if true.

**visibleLabels** Display axis labels, if true.

**visibleTicks** Display axis ticks, if true.

## `XValueAxis(ValueAxis)`

X/value axis

## Public Attributes

**avoidBoundFrac** Fraction of interval to allow above and below.

**forceZero** Ensure zero in range if true.

**gridEnd** End of grid lines wrt axis origin

**gridStart** Start of grid lines wrt axis origin

**gridStrokeColor** Color of grid lines.

**gridStrokeDashArray** Dash array used for grid lines.

**gridStrokeWidth** Width of grid lines.

**joinAxis** Join both axes if true.

**joinAxisMode** Mode used for connecting axis ('bottom', 'top', 'value', 'points', None).

**joinAxisPos** Position at which to join with other axis.

**labelTextFormat** Formatting string or function used for axis labels.

**labelTextPostFormat** Extra Formatting string.

**labelTextScale** Scaling for label tick values.

**labels** Handle of the axis labels.

**maximumTicks** Maximum number of ticks.

**minimumTickSpacing** Minimum value for distance between ticks.

**rangeRound** How to round the axis limits

**strokeColor** Color of axis line and ticks.

**strokeDashArray** Dash array used for axis line.

**strokeWidth** Width of axis line and ticks.

**style** How values are plotted!

**tickDown** Tick length down the axis.

**tickUp** Tick length up the axis.

**valueMax** Maximum value on axis.

**valueMin** Minimum value on axis.

**valueStep** Step size used between ticks.

**valueSteps** List of step sizes used between ticks.

**visible** Display entire object, if true.

**visibleAxis** Display axis line, if true.

**visibleGrid** Display axis grid, if true.

**visibleLabels** Display axis labels, if true.

**visibleTicks** Display axis ticks, if true.

**zrangePref** Zero range axis limit preference.

*Example*

```
def demo(self):
    self.setPosition(20, 50, 150)
    self.configure([(10,20,30,40,50)])

    d = Drawing(200, 100)
    d.add(self)
    return d
```

*Properties of Example Widget*

```
avoidBoundFrac = None
forceZero = 0
gridEnd = 0
gridStart = 0
gridStrokeColor = Color(0,0,0)
gridStrokeDashArray = None
gridStrokeWidth = 0.25
joinAxis = None
joinAxisMode = None
joinAxisPos = None
labelTextFormat = None
labelTextPostFormat = None
labelTextScale = None
labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x888288c>
maximumTicks = 7
minimumTickSpacing = 10
rangeRound = 'none'
```

```
strokeColor = Color(0,0,0)
strokeDashArray = None
strokeWidth = 1
style = 'normal'
tickDown = 5
tickUp = 0
valueMax = None
valueMin = None
valueStep = None
visible = 1
visibleAxis = 1
visibleGrid = 0
visibleLabels = 1
visibleTicks = 1
zrangePref = 0
```

```
strokeColor = Color(0,0,0)
```

## YCategoryAxis(CategoryAxis)

Y/category axis

## Public Attributes

**categoryNames** List of category names.

**gridEnd** End of grid lines wrt axis origin

**gridStart** Start of grid lines wrt axis origin

**gridStrokeColor** Color of grid lines.

**gridStrokeDashArray** Dash array used for grid lines.

**gridStrokeWidth** Width of grid lines.

**joinAxis** Join both axes if true.

**joinAxisMode** Mode used for connecting axis ('left', 'right', 'value', 'points', None).

**joinAxisPos** Position at which to join with other axis.

**labelAxisMode** Like joinAxisMode, but for the axis labels

**labels** Handle of the axis labels.

**reverseDirection** If true reverse category direction.

**strokeColor** Color of axis line and ticks.

**strokeDashArray** Dash array used for axis line.

**strokeWidth** Width of axis line and ticks.

**style** How common category bars are plotted

**tickLeft** Tick length left of the axis.

**tickRight** Tick length right of the axis.

**tickShift** Tick shift typically

**visible** Display entire object, if true.

**visibleAxis** Display axis line, if true.

**visibleGrid** Display axis grid, if true.

**visibleLabels** Display axis labels, if true.

**visibleTicks** Display axis ticks, if true.

## YValueAxis(ValueAxis)

Y/value axis

## Public Attributes

**avoidBoundFrac** Fraction of interval to allow above and below.

**forceZero** Ensure zero in range if true.

**gridEnd** End of grid lines wrt axis origin

**gridStart** Start of grid lines wrt axis origin

**gridStrokeColor** Color of grid lines.

**gridStrokeDashArray** Dash array used for grid lines.

**gridStrokeWidth** Width of grid lines.

**joinAxis** Join both axes if true.

**joinAxisMode** Mode used for connecting axis ('left', 'right', 'value', 'points', None).

**joinAxisPos** Position at which to join with other axis.

**labelTextFormat** Formatting string or function used for axis labels.

**labelTextPostFormat** Extra Formatting string.

**labelTextScale** Scaling for label tick values.

**labels** Handle of the axis labels.

**maximumTicks** Maximum number of ticks.

**minimumTickSpacing** Minimum value for distance between ticks.

**rangeRound** How to round the axis limits

**strokeColor** Color of axis line and ticks.

**strokeDashArray** Dash array used for axis line.

**strokeWidth** Width of axis line and ticks.

**style** How values are plotted!

**tickLeft** Tick length left of the axis.

**tickRight** Tick length right of the axis.

**valueMax** Maximum value on axis.

**valueMin** Minimum value on axis.

**valueStep** Step size used between ticks.

**valueSteps** List of step sizes used between ticks.

**visible** Display entire object, if true.

**visibleAxis** Display axis line, if true.

**visibleGrid** Display axis grid, if true.

**visibleLabels** Display axis labels, if true.

**visibleTicks** Display axis ticks, if true.

**zrangePref** Zero range axis limit preference.


*Example*

```
def demo(self):
    data = [(10, 20, 30, 42)]
    self.setPosition(100, 10, 80)
    self.configure(data)

    drawing = Drawing(200, 100)
    drawing.add(self)
    return drawing
```


*Properties of Example Widget*

```
avoidBoundFrac = None
forceZero = 0
gridEnd = 0
gridStart = 0
```

```
gridStrokeColor = Color(0,0,0)
gridStrokeDashArray = None
gridStrokeWidth = 0.25
joinAxis = None
joinAxisMode = None
joinAxisPos = None
labelTextFormat = None
labelTextPostFormat = None
labelTextScale = None
labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x888cd6c>
maximumTicks = 7
minimumTickSpacing = 10
rangeRound = 'none'
strokeColor = Color(0,0,0)
strokeDashArray = None
strokeWidth = 1
style = 'normal'
tickLeft = 5
tickRight = 0
valueMax = None
valueMin = None
valueStep = None
visible = 1
visibleAxis = 1
visibleGrid = 0
visibleLabels = 1
visibleTicks = 1
zrangePref = 0
```

## _AxisG(Widget)

## Public Attributes

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

## Functions

### sample0a( ... )

Sample drawing with one xcat axis and two buckets.

*Example*

```
def sample0a():
    "Sample drawing with one xcat axis and two buckets."

    drawing = Drawing(400, 200)

    data = [(10, 20)]

    xAxis = XCategoryAxis()
    xAxis.setPosition(75, 75, 300)
    xAxis.configure(data)
    xAxis.categoryNames = ['Ying', 'Yang']
    xAxis.labels.boxAnchor = 'n'

    drawing.add(xAxis)

    return drawing
```

```
         Ying                    Yang
```

**sample0b( ... )**

Sample drawing with one xcat axis and one bucket only.

*Example*

```
def sample0b():
    "Sample drawing with one xcat axis and one bucket only."

    drawing = Drawing(400, 200)

    data = [(10,)]

    xAxis = XCategoryAxis()
    xAxis.setPosition(75, 75, 300)
    xAxis.configure(data)
    xAxis.categoryNames = ['Ying']
    xAxis.labels.boxAnchor = 'n'

    drawing.add(xAxis)

    return drawing
```

Ying
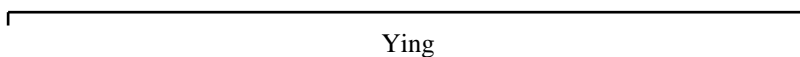
## **sample1( ... )**

Sample drawing containing two unconnected axes.

*Example*

```
def sample1():
    "Sample drawing containing two unconnected axes."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    xAxis = XCategoryAxis()
    xAxis.setPosition(75, 75, 300)
    xAxis.configure(data)
    xAxis.categoryNames = ['Beer','Wine','Meat','Cannelloni']
    xAxis.labels.boxAnchor = 'n'
    xAxis.labels[3].dy = -15
    xAxis.labels[3].angle = 30
    xAxis.labels[3].fontName = 'Times-Bold'

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

    return drawing
```

## sample4a( ... )

Sample drawing, xvalue/yvalue axes, y connected at 100 pts to x.

*Example*

```
def sample4a():
    "Sample drawing, xvalue/yvalue axes, y connected at 100 pts to x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

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

    xAxis = XValueAxis()
    xAxis._length = 300
    xAxis.joinAxis = yAxis
    xAxis.joinAxisMode = 'points'
    xAxis.joinAxisPos = 100
    xAxis.configure(data)

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

    return drawing
```

### sample4b( ... )

Sample drawing, xvalue/yvalue axes, y connected at value 35 of x.

*Example*

```
def sample4b():
    "Sample drawing, xvalue/yvalue axes, y connected at value 35 of x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

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

    xAxis = XValueAxis()
    xAxis._length = 300
    xAxis.joinAxis = yAxis
    xAxis.joinAxisMode = 'value'
    xAxis.joinAxisPos = 35
    xAxis.configure(data)

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

    return drawing
```

## sample4c( ... )

Sample drawing, xvalue/yvalue axes, y connected to bottom of x.

*Example*

```
def sample4c():
    "Sample drawing, xvalue/yvalue axes, y connected to bottom of x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

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

    xAxis = XValueAxis()
    xAxis._length = 300
    xAxis.joinAxis = yAxis
    xAxis.joinAxisMode = 'bottom'
    xAxis.configure(data)

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

    return drawing
```

### sample4c1( ... )

xvalue/yvalue axes, without drawing axis lines/ticks.

*Example*

```
def sample4c1():
    "xvalue/yvalue axes, without drawing axis lines/ticks."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    yAxis = YValueAxis()
    yAxis.setPosition(50, 50, 125)
    yAxis.configure(data)
    yAxis.visibleAxis = 0
    yAxis.visibleTicks = 0

    xAxis = XValueAxis()
    xAxis._length = 300
    xAxis.joinAxis = yAxis
    xAxis.joinAxisMode = 'bottom'
    xAxis.configure(data)
    xAxis.visibleAxis = 0
    xAxis.visibleTicks = 0

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

    return drawing
```

```
40



30



20



10
  10              20              30              40
```

## **sample4d( ... )**

Sample drawing, xvalue/yvalue axes, y connected to top of x.

*Example*

```
def sample4d():
    "Sample drawing, xvalue/yvalue axes, y connected to top of x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

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

    xAxis = XValueAxis()
    xAxis._length = 300
    xAxis.joinAxis = yAxis
    xAxis.joinAxisMode = 'top'
    xAxis.configure(data)

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

    return drawing
```
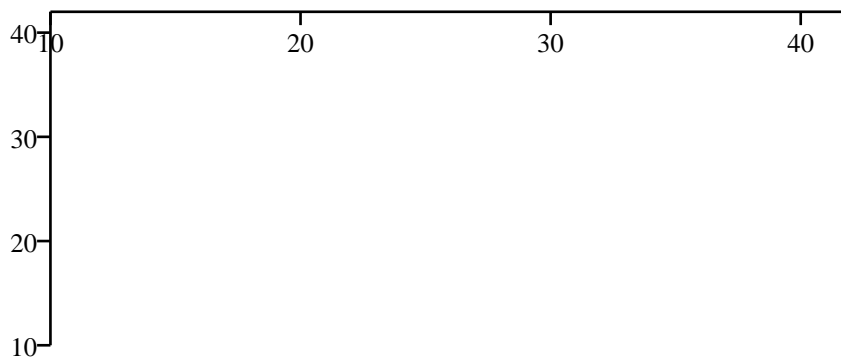
## sample5a( ... )

Sample drawing, xvalue/yvalue axes, y connected at 100 pts to x.

*Example*

```
def sample5a():
    "Sample drawing, xvalue/yvalue axes, y connected at 100 pts to x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    xAxis = XValueAxis()
    xAxis.setPosition(50, 50, 300)
    xAxis.configure(data)

    yAxis = YValueAxis()
    yAxis.setPosition(50, 50, 125)
    yAxis.joinAxis = xAxis
    yAxis.joinAxisMode = 'points'
    yAxis.joinAxisPos = 100
    yAxis.configure(data)

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

    return drawing
```
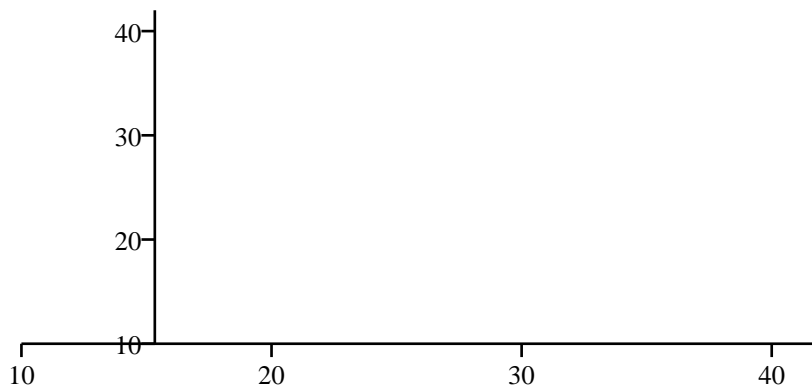
## sample5b( ... )

Sample drawing, xvalue/yvalue axes, y connected at value 35 of x.

*Example*

```
def sample5b():
    "Sample drawing, xvalue/yvalue axes, y connected at value 35 of x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    xAxis = XValueAxis()
    xAxis.setPosition(50, 50, 300)
    xAxis.configure(data)

    yAxis = YValueAxis()
    yAxis.setPosition(50, 50, 125)
    yAxis.joinAxis = xAxis
    yAxis.joinAxisMode = 'value'
    yAxis.joinAxisPos = 35
    yAxis.configure(data)

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

    return drawing
```
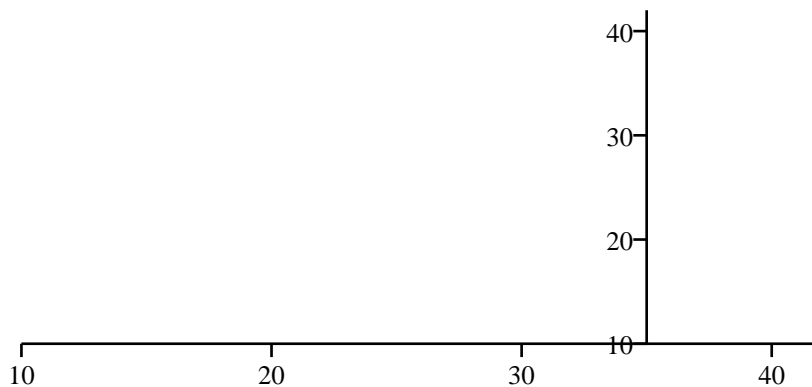
### sample5c( ... )

Sample drawing, xvalue/yvalue axes, y connected at right of x.

*Example*

```
def sample5c():
    "Sample drawing, xvalue/yvalue axes, y connected at right of x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    xAxis = XValueAxis()
    xAxis.setPosition(50, 50, 300)
    xAxis.configure(data)

    yAxis = YValueAxis()
    yAxis.setPosition(50, 50, 125)
    yAxis.joinAxis = xAxis
    yAxis.joinAxisMode = 'right'
    yAxis.configure(data)

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

    return drawing
```
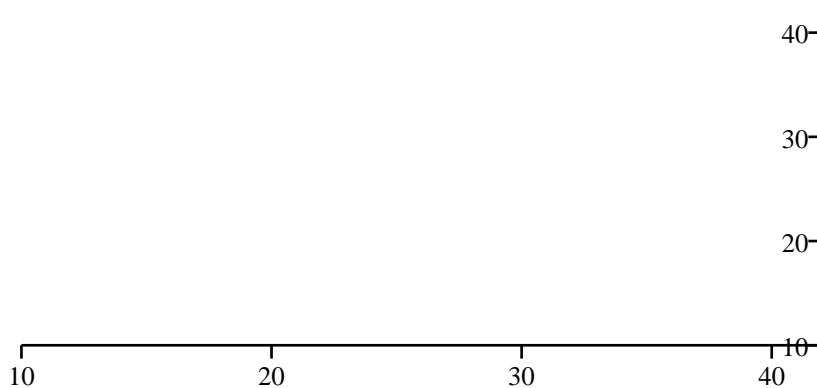
## sample5d( ... )

Sample drawing, xvalue/yvalue axes, y connected at left of x.

*Example*

```
def sample5d():
    "Sample drawing, xvalue/yvalue axes, y connected at left of x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    xAxis = XValueAxis()
    xAxis.setPosition(50, 50, 300)
    xAxis.configure(data)

    yAxis = YValueAxis()
    yAxis.setPosition(50, 50, 125)
    yAxis.joinAxis = xAxis
    yAxis.joinAxisMode = 'left'
    yAxis.configure(data)

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

    return drawing
```
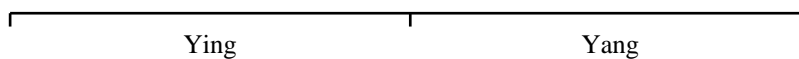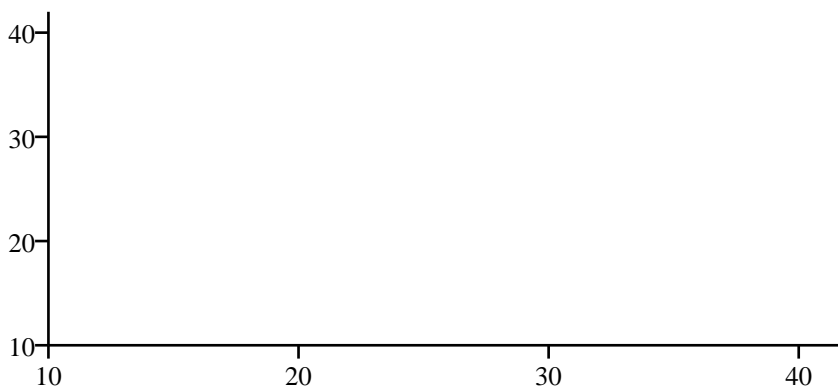
## sample6a( ... )

Sample drawing, xcat/yvalue axes, x connected at top of y.

*Example*

```
def sample6a():
    "Sample drawing, xcat/yvalue axes, x connected at top of y."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

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

    xAxis = XCategoryAxis()
    xAxis._length = 300
    xAxis.configure(data)
    xAxis.joinAxis = yAxis
    xAxis.joinAxisMode = 'top'
    xAxis.categoryNames = ['Beer', 'Wine', 'Meat', 'Cannelloni']
    xAxis.labels.boxAnchor = 'n'

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

    return drawing
```
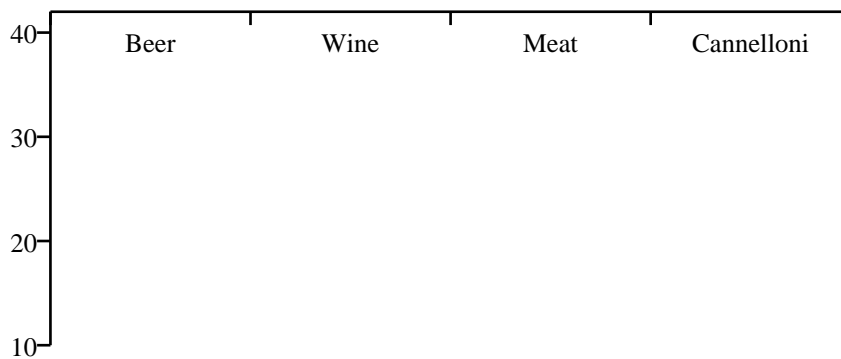
## sample6b( ... )

Sample drawing, xcat/yvalue axes, x connected at bottom of y.

*Example*

```
def sample6b():
    "Sample drawing, xcat/yvalue axes, x connected at bottom of y."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

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

    xAxis = XCategoryAxis()
    xAxis._length = 300
    xAxis.configure(data)
    xAxis.joinAxis = yAxis
    xAxis.joinAxisMode = 'bottom'
    xAxis.categoryNames = ['Beer', 'Wine', 'Meat', 'Cannelloni']
    xAxis.labels.boxAnchor = 'n'

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

    return drawing
```
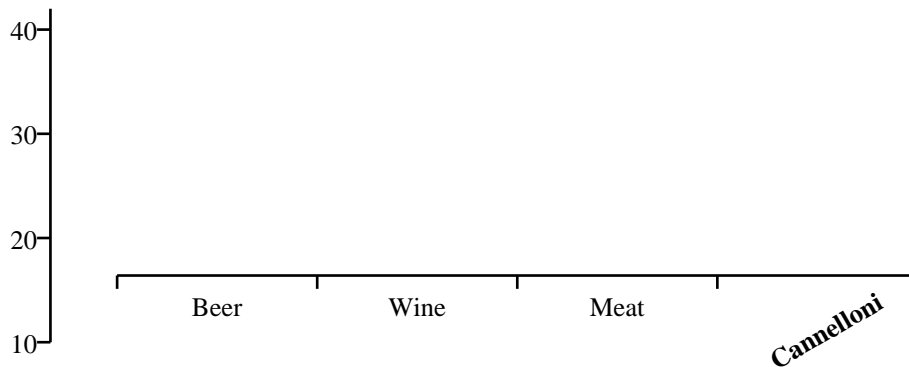
## sample6c( ... )

Sample drawing, xcat/yvalue axes, x connected at 100 pts to y.

*Example*

```
def sample6c():
    "Sample drawing, xcat/yvalue axes, x connected at 100 pts to y."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

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

    xAxis = XCategoryAxis()
    xAxis._length = 300
    xAxis.configure(data)
    xAxis.joinAxis = yAxis
    xAxis.joinAxisMode = 'points'
    xAxis.joinAxisPos = 100
    xAxis.categoryNames = ['Beer', 'Wine', 'Meat', 'Cannelloni']
    xAxis.labels.boxAnchor = 'n'

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

    return drawing
```
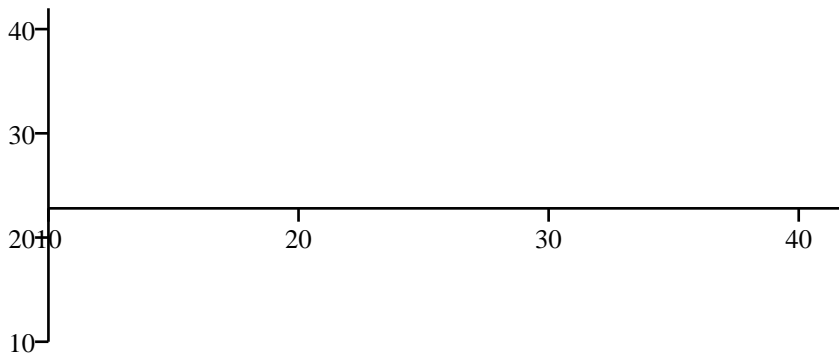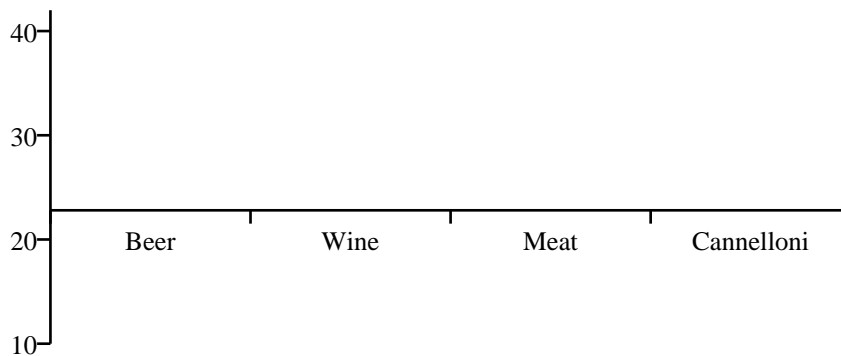
### sample6d( ... )

Sample drawing, xcat/yvalue axes, x connected at value 20 of y.

*Example*

```
def sample6d():
    "Sample drawing, xcat/yvalue axes, x connected at value 20 of y."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

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

    xAxis = XCategoryAxis()
    xAxis._length = 300
    xAxis.configure(data)
    xAxis.joinAxis = yAxis
    xAxis.joinAxisMode = 'value'
    xAxis.joinAxisPos = 20
    xAxis.categoryNames = ['Beer', 'Wine', 'Meat', 'Cannelloni']
    xAxis.labels.boxAnchor = 'n'

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

    return drawing
```
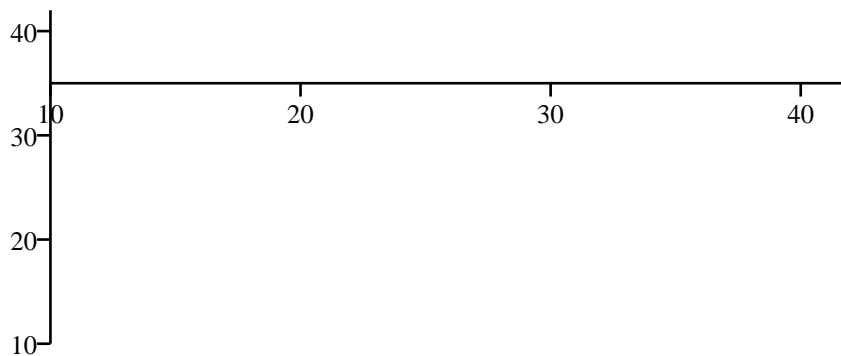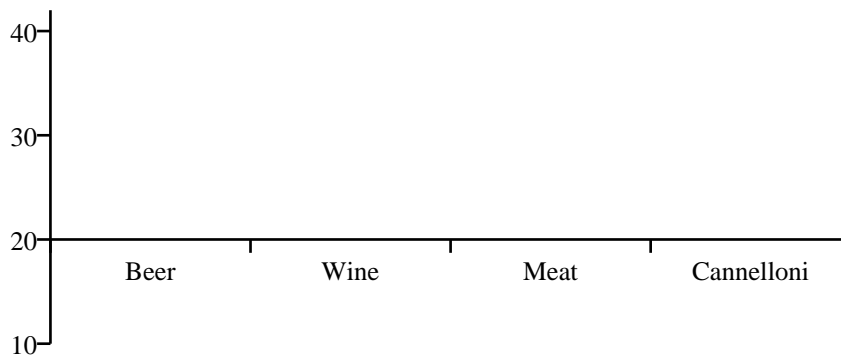
## sample7a( ... )

Sample drawing, xvalue/ycat axes, y connected at right of x.

*Example*

```
def sample7a():
    "Sample drawing, xvalue/ycat axes, y connected at right of x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    xAxis = XValueAxis()
    xAxis._length = 300
    xAxis.configure(data)

    yAxis = YCategoryAxis()
    yAxis.setPosition(50, 50, 125)
    yAxis.joinAxis = xAxis
    yAxis.joinAxisMode = 'right'
    yAxis.categoryNames = ['Beer', 'Wine', 'Meat', 'Cannelloni']
    yAxis.labels.boxAnchor = 'e'
    yAxis.configure(data)

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

    return drawing
```
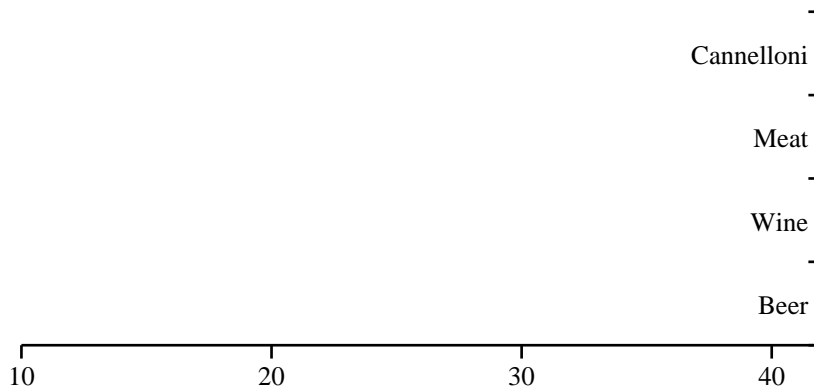
## **sample7b( ... )**

Sample drawing, xvalue/ycat axes, y connected at left of x.

*Example*

```
def sample7b():
    "Sample drawing, xvalue/ycat axes, y connected at left of x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    xAxis = XValueAxis()
    xAxis._length = 300
    xAxis.configure(data)

    yAxis = YCategoryAxis()
    yAxis.setPosition(50, 50, 125)
    yAxis.joinAxis = xAxis
    yAxis.joinAxisMode = 'left'
    yAxis.categoryNames = ['Beer', 'Wine', 'Meat', 'Cannelloni']
    yAxis.labels.boxAnchor = 'e'
    yAxis.configure(data)

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

    return drawing
```
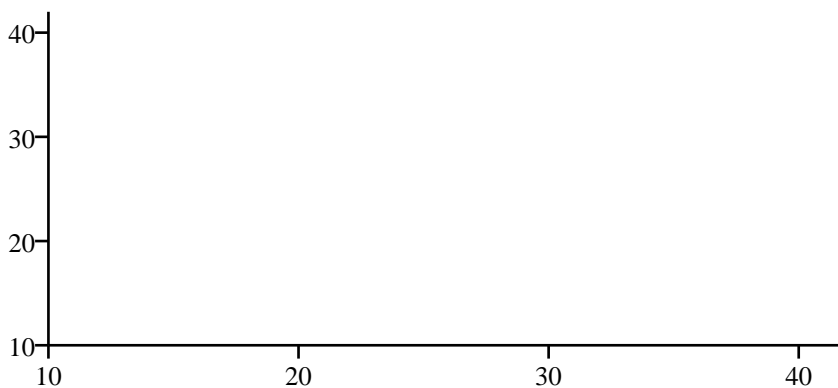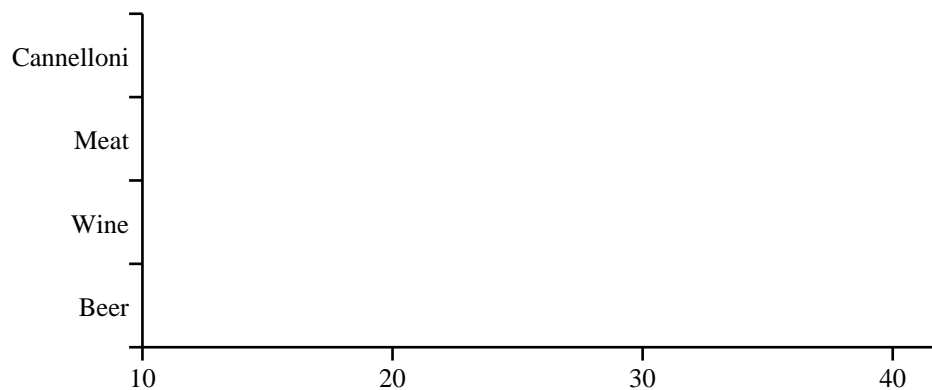
### sample7c( ... )

Sample drawing, xvalue/ycat axes, y connected at value 30 of x.

*Example*

```
def sample7c():
    "Sample drawing, xvalue/ycat axes, y connected at value 30 of x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    xAxis = XValueAxis()
    xAxis._length = 300
    xAxis.configure(data)

    yAxis = YCategoryAxis()
    yAxis.setPosition(50, 50, 125)
    yAxis.joinAxis = xAxis
    yAxis.joinAxisMode = 'value'
    yAxis.joinAxisPos = 30
    yAxis.categoryNames = ['Beer', 'Wine', 'Meat', 'Cannelloni']
    yAxis.labels.boxAnchor = 'e'
    yAxis.configure(data)

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

    return drawing
```
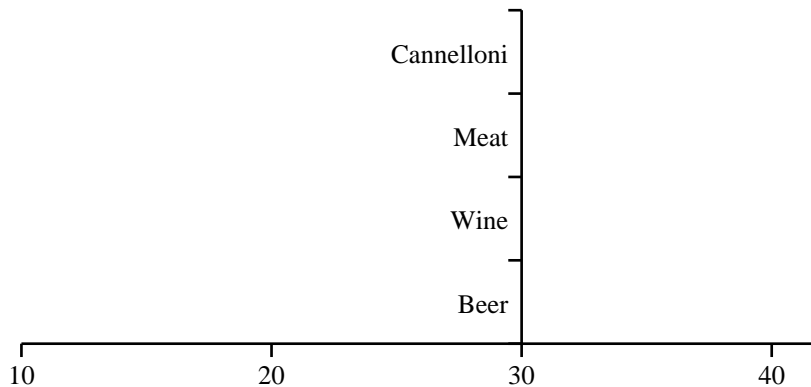
### sample7d( ... )

Sample drawing, xvalue/ycat axes, y connected at 200 pts to x.

*Example*

```
def sample7d():
    "Sample drawing, xvalue/ycat axes, y connected at 200 pts to x."

    drawing = Drawing(400, 200)

    data = [(10, 20, 30, 42)]

    xAxis = XValueAxis()
    xAxis._length = 300
    xAxis.configure(data)

    yAxis = YCategoryAxis()
    yAxis.setPosition(50, 50, 125)
    yAxis.joinAxis = xAxis
    yAxis.joinAxisMode = 'points'
    yAxis.joinAxisPos = 200
    yAxis.categoryNames = ['Beer', 'Wine', 'Meat', 'Cannelloni']
    yAxis.labels.boxAnchor = 'e'
    yAxis.configure(data)

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

    return drawing
```
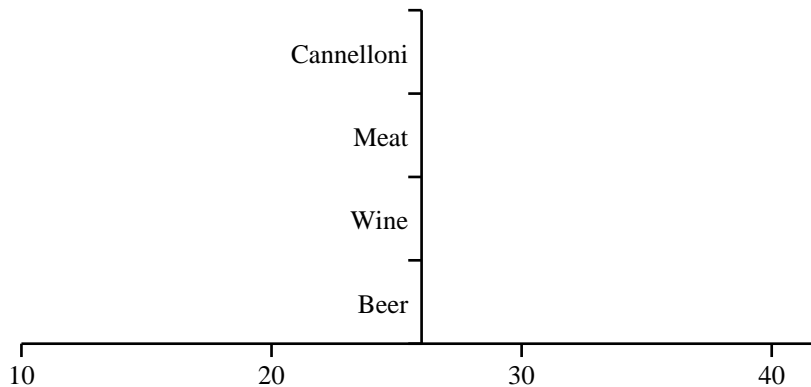
## `barcharts`

This module defines a variety of Bar Chart components.

The basic flavors are Side-by-side, available in horizontal and vertical versions.

Stacked and percentile bar charts to follow...

## Classes

### `BarChart(PlotArea)`

Abstract base class, unusable by itself.

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**barLabelArray** explicit array of bar label values, must match size of data if present.

**barLabelCallOut** Callout function(label) label._callOutInfo = (self,g,rowNo,colNo,x,y,width,height,x00,y00,x0,y0)

**barLabelFormat** Formatting string or function used for bar labels.

**barLabels** Handle to the list of bar labels.

**barSpacing** Width between individual bars.

**barWidth** The width of an individual bar.

**bars** Handle of the individual bars.

**categoryAxis** Handle of the category axis.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**groupSpacing** Width between groups of bars.

**height** Height of the chart.

**naLabel** Label to use for N/A values.

**reversePlotOrder** If true, reverse common category plot order.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**useAbsolute** Flag to use absolute spacing values.

**valueAxis** Handle of the value axis.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

### `BarChart3D(BarChart)`

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**barLabelArray** explicit array of bar label values, must match size of data if present.

**barLabelCallOut** Callout function(label) label._callOutInfo = (self,g,rowNo,colNo,x,y,width,height,x00,y00,x0,y0)

**barLabelFormat** Formatting string or function used for bar labels.

**barLabels** Handle to the list of bar labels.

**barSpacing** Width between individual bars.

**barWidth** The width of an individual bar.

**bars** Handle of the individual bars.

**categoryAxis** Handle of the category axis.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**groupSpacing** Width between groups of bars.

**height** Height of the chart.

**naLabel** Label to use for N/A values.

**reversePlotOrder** If true, reverse common category plot order.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**theta_x** dx/dz

**theta_y** dy/dz

**useAbsolute** Flag to use absolute spacing values.

**valueAxis** Handle of the value axis.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

**zDepth** depth of an individual series

**zSpace** z gap around series

## HorizontalBarChart(BarChart)

Horizontal bar chart with multiple side-by-side bars.

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**barLabelArray** explicit array of bar label values, must match size of data if present.

**barLabelCallOut** Callout function(label) label._callOutInfo = (self,g,rowNo,colNo,x,y,width,height,x00,y00,x0,y0)

**barLabelFormat** Formatting string or function used for bar labels.

**barLabels** Handle to the list of bar labels.

**barSpacing** Width between individual bars.

**barWidth** The width of an individual bar.

**bars** Handle of the individual bars.

**categoryAxis** Handle of the category axis.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**groupSpacing** Width between groups of bars.

**height** Height of the chart.

**naLabel** Label to use for N/A values.

**reversePlotOrder** If true, reverse common category plot order.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**useAbsolute** Flag to use absolute spacing values.

**valueAxis** Handle of the value axis.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

*Example*

```
def demo(self):
    """Shows basic use of a bar chart"""
    if self.__class__.__name__=='BarChart':
        raise NotImplementedError, 'Abstract Class BarChart has no demo'
    drawing = Drawing(200, 100)
    bc = self.__class__()
    drawing.add(bc)
    return drawing
```

*Properties of Example Widget*

```
background = None
barLabelArray = None
barLabelFormat = None
barLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x881d6ac>
barSpacing = 0
barWidth = 10
bars = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x881d6ec>
categoryAxis.categoryNames = None
categoryAxis.gridEnd = 0
categoryAxis.gridStart = 0
categoryAxis.gridStrokeColor = Color(0,0,0)
categoryAxis.gridStrokeDashArray = None
categoryAxis.gridStrokeWidth = 0.25
categoryAxis.joinAxis = None
categoryAxis.joinAxisMode = None
categoryAxis.joinAxisPos = None
categoryAxis.labelAxisMode = 'axis'
```

```
categoryAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x881d62c>
categoryAxis.reverseDirection = 0
categoryAxis.strokeColor = Color(0,0,0)
categoryAxis.strokeDashArray = None
categoryAxis.strokeWidth = 1
categoryAxis.style = 'parallel'
categoryAxis.tickLeft = 5
categoryAxis.tickRight = 0
categoryAxis.tickShift = 0
categoryAxis.visible = 1
categoryAxis.visibleAxis = 1
categoryAxis.visibleGrid = 0
categoryAxis.visibleLabels = 1
categoryAxis.visibleTicks = 1
data = [(100, 110, 120, 130), (70, 80, 85, 90)]
debug = 0
fillColor = None
groupSpacing = 5
height = 85
naLabel = None
reversePlotOrder = 0
strokeColor = None
strokeWidth = 1
useAbsolute = 0
valueAxis.avoidBoundFrac = None
valueAxis.forceZero = 0
valueAxis.gridEnd = 0
valueAxis.gridStart = 0
valueAxis.gridStrokeColor = Color(0,0,0)
valueAxis.gridStrokeDashArray = None
valueAxis.gridStrokeWidth = 0.25
valueAxis.joinAxis = None
valueAxis.joinAxisMode = None
valueAxis.joinAxisPos = None
valueAxis.labelTextFormat = None
valueAxis.labelTextPostFormat = None
valueAxis.labelTextScale = None
valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x881d58c>
valueAxis.maximumTicks = 7
valueAxis.minimumTickSpacing = 10
valueAxis.rangeRound = 'none'
valueAxis.strokeColor = Color(0,0,0)
valueAxis.strokeDashArray = None
valueAxis.strokeWidth = 1
valueAxis.style = 'normal'
valueAxis.tickDown = 5
valueAxis.tickUp = 0
valueAxis.valueMax = None
valueAxis.valueMin = None
valueAxis.valueStep = None
valueAxis.visible = 1
valueAxis.visibleAxis = 1
valueAxis.visibleGrid = 0
valueAxis.visibleLabels = 1
valueAxis.visibleTicks = 1
valueAxis.zrangePref = 0
width = 180
x = 20
y = 10
```

## `HorizontalBarChart3D(BarChart3D, HorizontalBarChart)`

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**barLabelArray** explicit array of bar label values, must match size of data if present.

**barLabelCallOut** Callout function(label) label._callOutInfo = (self,g,rowNo,colNo,x,y,width,height,x00,y00,x0,y0)

**barLabelFormat** Formatting string or function used for bar labels.

**barLabels** Handle to the list of bar labels.

**barSpacing** Width between individual bars.

**barWidth** The width of an individual bar.

**bars** Handle of the individual bars.

**categoryAxis** Handle of the category axis.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**groupSpacing** Width between groups of bars.

**height** Height of the chart.

**naLabel** Label to use for N/A values.

**reversePlotOrder** If true, reverse common category plot order.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**theta_x** dx/dz

**theta_y** dy/dz

**useAbsolute** Flag to use absolute spacing values.

**valueAxis** Handle of the value axis.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

**zDepth** depth of an individual series

**zSpace** z gap around series

*Example*

```
def demo(self):
    """Shows basic use of a bar chart"""
    if self.__class__.__name__=='BarChart':
        raise NotImplementedError, 'Abstract Class BarChart has no demo'
    drawing = Drawing(200, 100)
    bc = self.__class__()
    drawing.add(bc)
    return drawing
```

*Properties of Example Widget*

```
background = None
barLabelArray = None
barLabelFormat = None
barLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a03a8c>
barSpacing = 0
barWidth = 10
bars = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a03acc>
categoryAxis.categoryNames = None
categoryAxis.gridEnd = 0
categoryAxis.gridStart = 0
categoryAxis.gridStrokeColor = Color(0,0,0)
categoryAxis.gridStrokeDashArray = None
categoryAxis.gridStrokeWidth = 0.25
categoryAxis.joinAxis = None
categoryAxis.joinAxisMode = None
categoryAxis.joinAxisPos = None
categoryAxis.labelAxisMode = 'axis'
categoryAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a039cc>
categoryAxis.reverseDirection = 0
categoryAxis.strokeColor = Color(0,0,0)
categoryAxis.strokeDashArray = None
categoryAxis.strokeWidth = 1
categoryAxis.style = 'parallel'
categoryAxis.tickLeft = 5
categoryAxis.tickRight = 0
categoryAxis.tickShift = 0
categoryAxis.visible = 1
categoryAxis.visibleAxis = 1
categoryAxis.visibleGrid = 0
categoryAxis.visibleLabels = 1
categoryAxis.visibleTicks = 1
data = [(100, 110, 120, 130), (70, 80, 85, 90)]
debug = 0
fillColor = None
groupSpacing = 5
height = 85
naLabel = None
reversePlotOrder = 0
strokeColor = None
strokeWidth = 1
useAbsolute = 0
valueAxis.avoidBoundFrac = None
valueAxis.forceZero = 0
valueAxis.gridEnd = 0
valueAxis.gridStart = 0
valueAxis.gridStrokeColor = Color(0,0,0)
valueAxis.gridStrokeDashArray = None
valueAxis.gridStrokeWidth = 0.25
valueAxis.joinAxis = None
valueAxis.joinAxisMode = None
valueAxis.joinAxisPos = None
valueAxis.labelTextFormat = None
valueAxis.labelTextPostFormat = None
valueAxis.labelTextScale = None
valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a03a4c>
valueAxis.maximumTicks = 7
valueAxis.minimumTickSpacing = 10
valueAxis.rangeRound = 'none'
valueAxis.strokeColor = Color(0,0,0)
valueAxis.strokeDashArray = None
valueAxis.strokeWidth = 1
valueAxis.style = 'normal'
valueAxis.tickDown = 5
valueAxis.tickUp = 0
valueAxis.valueMax = None
valueAxis.valueMin = None
valueAxis.valueStep = None
valueAxis.visible = 1
valueAxis.visibleAxis = 1
valueAxis.visibleGrid = 0
valueAxis.visibleLabels = 1
valueAxis.visibleTicks = 1
valueAxis.zrangePref = 0
```

```
width = 180
x = 20
y = 10
```

53

## SampleH5c4(Drawing)

Simple bar chart with absolute spacing.

*Example*

```
def __init__(self,width=400,height=200,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    bc = HorizontalBarChart()
    bc.x = 50
    bc.y = 50
    bc.height = 125
    bc.width = 300
    bc.data = dataSample5
    bc.strokeColor = colors.black

    bc.useAbsolute = 1
    bc.barWidth = 10
    bc.groupSpacing = 20
    bc.barSpacing = 10

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    self.add(bc,name='HBC')
```
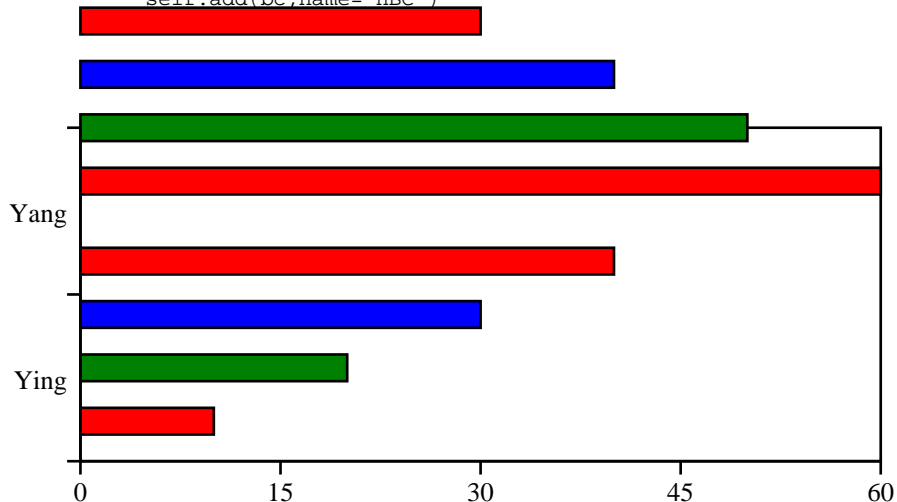
## VerticalBarChart(BarChart)

Vertical bar chart with multiple side-by-side bars.

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**barLabelArray** explicit array of bar label values, must match size of data if present.

**barLabelCallOut** Callout function(label) label._callOutInfo = (self,g,rowNo,colNo,x,y,width,height,x00,y00,x0,y0)

**barLabelFormat** Formatting string or function used for bar labels.

**barLabels** Handle to the list of bar labels.

**barSpacing** Width between individual bars.

**barWidth** The width of an individual bar.

**bars** Handle of the individual bars.

**categoryAxis** Handle of the category axis.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**groupSpacing** Width between groups of bars.

**height** Height of the chart.

**naLabel** Label to use for N/A values.

**reversePlotOrder** If true, reverse common category plot order.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**useAbsolute** Flag to use absolute spacing values.

**valueAxis** Handle of the value axis.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

*Example*

```
def demo(self):
    """Shows basic use of a bar chart"""
    if self.__class__.__name__=='BarChart':
        raise NotImplementedError, 'Abstract Class BarChart has no demo'
    drawing = Drawing(200, 100)
    bc = self.__class__()
    drawing.add(bc)
    return drawing
```

*Properties of Example Widget*

```
background = None
barLabelArray = None
barLabelFormat = None
barLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a0e36c>
barSpacing = 0
barWidth = 10
bars = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a0e3ac>
categoryAxis.categoryNames = None
categoryAxis.gridEnd = 0
categoryAxis.gridStart = 0
categoryAxis.gridStrokeColor = Color(0,0,0)
categoryAxis.gridStrokeDashArray = None
categoryAxis.gridStrokeWidth = 0.25
categoryAxis.joinAxis = None
categoryAxis.joinAxisMode = None
categoryAxis.joinAxisPos = None
categoryAxis.labelAxisMode = 'axis'
categoryAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a0e2ec>
categoryAxis.reverseDirection = 0
categoryAxis.strokeColor = Color(0,0,0)
```

```
categoryAxis.strokeDashArray = None
categoryAxis.strokeWidth = 1
categoryAxis.style = 'parallel'
categoryAxis.tickDown = 5
categoryAxis.tickShift = 0
categoryAxis.tickUp = 0
categoryAxis.visible = 1
categoryAxis.visibleAxis = 1
categoryAxis.visibleGrid = 0
categoryAxis.visibleLabels = 1
categoryAxis.visibleTicks = 1
data = [(100, 110, 120, 130), (70, 80, 85, 90)]
debug = 0
fillColor = None
groupSpacing = 5
height = 85
naLabel = None
reversePlotOrder = 0
strokeColor = None
strokeWidth = 1
useAbsolute = 0
valueAxis.avoidBoundFrac = None
valueAxis.forceZero = 0
valueAxis.gridEnd = 0
valueAxis.gridStart = 0
valueAxis.gridStrokeColor = Color(0,0,0)
valueAxis.gridStrokeDashArray = None
valueAxis.gridStrokeWidth = 0.25
valueAxis.joinAxis = None
valueAxis.joinAxisMode = None
valueAxis.joinAxisPos = None
valueAxis.labelTextFormat = None
valueAxis.labelTextPostFormat = None
valueAxis.labelTextScale = None
valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a0e24c>
valueAxis.maximumTicks = 7
valueAxis.minimumTickSpacing = 10
valueAxis.rangeRound = 'none'
valueAxis.strokeColor = Color(0,0,0)
valueAxis.strokeDashArray = None
valueAxis.strokeWidth = 1
valueAxis.style = 'normal'
valueAxis.tickLeft = 5
valueAxis.tickRight = 0
valueAxis.valueMax = None
valueAxis.valueMin = None
valueAxis.valueStep = None
valueAxis.visible = 1
valueAxis.visibleAxis = 1
valueAxis.visibleGrid = 0
valueAxis.visibleLabels = 1
valueAxis.visibleTicks = 1
valueAxis.zrangePref = 0
width = 180
x = 20
y = 10
```

## **VerticalBarChart3D(BarChart3D, VerticalBarChart)**

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**barLabelArray** explicit array of bar label values, must match size of data if present.

**barLabelCallOut** Callout function(label) label._callOutInfo = (self,g,rowNo,colNo,x,y,width,height,x00,y00,x0,y0)

**barLabelFormat** Formatting string or function used for bar labels.

**barLabels** Handle to the list of bar labels.

**barSpacing** Width between individual bars.

**barWidth** The width of an individual bar.

**bars** Handle of the individual bars.

**categoryAxis** Handle of the category axis.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**groupSpacing** Width between groups of bars.

**height** Height of the chart.

**naLabel** Label to use for N/A values.

**reversePlotOrder** If true, reverse common category plot order.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**theta_x** dx/dz

**theta_y** dy/dz

**useAbsolute** Flag to use absolute spacing values.

**valueAxis** Handle of the value axis.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

**zDepth** depth of an individual series

**zSpace** z gap around series

*Example*

```
def demo(self):
    """Shows basic use of a bar chart"""
    if self.__class__.__name__=='BarChart':
        raise NotImplementedError, 'Abstract Class BarChart has no demo'
    drawing = Drawing(200, 100)
    bc = self.__class__()
    drawing.add(bc)
    return drawing
```

*Properties of Example Widget*

```
background = None
barLabelArray = None
barLabelFormat = None
barLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a157ac>
barSpacing = 0
barWidth = 10
bars = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a157ec>
categoryAxis.categoryNames = None
categoryAxis.gridEnd = 0
categoryAxis.gridStart = 0
categoryAxis.gridStrokeColor = Color(0,0,0)
categoryAxis.gridStrokeDashArray = None
categoryAxis.gridStrokeWidth = 0.25
categoryAxis.joinAxis = None
categoryAxis.joinAxisMode = None
categoryAxis.joinAxisPos = None
categoryAxis.labelAxisMode = 'axis'
categoryAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a1570c>
categoryAxis.reverseDirection = 0
categoryAxis.strokeColor = Color(0,0,0)
categoryAxis.strokeDashArray = None
categoryAxis.strokeWidth = 1
categoryAxis.style = 'parallel'
categoryAxis.tickDown = 5
categoryAxis.tickShift = 0
categoryAxis.tickUp = 0
categoryAxis.visible = 1
categoryAxis.visibleAxis = 1
categoryAxis.visibleGrid = 0
categoryAxis.visibleLabels = 1
categoryAxis.visibleTicks = 1
data = [(100, 110, 120, 130), (70, 80, 85, 90)]
debug = 0
fillColor = None
groupSpacing = 5
height = 85
naLabel = None
reversePlotOrder = 0
strokeColor = None
strokeWidth = 1
useAbsolute = 0
valueAxis.avoidBoundFrac = None
valueAxis.forceZero = 0
valueAxis.gridEnd = 0
valueAxis.gridStart = 0
valueAxis.gridStrokeColor = Color(0,0,0)
valueAxis.gridStrokeDashArray = None
valueAxis.gridStrokeWidth = 0.25
valueAxis.joinAxis = None
valueAxis.joinAxisMode = None
valueAxis.joinAxisPos = None
valueAxis.labelTextFormat = None
valueAxis.labelTextPostFormat = None
valueAxis.labelTextScale = None
valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8a1576c>
valueAxis.maximumTicks = 7
valueAxis.minimumTickSpacing = 10
valueAxis.rangeRound = 'none'
valueAxis.strokeColor = Color(0,0,0)
valueAxis.strokeDashArray = None
valueAxis.strokeWidth = 1
valueAxis.style = 'normal'
valueAxis.tickLeft = 5
valueAxis.tickRight = 0
valueAxis.valueMax = None
valueAxis.valueMin = None
valueAxis.valueStep = None
valueAxis.visible = 1
valueAxis.visibleAxis = 1
valueAxis.visibleGrid = 0
valueAxis.visibleLabels = 1
valueAxis.visibleTicks = 1
valueAxis.zrangePref = 0
width = 180
```

```
x = 20
y = 10
```

```
x = 20
y = 10
```

## Functions

### **sampleH0a( ... )**

Make a slightly pathologic bar chart with only TWO data items.

*Example*

```
def sampleH0a():
    "Make a slightly pathologic bar chart with only TWO data items."

    drawing = Drawing(400, 200)

    data = [(13, 20)]

    bc = HorizontalBarChart()
    bc.x = 50
    bc.y = 50
    bc.height = 125
    bc.width = 300
    bc.data = data

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'se'
    bc.categoryAxis.labels.angle = 30
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```

### sampleH0b( ... )

Make a pathologic bar chart with only ONE data item.

*Example*

```
def sampleH0b():
    "Make a pathologic bar chart with only ONE data item."

    drawing = Drawing(400, 200)

    data = [(42,)]

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

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 50
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'se'
    bc.categoryAxis.labels.angle = 30
    bc.categoryAxis.categoryNames = ['Jan-99']

    drawing.add(bc)

    return drawing
```

## sampleH0c( ... )

Make a really pathologic bar chart with NO data items at all!

*Example*

```
def sampleH0c():
    "Make a really pathologic bar chart with NO data items at all!"

    drawing = Drawing(400, 200)

    data = [()]

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

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'se'
    bc.categoryAxis.labels.angle = 30
    bc.categoryAxis.categoryNames = []

    drawing.add(bc)

    return drawing
```

**sampleH1( ... )**

Sample of multi-series bar chart.

*Example*

```
def sampleH1():
    "Sample of multi-series bar chart."

    drawing = Drawing(400, 200)

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

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

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    catNames = string.split('Jan Feb Mar Apr May Jun Jul Aug', ' ')
    catNames = map(lambda n:n+'-99', catNames)
    bc.categoryAxis.categoryNames = catNames
    drawing.add(bc, 'barchart')

    return drawing
```
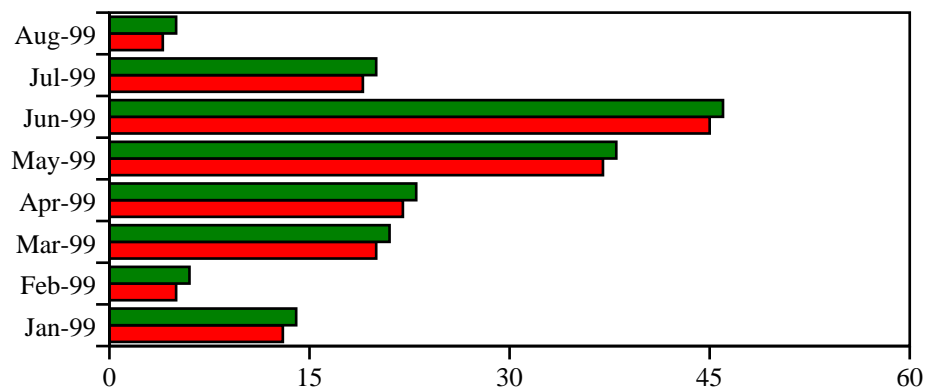
## **sampleH2a( ... )**

Sample of multi-series bar chart.

*Example*

```
def sampleH2a():
    "Sample of multi-series bar chart."

    data = [(2.4, -5.7, 2, 5, 9.2),
            (0.6, -4.9, -3, 4, 6.8)
            ]

    labels = ("Q3 2000", "Year to Date", "12 months",
              "Annualised\n3 years", "Since 07.10.99")

    drawing = Drawing(400, 200)

    bc = HorizontalBarChart()
    bc.x = 80
    bc.y = 50
    bc.height = 120
    bc.width = 300
    bc.data = data

    bc.barSpacing = 0
    bc.groupSpacing = 10
    bc.barWidth = 10

    bc.valueAxis.valueMin = -15
    bc.valueAxis.valueMax = +15
    bc.valueAxis.valueStep = 5
    bc.valueAxis.labels.fontName = 'Helvetica'
    bc.valueAxis.labels.fontSize = 8
    bc.valueAxis.labels.boxAnchor = 'n'    # irrelevant (becomes 'c')
    bc.valueAxis.labels.textAnchor = 'middle'
    bc.valueAxis.configure(bc.data)

    bc.categoryAxis.categoryNames = labels
    bc.categoryAxis.labels.fontName = 'Helvetica'
    bc.categoryAxis.labels.fontSize = 8
    bc.categoryAxis.labels.dx = -150

    drawing.add(bc)

    return drawing
```
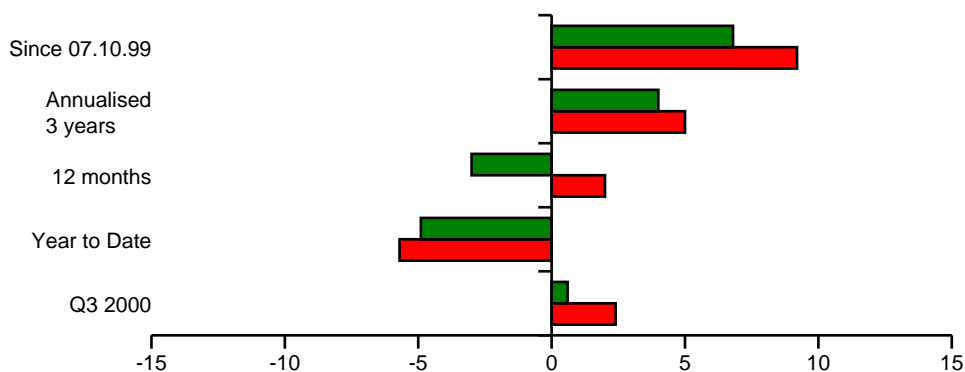
## sampleH2b( ... )

Sample of multi-series bar chart.

*Example*

```
def sampleH2b():
    "Sample of multi-series bar chart."

    data = [(2.4, -5.7, 2, 5, 9.2),
            (0.6, -4.9, -3, 4, 6.8)
            ]

    labels = ("Q3 2000", "Year to Date", "12 months",
              "Annualised\n3 years", "Since 07.10.99")

    drawing = Drawing(400, 200)

    bc = HorizontalBarChart()
    bc.x = 80
    bc.y = 50
    bc.height = 120
    bc.width = 300
    bc.data = data

    bc.barSpacing = 5
    bc.groupSpacing = 10
    bc.barWidth = 10

    bc.valueAxis.valueMin = -15
    bc.valueAxis.valueMax = +15
    bc.valueAxis.valueStep = 5
    bc.valueAxis.labels.fontName = 'Helvetica'
    bc.valueAxis.labels.fontSize = 8
    bc.valueAxis.labels.boxAnchor = 'n'    # irrelevant (becomes 'c')
    bc.valueAxis.labels.textAnchor = 'middle'

    bc.categoryAxis.categoryNames = labels
    bc.categoryAxis.labels.fontName = 'Helvetica'
    bc.categoryAxis.labels.fontSize = 8
    bc.categoryAxis.labels.dx = -150

    drawing.add(bc)

    return drawing
```

### sampleH2c( ... )

Sample of multi-series bar chart.

*Example*

```
def sampleH2c():
    "Sample of multi-series bar chart."

    data = [(2.4, -5.7, 2, 5, 9.99),
            (0.6, -4.9, -3, 4, 9.99)
            ]

    labels = ("Q3 2000", "Year to Date", "12 months",
              "Annualised\n3 years", "Since 07.10.99")

    drawing = Drawing(400, 200)

    bc = HorizontalBarChart()
    bc.x = 80
    bc.y = 50
    bc.height = 120
    bc.width = 300
    bc.data = data

    bc.barSpacing = 2
    bc.groupSpacing = 10
    bc.barWidth = 10

    bc.valueAxis.valueMin = -15
    bc.valueAxis.valueMax = +15
    bc.valueAxis.valueStep = 5
    bc.valueAxis.labels.fontName = 'Helvetica'
    bc.valueAxis.labels.fontSize = 8
    bc.valueAxis.labels.boxAnchor = 'n'
    bc.valueAxis.labels.textAnchor = 'middle'

    bc.categoryAxis.categoryNames = labels
    bc.categoryAxis.labels.fontName = 'Helvetica'
    bc.categoryAxis.labels.fontSize = 8
    bc.categoryAxis.labels.dx = -150

    bc.barLabels.nudge = 10

    bc.barLabelFormat = '%0.2f'
    bc.barLabels.dx = 0
    bc.barLabels.dy = 0
    bc.barLabels.boxAnchor = 'n'  # irrelevant (becomes 'c')
    bc.barLabels.fontName = 'Helvetica'
    bc.barLabels.fontSize = 6

    drawing.add(bc)

    return drawing
```

| | | | | | | |
|---|---|---|---|---|---|---|
| Since 07.10.99 | | | | 9.99 | | |
| | | | | 9.99 | | |
| Annualised 3 years | | | | 4.00 | | |
| | | | | 5.00 | | |
| 12 months | -3.00 | | | | | |
| | | | 2.00 | | | |
| Year to Date | -4.90 | | | | | |
| | -5.70 | | | | | |
| Q3 2000 | | | 0.60 | | | |
| | | | 2.40 | | | |

-15   -10   -5   0   5   10   15
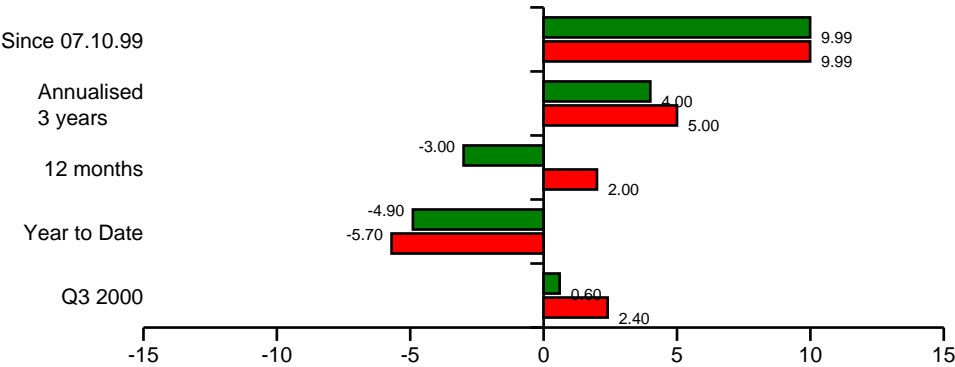
### sampleH3( ... )

A really horizontal bar chart (compared to the equivalent faked one).

*Example*

```
def sampleH3():
    "A really horizontal bar chart (compared to the equivalent faked one)."

    names = ("UK Equities", "US Equities", "European Equities", "Japanese Equities",
             "Pacific (ex Japan) Equities", "Emerging Markets Equities",
             "UK Bonds", "Overseas Bonds", "UK Index-Linked", "Cash")

    series1 = (-1.5, 0.3, 0.5, 1.0, 0.8, 0.7, 0.4, 0.1, 1.0, 0.3)
    series2 = (0.0, 0.33, 0.55, 1.1, 0.88, 0.77, 0.44, 0.11, 1.10, 0.33)

    assert len(names) == len(series1), "bad data"
    assert len(names) == len(series2), "bad data"

    drawing = Drawing(400, 200)

    bc = HorizontalBarChart()
    bc.x = 100
    bc.y = 20
    bc.height = 150
    bc.width = 250
    bc.data = (series1,)
    bc.bars.fillColor = colors.green

    bc.barLabelFormat = '%0.2f'
    bc.barLabels.dx = 0
    bc.barLabels.dy = 0
    bc.barLabels.boxAnchor = 'w' # irrelevant (becomes 'c')
    bc.barLabels.fontName = 'Helvetica'
    bc.barLabels.fontSize = 6
    bc.barLabels.nudge = 10

    bc.valueAxis.visible = 0
    bc.valueAxis.valueMin = -2
    bc.valueAxis.valueMax = +2
    bc.valueAxis.valueStep = 1

    bc.categoryAxis.tickLeft = 0
    bc.categoryAxis.tickRight = 0
    bc.categoryAxis.categoryNames = names
    bc.categoryAxis.labels.boxAnchor = 'w'
    bc.categoryAxis.labels.dx = -170
    bc.categoryAxis.labels.fontName = 'Helvetica'
    bc.categoryAxis.labels.fontSize = 6

    g = Group(bc)
    drawing.add(g)

    return drawing
```
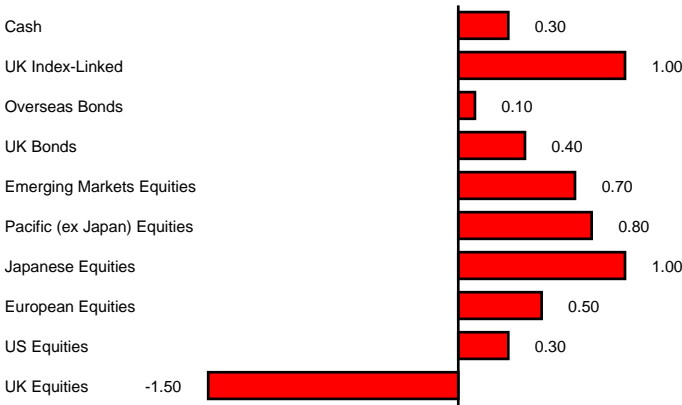
| | |
|---|---|
| Cash | 0.30 |
| UK Index-Linked | 1.00 |
| Overseas Bonds | 0.10 |
| UK Bonds | 0.40 |
| Emerging Markets Equities | 0.70 |
| Pacific (ex Japan) Equities | 0.80 |
| Japanese Equities | 1.00 |
| European Equities | 0.50 |
| US Equities | 0.30 |
| UK Equities | -1.50 |

## sampleH4a( ... )

A bar chart showing value axis region starting at *exactly* zero.

*Example*

```
def sampleH4a():
    "A bar chart showing value axis region starting at *exactly* zero."

    drawing = Drawing(400, 200)

    data = [(13, 20)]

    bc = HorizontalBarChart()
    bc.x = 50
    bc.y = 50
    bc.height = 125
    bc.width = 300
    bc.data = data

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
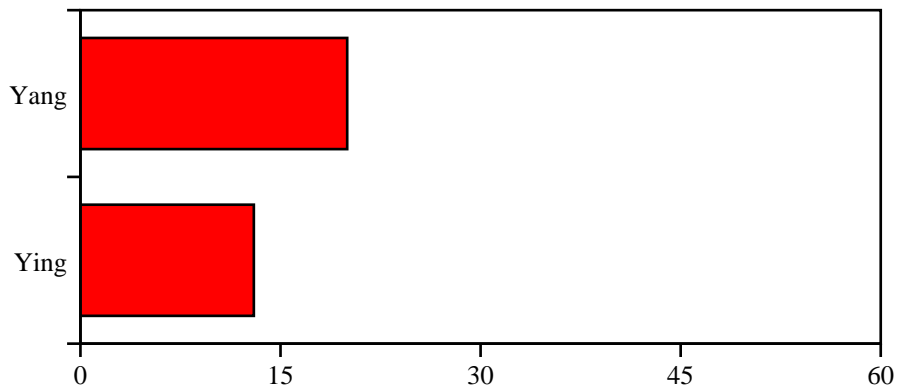
### sampleH4b( ... )

A bar chart showing value axis region starting *below* zero.

*Example*

```
def sampleH4b():
    "A bar chart showing value axis region starting *below* zero."

    drawing = Drawing(400, 200)

    data = [(13, 20)]

    bc = HorizontalBarChart()
    bc.x = 50
    bc.y = 50
    bc.height = 125
    bc.width = 300
    bc.data = data

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = -10
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
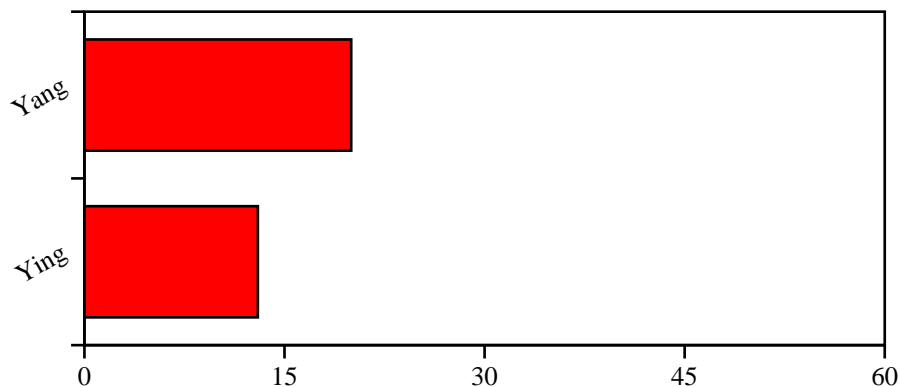
### sampleH4c( ... )

A bar chart showing value axis region starting *above* zero.

*Example*

```
def sampleH4c():
    "A bar chart showing value axis region starting *above* zero."

    drawing = Drawing(400, 200)

    data = [(13, 20)]

    bc = HorizontalBarChart()
    bc.x = 50
    bc.y = 50
    bc.height = 125
    bc.width = 300
    bc.data = data

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = 10
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
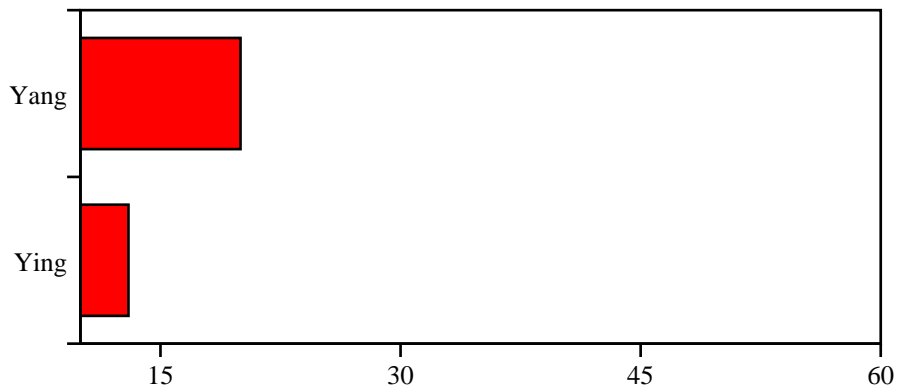
### sampleH4d( ... )

A bar chart showing value axis region entirely *below* zero.

*Example*

```
def sampleH4d():
    "A bar chart showing value axis region entirely *below* zero."

    drawing = Drawing(400, 200)

    data = [(-13, -20)]

    bc = HorizontalBarChart()
    bc.x = 50
    bc.y = 50
    bc.height = 125
    bc.width = 300
    bc.data = data

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = -30
    bc.valueAxis.valueMax = -10
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
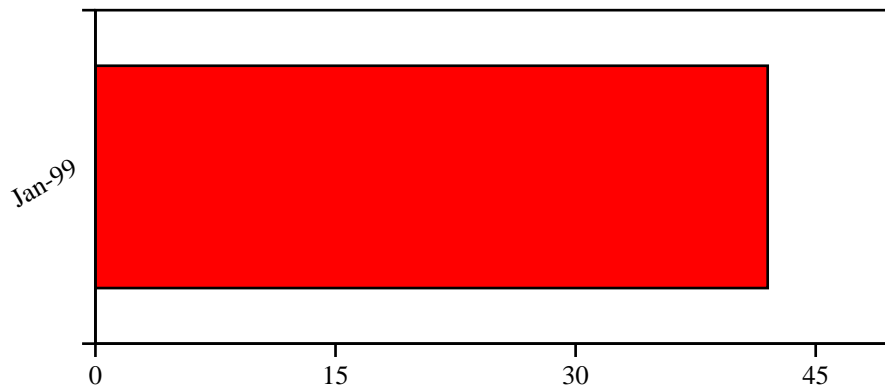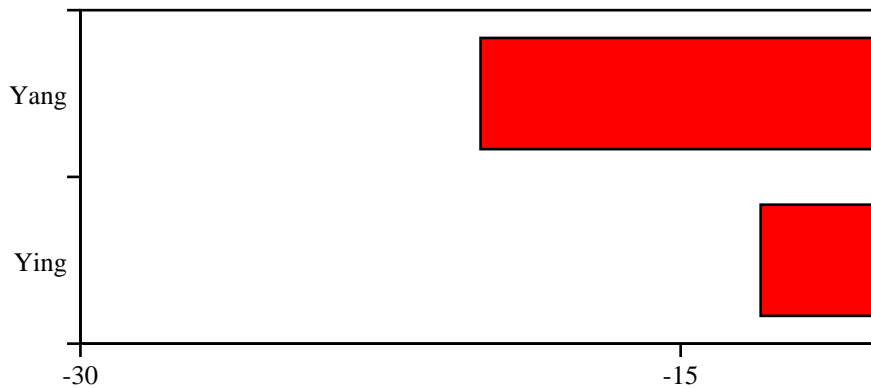
## sampleH5a( ... )

A simple bar chart with no expressed spacing attributes.

*Example*

```
def sampleH5a():
    "A simple bar chart with no expressed spacing attributes."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
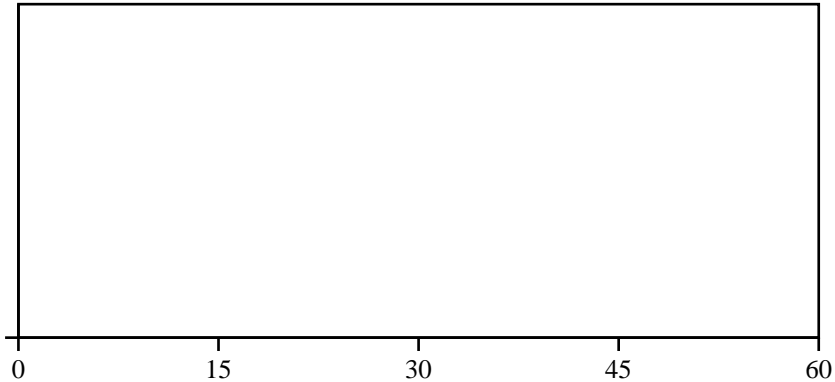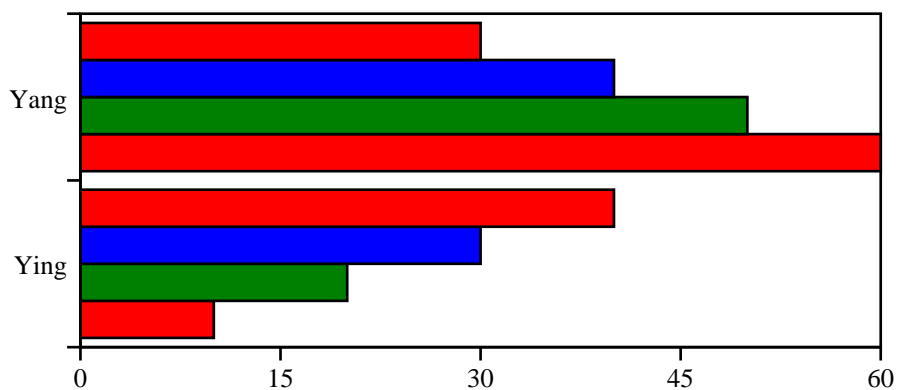
### sampleH5b( ... )

A simple bar chart with proportional spacing.

*Example*

```
def sampleH5b():
    "A simple bar chart with proportional spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.useAbsolute = 0
    bc.barWidth = 40
    bc.groupSpacing = 20
    bc.barSpacing = 10

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
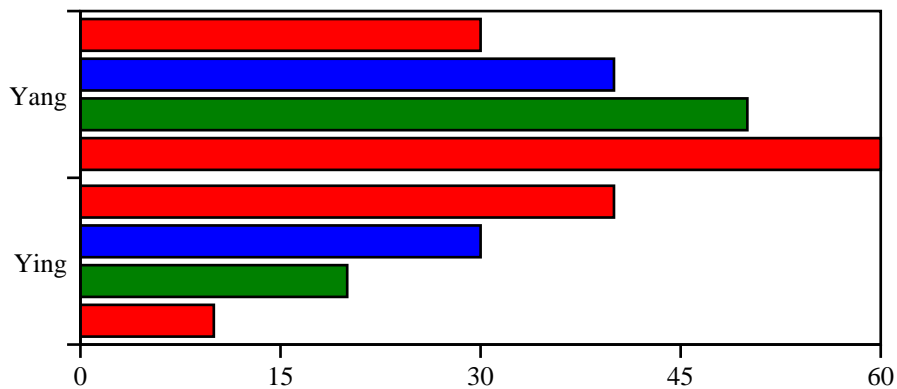
### sampleH5c1( ... )

A simple bar chart with absolute spacing.

*Example*

```
def sampleH5c1():
    "A simple bar chart with absolute spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.useAbsolute = 1
    bc.barWidth = 10
    bc.groupSpacing = 0
    bc.barSpacing = 0

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
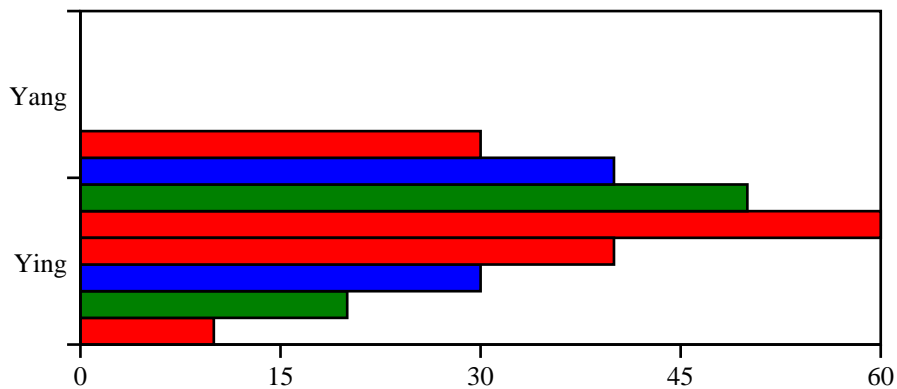
## **sampleH5c2( ... )**

Simple bar chart with absolute spacing.

*Example*

```
def sampleH5c2():
    "Simple bar chart with absolute spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.useAbsolute = 1
    bc.barWidth = 10
    bc.groupSpacing = 20
    bc.barSpacing = 0

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
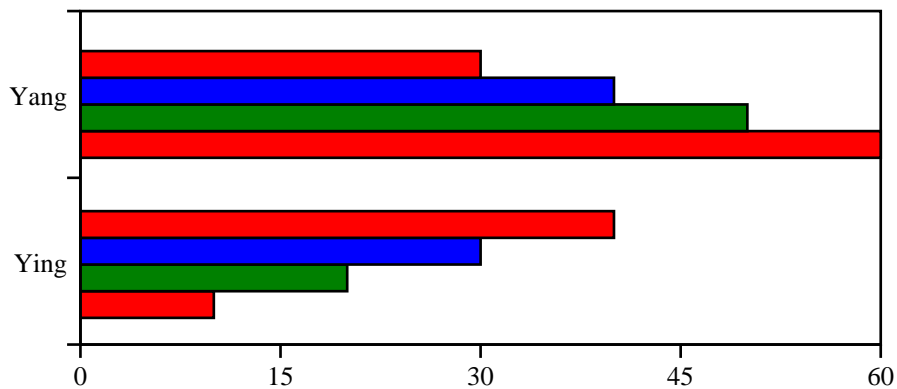
### sampleH5c3( ... )

Simple bar chart with absolute spacing.

*Example*

```
def sampleH5c3():
    "Simple bar chart with absolute spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

    bc = HorizontalBarChart()
    bc.x = 50
    bc.y = 20
    bc.height = 155
    bc.width = 300
    bc.data = data
    bc.strokeColor = colors.black

    bc.useAbsolute = 1
    bc.barWidth = 10
    bc.groupSpacing = 0
    bc.barSpacing = 2

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
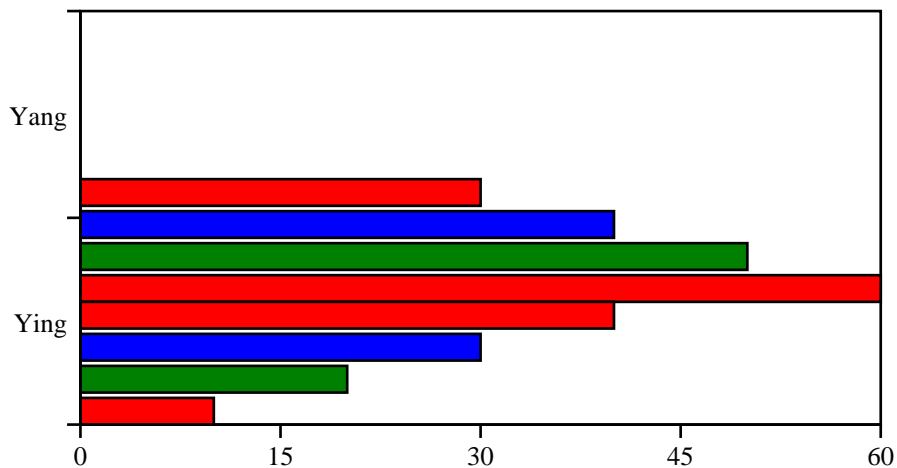
### **sampleH5c4( ... )**

Simple bar chart with absolute spacing.

*Example*

```
def sampleH5c4():
    "Simple bar chart with absolute spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.useAbsolute = 1
    bc.barWidth = 10
    bc.groupSpacing = 20
    bc.barSpacing = 10

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
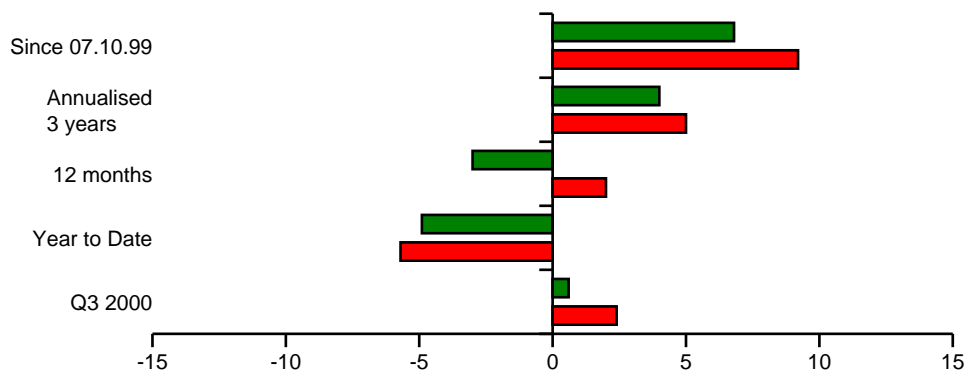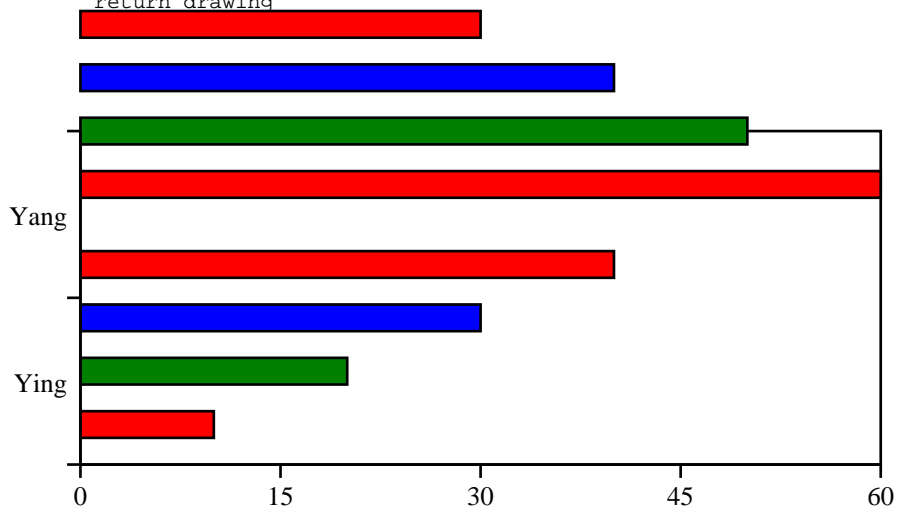
## sampleStacked1( ... )

Simple bar chart using symbol attribute.

*Example*

```
def sampleStacked1():
    "Simple bar chart using symbol attribute."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.barWidth = 10
    bc.groupSpacing = 15
    bc.valueAxis.valueMin = 0

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    from reportlab.graphics.widgets.grids import ShadedRect
    bc.bars.symbol = ShadedRect()
    bc.bars.symbol.fillColorStart = colors.red
    bc.bars.symbol.fillColorEnd = colors.white
    bc.bars.symbol.orientation = 'vertical'
    bc.bars.symbol.cylinderMode = 1
    bc.bars.symbol.strokeWidth = 0

    bc.bars[1].symbol = ShadedRect()
    bc.bars[1].symbol.fillColorStart = colors.magenta
    bc.bars[1].symbol.fillColorEnd = colors.white
    bc.bars[1].symbol.orientation = 'vertical'
    bc.bars[1].symbol.cylinderMode = 1
    bc.bars[1].symbol.strokeWidth = 0

    bc.bars[2].symbol = ShadedRect()
    bc.bars[2].symbol.fillColorStart = colors.green
    bc.bars[2].symbol.fillColorEnd = colors.white
    bc.bars[2].symbol.orientation = 'vertical'
    bc.bars[2].symbol.cylinderMode = 1
    bc.bars[2].symbol.strokeWidth = 0

    bc.bars[3].symbol = ShadedRect()
    bc.bars[3].symbol.fillColorStart = colors.blue
    bc.bars[3].symbol.fillColorEnd = colors.white
    bc.bars[3].symbol.orientation = 'vertical'
    bc.bars[3].symbol.cylinderMode = 1
    bc.bars[3].symbol.strokeWidth = 0

    drawing.add(bc)

    return drawing
```
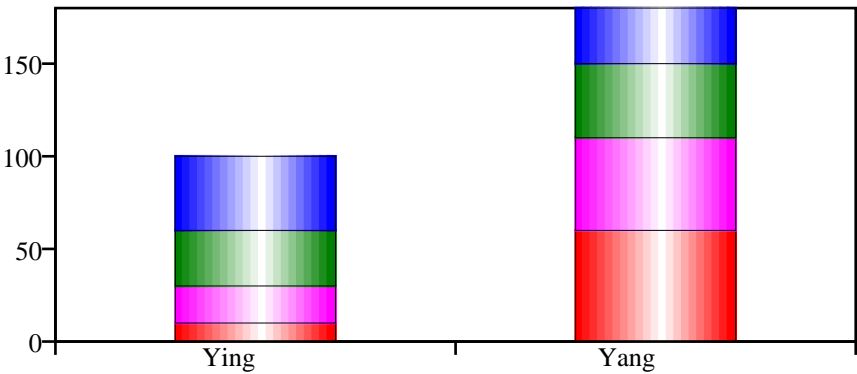
### sampleSymbol1( ... )

Simple bar chart using symbol attribute.

*Example*

```
def sampleSymbol1():
    "Simple bar chart using symbol attribute."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.barWidth = 10
    bc.groupSpacing = 15
    bc.barSpacing = 3

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'e'
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    from reportlab.graphics.widgets.grids import ShadedRect
    sym1 = ShadedRect()
    sym1.fillColorStart = colors.black
    sym1.fillColorEnd = colors.blue
    sym1.orientation = 'horizontal'
    sym1.strokeWidth = 0

    sym2 = ShadedRect()
    sym2.fillColorStart = colors.black
    sym2.fillColorEnd = colors.pink
    sym2.orientation = 'horizontal'
    sym2.strokeWidth = 0

    sym3 = ShadedRect()
    sym3.fillColorStart = colors.blue
    sym3.fillColorEnd = colors.white
    sym3.orientation = 'vertical'
    sym3.cylinderMode = 1
    sym3.strokeWidth = 0

    bc.bars.symbol = sym1
    bc.bars[2].symbol = sym2
    bc.bars[3].symbol = sym3

    drawing.add(bc)

    return drawing
```
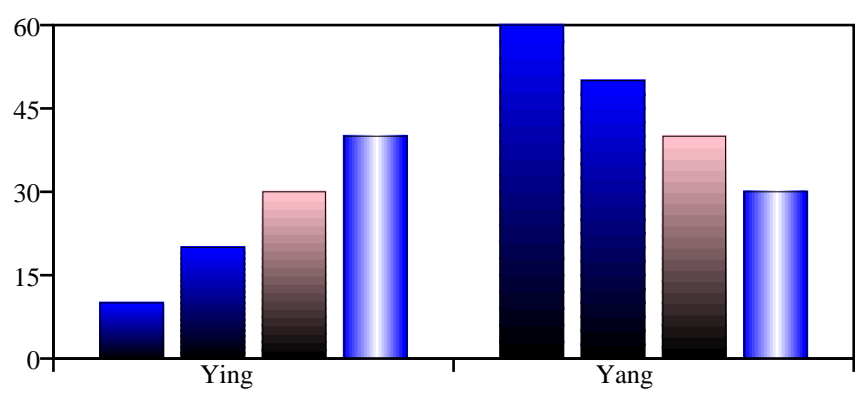
### sampleV0a( ... )

A slightly pathologic bar chart with only TWO data items.

*Example*

```
def sampleV0a():
    "A slightly pathologic bar chart with only TWO data items."

    drawing = Drawing(400, 200)

    data = [(13, 20)]

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

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'ne'
    bc.categoryAxis.labels.dx = 8
    bc.categoryAxis.labels.dy = -2
    bc.categoryAxis.labels.angle = 30
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
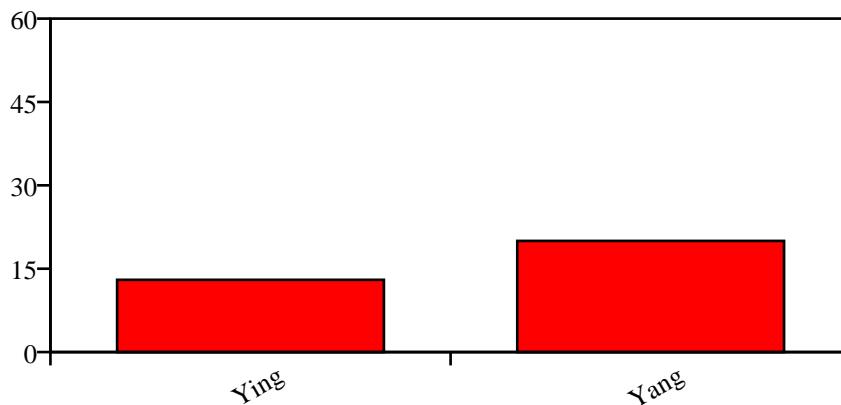
## sampleV0b( ... )

A pathologic bar chart with only ONE data item.

*Example*

```
def sampleV0b():
    "A pathologic bar chart with only ONE data item."

    drawing = Drawing(400, 200)

    data = [(42,)]

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

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 50
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'ne'
    bc.categoryAxis.labels.dx = 8
    bc.categoryAxis.labels.dy = -2
    bc.categoryAxis.labels.angle = 30
    bc.categoryAxis.categoryNames = ['Jan-99']

    drawing.add(bc)

    return drawing
```
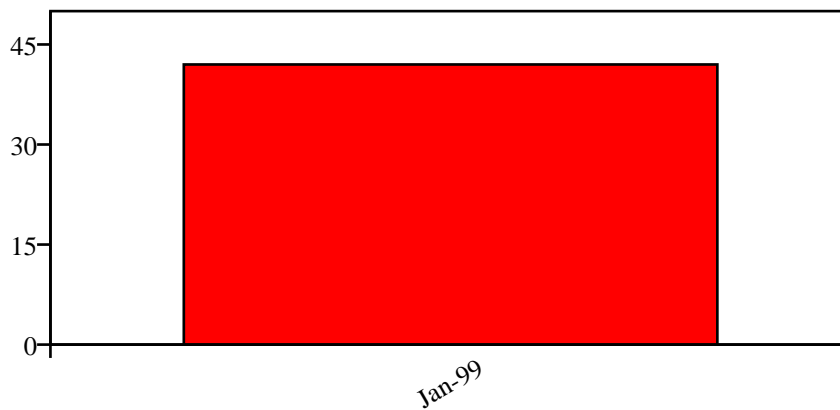
### sampleV0c( ... )

A really pathologic bar chart with NO data items at all!

*Example*

```
def sampleV0c():
    "A really pathologic bar chart with NO data items at all!"

    drawing = Drawing(400, 200)

    data = [()]

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

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'ne'
    bc.categoryAxis.labels.dx = 8
    bc.categoryAxis.labels.dy = -2
    bc.categoryAxis.categoryNames = []

    drawing.add(bc)

    return drawing
```
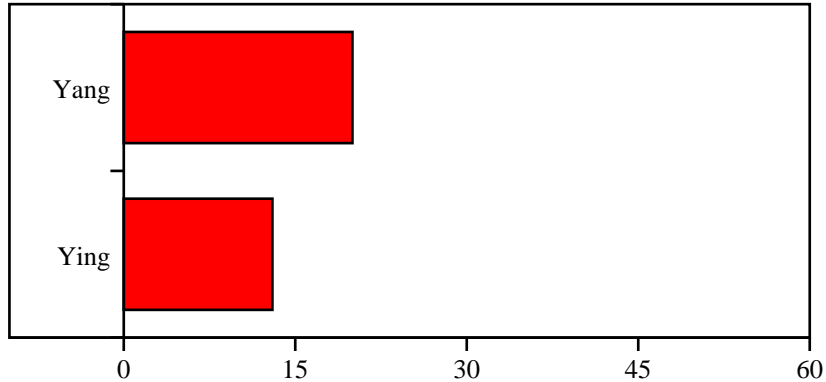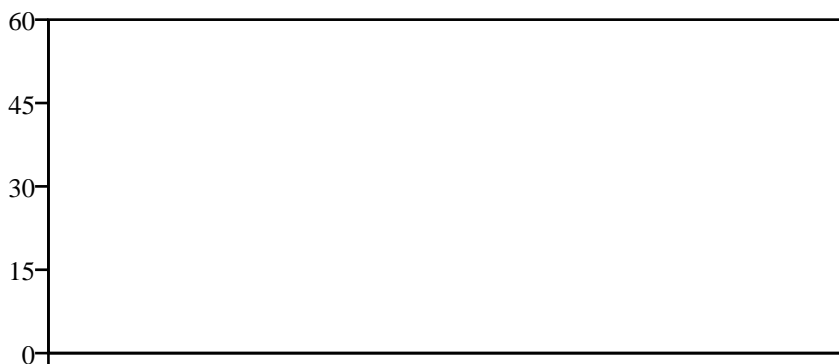
### **sampleV1( ... )**

Sample of multi-series bar chart.

*Example*

```
def sampleV1():
    "Sample of multi-series bar chart."

    drawing = Drawing(400, 200)

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

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

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'ne'
    bc.categoryAxis.labels.dx = 8
    bc.categoryAxis.labels.dy = -2
    bc.categoryAxis.labels.angle = 30

    catNames = string.split('Jan Feb Mar Apr May Jun Jul Aug', ' ')
    catNames = map(lambda n:n+'-99', catNames)
    bc.categoryAxis.categoryNames = catNames
    drawing.add(bc)

    return drawing
```
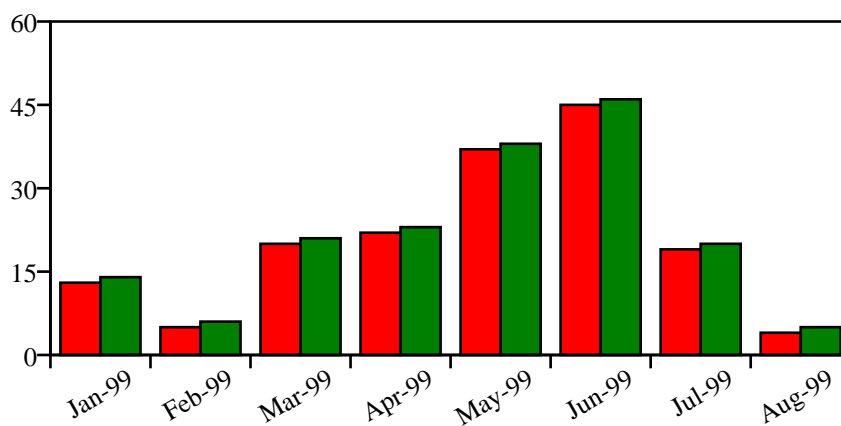
## sampleV2a( ... )

Sample of multi-series bar chart.

*Example*

```
def sampleV2a():
    "Sample of multi-series bar chart."

    data = [(2.4, -5.7, 2, 5, 9.2),
            (0.6, -4.9, -3, 4, 6.8)
            ]

    labels = ("Q3 2000", "Year to Date", "12 months",
              "Annualised\n3 years", "Since 07.10.99")

    drawing = Drawing(400, 200)

    bc = VerticalBarChart()
    bc.x = 50
    bc.y = 50
    bc.height = 120
    bc.width = 300
    bc.data = data

    bc.barSpacing = 0
    bc.groupSpacing = 10
    bc.barWidth = 10

    bc.valueAxis.valueMin = -15
    bc.valueAxis.valueMax = +15
    bc.valueAxis.valueStep = 5
    bc.valueAxis.labels.fontName = 'Helvetica'
    bc.valueAxis.labels.fontSize = 8
    bc.valueAxis.labels.boxAnchor = 'n'    # irrelevant (becomes 'c')
    bc.valueAxis.labels.textAnchor = 'middle'

    bc.categoryAxis.categoryNames = labels
    bc.categoryAxis.labels.fontName = 'Helvetica'
    bc.categoryAxis.labels.fontSize = 8
    bc.categoryAxis.labels.dy = -60

    drawing.add(bc)

    return drawing
```

## sampleV2b( ... )

Sample of multi-series bar chart.

*Example*

```
def sampleV2b():
    "Sample of multi-series bar chart."

    data = [(2.4, -5.7, 2, 5, 9.2),
            (0.6, -4.9, -3, 4, 6.8)
            ]

    labels = ("Q3 2000", "Year to Date", "12 months",
              "Annualised\n3 years", "Since 07.10.99")

    drawing = Drawing(400, 200)

    bc = VerticalBarChart()
    bc.x = 50
    bc.y = 50
    bc.height = 120
    bc.width = 300
    bc.data = data

    bc.barSpacing = 5
    bc.groupSpacing = 10
    bc.barWidth = 10

    bc.valueAxis.valueMin = -15
    bc.valueAxis.valueMax = +15
    bc.valueAxis.valueStep = 5
    bc.valueAxis.labels.fontName = 'Helvetica'
    bc.valueAxis.labels.fontSize = 8
    bc.valueAxis.labels.boxAnchor = 'n'    # irrelevant (becomes 'c')
    bc.valueAxis.labels.textAnchor = 'middle'

    bc.categoryAxis.categoryNames = labels
    bc.categoryAxis.labels.fontName = 'Helvetica'
    bc.categoryAxis.labels.fontSize = 8
    bc.categoryAxis.labels.dy = -60

    drawing.add(bc)

    return drawing
```

## sampleV2c( ... )

Sample of multi-series bar chart.

*Example*

```
def sampleV2c():
    "Sample of multi-series bar chart."

    data = [(2.4, -5.7, 2, 5, 9.99),
            (0.6, -4.9, -3, 4, 9.99)
            ]

    labels = ("Q3 2000", "Year to Date", "12 months",
              "Annualised\n3 years", "Since 07.10.99")

    drawing = Drawing(400, 200)

    bc = VerticalBarChart()
    bc.x = 50
    bc.y = 50
    bc.height = 120
    bc.width = 300
    bc.data = data

    bc.barSpacing = 2
    bc.groupSpacing = 10
    bc.barWidth = 10

    bc.valueAxis.valueMin = -15
    bc.valueAxis.valueMax = +15
    bc.valueAxis.valueStep = 5
    bc.valueAxis.labels.fontName = 'Helvetica'
    bc.valueAxis.labels.fontSize = 8

    bc.categoryAxis.categoryNames = labels
    bc.categoryAxis.labels.fontName = 'Helvetica'
    bc.categoryAxis.labels.fontSize = 8
    bc.valueAxis.labels.boxAnchor = 'n'
    bc.valueAxis.labels.textAnchor = 'middle'
    bc.categoryAxis.labels.dy = -60

    bc.barLabels.nudge = 10

    bc.barLabelFormat = '%0.2f'
    bc.barLabels.dx = 0
    bc.barLabels.dy = 0
    bc.barLabels.boxAnchor = 'n'  # irrelevant (becomes 'c')
    bc.barLabels.fontName = 'Helvetica'
    bc.barLabels.fontSize = 6

    drawing.add(bc)

    return drawing
```
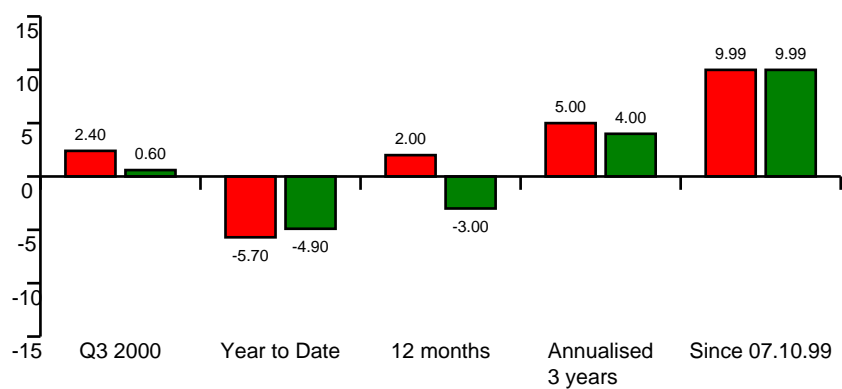
## sampleV3( ... )

Faked horizontal bar chart using a vertical real one (deprecated).

*Example*

```
def sampleV3():
    "Faked horizontal bar chart using a vertical real one (deprecated)."

    names = ("UK Equities", "US Equities", "European Equities", "Japanese Equities",
             "Pacific (ex Japan) Equities", "Emerging Markets Equities",
             "UK Bonds", "Overseas Bonds", "UK Index-Linked", "Cash")

    series1 = (-1.5, 0.3, 0.5, 1.0, 0.8, 0.7, 0.4, 0.1, 1.0, 0.3)
    series2 = (0.0, 0.33, 0.55, 1.1, 0.88, 0.77, 0.44, 0.11, 1.10, 0.33)

    assert len(names) == len(series1), "bad data"
    assert len(names) == len(series2), "bad data"

    drawing = Drawing(400, 200)

    bc = VerticalBarChart()
    bc.x = 0
    bc.y = 0
    bc.height = 100
    bc.width = 150
    bc.data = (series1,)
    bc.bars.fillColor = colors.green

    bc.barLabelFormat = '%0.2f'
    bc.barLabels.dx = 0
    bc.barLabels.dy = 0
    bc.barLabels.boxAnchor = 'w' # irrelevant (becomes 'c')
    bc.barLabels.angle = 90
    bc.barLabels.fontName = 'Helvetica'
    bc.barLabels.fontSize = 6
    bc.barLabels.nudge = 10

    bc.valueAxis.visible = 0
    bc.valueAxis.valueMin = -2
    bc.valueAxis.valueMax = +2
    bc.valueAxis.valueStep = 1

    bc.categoryAxis.tickUp = 0
    bc.categoryAxis.tickDown = 0
    bc.categoryAxis.categoryNames = names
    bc.categoryAxis.labels.angle = 90
    bc.categoryAxis.labels.boxAnchor = 'w'
    bc.categoryAxis.labels.dx = 0
    bc.categoryAxis.labels.dy = -125
    bc.categoryAxis.labels.fontName = 'Helvetica'
    bc.categoryAxis.labels.fontSize = 6

    g = Group(bc)
    g.translate(100, 175)
    g.rotate(-90)

    drawing.add(g)

    return drawing
```
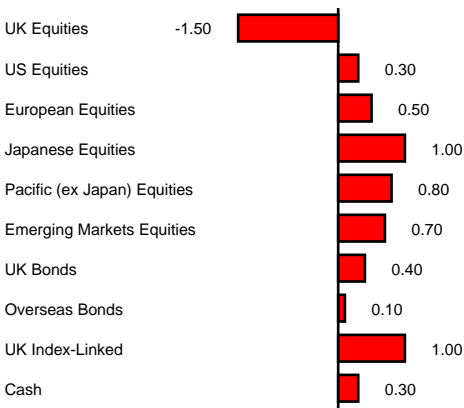
| | | |
|---|---|---|
| UK Equities | -1.50 | |
| US Equities | | 0.30 |
| European Equities | | 0.50 |
| Japanese Equities | | 1.00 |
| Pacific (ex Japan) Equities | | 0.80 |
| Emerging Markets Equities | | 0.70 |
| UK Bonds | | 0.40 |
| Overseas Bonds | | 0.10 |
| UK Index-Linked | | 1.00 |
| Cash | | 0.30 |

### sampleV4a( ... )

A bar chart showing value axis region starting at *exactly* zero.

*Example*

```
def sampleV4a():
    "A bar chart showing value axis region starting at *exactly* zero."

    drawing = Drawing(400, 200)

    data = [(13, 20)]

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

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
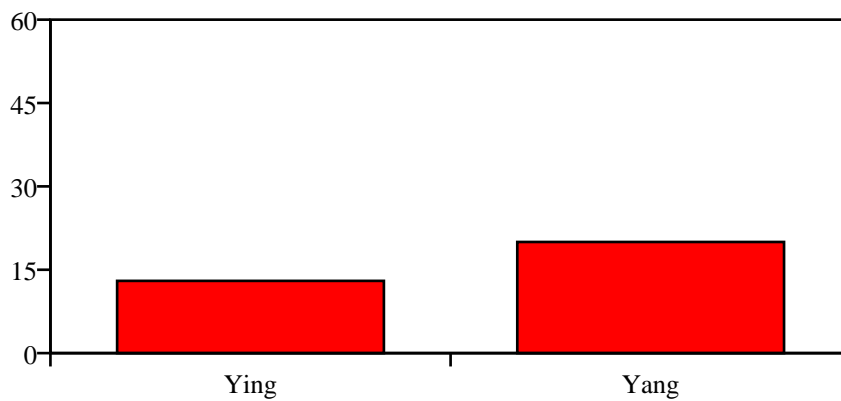
### sampleV4b( ... )

A bar chart showing value axis region starting *below* zero.

*Example*

```
def sampleV4b():
    "A bar chart showing value axis region starting *below* zero."

    drawing = Drawing(400, 200)

    data = [(13, 20)]

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

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = -10
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
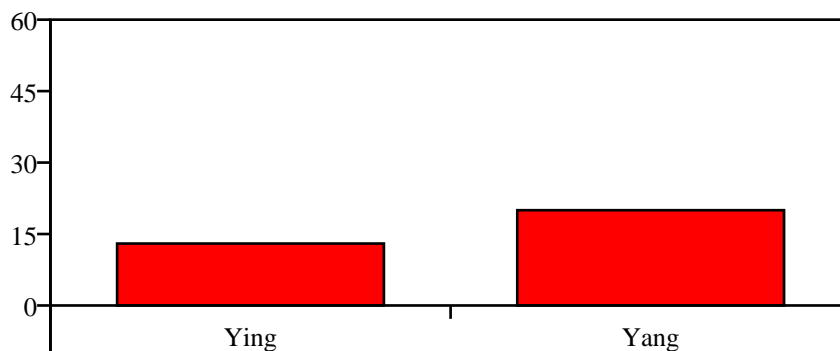
## sampleV4c( ... )

A bar chart showing value axis region staring *above* zero.

*Example*

```
def sampleV4c():
    "A bar chart showing value axis region staring *above* zero."

    drawing = Drawing(400, 200)

    data = [(13, 20)]

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

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = 10
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
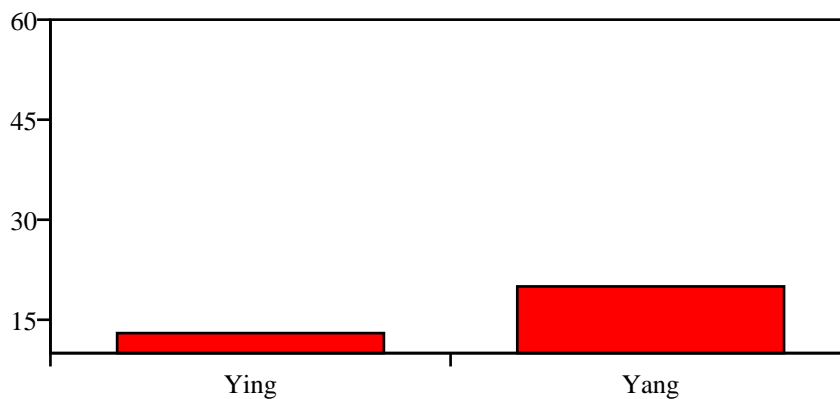
### sampleV4d( ... )

A bar chart showing value axis region entirely *below* zero.

*Example*

```
def sampleV4d():
    "A bar chart showing value axis region entirely *below* zero."

    drawing = Drawing(400, 200)

    data = [(-13, -20)]

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

    bc.strokeColor = colors.black

    bc.valueAxis.valueMin = -30
    bc.valueAxis.valueMax = -10
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
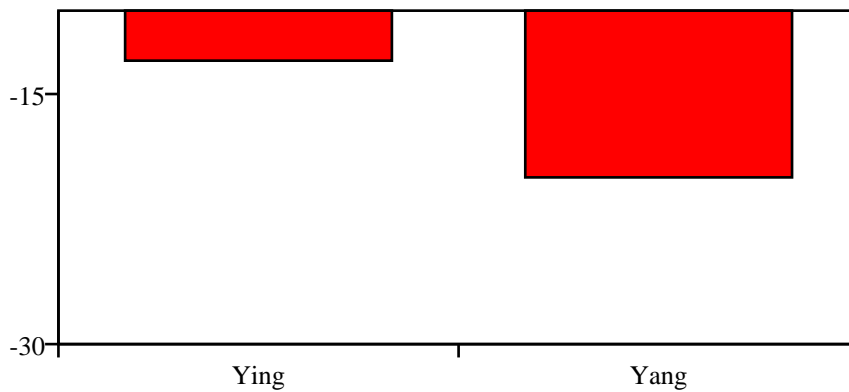
### sampleV5a( ... )

A simple bar chart with no expressed spacing attributes.

*Example*

```
def sampleV5a():
    "A simple bar chart with no expressed spacing attributes."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
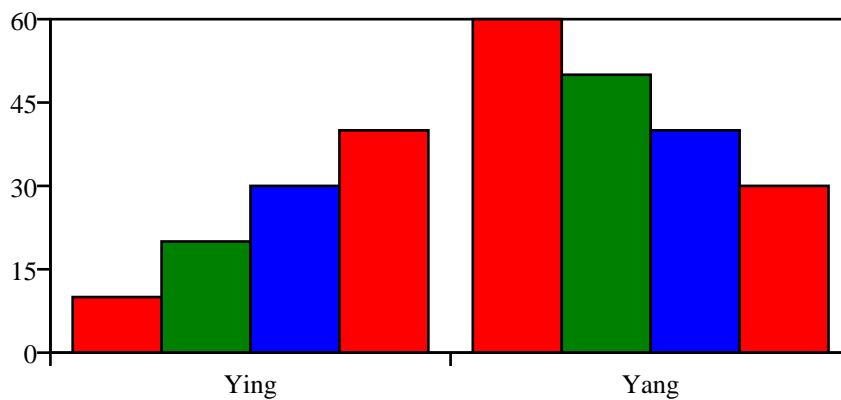
### sampleV5b( ... )

A simple bar chart with proportional spacing.

*Example*

```
def sampleV5b():
    "A simple bar chart with proportional spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.useAbsolute = 0
    bc.barWidth = 40
    bc.groupSpacing = 20
    bc.barSpacing = 10

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
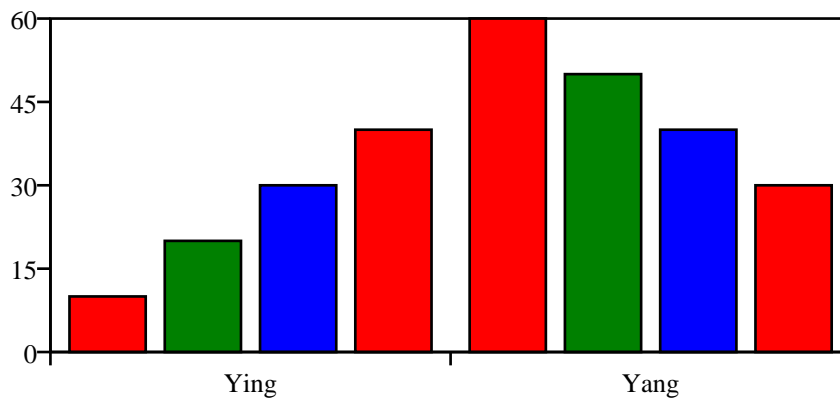
### sampleV5c1( ... )

Make sampe simple bar chart but with absolute spacing.

*Example*

```
def sampleV5c1():
    "Make sampe simple bar chart but with absolute spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.useAbsolute = 1
    bc.barWidth = 40
    bc.groupSpacing = 0
    bc.barSpacing = 0

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
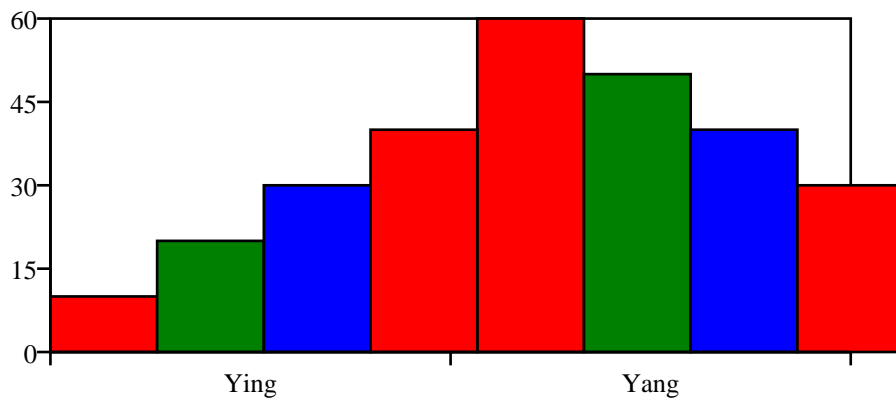
### sampleV5c2( ... )

Make sampe simple bar chart but with absolute spacing.

*Example*

```
def sampleV5c2():
    "Make sampe simple bar chart but with absolute spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.useAbsolute = 1
    bc.barWidth = 40
    bc.groupSpacing = 20
    bc.barSpacing = 0

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
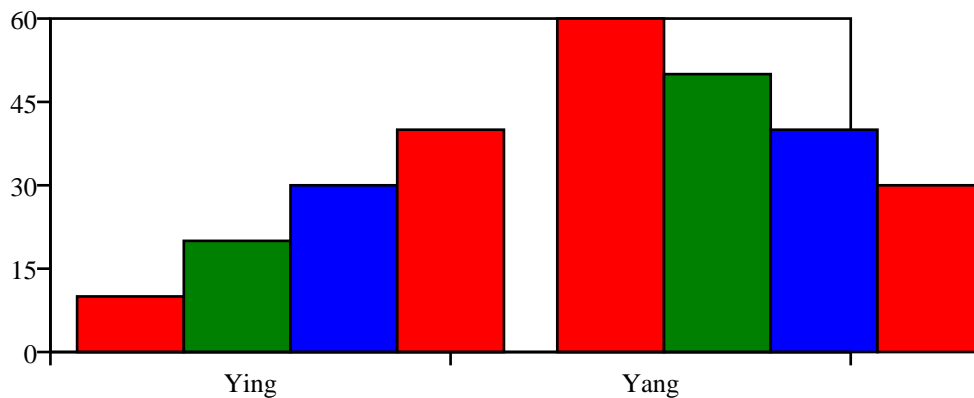
## sampleV5c3( ... )

Make sampe simple bar chart but with absolute spacing.

*Example*

```
def sampleV5c3():
    "Make sampe simple bar chart but with absolute spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.useAbsolute = 1
    bc.barWidth = 40
    bc.groupSpacing = 0
    bc.barSpacing = 10

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
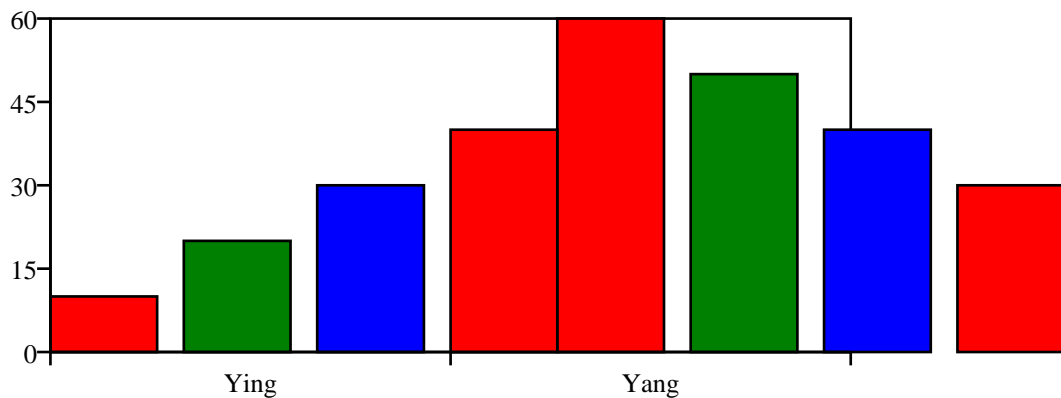
### sampleV5c4( ... )

Make sampe simple bar chart but with absolute spacing.

*Example*

```
def sampleV5c4():
    "Make sampe simple bar chart but with absolute spacing."

    drawing = Drawing(400, 200)

    data = dataSample5

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

    bc.useAbsolute = 1
    bc.barWidth = 40
    bc.groupSpacing = 20
    bc.barSpacing = 10

    bc.valueAxis.valueMin = 0
    bc.valueAxis.valueMax = 60
    bc.valueAxis.valueStep = 15

    bc.categoryAxis.labels.boxAnchor = 'n'
    bc.categoryAxis.labels.dy = -5
    bc.categoryAxis.categoryNames = ['Ying', 'Yang']

    drawing.add(bc)

    return drawing
```
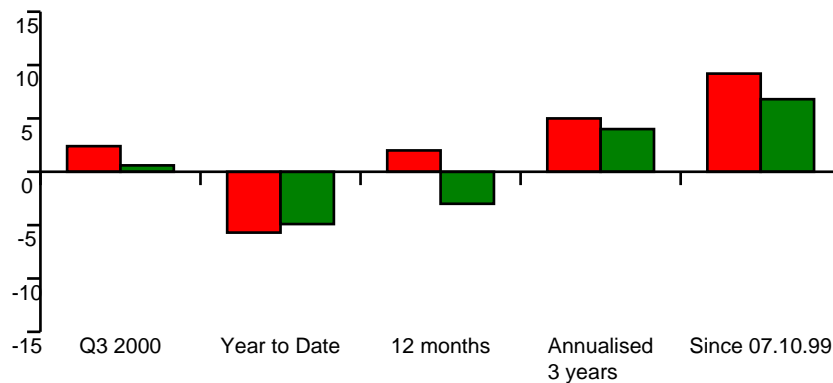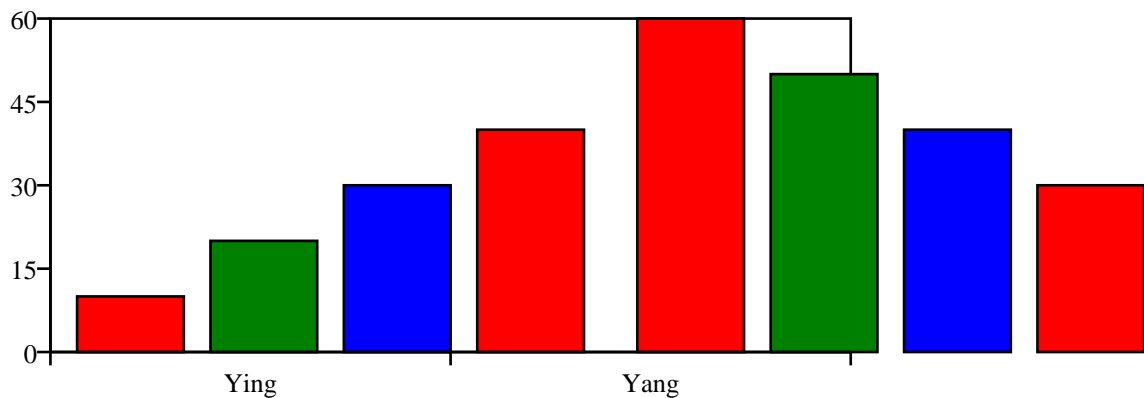
## dotbox

## Classes

### DotBox(Widget)

Returns a dotbox widget.

## Public Attributes

**dotColor** Colour of the circle on the box

**dotDiameter** Diameter of the circle used for the 'dot'

**dotXPosition** X Position of the circle

**dotYPosition** X Position of the circle

**gridColor** Colour for the box and gridding

**gridDivWidth** Width of each 'box'

**labelFontName** Name of font used for the labels

**labelFontSize** Size of font used for the labels

**labelOffset** Space between label text and grid edge

**strokeWidth** Width of the grid and dot outline

**x** X Position of dotbox

**xlabels** List of text labels for boxes on left hand side

**y** Y Position of dotbox

**ylabels** Text label for second box on left hand side

*Example*

```
def demo(self,drawing=None):
    if not drawing:
        tx,ty=self._getDrawingDimensions()
        drawing = Drawing(tx,ty)
    drawing.add(self.draw())
    return drawing
```
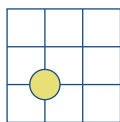


*Properties of Example Widget*

```
dotColor = Color(.909804,.878431,.466667)
dotDiameter = 11.338582677165356
dotXPosition = 1
dotYPosition = 1
gridColor = Color(.098039,.301961,.529412)
gridDivWidth = 14.173228346456693
labelFontName = 'Helvetica'
labelFontSize = 6
```

```
labelOffset = 5
strokeWidth = 0.5
x = 30
xlabels = ['Value', 'Blend', 'Growth']
y = 5
ylabels = ['Small', 'Medium', 'Large']
```

## **doughnut**

Doughnut chart

Produces a circular chart like the doughnut charts produced by Excel.
Can handle multiple series (which produce concentric 'rings' in the chart).

## Classes

## **Doughnut(AbstractPieChart)**

## Public Attributes

**data** list of numbers defining sector sizes; need not sum to 1

**direction** 'clockwise' or 'anticlockwise'

**height** height of doughnut bounding box. Need not be same as height.

**labels** optional list of labels to use for each data point

**simpleLabels** If true(default) use String not super duper WedgeLabel

**slices** collection of sector descriptor objects

**startAngle** angle of first slice; like the compass, 0 is due North

**width** width of doughnut bounding box. Need not be same as width.

**x** X position of the chart within its container.

**y** Y position of the chart within its container.

*Example*

```
def demo(self):
    d = Drawing(200, 100)

    dn = Doughnut()
    dn.x = 50
    dn.y = 10
    dn.width = 100
    dn.height = 80
    dn.data = [10,20,30,40,50,60]
    dn.labels = ['a','b','c','d','e','f']

    dn.slices.strokeWidth=0.5
    dn.slices[3].popout = 10
    dn.slices[3].strokeWidth = 2
    dn.slices[3].strokeDashArray = [2,2]
    dn.slices[3].labelRadius = 1.75
    dn.slices[3].fontColor = colors.red
    dn.slices[0].fillColor = colors.darkcyan
    dn.slices[1].fillColor = colors.blueviolet
    dn.slices[2].fillColor = colors.blue
    dn.slices[3].fillColor = colors.cyan
    dn.slices[4].fillColor = colors.aquamarine
    dn.slices[5].fillColor = colors.cadetblue
    dn.slices[6].fillColor = colors.lightcoral

    d.add(dn)
    return d
```
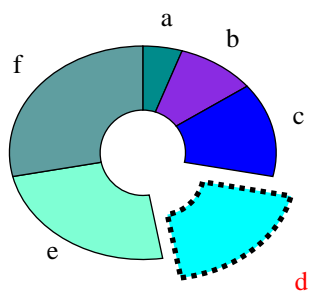
*Properties of Example Widget*

```
data = [1, 1]
direction = 'clockwise'
height = 100
labels = None
simpleLabels = 1
slices = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8ba7aec>
startAngle = 90
width = 100
x = 0
y = 0
```

## Functions

**sample1( ... )**

Make up something from the individual Sectors

*Example*

```
def sample1():
    "Make up something from the individual Sectors"

    d = Drawing(400, 400)
    g = Group()

    s1 = Wedge(centerx=200, centery=200, radius=150, startangledegrees=0, endangledegrees=120, radius
    s1.fillColor=colors.red
    s1.strokeColor=None
    d.add(s1)
    s2 = Wedge(centerx=200, centery=200, radius=150, startangledegrees=120, endangledegrees=240, radi
    s2.fillColor=colors.green
    s2.strokeColor=None
    d.add(s2)
    s3 = Wedge(centerx=200, centery=200, radius=150, startangledegrees=240, endangledegrees=260, radi
    s3.fillColor=colors.blue
    s3.strokeColor=None
    d.add(s3)
    s4 = Wedge(centerx=200, centery=200, radius=150, startangledegrees=260, endangledegrees=360, radi
    s4.fillColor=colors.gray
    s4.strokeColor=None
    d.add(s4)

    return d
```
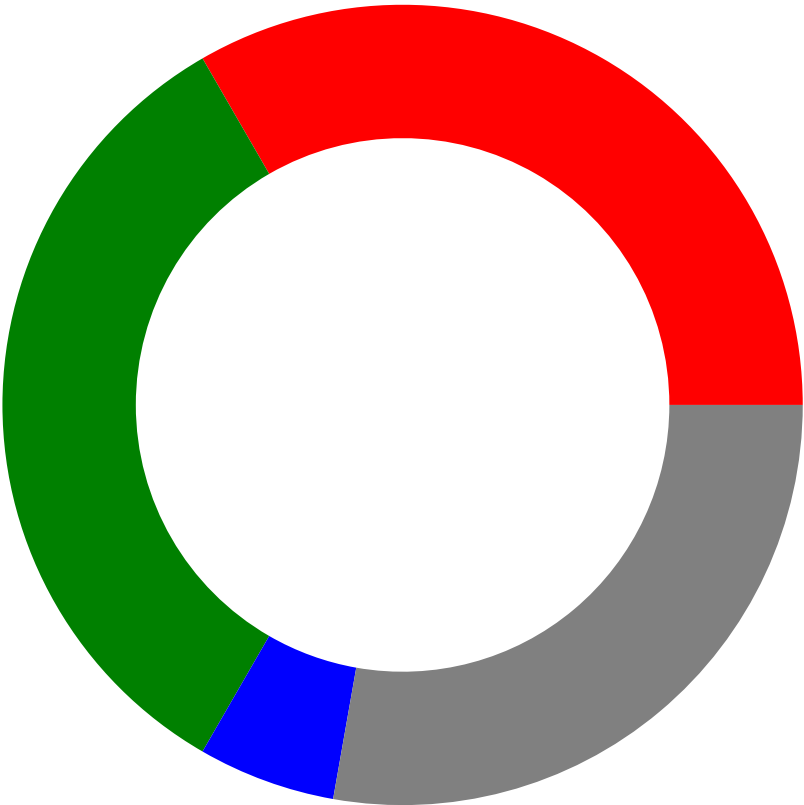
## sample2( ... )

Make a simple demo

*Example*

```
def sample2():
    "Make a simple demo"

    d = Drawing(400, 400)

    dn = Doughnut()
    dn.x = 50
    dn.y = 50
    dn.width = 300
    dn.height = 300
    dn.data = [10,20,30,40,50,60]

    d.add(dn)

    return d
```
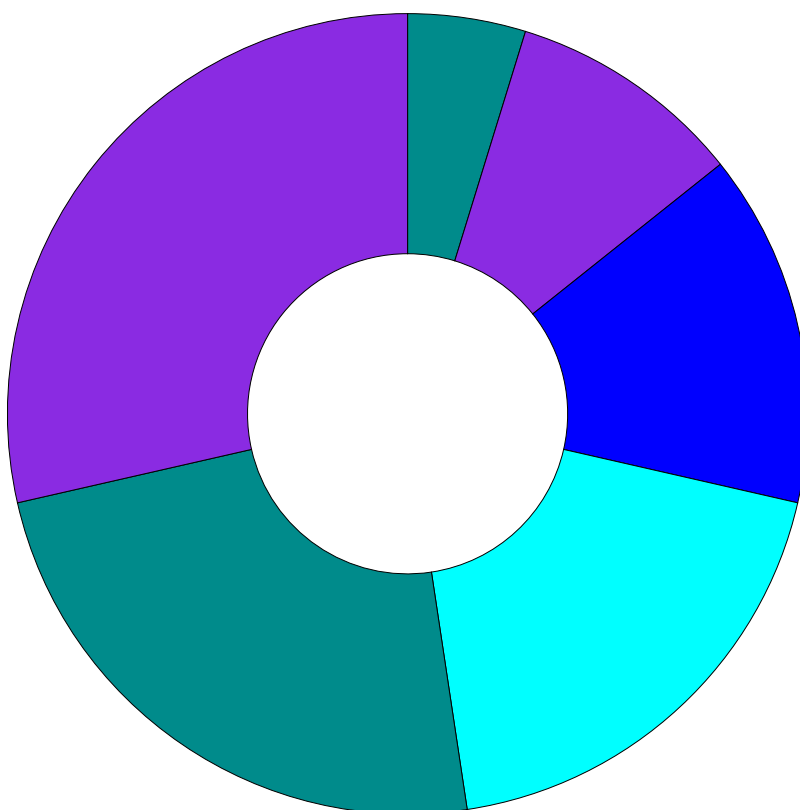
## sample3( ... )

Make a more complex demo

*Example*

```
def sample3():
    "Make a more complex demo"

    d = Drawing(400, 400)
    dn = Doughnut()
    dn.x = 50
    dn.y = 50
    dn.width = 300
    dn.height = 300
    dn.data = [[10,20,30,40,50,60], [10,20,30,40]]
    dn.labels = ['a','b','c','d','e','f']

    d.add(dn)

    return d
```
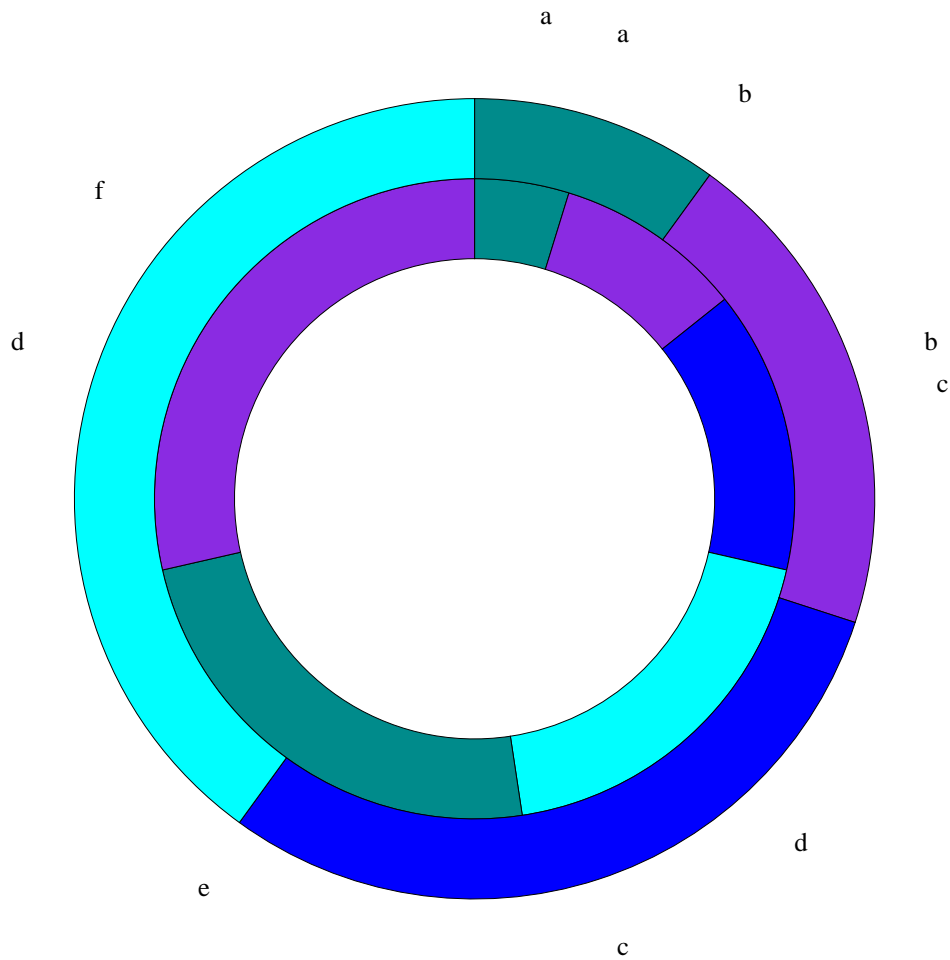
## `legends`

This will be a collection of legends to be used with charts.

## Classes

### `Legend(Widget)`

A simple legend containing rectangular swatches and strings.

The swatches are filled rectangles whenever the respective color object in 'colorNamePairs' is a subclass of Color in reportlab.lib.colors. Otherwise the object passed instead is assumed to have 'x', 'y', 'width' and 'height' attributes.
A legend then tries to set them or catches any error. This lets you plug-in any widget you like as a replacement for the default rectangular swatches.

Strings can be nicely aligned left or right to the swatches.

## Public Attributes

**alignment** Alignment of text with respect to swatches

**autoXPadding** x Padding between columns if deltax=None

**autoYPadding** y Padding between rows if deltay=None

**boxAnchor** Anchor point for the legend area

**callout** a user callout(self,g,x,y,(color,text))

**colEndCallout** a user callout(self,g, x, xt, y,width, lWidth)

**colorNamePairs** List of color/name tuples (color can also be widget)

**columnMaximum** Max. number of items per column

**deltax** x-distance between neighbouring swatches

**deltay** y-distance between neighbouring swatches

**dividerColor** dividerLines color

**dividerDashArray** Dash array for dividerLines.

**dividerLines** If 1 we have dividers between the rows | 2 for extra top | 4 for bottom

**dividerOffsX** divider lines X offsets

**dividerOffsY** dividerLines Y offset

**dividerWidth** dividerLines width

**dx** Width of swatch rectangle

**dxTextSpace** Distance between swatch rectangle and text

**dy** Height of swatch rectangle

**fillColor**

**fontName** Font name of the strings

**fontSize** Font size of the strings

**sepSpace** separator spacing

**strokeColor** Border color of the swatches

**strokeWidth** Width of the border color of the swatches

**swatchMarker** None, Auto() or makeMarker('Diamond') ...

**variColumn** If true column widths may vary (default is false)

**x** x-coordinate of upper-left reference point

**y** y-coordinate of upper-left reference point

**yGap** Additional gap between rows

*Example*

```
def demo(self):
    "Make sample legend."

    d = Drawing(200, 100)

    legend = Legend()
    legend.alignment = 'left'
    legend.x = 0
    legend.y = 100
    legend.dxTextSpace = 5
    items = 'red green blue yellow pink black white'.split()
    items = map(lambda i:(getattr(colors, i), i), items)
    legend.colorNamePairs = items

    d.add(legend, 'legend')

    return d
```

*Properties of Example Widget*

```
alignment = 'left'
autoXPadding = 5
autoYPadding = 2
boxAnchor = 'nw'
colEndCallout = None
colorNamePairs = [(Color(1,0,0), 'red'),
                  (Color(0,0,1), 'blue'),
                  (Color(0,.501961,0), 'green'),
                  (Color(1,.752941,.796078), 'pink'),
                  (Color(1,1,0), 'yellow')]
columnMaximum = 3
deltax = 75
deltay = 20
dividerColor = Color(0,0,0)
dividerDashArray = None
dividerLines = 0
dividerOffsX = (0, 0)
dividerOffsY = 0
dividerWidth = 0.5
dx = 10
dxTextSpace = 10
dy = 10
fillColor = Color(0,0,0)
fontName = 'Times-Roman'
fontSize = 10
sepSpace = 0
strokeColor = Color(0,0,0)
strokeWidth = 1
swatchMarker = None
variColumn = 0
x = 0
y = 0
yGap = 0
```

## `LineLegend(Legend)`

A subclass of Legend for drawing legends with lines as the swatches rather than rectangles. Useful for lineCharts and linePlots. Should be similar in all other ways the the standard Legend class.

# Public Attributes

**alignment** Alignment of text with respect to swatches

**autoXPadding** x Padding between columns if deltax=None

**autoYPadding** y Padding between rows if deltay=None

**boxAnchor** Anchor point for the legend area

**callout** a user callout(self,g,x,y,(color,text))

**colEndCallout** a user callout(self,g, x, xt, y,width, lWidth)

**colorNamePairs** List of color/name tuples (color can also be widget)

**columnMaximum** Max. number of items per column

**deltax** x-distance between neighbouring swatches

**deltay** y-distance between neighbouring swatches

**dividerColor** dividerLines color

**dividerDashArray** Dash array for dividerLines.

**dividerLines** If 1 we have dividers between the rows | 2 for extra top | 4 for bottom

**dividerOffsX** divider lines X offsets

**dividerOffsY** dividerLines Y offset

**dividerWidth** dividerLines width

**dx** Width of swatch rectangle

**dxTextSpace** Distance between swatch rectangle and text

**dy** Height of swatch rectangle

**fillColor**

**fontName** Font name of the strings

**fontSize** Font size of the strings

**sepSpace** separator spacing

**strokeColor** Border color of the swatches

**strokeWidth** Width of the border color of the swatches

**swatchMarker** None, Auto() or makeMarker('Diamond') ...

**variColumn** If true column widths may vary (default is false)

**x** x-coordinate of upper-left reference point

**y** y-coordinate of upper-left reference point

**yGap** Additional gap between rows

*Example*

```
def demo(self):
```

```
            "Make sample legend."

            d = Drawing(200, 100)

            legend = Legend()
            legend.alignment = 'left'
            legend.x = 0
            legend.y = 100
            legend.dxTextSpace = 5
            items = 'red green blue yellow pink black white'.split()
            items = map(lambda i:(getattr(colors, i), i), items)
            legend.colorNamePairs = items

            d.add(legend, 'legend')

            return d
```

*Properties of Example Widget*

```
        alignment = 'left'
        autoXPadding = 5
        autoYPadding = 2
        boxAnchor = 'nw'
        colEndCallout = None
        colorNamePairs = [(Color(1,0,0), 'red'),
                          (Color(0,0,1), 'blue'),
                          (Color(0,.501961,0), 'green'),
                          (Color(1,.752941,.796078), 'pink'),
                          (Color(1,1,0), 'yellow')]
        columnMaximum = 3
        deltax = 75
        deltay = 20
        dividerColor = Color(0,0,0)
        dividerDashArray = None
        dividerLines = 0
        dividerOffsX = (0, 0)
        dividerOffsY = 0
        dividerWidth = 0.5
        dx = 10
        dxTextSpace = 10
        dy = 2
        fillColor = Color(0,0,0)
        fontName = 'Times-Roman'
        fontSize = 10
        sepSpace = 0
        strokeColor = Color(0,0,0)
        strokeWidth = 1
        swatchMarker = None
        variColumn = 0
        x = 0
        y = 0
        yGap = 0
```

## `LineSwatch(Widget)`

basically a Line with properties added so it can be used in a LineLegend

## Public Attributes

**height** used for line strokeWidth

**strokeColor** color of swatch line

**strokeDashArray** dash array for swatch line

**width** length of swatch line

**x** x-coordinate for swatch line start point

**y** y-coordinate for swatch line start point

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

```
height = 1
strokeColor = Color(1,0,0)
strokeDashArray = None
width = 20
x = 0
y = 0
```

## Functions

### sample1c( ... )

Make sample legend.

*Example*

```
def sample1c():
    "Make sample legend."

    d = Drawing(200, 100)

    legend = Legend()
    legend.alignment = 'right'
    legend.x = 0
    legend.y = 100
    legend.dxTextSpace = 5
    items = 'red green blue yellow pink black white'.split()
    items = map(lambda i:(getattr(colors, i), i), items)
    legend.colorNamePairs = items

    d.add(legend, 'legend')

    return d
```

■ red         ■ yellow         □ white

■ green       □ pink

■ blue        ■ black

## sample2c( ... )

Make sample legend.

*Example*

```
def sample2c():
    "Make sample legend."

    d = Drawing(200, 100)

    legend = Legend()
    legend.alignment = 'right'
    legend.x = 20
    legend.y = 90
    legend.deltax = 60
    legend.dxTextSpace = 10
    legend.columnMaximum = 4
    items = 'red green blue yellow pink black white'.split()
    items = map(lambda i:(getattr(colors, i), i), items)
    legend.colorNamePairs = items

    d.add(legend, 'legend')

    return d
```

■ red          ■ pink

■ green        ■ black

■ blue         □ white

■ yellow

## sample3( ... )

Make sample legend with line swatches.

*Example*

```
def sample3():
    "Make sample legend with line swatches."

    d = Drawing(200, 100)

    legend = LineLegend()
    legend.alignment = 'right'
    legend.x = 20
    legend.y = 90
    legend.deltax = 60
    legend.dxTextSpace = 10
    legend.columnMaximum = 4
    items = 'red green blue yellow pink black white'.split()
    items = map(lambda i:(getattr(colors, i), i), items)
    legend.colorNamePairs = items
    d.add(legend, 'legend')

    return d
```

— red    — pink

— green   — black

— blue     white

— yellow

### sample3a( ... )

Make sample legend with line swatches and dasharrays on the lines.

*Example*

```
def sample3a():
    "Make sample legend with line swatches and dasharrays on the lines."

    d = Drawing(200, 100)

    legend = LineLegend()
    legend.alignment = 'right'
    legend.x = 20
    legend.y = 90
    legend.deltax = 60
    legend.dxTextSpace = 10
    legend.columnMaximum = 4
    items = 'red green blue yellow pink black white'.split()
    darrays = ([2,1], [2,5], [2,2,5,5], [1,2,3,4], [4,2,3,4], [1,2,3,4,5,6], [1])
    cnp = []
    for i in range(0, len(items)):
        l =  LineSwatch()
        l.strokeColor = getattr(colors, items[i])
        l.strokeDashArray = darrays[i]
        cnp.append((l, items[i]))
    legend.colorNamePairs = cnp
    d.add(legend, 'legend')

    return d
```

## linecharts

This modules defines a very preliminary Line Chart example.

# Classes

## AbstractLineChart(PlotArea)

# Public Attributes

**background** Handle to background object.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**height** Height of the chart.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

```
background = None
debug = 0
fillColor = None
height = 85
strokeColor = None
strokeWidth = 1
width = 180
x = 20
y = 10
```

### **`HorizontalLineChart(LineChart)`**

Line chart with multiple lines.

A line chart is assumed to have one category and one value axis.
Despite its generic name this particular line chart class has
a vertical value axis and a horizontal category one. It may
evolve into individual horizontal and vertical variants (like
with the existing bar charts).

Available attributes are:

    x: x-position of lower-left chart origin
    y: y-position of lower-left chart origin
    width: chart width
    height: chart height

    useAbsolute: disables auto-scaling of chart elements (?)
    lineLabelNudge: distance of data labels to data points
    lineLabels: labels associated with data values
    lineLabelFormat: format string or callback function
    groupSpacing: space between categories

    joinedLines: enables drawing of lines

    strokeColor: color of chart lines (?)
    fillColor: color for chart background (?)
    lines: style list, used cyclically for data series

    valueAxis: value axis object
    categoryAxis: category axis object
    categoryNames: category names

    data: chart data, a list of data series of equal length

## **Public Attributes**

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**categoryAxis** Handle of the category axis.

**categoryNames** List of category names.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**groupSpacing** ? - Likely to disappear.

**height** Height of the chart.

**inFill** Whether infilling should be done.

**joinedLines** Display data points joined with lines if true.

**lineLabelArray** explicit array of line label values, must match size of data if present.

**lineLabelFormat** Formatting string or function used for data point labels.

**lineLabelNudge** Distance between a data point and its label.

---

**lineLabels** Handle to the list of data point labels.

**lines** Handle of the lines.

**reversePlotOrder** If true reverse plot order.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**useAbsolute** Flag to use absolute spacing values.

**valueAxis** Handle of the value axis.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

*Example*

```
def demo(self):
    """Shows basic use of a line chart."""

    drawing = Drawing(200, 100)

    data = [
            (13, 5, 20, 22, 37, 45, 19, 4),
            (14, 10, 21, 28, 38, 46, 25, 5)
            ]

    lc = HorizontalLineChart()

    lc.x = 20
    lc.y = 10
    lc.height = 85
    lc.width = 170
    lc.data = data
    lc.lines.symbol = makeMarker('Circle')

    drawing.add(lc)

    return drawing
```

*Properties of Example Widget*

```
background = None
categoryAxis.categoryNames = None
categoryAxis.gridEnd = 0
categoryAxis.gridStart = 0
categoryAxis.gridStrokeColor = Color(0,0,0)
categoryAxis.gridStrokeDashArray = None
categoryAxis.gridStrokeWidth = 0.25
categoryAxis.joinAxis = None
categoryAxis.joinAxisMode = None
categoryAxis.joinAxisPos = None
categoryAxis.labelAxisMode = 'axis'
categoryAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8c8e2ac>
categoryAxis.reverseDirection = 0
categoryAxis.strokeColor = Color(0,0,0)
categoryAxis.strokeDashArray = None
categoryAxis.strokeWidth = 1
categoryAxis.style = 'parallel'
categoryAxis.tickDown = 5
categoryAxis.tickShift = 0
categoryAxis.tickUp = 0
categoryAxis.visible = 1
categoryAxis.visibleAxis = 1
categoryAxis.visibleGrid = 0
categoryAxis.visibleLabels = 1
categoryAxis.visibleTicks = 1
categoryNames = ('North', 'South', 'East', 'West')
data = [(100, 110, 120, 130), (70, 80, 80, 90)]
```

```
debug = 0
fillColor = None
groupSpacing = 1
height = 85
inFill = 0
joinedLines = 1
lineLabelArray = None
lineLabelFormat = None
lineLabelNudge = 10
lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8c8ed8c>
lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8c8ec0c>
reversePlotOrder = 0
strokeColor = None
strokeWidth = 1
useAbsolute = 0
valueAxis.avoidBoundFrac = None
valueAxis.forceZero = 0
valueAxis.gridEnd = 0
valueAxis.gridStart = 0
valueAxis.gridStrokeColor = Color(0,0,0)
valueAxis.gridStrokeDashArray = None
valueAxis.gridStrokeWidth = 0.25
valueAxis.joinAxis = None
valueAxis.joinAxisMode = None
valueAxis.joinAxisPos = None
valueAxis.labelTextFormat = None
valueAxis.labelTextPostFormat = None
valueAxis.labelTextScale = None
valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8c8ed2c>
valueAxis.maximumTicks = 7
valueAxis.minimumTickSpacing = 10
valueAxis.rangeRound = 'none'
valueAxis.strokeColor = Color(0,0,0)
valueAxis.strokeDashArray = None
valueAxis.strokeWidth = 1
valueAxis.style = 'normal'
valueAxis.tickLeft = 5
valueAxis.tickRight = 0
valueAxis.valueMax = None
valueAxis.valueMin = None
valueAxis.valueStep = None
valueAxis.visible = 1
valueAxis.visibleAxis = 1
valueAxis.visibleGrid = 0
valueAxis.visibleLabels = 1
valueAxis.visibleTicks = 1
valueAxis.zrangePref = 0
width = 180
x = 20
y = 10
```

## **HorizontalLineChart3D(HorizontalLineChart)**

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**categoryAxis** Handle of the category axis.

**categoryNames** List of category names.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**groupSpacing** ? - Likely to disappear.

**height** Height of the chart.

**inFill** Whether infilling should be done.

**joinedLines** Display data points joined with lines if true.

**lineLabelArray** explicit array of line label values, must match size of data if present.

**lineLabelFormat** Formatting string or function used for data point labels.

**lineLabelNudge** Distance between a data point and its label.

**lineLabels** Handle to the list of data point labels.

**lines** Handle of the lines.

**reversePlotOrder** If true reverse plot order.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**theta_x** dx/dz

**theta_y** dy/dz

**useAbsolute** Flag to use absolute spacing values.

**valueAxis** Handle of the value axis.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

**zDepth** depth of an individual series

**zSpace** z gap around series

*Example*

```
def demo(self):
    """Shows basic use of a line chart."""

    drawing = Drawing(200, 100)

    data = [
            (13, 5, 20, 22, 37, 45, 19, 4),
            (14, 10, 21, 28, 38, 46, 25, 5)
            ]

    lc = HorizontalLineChart()
    lc.x = 20
```

```
                        lc.y = 10
                        lc.height = 85
                        lc.width = 170
                        lc.data = data
                        lc.lines.symbol = makeMarker('Circle')

                        drawing.add(lc)

                        return drawing
```

*Properties of Example Widget*

```
        background = None
        categoryAxis.categoryNames = None
        categoryAxis.gridEnd = 0
        categoryAxis.gridStart = 0
        categoryAxis.gridStrokeColor = Color(0,0,0)
        categoryAxis.gridStrokeDashArray = None
        categoryAxis.gridStrokeWidth = 0.25
        categoryAxis.joinAxis = None
        categoryAxis.joinAxisMode = None
        categoryAxis.joinAxisPos = None
        categoryAxis.labelAxisMode = 'axis'
        categoryAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8c96dcc>
        categoryAxis.reverseDirection = 0
        categoryAxis.strokeColor = Color(0,0,0)
        categoryAxis.strokeDashArray = None
        categoryAxis.strokeWidth = 1
        categoryAxis.style = 'parallel'
        categoryAxis.tickDown = 5
        categoryAxis.tickShift = 0
        categoryAxis.tickUp = 0
        categoryAxis.visible = 1
        categoryAxis.visibleAxis = 1
        categoryAxis.visibleGrid = 0
        categoryAxis.visibleLabels = 1
        categoryAxis.visibleTicks = 1
        categoryNames = ('North', 'South', 'East', 'West')
        data = [(100, 110, 120, 130), (70, 80, 80, 90)]
        debug = 0
        fillColor = None
        groupSpacing = 1
        height = 85
        inFill = 0
        joinedLines = 1
        lineLabelArray = None
        lineLabelFormat = None
        lineLabelNudge = 10
        lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8c96eac>
        lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8c96e6c>
        reversePlotOrder = 0
        strokeColor = None
        strokeWidth = 1
        useAbsolute = 0
        valueAxis.avoidBoundFrac = None
        valueAxis.forceZero = 0
        valueAxis.gridEnd = 0
        valueAxis.gridStart = 0
        valueAxis.gridStrokeColor = Color(0,0,0)
        valueAxis.gridStrokeDashArray = None
        valueAxis.gridStrokeWidth = 0.25
        valueAxis.joinAxis = None
        valueAxis.joinAxisMode = None
        valueAxis.joinAxisPos = None
        valueAxis.labelTextFormat = None
        valueAxis.labelTextPostFormat = None
        valueAxis.labelTextScale = None
        valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8c96e2c>
        valueAxis.maximumTicks = 7
        valueAxis.minimumTickSpacing = 10
        valueAxis.rangeRound = 'none'
        valueAxis.strokeColor = Color(0,0,0)
        valueAxis.strokeDashArray = None
        valueAxis.strokeWidth = 1
        valueAxis.style = 'normal'
        valueAxis.tickLeft = 5
```

```
valueAxis.tickRight = 0
valueAxis.valueMax = None
valueAxis.valueMin = None
valueAxis.valueStep = None
valueAxis.visible = 1
valueAxis.visibleAxis = 1
valueAxis.visibleGrid = 0
valueAxis.visibleLabels = 1
valueAxis.visibleTicks = 1
valueAxis.zrangePref = 0
width = 180
x = 20
y = 10
```

## LineChart(AbstractLineChart)

## Public Attributes

**background** Handle to background object.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**height** Height of the chart.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

```
background = None
debug = 0
fillColor = None
height = 85
strokeColor = None
strokeWidth = 1
width = 180
x = 20
y = 10
```

## **`SampleHorizontalLineChart(HorizontalLineChart)`**

Sample class overwriting one method to draw additional horizontal lines.

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**categoryAxis** Handle of the category axis.

**categoryNames** List of category names.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**groupSpacing** ? - Likely to disappear.

**height** Height of the chart.

**inFill** Whether infilling should be done.

**joinedLines** Display data points joined with lines if true.

**lineLabelArray** explicit array of line label values, must match size of data if present.

**lineLabelFormat** Formatting string or function used for data point labels.

**lineLabelNudge** Distance between a data point and its label.

**lineLabels** Handle to the list of data point labels.

**lines** Handle of the lines.

**reversePlotOrder** If true reverse plot order.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**useAbsolute** Flag to use absolute spacing values.

**valueAxis** Handle of the value axis.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

*Example*

```
def demo(self):
    """Shows basic use of a line chart."""

    drawing = Drawing(200, 100)

    data = [
            (13, 5, 20, 22, 37, 45, 19, 4),
            (14, 10, 21, 28, 38, 46, 25, 5)
            ]

    lc = SampleHorizontalLineChart()

    lc.x = 20
    lc.y = 10
    lc.height = 85
    lc.width = 170
    lc.data = data
```

```
                    lc.strokeColor = colors.white
                    lc.fillColor = colors.HexColor(0xCCCCCC)

                    drawing.add(lc)

                    return drawing
```

*Properties of Example Widget*

```
        background = None
        categoryAxis.categoryNames = None
        categoryAxis.gridEnd = 0
        categoryAxis.gridStart = 0
        categoryAxis.gridStrokeColor = Color(0,0,0)
        categoryAxis.gridStrokeDashArray = None
        categoryAxis.gridStrokeWidth = 0.25
        categoryAxis.joinAxis = None
        categoryAxis.joinAxisMode = None
        categoryAxis.joinAxisPos = None
        categoryAxis.labelAxisMode = 'axis'
        categoryAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8ca278c>
        categoryAxis.reverseDirection = 0
        categoryAxis.strokeColor = Color(0,0,0)
        categoryAxis.strokeDashArray = None
        categoryAxis.strokeWidth = 1
        categoryAxis.style = 'parallel'
        categoryAxis.tickDown = 5
        categoryAxis.tickShift = 0
        categoryAxis.tickUp = 0
        categoryAxis.visible = 1
        categoryAxis.visibleAxis = 1
        categoryAxis.visibleGrid = 0
        categoryAxis.visibleLabels = 1
        categoryAxis.visibleTicks = 1
        categoryNames = ('North', 'South', 'East', 'West')
        data = [(100, 110, 120, 130), (70, 80, 80, 90)]
        debug = 0
        fillColor = None
        groupSpacing = 1
        height = 85
        inFill = 0
        joinedLines = 1
        lineLabelArray = None
        lineLabelFormat = None
        lineLabelNudge = 10
        lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8ca284c>
        lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8ca280c>
        reversePlotOrder = 0
        strokeColor = None
        strokeWidth = 1
        useAbsolute = 0
        valueAxis.avoidBoundFrac = None
        valueAxis.forceZero = 0
        valueAxis.gridEnd = 0
        valueAxis.gridStart = 0
        valueAxis.gridStrokeColor = Color(0,0,0)
        valueAxis.gridStrokeDashArray = None
        valueAxis.gridStrokeWidth = 0.25
        valueAxis.joinAxis = None
        valueAxis.joinAxisMode = None
        valueAxis.joinAxisPos = None
        valueAxis.labelTextFormat = None
        valueAxis.labelTextPostFormat = None
        valueAxis.labelTextScale = None
        valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8ca26ec>
        valueAxis.maximumTicks = 7
        valueAxis.minimumTickSpacing = 10
        valueAxis.rangeRound = 'none'
        valueAxis.strokeColor = Color(0,0,0)
        valueAxis.strokeDashArray = None
        valueAxis.strokeWidth = 1
        valueAxis.style = 'normal'
        valueAxis.tickLeft = 5
        valueAxis.tickRight = 0
        valueAxis.valueMax = None
        valueAxis.valueMin = None
```

```
valueAxis.valueStep = None
valueAxis.visible = 1
valueAxis.visibleAxis = 1
valueAxis.visibleGrid = 0
valueAxis.visibleLabels = 1
valueAxis.visibleTicks = 1
valueAxis.zrangePref = 0
width = 180
x = 20
y = 10
```

## `VerticalLineChart(LineChart)`

## Public Attributes

**background** Handle to background object.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**height** Height of the chart.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

```
background = None
debug = 0
fillColor = None
height = 85
strokeColor = None
strokeWidth = 1
width = 180
x = 20
y = 10
```

## Functions

### sample1( ... )

*Example*

```
def sample1():
    drawing = Drawing(400, 200)

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

    lc = HorizontalLineChart()

    lc.x = 50
    lc.y = 50
    lc.height = 125
    lc.width = 300
    lc.data = data
    lc.joinedLines = 1
    lc.lines.symbol = makeMarker('FilledDiamond')
    lc.lineLabelFormat = '%2.0f'

    catNames = string.split('Jan Feb Mar Apr May Jun Jul Aug', ' ')
    lc.categoryAxis.categoryNames = catNames
    lc.categoryAxis.labels.boxAnchor = 'n'

    lc.valueAxis.valueMin = 0
    lc.valueAxis.valueMax = 60
    lc.valueAxis.valueStep = 15

    drawing.add(lc)

    return drawing
```
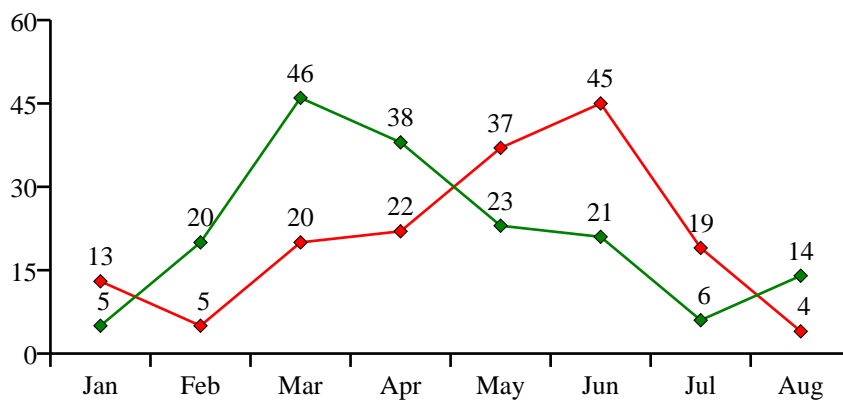
**sample1a( ... )**

*Example*

```
def sample1a():
    drawing = Drawing(400, 200)

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

    lc = SampleHorizontalLineChart()

    lc.x = 50
    lc.y = 50
    lc.height = 125
    lc.width = 300
    lc.data = data
    lc.joinedLines = 1
    lc.strokeColor = colors.white
    lc.fillColor = colors.HexColor(0xCCCCCC)
    lc.lines.symbol = makeMarker('FilledDiamond')
    lc.lineLabelFormat = '%2.0f'

    catNames = string.split('Jan Feb Mar Apr May Jun Jul Aug', ' ')
    lc.categoryAxis.categoryNames = catNames
    lc.categoryAxis.labels.boxAnchor = 'n'

    lc.valueAxis.valueMin = 0
    lc.valueAxis.valueMax = 60
    lc.valueAxis.valueStep = 15

    drawing.add(lc)

    return drawing
```
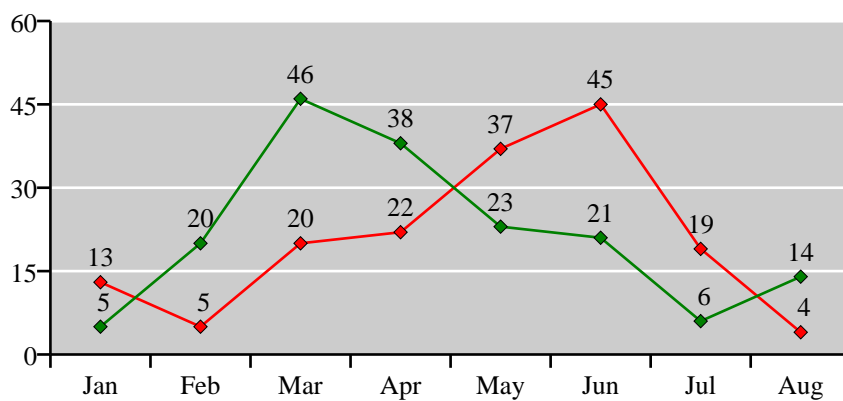
## sample2( ... )

*Example*

```
def sample2():
    drawing = Drawing(400, 200)

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

    lc = HorizontalLineChart()

    lc.x = 50
    lc.y = 50
    lc.height = 125
    lc.width = 300
    lc.data = data
    lc.joinedLines = 1
    lc.lines.symbol = makeMarker('Smiley')
    lc.lineLabelFormat = '%2.0f'
    lc.strokeColor = colors.black
    lc.fillColor = colors.lightblue

    catNames = string.split('Jan Feb Mar Apr May Jun Jul Aug', ' ')
    lc.categoryAxis.categoryNames = catNames
    lc.categoryAxis.labels.boxAnchor = 'n'

    lc.valueAxis.valueMin = 0
    lc.valueAxis.valueMax = 60
    lc.valueAxis.valueStep = 15

    drawing.add(lc)

    return drawing
```
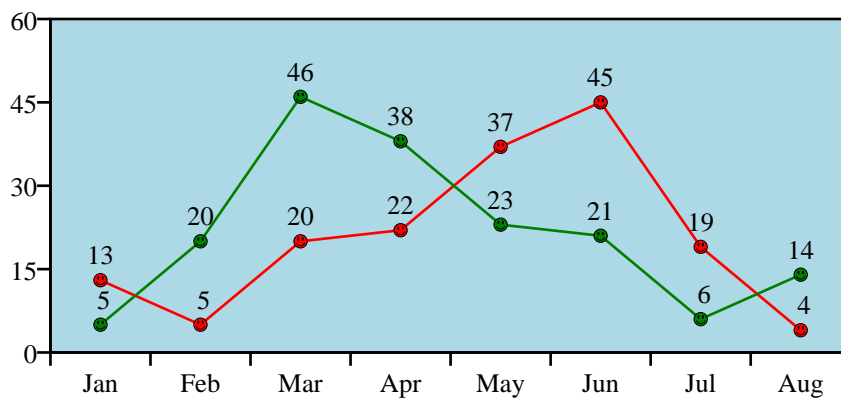
## sample3( ... )

*Example*

```
def sample3():
    drawing = Drawing(400, 200)

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

    lc = HorizontalLineChart()

    lc.x = 50
    lc.y = 50
    lc.height = 125
    lc.width = 300
    lc.data = data
    lc.joinedLines = 1
    lc.lineLabelFormat = '%2.0f'
    lc.strokeColor = colors.black

    lc.lines[0].symbol = makeMarker('Smiley')
    lc.lines[1].symbol = NoEntry
    lc.lines[0].strokeWidth = 2
    lc.lines[1].strokeWidth = 4

    catNames = string.split('Jan Feb Mar Apr May Jun Jul Aug', ' ')
    lc.categoryAxis.categoryNames = catNames
    lc.categoryAxis.labels.boxAnchor = 'n'

    lc.valueAxis.valueMin = 0
    lc.valueAxis.valueMax = 60
    lc.valueAxis.valueStep = 15

    drawing.add(lc)

    return drawing
```
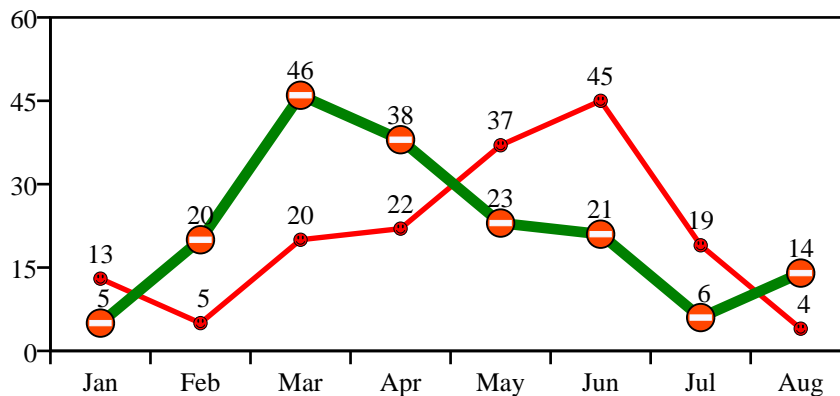
## `lineplots`

This module defines a very preliminary Line Plot example.

## Classes

### `AreaLinePlot(LinePlot)`

we're given data in the form [(X1,Y11,..Y1M)....(Xn,Yn1,...YnM)]

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**data** Data to be plotted, list of (lists of) x/y tuples.

**debug** Used only for debugging.

**fillColor** Color used for background interior of plot area.

**height** Height of the chart.

**joinedLines** Display data points joined with lines if true.

**lineLabelArray** explicit array of line label values, must match size of data if present.

**lineLabelFormat** Formatting string or function used for data point labels.

**lineLabelNudge** Distance between a data point and its label.

**lineLabels** Handle to the list of data point labels.

**lines** Handle of the lines.

**reversePlotOrder** If true reverse plot order.

**strokeColor** Color used for background border of plot area.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**xValueAxis** Handle of the x axis.

**y** Y position of the lower-left corner of the chart.

**yValueAxis** Handle of the y axis.

*Example*

```
def demo(self):
    """Shows basic use of a line chart."""

    drawing = Drawing(400, 200)

    data = [
        ((1,1), (2,2), (2.5,1), (3,3), (4,5)),
        ((1,2), (2,3), (2.5,2), (3.5,5), (4,6))
        ]

    lp = LinePlot()

    lp.x = 50
    lp.y = 50
    lp.height = 125
    lp.width = 300
```

```
        lp.data = data
        lp.joinedLines = 1
        lp.lineLabelFormat = '%2.0f'
        lp.strokeColor = colors.black

        lp.lines[0].strokeColor = colors.red
        lp.lines[0].symbol = makeMarker('FilledCircle')
        lp.lines[1].strokeColor = colors.blue
        lp.lines[1].symbol = makeMarker('FilledDiamond')

        lp.xValueAxis.valueMin = 0
        lp.xValueAxis.valueMax = 5
        lp.xValueAxis.valueStep = 1

        lp.yValueAxis.valueMin = 0
        lp.yValueAxis.valueMax = 7
        lp.yValueAxis.valueStep = 1

        drawing.add(lp)

        return drawing
```

*Properties of Example Widget*

```
background = None
data = [(1, 20, 100, 30), (2, 11, 50, 15), (3, 15, 70, 40)]
debug = 0
fillColor = None
height = 85
joinedLines = 1
lineLabelArray = None
lineLabelFormat = None
lineLabelNudge = 10
lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d38c8c>
lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d38c4c>
reversePlotOrder = 1
strokeColor = None
strokeWidth = 1
width = 180
x = 20
xValueAxis.avoidBoundFrac = None
xValueAxis.forceZero = 0
xValueAxis.gridEnd = 0
xValueAxis.gridStart = 0
xValueAxis.gridStrokeColor = Color(0,0,0)
xValueAxis.gridStrokeDashArray = None
xValueAxis.gridStrokeWidth = 0.25
xValueAxis.joinAxis = None
xValueAxis.joinAxisMode = None
xValueAxis.joinAxisPos = None
xValueAxis.labelTextFormat = None
xValueAxis.labelTextPostFormat = None
xValueAxis.labelTextScale = None
xValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d3830c>
xValueAxis.maximumTicks = 7
xValueAxis.minimumTickSpacing = 10
xValueAxis.rangeRound = 'none'
xValueAxis.strokeColor = Color(0,0,0)
xValueAxis.strokeDashArray = None
xValueAxis.strokeWidth = 1
xValueAxis.style = 'normal'
xValueAxis.tickDown = 5
xValueAxis.tickUp = 0
xValueAxis.valueMax = None
xValueAxis.valueMin = None
xValueAxis.valueStep = None
xValueAxis.visible = 1
xValueAxis.visibleAxis = 1
xValueAxis.visibleGrid = 0
xValueAxis.visibleLabels = 1
xValueAxis.visibleTicks = 1
xValueAxis.zrangePref = 0
y = 10
yValueAxis.avoidBoundFrac = None
yValueAxis.forceZero = 0
yValueAxis.gridEnd = 0
```

```
yValueAxis.gridStart = 0
yValueAxis.gridStrokeColor = Color(0,0,0)
yValueAxis.gridStrokeDashArray = None
yValueAxis.gridStrokeWidth = 0.25
yValueAxis.joinAxis = None
yValueAxis.joinAxisMode = None
yValueAxis.joinAxisPos = None
yValueAxis.labelTextFormat = None
yValueAxis.labelTextPostFormat = None
yValueAxis.labelTextScale = None
yValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d38c0c>
yValueAxis.maximumTicks = 7
yValueAxis.minimumTickSpacing = 10
yValueAxis.rangeRound = 'none'
yValueAxis.strokeColor = Color(0,0,0)
yValueAxis.strokeDashArray = None
yValueAxis.strokeWidth = 1
yValueAxis.style = 'normal'
yValueAxis.tickLeft = 5
yValueAxis.tickRight = 0
yValueAxis.valueMax = None
yValueAxis.valueMin = None
yValueAxis.valueStep = None
yValueAxis.visible = 1
yValueAxis.visibleAxis = 1
yValueAxis.visibleGrid = 0
yValueAxis.visibleLabels = 1
yValueAxis.visibleTicks = 1
yValueAxis.zrangePref = 0
```

## **GridLinePlot(LinePlot)**

A customized version of LinePlot.
It uses NormalDateXValueAxis() and AdjYValueAxis() for the X and Y axes.
The chart has a default grid background with thin horizontal lines
aligned with the tickmarks (and labels). You can change the back-
ground to be any Grid or ShadedRect, or scale the whole chart.
If you do provide a background, you can specify the colours of the
stripes with 'background.stripeColors'.

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Background for chart area (now Grid or ShadedRect).

**data** Data to be plotted, list of (lists of) x/y tuples.

**debug** Used only for debugging.

**fillColor** Color used for background interior of plot area.

**height** Height of the chart.

**joinedLines** Display data points joined with lines if true.

**lineLabelArray** explicit array of line label values, must match size of data if present.

**lineLabelFormat** Formatting string or function used for data point labels.

**lineLabelNudge** Distance between a data point and its label.

**lineLabels** Handle to the list of data point labels.

**lines** Handle of the lines.

**reversePlotOrder** If true reverse plot order.

**scaleFactor** Scalefactor to apply to whole drawing.

**strokeColor** Color used for background border of plot area.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**xValueAxis** Handle of the x axis.

**y** Y position of the lower-left corner of the chart.

**yValueAxis** Handle of the y axis.

*Example*

```
def demo(self,drawing=None):
    from reportlab.lib import colors
    if not drawing:
        drawing = Drawing(400, 200)
    lp = AdjLinePlot()
    lp.x = 50
    lp.y = 50
    lp.height = 125
    lp.width = 300
    lp.data = _monthlyIndexData
    lp.joinedLines = 1
    lp.strokeColor = colors.black
    c0 = colors.PCMYKColor(100,65,0,30, spotName='PANTONE 288 CV', density=100)
    lp.lines[0].strokeColor = c0
    lp.lines[0].strokeWidth = 2
```

```
lp.lines[0].strokeDashArray = None
c1 = colors.PCMYKColor(0,79,91,0, spotName='PANTONE Wm Red CV', density=100)
lp.lines[1].strokeColor = c1
lp.lines[1].strokeWidth = 1
lp.lines[1].strokeDashArray = [3,1]
lp.xValueAxis.labels.fontSize = 10
lp.xValueAxis.labels.textAnchor = 'start'
lp.xValueAxis.labels.boxAnchor = 'w'
lp.xValueAxis.labels.angle = -45
lp.xValueAxis.labels.dx = 0
lp.xValueAxis.labels.dy = -8
lp.xValueAxis.xLabelFormat = '{mm}/{yy}'
lp.yValueAxis.labelTextFormat = '%5d%% '
lp.yValueAxis.tickLeft = 5
lp.yValueAxis.labels.fontSize = 10
lp.background = Grid()
lp.background.stripeColors = [colors.pink, colors.lightblue]
lp.background.orientation = 'vertical'
drawing.add(lp,'plot')
return drawing
```

*Properties of Example Widget*

```
background.delta = 20
background.delta0 = 0
background.deltaSteps = []
background.fillColor = Color(1,1,1)
background.height = 100
background.orientation = 'horizontal'
background.stripeColors = [Color(1,0,0), Color(0,.501961,0), Color(0,0,1)]
background.strokeColor = Color(0,0,0)
background.strokeWidth = 0.5
background.useLines = 1
background.useRects = 0
background.width = 100
background.x = 0
background.y = 0
data = [[(19971202, 100.0),
        (19971231, 100.1704367),
        (19980131, 101.5639577),
        (19980228, 102.1879927),
        (19980331, 101.6337257),
        (19980430, 102.76404460000001),
        (19980531, 102.9198038),
        (19980630, 103.25938789999999),
        (19980731, 103.2516421),
        (19980831, 105.4744329),
        (19980930, 109.3242705),
        (19981031, 111.98592910000001),
        (19981130, 110.9184642),
        (19981231, 110.9184642),
        (19990131, 111.9882532),
        (19990228, 109.7912614),
        (19990331, 110.24189629999999),
        (19990430, 110.42793210000001),
        (19990531, 109.33955469999999),
        (19990630, 108.23417480000001),
        (19990731, 110.21294469999999),
        (19990831, 110.9683062),
        (19990930, 112.4425371),
        (19991031, 112.7314032),
        (19991130, 112.3509645),
        (19991231, 112.3660659),
        (20000131, 110.92552480000001),
        (20000229, 110.5266306),
        (20000331, 113.3116101),
        (20000430, 111.0449133),
        (20000531, 111.70271700000001),
        (20000630, 113.5832178)],
       [(19971202, 100.0),
        (19971231, 100.0),
        (19980131, 100.8),
        (19980228, 102.0),
        (19980331, 101.90000000000001),
        (19980430, 103.0),
        (19980531, 103.0),
```

```
                (19980630, 103.09999999999999),
                (19980731, 103.09999999999999),
                (19980831, 102.8),
                (19980930, 105.59999999999999),
                (19981031, 108.3),
                (19981130, 108.09999999999999),
                (19981231, 111.90000000000001),
                (19990131, 113.09999999999999),
                (19990228, 110.2),
                (19990331, 111.8),
                (19990430, 112.3),
                (19990531, 110.09999999999999),
                (19990630, 109.3),
                (19990731, 111.2),
                (19990831, 111.7),
                (19990930, 112.59999999999999),
                (19991031, 113.2),
                (19991130, 113.90000000000001),
                (19991231, 115.40000000000001),
                (20000131, 112.7),
                (20000229, 113.90000000000001),
                (20000331, 115.8),
                (20000430, 112.2),
                (20000531, 112.59999999999999),
                (20000630, 114.59999999999999)]]
debug = 0
fillColor = None
height = 85
joinedLines = 1
lineLabelArray = None
lineLabelFormat = None
lineLabelNudge = 10
lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d428ec>
lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d428ac>
reversePlotOrder = 0
scaleFactor = None
strokeColor = None
strokeWidth = 1
width = 180
x = 20
xValueAxis.avoidBoundFrac = None
xValueAxis.bottomAxisLabelSlack = 0.10000000000000001
xValueAxis.dailyFreq = 0
xValueAxis.dayOfWeekName = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sund
xValueAxis.forceEndDate = 0
xValueAxis.forceFirstDate = 0
xValueAxis.forceZero = 0
xValueAxis.gridEnd = 0
xValueAxis.gridStart = 0
xValueAxis.gridStrokeColor = Color(0,0,0)
xValueAxis.gridStrokeDashArray = None
xValueAxis.gridStrokeWidth = 0.25
xValueAxis.joinAxis = None
xValueAxis.joinAxisMode = None
xValueAxis.joinAxisPos = None
xValueAxis.labelTextFormat = None
xValueAxis.labelTextPostFormat = None
xValueAxis.labelTextScale = None
xValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d4298c>
xValueAxis.maximumTicks = 7
xValueAxis.minimumTickSpacing = 10
xValueAxis.monthName = ['January',
                        'February',
                        'March',
                        'April',
                        'May',
                        'June',
                        'July',
                        'August',
                        'September',
                        'October',
                        'November',
                        'December']
xValueAxis.niceMonth = 1
xValueAxis.rangeRound = 'none'
xValueAxis.strokeColor = Color(0,0,0)
xValueAxis.strokeDashArray = None
xValueAxis.strokeWidth = 1
xValueAxis.style = 'normal'
```

```
xValueAxis.tickDown = 5
xValueAxis.tickUp = 0
xValueAxis.valueMax = None
xValueAxis.valueMin = None
xValueAxis.valueStep = None
xValueAxis.valueSteps = None
xValueAxis.visible = 1
xValueAxis.visibleAxis = 1
xValueAxis.visibleGrid = 0
xValueAxis.visibleLabels = 1
xValueAxis.visibleTicks = 1
xValueAxis.xLabelFormat = '{mm}/{yy}'
xValueAxis.zrangePref = 0
y = 10
yValueAxis.avoidBoundFrac = None
yValueAxis.forceZero = 0
yValueAxis.gridEnd = 0
yValueAxis.gridStart = 0
yValueAxis.gridStrokeColor = Color(0,0,0)
yValueAxis.gridStrokeDashArray = None
yValueAxis.gridStrokeWidth = 0.25
yValueAxis.joinAxis = None
yValueAxis.joinAxisMode = None
yValueAxis.joinAxisPos = None
yValueAxis.labelTextFormat = None
yValueAxis.labelTextPostFormat = None
yValueAxis.labelTextScale = None
yValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d420ec>
yValueAxis.leftAxisOrigShiftIPC = 0.14999999999999999
yValueAxis.leftAxisOrigShiftMin = 12
yValueAxis.leftAxisPercent = 1
yValueAxis.leftAxisSkipLL0 = 0
yValueAxis.maximumTicks = 7
yValueAxis.minimumTickSpacing = 10
yValueAxis.rangeRound = 'none'
yValueAxis.requiredRange = 30
yValueAxis.strokeColor = Color(0,0,0)
yValueAxis.strokeDashArray = None
yValueAxis.strokeWidth = 1
yValueAxis.style = 'normal'
yValueAxis.tickLeft = 5
yValueAxis.tickRight = 0
yValueAxis.valueMax = None
yValueAxis.valueMin = None
yValueAxis.valueStep = None
yValueAxis.valueSteps = None
yValueAxis.visible = 1
yValueAxis.visibleAxis = 1
yValueAxis.visibleGrid = 0
yValueAxis.visibleLabels = 1
yValueAxis.visibleTicks = 1
yValueAxis.zrangePref = 0
```

## `LinePlot(AbstractLineChart)`

Line plot with multiple lines.

Both x- and y-axis are value axis (so there are no seperate
X and Y versions of this class).

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**data** Data to be plotted, list of (lists of) x/y tuples.

**debug** Used only for debugging.

**fillColor** Color used for background interior of plot area.

**height** Height of the chart.

**joinedLines** Display data points joined with lines if true.

**lineLabelArray** explicit array of line label values, must match size of data if present.

**lineLabelFormat** Formatting string or function used for data point labels.

**lineLabelNudge** Distance between a data point and its label.

**lineLabels** Handle to the list of data point labels.

**lines** Handle of the lines.

**reversePlotOrder** If true reverse plot order.

**strokeColor** Color used for background border of plot area.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**xValueAxis** Handle of the x axis.

**y** Y position of the lower-left corner of the chart.

**yValueAxis** Handle of the y axis.

*Example*

```
def demo(self):
    """Shows basic use of a line chart."""

    drawing = Drawing(400, 200)

    data = [
        ((1,1), (2,2), (2.5,1), (3,3), (4,5)),
        ((1,2), (2,3), (2.5,2), (3.5,5), (4,6))
        ]

    lp = LinePlot()

    lp.x = 50
    lp.y = 50
    lp.height = 125
    lp.width = 300
    lp.data = data
    lp.joinedLines = 1
    lp.lineLabelFormat = '%2.0f'
    lp.strokeColor = colors.black
    lp.lines[0].strokeColor = colors.red
```

```
                    lp.lines[0].symbol = makeMarker('FilledCircle')
                    lp.lines[1].strokeColor = colors.blue
                    lp.lines[1].symbol = makeMarker('FilledDiamond')

                    lp.xValueAxis.valueMin = 0
                    lp.xValueAxis.valueMax = 5
                    lp.xValueAxis.valueStep = 1

                    lp.yValueAxis.valueMin = 0
                    lp.yValueAxis.valueMax = 7
                    lp.yValueAxis.valueStep = 1

                    drawing.add(lp)

                    return drawing
```

*Properties of Example Widget*

```
        background = None
        data = [((1, 1), (2, 2), (2.5, 1), (3, 3), (4, 5)),
                ((1, 2), (2, 3), (2.5, 2), (3, 4), (4, 6))]
        debug = 0
        fillColor = None
        height = 85
        joinedLines = 1
        lineLabelArray = None
        lineLabelFormat = None
        lineLabelNudge = 10
        lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d4a6ec>
        lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d4a6ac>
        reversePlotOrder = 0
        strokeColor = None
        strokeWidth = 1
        width = 180
        x = 20
        xValueAxis.avoidBoundFrac = None
        xValueAxis.forceZero = 0
        xValueAxis.gridEnd = 0
        xValueAxis.gridStart = 0
        xValueAxis.gridStrokeColor = Color(0,0,0)
        xValueAxis.gridStrokeDashArray = None
        xValueAxis.gridStrokeWidth = 0.25
        xValueAxis.joinAxis = None
        xValueAxis.joinAxisMode = None
        xValueAxis.joinAxisPos = None
        xValueAxis.labelTextFormat = None
        xValueAxis.labelTextPostFormat = None
        xValueAxis.labelTextScale = None
        xValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d4a60c>
        xValueAxis.maximumTicks = 7
        xValueAxis.minimumTickSpacing = 10
        xValueAxis.rangeRound = 'none'
        xValueAxis.strokeColor = Color(0,0,0)
        xValueAxis.strokeDashArray = None
        xValueAxis.strokeWidth = 1
        xValueAxis.style = 'normal'
        xValueAxis.tickDown = 5
        xValueAxis.tickUp = 0
        xValueAxis.valueMax = None
        xValueAxis.valueMin = None
        xValueAxis.valueStep = None
        xValueAxis.visible = 1
        xValueAxis.visibleAxis = 1
        xValueAxis.visibleGrid = 0
        xValueAxis.visibleLabels = 1
        xValueAxis.visibleTicks = 1
        xValueAxis.zrangePref = 0
        y = 10
        yValueAxis.avoidBoundFrac = None
        yValueAxis.forceZero = 0
        yValueAxis.gridEnd = 0
        yValueAxis.gridStart = 0
        yValueAxis.gridStrokeColor = Color(0,0,0)
        yValueAxis.gridStrokeDashArray = None
        yValueAxis.gridStrokeWidth = 0.25
        yValueAxis.joinAxis = None
```

```
yValueAxis.joinAxisMode = None
yValueAxis.joinAxisPos = None
yValueAxis.labelTextFormat = None
yValueAxis.labelTextPostFormat = None
yValueAxis.labelTextScale = None
yValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d4a66c>
yValueAxis.maximumTicks = 7
yValueAxis.minimumTickSpacing = 10
yValueAxis.rangeRound = 'none'
yValueAxis.strokeColor = Color(0,0,0)
yValueAxis.strokeDashArray = None
yValueAxis.strokeWidth = 1
yValueAxis.style = 'normal'
yValueAxis.tickLeft = 5
yValueAxis.tickRight = 0
yValueAxis.valueMax = None
yValueAxis.valueMin = None
yValueAxis.valueStep = None
yValueAxis.visible = 1
yValueAxis.visibleAxis = 1
yValueAxis.visibleGrid = 0
yValueAxis.visibleLabels = 1
yValueAxis.visibleTicks = 1
yValueAxis.zrangePref = 0
```

## `LinePlot3D(LinePlot)`

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**data** Data to be plotted, list of (lists of) x/y tuples.

**debug** Used only for debugging.

**fillColor** Color used for background interior of plot area.

**height** Height of the chart.

**joinedLines** Display data points joined with lines if true.

**lineLabelArray** explicit array of line label values, must match size of data if present.

**lineLabelFormat** Formatting string or function used for data point labels.

**lineLabelNudge** Distance between a data point and its label.

**lineLabels** Handle to the list of data point labels.

**lines** Handle of the lines.

**reversePlotOrder** If true reverse plot order.

**strokeColor** Color used for background border of plot area.

**strokeWidth** Width plot area border.

**theta_x** dx/dz

**theta_y** dy/dz

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**xValueAxis** Handle of the x axis.

**y** Y position of the lower-left corner of the chart.

**yValueAxis** Handle of the y axis.

**zDepth** depth of an individual series

**zSpace** z gap around series

*Example*

```
def demo(self):
    """Shows basic use of a line chart."""

    drawing = Drawing(400, 200)

    data = [
        ((1,1), (2,2), (2.5,1), (3,3), (4,5)),
        ((1,2), (2,3), (2.5,2), (3.5,5), (4,6))
        ]

    lp = LinePlot()

    lp.x = 50
    lp.y = 50
    lp.height = 125
    lp.width = 300
    lp.data = data
    lp.joinedLines = 1
    lp.lineLabelFormat = '%2.0f'
    lp.strokeColor = colors.black
```

```
                lp.lines[0].strokeColor = colors.red
                lp.lines[0].symbol = makeMarker('FilledCircle')
                lp.lines[1].strokeColor = colors.blue
                lp.lines[1].symbol = makeMarker('FilledDiamond')

                lp.xValueAxis.valueMin = 0
                lp.xValueAxis.valueMax = 5
                lp.xValueAxis.valueStep = 1

                lp.yValueAxis.valueMin = 0
                lp.yValueAxis.valueMax = 7
                lp.yValueAxis.valueStep = 1

                drawing.add(lp)

                return drawing
```

*Properties of Example Widget*

```
        background = None
        data = [((1, 1), (2, 2), (2.5, 1), (3, 3), (4, 5)),
                ((1, 2), (2, 3), (2.5, 2), (3, 4), (4, 6))]
        debug = 0
        fillColor = None
        height = 85
        joinedLines = 1
        lineLabelArray = None
        lineLabelFormat = None
        lineLabelNudge = 10
        lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d518cc>
        lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d5188c>
        reversePlotOrder = 0
        strokeColor = None
        strokeWidth = 1
        width = 180
        x = 20
        xValueAxis.avoidBoundFrac = None
        xValueAxis.forceZero = 0
        xValueAxis.gridEnd = 0
        xValueAxis.gridStart = 0
        xValueAxis.gridStrokeColor = Color(0,0,0)
        xValueAxis.gridStrokeDashArray = None
        xValueAxis.gridStrokeWidth = 0.25
        xValueAxis.joinAxis = None
        xValueAxis.joinAxisMode = None
        xValueAxis.joinAxisPos = None
        xValueAxis.labelTextFormat = None
        xValueAxis.labelTextPostFormat = None
        xValueAxis.labelTextScale = None
        xValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d517ec>
        xValueAxis.maximumTicks = 7
        xValueAxis.minimumTickSpacing = 10
        xValueAxis.rangeRound = 'none'
        xValueAxis.strokeColor = Color(0,0,0)
        xValueAxis.strokeDashArray = None
        xValueAxis.strokeWidth = 1
        xValueAxis.style = 'normal'
        xValueAxis.tickDown = 5
        xValueAxis.tickUp = 0
        xValueAxis.valueMax = None
        xValueAxis.valueMin = None
        xValueAxis.valueStep = None
        xValueAxis.visible = 1
        xValueAxis.visibleAxis = 1
        xValueAxis.visibleGrid = 0
        xValueAxis.visibleLabels = 1
        xValueAxis.visibleTicks = 1
        xValueAxis.zrangePref = 0
        y = 10
        yValueAxis.avoidBoundFrac = None
        yValueAxis.forceZero = 0
        yValueAxis.gridEnd = 0
        yValueAxis.gridStart = 0
        yValueAxis.gridStrokeColor = Color(0,0,0)
        yValueAxis.gridStrokeDashArray = None
        yValueAxis.gridStrokeWidth = 0.25
```

```
yValueAxis.joinAxis = None
yValueAxis.joinAxisMode = None
yValueAxis.joinAxisPos = None
yValueAxis.labelTextFormat = None
yValueAxis.labelTextPostFormat = None
yValueAxis.labelTextScale = None
yValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d5184c>
yValueAxis.maximumTicks = 7
yValueAxis.minimumTickSpacing = 10
yValueAxis.rangeRound = 'none'
yValueAxis.strokeColor = Color(0,0,0)
yValueAxis.strokeDashArray = None
yValueAxis.strokeWidth = 1
yValueAxis.style = 'normal'
yValueAxis.tickLeft = 5
yValueAxis.tickRight = 0
yValueAxis.valueMax = None
yValueAxis.valueMin = None
yValueAxis.valueStep = None
yValueAxis.visible = 1
yValueAxis.visibleAxis = 1
yValueAxis.visibleGrid = 0
yValueAxis.visibleLabels = 1
yValueAxis.visibleTicks = 1
yValueAxis.zrangePref = 0
```

## **ScatterPlot(LinePlot)**

A scatter plot widget

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**axisStrokeWidth** Stroke width for both axes

**axisTickLengths** Lenth of the ticks on both axes

**background** Background color (if any)

**bottomPadding** Padding at bottom of drawing

**data** Data points - a list of x/y tuples.

**debug** Used only for debugging.

**fillColor** Color used for background interior of plot area.

**height** Height of the area inside the axes

**joinedLines** Display data points joined with lines if true.

**labelOffset** Space between label and Axis (or other labels)

**leftPadding** Padding on left of drawing

**lineLabelArray** explicit array of line label values, must match size of data if present.

**lineLabelFormat** Formatting string or function used for data point labels.

**lineLabelNudge** Distance between a data point and its label.

**lineLabels** Handle to the list of data point labels.

**lines** Handle of the lines.

**outerBorderColor** Color of outer border (if any)

**outerBorderOn** Is there an outer border (continuation of axes)

**reversePlotOrder** If true reverse plot order.

**rightPadding** Padding on right of drawing

**strokeColor** Color used for border of plot area.

**strokeWidth** Width plot area border.

**topPadding** Padding at top of drawing

**width** Width of the area inside the axes

**x** X position of the lower-left corner of the chart.

**xLabel** Label for the whole X-Axis

**xValueAxis** Handle of the x axis.

**y** Y position of the lower-left corner of the chart.

**yLabel** Label for the whole Y-Axis

**yValueAxis** Handle of the y axis.

*Example*
```
def demo(self,drawing=None):
    if not drawing:
        tx,ty=self._getDrawingDimensions()
```

```
                    drawing = Drawing(tx,ty)
                drawing.add(self.draw())
                return drawing
```

*Properties of Example Widget*

```
        background = None
        bottomPadding = 5
        data = [((0.029999999999999999, 62.729999999999997),
                (0.073999999999999996, 54.363),
                (1.216, 17.963999999999999)),
                ((1.3600000000000001, 11.621),
                (1.387, 50.011000000000003),
                (1.4279999999999999, 68.953000000000003)),
                ((1.444, 86.888000000000005),
                (1.754, 35.579999999999998),
                (1.766, 36.049999999999997))]
        debug = 0
        fillColor = None
        height = 77
        joinedLines = 0
        leftPadding = 5
        lineLabelArray = None
        lineLabelFormat = '%.2f'
        lineLabelNudge = 0
        lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d86ccc>
        lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d86c8c>
        outerBorderColor = Color(0,0,0)
        outerBorderOn = 1
        reversePlotOrder = 0
        rightPadding = 10
        strokeColor = None
        strokeWidth = 1
        topPadding = 5
        width = 142
        x = 25.995999999999999
        xLabel = 'X Lable'
        xValueAxis.avoidBoundFrac = None
        xValueAxis.forceZero = 0
        xValueAxis.gridEnd = 0
        xValueAxis.gridStart = 0
        xValueAxis.gridStrokeColor = Color(0,0,0)
        xValueAxis.gridStrokeDashArray = None
        xValueAxis.gridStrokeWidth = 0.25
        xValueAxis.joinAxis = None
        xValueAxis.joinAxisMode = None
        xValueAxis.joinAxisPos = None
        xValueAxis.labelTextFormat = None
        xValueAxis.labelTextPostFormat = None
        xValueAxis.labelTextScale = None
        xValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d86bec>
        xValueAxis.maximumTicks = 7
        xValueAxis.minimumTickSpacing = 10
        xValueAxis.rangeRound = 'both'
        xValueAxis.strokeColor = Color(0,0,0)
        xValueAxis.strokeDashArray = None
        xValueAxis.strokeWidth = 0.5
        xValueAxis.style = 'normal'
        xValueAxis.tickDown = 2
        xValueAxis.tickUp = 0
        xValueAxis.valueMax = None
        xValueAxis.valueMin = None
        xValueAxis.valueStep = None
        xValueAxis.visible = 1
        xValueAxis.visibleAxis = 1
        xValueAxis.visibleGrid = 0
        xValueAxis.visibleLabels = 1
        xValueAxis.visibleTicks = 1
        xValueAxis.zrangePref = 0
        y = 16
        yLabel = 'Y Lable'
        yValueAxis.avoidBoundFrac = None
        yValueAxis.forceZero = 0
        yValueAxis.gridEnd = 0
        yValueAxis.gridStart = 0
        yValueAxis.gridStrokeColor = Color(0,0,0)
```

```
yValueAxis.gridStrokeDashArray = None
yValueAxis.gridStrokeWidth = 0.25
yValueAxis.joinAxis = None
yValueAxis.joinAxisMode = None
yValueAxis.joinAxisPos = None
yValueAxis.labelTextFormat = '%s'
yValueAxis.labelTextPostFormat = None
yValueAxis.labelTextScale = None
yValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d86c4c>
yValueAxis.maximumTicks = 7
yValueAxis.minimumTickSpacing = 10
yValueAxis.rangeRound = 'both'
yValueAxis.strokeColor = Color(0,0,0)
yValueAxis.strokeDashArray = None
yValueAxis.strokeWidth = 0.5
yValueAxis.style = 'normal'
yValueAxis.tickLeft = 2
yValueAxis.tickRight = 0
yValueAxis.valueMax = None
yValueAxis.valueMin = None
yValueAxis.valueStep = None
yValueAxis.visible = 1
yValueAxis.visibleAxis = 1
yValueAxis.visibleGrid = 0
yValueAxis.visibleLabels = 1
yValueAxis.visibleTicks = 1
yValueAxis.zrangePref = 0
```

## ShadedPolyFiller(Filler, ShadedPolygon)

## Public Attributes

**fillColor** filler interior color

**strokeColor** filler edge color

**strokeWidth** filler edge width

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

## `SplitLinePlot(AreaLinePlot)`

## Public Attributes

**annotations** list of callables, will be called with self, xscale, yscale.

**background** Handle to background object.

**data** Data to be plotted, list of (lists of) x/y tuples.

**debug** Used only for debugging.

**fillColor** Color used for background interior of plot area.

**height** Height of the chart.

**joinedLines** Display data points joined with lines if true.

**lineLabelArray** explicit array of line label values, must match size of data if present.

**lineLabelFormat** Formatting string or function used for data point labels.

**lineLabelNudge** Distance between a data point and its label.

**lineLabels** Handle to the list of data point labels.

**lines** Handle of the lines.

**reversePlotOrder** If true reverse plot order.

**strokeColor** Color used for background border of plot area.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**xValueAxis** Handle of the x axis.

**y** Y position of the lower-left corner of the chart.

**yValueAxis** Handle of the y axis.

*Example*

```
def demo(self):
    """Shows basic use of a line chart."""

    drawing = Drawing(400, 200)

    data = [
        ((1,1), (2,2), (2.5,1), (3,3), (4,5)),
        ((1,2), (2,3), (2.5,2), (3.5,5), (4,6))
        ]

    lp = LinePlot()

    lp.x = 50
    lp.y = 50
    lp.height = 125
    lp.width = 300
    lp.data = data
    lp.joinedLines = 1
    lp.lineLabelFormat = '%2.0f'
    lp.strokeColor = colors.black

    lp.lines[0].strokeColor = colors.red
    lp.lines[0].symbol = makeMarker('FilledCircle')
    lp.lines[1].strokeColor = colors.blue
    lp.lines[1].symbol = makeMarker('FilledDiamond')

    lp.xValueAxis.valueMin = 0
    lp.xValueAxis.valueMax = 5
```

```
                    lp.xValueAxis.valueStep = 1

                    lp.yValueAxis.valueMin = 0
                    lp.yValueAxis.valueMax = 7
                    lp.yValueAxis.valueStep = 1

                    drawing.add(lp)

                    return drawing
```

*Properties of Example Widget*

```
        background = None
        data = [(20030601, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20030701, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20030801, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20030901, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20031001, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20031101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20031201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20040101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20040201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20040301, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20040401, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20040501, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20040601, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20040701, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20040801, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20040901, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20041001, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20041101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20041201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20050101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20050201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20050301, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20050401, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20050501, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20050601, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20050701, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20050801, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20050901, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20051001, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20051101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20051201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20060101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20060201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20060301, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20060401, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20060501, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20060601, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20060701, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20060801, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20060901, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20061001, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20061101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20061201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20070101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20070201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20070301, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20070401, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20070501, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20070601, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20070701, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20070801, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20070901, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20071001, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20071101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20071201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20080101, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20080201, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20080301, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20080401, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20080501, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20080601, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20080701, 0.94999999999999996, 0.050000000000000003, 0.0),
                (20080801, 0.94999999999999996, 0.050000000000000003, 0.0),
```

```
(20080901, 0.94999999999999996, 0.050000000000000003, 0.0),
(20081001, 0.94999999999999996, 0.050000000000000003, 0.0),
(20081101, 0.94999999999999996, 0.050000000000000003, 0.0),
(20081201, 0.94999999999999996, 0.050000000000000003, 0.0),
(20090101, 0.94999999999999996, 0.050000000000000003, 0.0),
(20090201, 0.91000000000000003, 0.089999999999999997, 0.0),
(20090301, 0.91000000000000003, 0.089999999999999997, 0.0),
(20090401, 0.91000000000000003, 0.089999999999999997, 0.0),
(20090501, 0.91000000000000003, 0.089999999999999997, 0.0),
(20090601, 0.91000000000000003, 0.089999999999999997, 0.0),
(20090701, 0.91000000000000003, 0.089999999999999997, 0.0),
(20090801, 0.91000000000000003, 0.089999999999999997, 0.0),
(20090901, 0.91000000000000003, 0.089999999999999997, 0.0),
(20091001, 0.91000000000000003, 0.089999999999999997, 0.0),
(20091101, 0.91000000000000003, 0.089999999999999997, 0.0),
(20091201, 0.91000000000000003, 0.089999999999999997, 0.0),
(20100101, 0.91000000000000003, 0.089999999999999997, 0.0),
(20100201, 0.81000000000000005, 0.19, 0.0),
(20100301, 0.81000000000000005, 0.19, 0.0),
(20100401, 0.81000000000000005, 0.19, 0.0),
(20100501, 0.81000000000000005, 0.19, 0.0),
(20100601, 0.81000000000000005, 0.19, 0.0),
(20100701, 0.81000000000000005, 0.19, 0.0),
(20100801, 0.81000000000000005, 0.19, 0.0),
(20100901, 0.81000000000000005, 0.19, 0.0),
(20101001, 0.81000000000000005, 0.19, 0.0),
(20101101, 0.81000000000000005, 0.19, 0.0),
(20101201, 0.81000000000000005, 0.19, 0.0),
(20110101, 0.81000000000000005, 0.19, 0.0),
(20110201, 0.71999999999999997, 0.28000000000000003, 0.0),
(20110301, 0.71999999999999997, 0.28000000000000003, 0.0),
(20110401, 0.71999999999999997, 0.28000000000000003, 0.0),
(20110501, 0.71999999999999997, 0.28000000000000003, 0.0),
(20110601, 0.71999999999999997, 0.28000000000000003, 0.0),
(20110701, 0.71999999999999997, 0.28000000000000003, 0.0),
(20110801, 0.71999999999999997, 0.28000000000000003, 0.0),
(20110901, 0.71999999999999997, 0.28000000000000003, 0.0),
(20111001, 0.71999999999999997, 0.28000000000000003, 0.0),
(20111101, 0.71999999999999997, 0.28000000000000003, 0.0),
(20111201, 0.71999999999999997, 0.28000000000000003, 0.0),
(20120101, 0.71999999999999997, 0.28000000000000003, 0.0),
(20120201, 0.53000000000000003, 0.46999999999999997, 0.0),
(20120301, 0.53000000000000003, 0.46999999999999997, 0.0),
(20120401, 0.53000000000000003, 0.46999999999999997, 0.0),
(20120501, 0.53000000000000003, 0.46999999999999997, 0.0),
(20120601, 0.53000000000000003, 0.46999999999999997, 0.0),
(20120701, 0.53000000000000003, 0.46999999999999997, 0.0),
(20120801, 0.53000000000000003, 0.46999999999999997, 0.0),
(20120901, 0.53000000000000003, 0.46999999999999997, 0.0),
(20121001, 0.53000000000000003, 0.46999999999999997, 0.0),
(20121101, 0.53000000000000003, 0.46999999999999997, 0.0),
(20121201, 0.53000000000000003, 0.46999999999999997, 0.0),
(20130101, 0.53000000000000003, 0.46999999999999997, 0.0),
(20130201, 0.44, 0.56000000000000005, 0.0),
(20130301, 0.44, 0.56000000000000005, 0.0),
(20130401, 0.44, 0.56000000000000005, 0.0),
(20130501, 0.44, 0.56000000000000005, 0.0),
(20130601, 0.44, 0.56000000000000005, 0.0),
(20130701, 0.44, 0.56000000000000005, 0.0),
(20130801, 0.44, 0.56000000000000005, 0.0),
(20130901, 0.44, 0.56000000000000005, 0.0),
(20131001, 0.44, 0.56000000000000005, 0.0),
(20131101, 0.44, 0.56000000000000005, 0.0),
(20131201, 0.44, 0.56000000000000005, 0.0),
(20140101, 0.44, 0.56000000000000005, 0.0),
(20140201, 0.35999999999999999, 0.5, 0.14000000000000001),
(20140301, 0.35999999999999999, 0.5, 0.14000000000000001),
(20140401, 0.35999999999999999, 0.5, 0.14000000000000001),
(20140501, 0.35999999999999999, 0.5, 0.14000000000000001),
(20140601, 0.35999999999999999, 0.5, 0.14000000000000001),
(20140701, 0.35999999999999999, 0.5, 0.14000000000000001),
(20140801, 0.35999999999999999, 0.5, 0.14000000000000001),
(20140901, 0.35999999999999999, 0.5, 0.14000000000000001),
(20141001, 0.35999999999999999, 0.5, 0.14000000000000001),
(20141101, 0.35999999999999999, 0.5, 0.14000000000000001),
(20141201, 0.35999999999999999, 0.5, 0.14000000000000001),
(20150101, 0.35999999999999999, 0.5, 0.14000000000000001),
(20150201, 0.29999999999999999, 0.40999999999999998, 0.28999999999999998),
(20150301, 0.29999999999999999, 0.40999999999999998, 0.28999999999999998),
```

```
                    (20150401, 0.2999999999999999, 0.4099999999999998, 0.2899999999999998),
                    (20150501, 0.2999999999999999, 0.4099999999999998, 0.2899999999999998),
                    (20150601, 0.2999999999999999, 0.4099999999999998, 0.2899999999999998),
                    (20150701, 0.2999999999999999, 0.4099999999999998, 0.2899999999999998),
                    (20150801, 0.2999999999999999, 0.4099999999999998, 0.2899999999999998),
                    (20150901, 0.2999999999999999, 0.4099999999999998, 0.2899999999999998),
                    (20151001, 0.2999999999999999, 0.4099999999999998, 0.2899999999999998),
                    (20151101, 0.2999999999999999, 0.4099999999999998, 0.2899999999999998),
                    (20151201, 0.2999999999999999, 0.4099999999999998, 0.2899999999999998),
                    (20160101, 0.2999999999999999, 0.4099999999999998, 0.28999999999999998),
                    (20160201, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20160301, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20160401, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20160501, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20160601, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20160701, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20160801, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20160901, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20161001, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20161101, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20161201, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20170101, 0.26000000000000001, 0.3599999999999999, 0.38),
                    (20170201, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20170301, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20170401, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20170501, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20170601, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20170701, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20170801, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20170901, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20171001, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20171101, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20171201, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20180101, 0.20000000000000001, 0.2999999999999999, 0.5),
                    (20180201, 0.13, 0.37, 0.5),
                    (20180301, 0.13, 0.37, 0.5),
                    (20180401, 0.13, 0.37, 0.5),
                    (20180501, 0.13, 0.37, 0.5),
                    (20180601, 0.13, 0.37, 0.5),
                    (20180701, 0.13, 0.37, 0.5),
                    (20180801, 0.13, 0.37, 0.5),
                    (20180901, 0.13, 0.37, 0.5),
                    (20181001, 0.13, 0.37, 0.5),
                    (20181101, 0.13, 0.37, 0.5),
                    (20181201, 0.13, 0.37, 0.5),
                    (20190101, 0.13, 0.37, 0.5),
                    (20190201, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20190301, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20190401, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20190501, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20190601, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20190701, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20190801, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20190901, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20191001, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20191101, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20191201, 0.10000000000000001, 0.40000000000000002, 0.5),
                    (20200101, 0.10000000000000001, 0.40000000000000002, 0.5)]
        debug = 0
        fillColor = None
        height = 85
        joinedLines = 1
        lineLabelArray = None
        lineLabelFormat = None
        lineLabelNudge = 10
        lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d9120c>
        lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d911cc>
        reversePlotOrder = 1
        strokeColor = None
        strokeWidth = 1
        width = 180
        x = 20
        xValueAxis.avoidBoundFrac = None
        xValueAxis.bottomAxisLabelSlack = 0.10000000000000001
        xValueAxis.dailyFreq = 0
        xValueAxis.dayOfWeekName = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sund
        xValueAxis.forceEndDate = 0
        xValueAxis.forceFirstDate = 0
        xValueAxis.forceZero = 0
```

```
            xValueAxis.gridEnd = 0
            xValueAxis.gridStart = 0
            xValueAxis.gridStrokeColor = Color(0,0,0)
            xValueAxis.gridStrokeDashArray = None
            xValueAxis.gridStrokeWidth = 0.25
            xValueAxis.joinAxis = None
            xValueAxis.joinAxisMode = None
            xValueAxis.joinAxisPos = None
            xValueAxis.labelTextFormat = None
            xValueAxis.labelTextPostFormat = None
            xValueAxis.labelTextScale = None
            xValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d912ac>
            xValueAxis.maximumTicks = 7
            xValueAxis.minimumTickSpacing = 10
            xValueAxis.monthName = ['January',
                                    'February',
                                    'March',
                                    'April',
                                    'May',
                                    'June',
                                    'July',
                                    'August',
                                    'September',
                                    'October',
                                    'November',
                                    'December']
            xValueAxis.niceMonth = 1
            xValueAxis.rangeRound = 'none'
            xValueAxis.strokeColor = Color(0,0,0)
            xValueAxis.strokeDashArray = None
            xValueAxis.strokeWidth = 1
            xValueAxis.style = 'normal'
            xValueAxis.tickDown = 5
            xValueAxis.tickUp = 0
            xValueAxis.valueMax = None
            xValueAxis.valueMin = None
            xValueAxis.valueStep = None
            xValueAxis.valueSteps = None
            xValueAxis.visible = 1
            xValueAxis.visibleAxis = 1
            xValueAxis.visibleGrid = 0
            xValueAxis.visibleLabels = 1
            xValueAxis.visibleTicks = 1
            xValueAxis.xLabelFormat = '{mm}/{yy}'
            xValueAxis.zrangePref = 0
        y = 10
            yValueAxis.avoidBoundFrac = None
            yValueAxis.forceZero = 0
            yValueAxis.gridEnd = 0
            yValueAxis.gridStart = 0
            yValueAxis.gridStrokeColor = Color(0,0,0)
            yValueAxis.gridStrokeDashArray = None
            yValueAxis.gridStrokeWidth = 0.25
            yValueAxis.joinAxis = None
            yValueAxis.joinAxisMode = None
            yValueAxis.joinAxisPos = None
            yValueAxis.labelTextFormat = None
            yValueAxis.labelTextPostFormat = None
            yValueAxis.labelTextScale = None
            yValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8d9112c>
            yValueAxis.leftAxisOrigShiftIPC = 0
            yValueAxis.leftAxisOrigShiftMin = 0
            yValueAxis.leftAxisPercent = 0
            yValueAxis.leftAxisSkipLL0 = 0
            yValueAxis.maximumTicks = 7
            yValueAxis.minimumTickSpacing = 10
            yValueAxis.rangeRound = 'none'
            yValueAxis.requiredRange = None
            yValueAxis.strokeColor = Color(0,0,0)
            yValueAxis.strokeDashArray = None
            yValueAxis.strokeWidth = 1
            yValueAxis.style = 'normal'
            yValueAxis.tickLeft = 5
            yValueAxis.tickRight = 0
            yValueAxis.valueMax = None
            yValueAxis.valueMin = None
            yValueAxis.valueStep = None
            yValueAxis.valueSteps = None
            yValueAxis.visible = 1
```

```
yValueAxis.visibleAxis = 1
yValueAxis.visibleGrid = 0
yValueAxis.visibleLabels = 1
yValueAxis.visibleTicks = 1
yValueAxis.zrangePref = 0
```

## Functions

### **sample1a( ... )**

A line plot with non-equidistant points in x-axis.

*Example*

```
def sample1a():
    "A line plot with non-equidistant points in x-axis."

    drawing = Drawing(400, 200)

    data = [
            ((1,1), (2,2), (2.5,1), (3,3), (4,5)),
            ((1,2), (2,3), (2.5,2), (3.5,5), (4,6))
        ]

    lp = LinePlot()

    lp.x = 50
    lp.y = 50
    lp.height = 125
    lp.width = 300
    lp.data = data
    lp.joinedLines = 1
    lp.strokeColor = colors.black

    lp.lines.symbol = makeMarker('UK_Flag')

    lp.lines[0].strokeWidth = 2
    lp.lines[1].strokeWidth = 4

    lp.xValueAxis.valueMin = 0
    lp.xValueAxis.valueMax = 5
    lp.xValueAxis.valueStep = 1

    lp.yValueAxis.valueMin = 0
    lp.yValueAxis.valueMax = 7
    lp.yValueAxis.valueStep = 1

    drawing.add(lp)

    return drawing
```
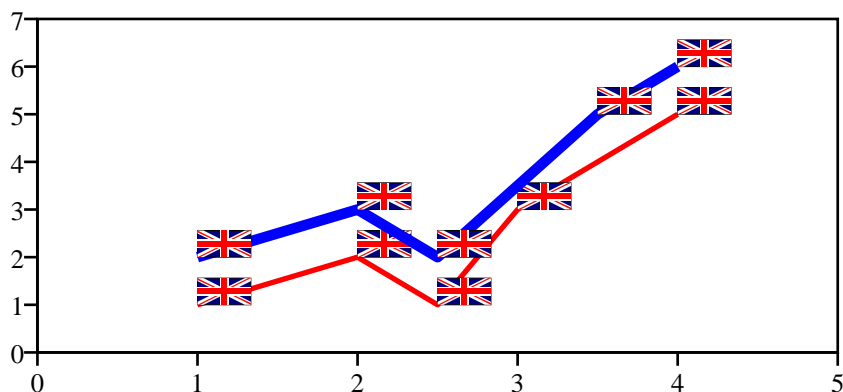
## **sample1b( ... )**

A line plot with non-equidistant points in x-axis.

*Example*

```
def sample1b():
    "A line plot with non-equidistant points in x-axis."

    drawing = Drawing(400, 200)

    data = [
            ((1,1), (2,2), (2.5,1), (3,3), (4,5)),
            ((1,2), (2,3), (2.5,2), (3.5,5), (4,6))
        ]

    lp = LinePlot()

    lp.x = 50
    lp.y = 50
    lp.height = 125
    lp.width = 300
    lp.data = data
    lp.joinedLines = 1
    lp.lines.symbol = makeMarker('Circle')
    lp.lineLabelFormat = '%2.0f'
    lp.strokeColor = colors.black

    lp.xValueAxis.valueMin = 0
    lp.xValueAxis.valueMax = 5
    lp.xValueAxis.valueSteps = [1, 2, 2.5, 3, 4, 5]
    lp.xValueAxis.labelTextFormat = '%2.1f'

    lp.yValueAxis.valueMin = 0
    lp.yValueAxis.valueMax = 7
    lp.yValueAxis.valueStep = 1

    drawing.add(lp)

    return drawing
```

## **sample1c( ... )**

A line plot with non-equidistant points in x-axis.

*Example*

```
def sample1c():
    "A line plot with non-equidistant points in x-axis."

    drawing = Drawing(400, 200)

    data = [
            ((1,1), (2,2), (2.5,1), (3,3), (4,5)),
            ((1,2), (2,3), (2.5,2), (3.5,5), (4,6))
        ]

    lp = LinePlot()

    lp.x = 50
    lp.y = 50
    lp.height = 125
    lp.width = 300
    lp.data = data
    lp.joinedLines = 1
    lp.lines[0].symbol = makeMarker('FilledCircle')
    lp.lines[1].symbol = makeMarker('Circle')
    lp.lineLabelFormat = '%2.0f'
    lp.strokeColor = colors.black

    lp.xValueAxis.valueMin = 0
    lp.xValueAxis.valueMax = 5
    lp.xValueAxis.valueSteps = [1, 2, 2.5, 3, 4, 5]
    lp.xValueAxis.labelTextFormat = '%2.1f'

    lp.yValueAxis.valueMin = 0
    lp.yValueAxis.valueMax = 7
    lp.yValueAxis.valueSteps = [1, 2, 3, 5, 6]

    drawing.add(lp)

    return drawing
```



---
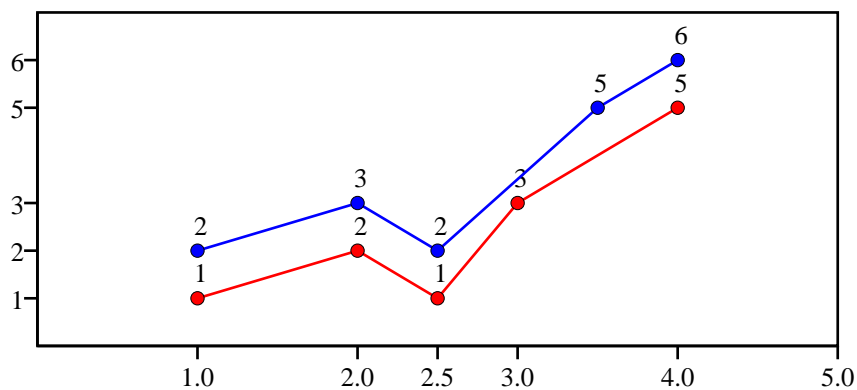
### sample2( ... )

A line plot with non-equidistant points in x-axis.

*Example*

```
def sample2():
    "A line plot with non-equidistant points in x-axis."

    drawing = Drawing(400, 200)

    data = [
        (('25/11/1991',1),
         ('30/11/1991',1.000933333),
         ('31/12/1991',1.0062),
         ('31/01/1992',1.0112),
         ('29/02/1992',1.0158),
         ('31/03/1992',1.020733333),
         ('30/04/1992',1.026133333),
         ('31/05/1992',1.030266667),
         ('30/06/1992',1.034466667),
         ('31/07/1992',1.038733333),
         ('31/08/1992',1.0422),
         ('30/09/1992',1.045533333),
         ('31/10/1992',1.049866667),
         ('30/11/1992',1.054733333),
         ('31/12/1992',1.061),
         ),
        ]

    data[0] = preprocessData(data[0])

    lp = LinePlot()

    lp.x = 50
    lp.y = 50
    lp.height = 125
    lp.width = 300
    lp.data = data
    lp.joinedLines = 1
    lp.lines.symbol = makeMarker('FilledDiamond')
    lp.strokeColor = colors.black

    start = mktime(mkTimeTuple('25/11/1991'))
    t0 = mktime(mkTimeTuple('30/11/1991'))
    t1 = mktime(mkTimeTuple('31/12/1991'))
    t2 = mktime(mkTimeTuple('31/03/1992'))
    t3 = mktime(mkTimeTuple('30/06/1992'))
    t4 = mktime(mkTimeTuple('30/09/1992'))
    end = mktime(mkTimeTuple('31/12/1992'))
    lp.xValueAxis.valueMin = start
    lp.xValueAxis.valueMax = end
    lp.xValueAxis.valueSteps = [start, t0, t1, t2, t3, t4, end]
    lp.xValueAxis.labelTextFormat = seconds2str
    lp.xValueAxis.labels[1].dy = -20
    lp.xValueAxis.labels[2].dy = -35

    lp.yValueAxis.labelTextFormat = '%4.2f'
    lp.yValueAxis.valueMin = 100
    lp.yValueAxis.valueMax = 110
    lp.yValueAxis.valueStep = 2

    drawing.add(lp)

    return drawing
```
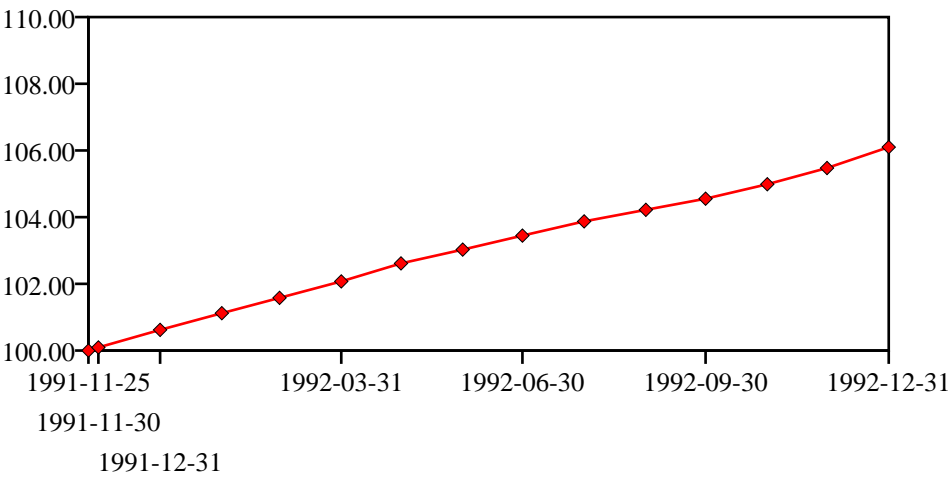
## **piecharts**

Basic Pie Chart class.

This permits you to customize and pop out individual wedges;
supports elliptical and circular pies.

## Classes

## **AbstractPieChart(PlotArea)**

## Public Attributes

**background** Handle to background object.

**debug** Used only for debugging.

**fillColor** Color of the plot area interior.

**height** Height of the chart.

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.


*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```


*Properties of Example Widget*

```
background = None
debug = 0
fillColor = None
height = 85
strokeColor = None
strokeWidth = 1
width = 180
x = 20
y = 10
```

## `LegendedPie(Pie)`

Pie with a two part legend (one editable with swatches, one hidden without swatches).

## Public Attributes

**background** Handle to background object.

**bottomPadding** Padding at bottom of drawing

**checkLabelOverlap** If true check and attempt to fix standard label overlaps(default off)

**data** list of numbers defining wedge sizes; need not sum to 1

**debug** Used only for debugging.

**direction** 'clockwise' or 'anticlockwise'

**drawLegend** If true then create and draw legend

**fillColor** Color of the plot area interior.

**height** Height of the chart.

**labels** optional list of labels to use for each data point

**leftPadding** Padding on left of drawing

**legend1** Handle to legend for pie

**legendNumberFormat** Formatting routine for number on right hand side of legend.

**legendNumberOffset** Horizontal space between legend and numbers on r/hand side

**legend_data** Numbers used on r/hand side of legend (or None)

**legend_names** Names used in legend (or None)

**orderMode** None

**other_threshold** A value for doing threshholding, not used yet.

**pieAndLegend_colors** Colours used for both swatches and pie

**pointerLabelMode**

**rightPadding** Padding on right of drawing

**sameRadii** If true make x/y radii the same(default off)

**simpleLabels** If true(default) use String not super duper WedgeLabel

**slices** collection of wedge descriptor objects

**startAngle** angle of first slice; like the compass, 0 is due North

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**topPadding** Padding at top of drawing

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**xradius** X direction Radius

**y** Y position of the lower-left corner of the chart.

**yradius** Y direction Radius

*Example*

```
def demo(self, drawing=None):
    if not drawing:
        tx,ty = self._getDrawingDimensions()
        drawing = Drawing(tx, ty)
    drawing.add(self.draw())
    return drawing
```

*Properties of Example Widget*

```
background = None
bottomPadding = 5
checkLabelOverlap = 0
data = [38.399999999999999,
        20.699999999999999,
        18.899999999999999,
        15.4,
        6.5999999999999996]
debug = 0
direction = 'clockwise'
drawLegend = 1
fillColor = None
height = 100
labels = None
leftPadding = 5
legend1.alignment = 'right'
legend1.autoXPadding = 5
legend1.autoYPadding = 2
legend1.boxAnchor = 'nw'
legend1.colEndCallout = None
legend1.colorNamePairs = [(Color(1,0,0), 'red'),
                          (Color(0,0,1), 'blue'),
                          (Color(0,.501961,0), 'green'),
                          (Color(1,.752941,.796078), 'pink'),
                          (Color(1,1,0), 'yellow'),
                          (PCMYKColor(11,11,72,0,spotName='PANTONE 458 CV'), 'AAA:'),
                          (PCMYKColor(100,65,0,30,spotName='PANTONE 288 CV'), 'AA:'),
                          (PCMYKColor(11,11,72,0,spotName='PANTONE 458 CV',density=75), 'A:'),
                          (PCMYKColor(100,65,0,30,spotName='PANTONE 288 CV',density=75), 'BBB:'),
                          (PCMYKColor(11,11,72,0,spotName='PANTONE 458 CV',density=50), 'NR:')]
legend1.columnMaximum = 7
legend1.deltax = 5.6699999999999999
legend1.deltay = 14.17
legend1.dividerColor = Color(0,0,0)
legend1.dividerDashArray = None
legend1.dividerLines = 0
legend1.dividerOffsX = (0, 0)
legend1.dividerOffsY = 0
legend1.dividerWidth = 0.5
legend1.dx = 5.6699999999999999
legend1.dxTextSpace = 11.390000000000001
legend1.dy = 5.6699999999999999
legend1.fillColor = Color(0,0,0)
legend1.fontName = 'Helvetica-Bold'
legend1.fontSize = 6
legend1.sepSpace = 0
legend1.strokeColor = Color(0,0,0)
legend1.strokeWidth = 0.5
legend1.swatchMarker = None
legend1.variColumn = 0
legend1.x = 117
legend1.y = 100
legend1.yGap = 0
legendNumberFormat = '%.1f%%'
legendNumberOffset = 51
legend_data = [38.399999999999999,
               20.699999999999999,
               18.899999999999999,
               15.4,
               6.5999999999999996]
legend_names = ['AAA:', 'AA:', 'A:', 'BBB:', 'NR:']
orderMode = 'fixed'
pieAndLegend_colors = [PCMYKColor(11,11,72,0,spotName='PANTONE 458 CV'),
                       PCMYKColor(100,65,0,30,spotName='PANTONE 288 CV'),
                       PCMYKColor(11,11,72,0,spotName='PANTONE 458 CV',density=75),
```

```
                                    PCMYKColor(100,65,0,30,spotName='PANTONE 288 CV',density=75),
                                    PCMYKColor(11,11,72,0,spotName='PANTONE 458 CV',density=50),
                                    PCMYKColor(100,65,0,30,spotName='PANTONE 288 CV',density=50)]
        pointerLabelMode = None
        rightPadding = 5
        sameRadii = False
        simpleLabels = 1
        slices = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8e1b72c>
        startAngle = 90
        strokeColor = None
        strokeWidth = 1
        topPadding = 5
        width = 100
        x = 0
        xradius = None
        y = 0
        yradius = None
```

## Pie(AbstractPieChart)

## Public Attributes

**background** Handle to background object.

**checkLabelOverlap** If true check and attempt to fix standard label overlaps(default off)

**data** list of numbers defining wedge sizes; need not sum to 1

**debug** Used only for debugging.

**direction** 'clockwise' or 'anticlockwise'

**fillColor** Color of the plot area interior.

**height** Height of the chart.

**labels** optional list of labels to use for each data point

**orderMode** None

**other_threshold** A value for doing threshholding, not used yet.

**pointerLabelMode**

**sameRadii** If true make x/y radii the same(default off)

**simpleLabels** If true(default) use String not super duper WedgeLabel

**slices** collection of wedge descriptor objects

**startAngle** angle of first slice; like the compass, 0 is due North

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**xradius** X direction Radius

**y** Y position of the lower-left corner of the chart.

**yradius** Y direction Radius

*Example*

```
def demo(self):
    d = Drawing(200, 100)

    pc = Pie()
    pc.x = 50
    pc.y = 10
    pc.width = 100
    pc.height = 80
    pc.data = [10,20,30,40,50,60]
    pc.labels = ['a','b','c','d','e','f']

    pc.slices.strokeWidth=0.5
    pc.slices[3].popout = 10
    pc.slices[3].strokeWidth = 2
    pc.slices[3].strokeDashArray = [2,2]
    pc.slices[3].labelRadius = 1.75
    pc.slices[3].fontColor = colors.red
    pc.slices[0].fillColor = colors.darkcyan
    pc.slices[1].fillColor = colors.blueviolet
    pc.slices[2].fillColor = colors.blue
    pc.slices[3].fillColor = colors.cyan
    pc.slices[4].fillColor = colors.aquamarine
    pc.slices[5].fillColor = colors.cadetblue
    pc.slices[6].fillColor = colors.lightcoral
```

```
        d.add(pc)
        return d
```

*Properties of Example Widget*

```
background = None
checkLabelOverlap = 0
data = [1, 2.2999999999999998, 1.7, 4.2000000000000002]
debug = 0
direction = 'clockwise'
fillColor = None
height = 100
labels = None
orderMode = 'fixed'
pointerLabelMode = None
sameRadii = False
simpleLabels = 1
slices = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8e2a3ec>
startAngle = 90
strokeColor = None
strokeWidth = 1
width = 100
x = 0
xradius = None
y = 0
yradius = None
```

### `Pie3d(Pie)`

## Public Attributes

**angle_3d** The view angle.

**background** Handle to background object.

**checkLabelOverlap** If true check and attempt to fix standard label overlaps(default off)

**data** list of numbers defining wedge sizes; need not sum to 1

**debug** Used only for debugging.

**depth_3d** depth of the pie.

**direction** 'clockwise' or 'anticlockwise'

**fillColor** Color of the plot area interior.

**height** Height of the chart.

**labels** optional list of labels to use for each data point

**orderMode** None

**other_threshold** A value for doing threshholding, not used yet.

**perspective** A flattening parameter.

**pointerLabelMode**

**sameRadii** If true make x/y radii the same(default off)

**simpleLabels** If true(default) use String not super duper WedgeLabel

**slices** collection of wedge descriptor objects

**startAngle** angle of first slice; like the compass, 0 is due North

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**xradius** X direction Radius

**y** Y position of the lower-left corner of the chart.

**yradius** Y direction Radius

*Example*

```
def demo(self):
    d = Drawing(200, 100)

    pc = Pie()
    pc.x = 50
    pc.y = 10
    pc.width = 100
    pc.height = 80
    pc.data = [10,20,30,40,50,60]
    pc.labels = ['a','b','c','d','e','f']

    pc.slices.strokeWidth=0.5
    pc.slices[3].popout = 10
    pc.slices[3].strokeWidth = 2
    pc.slices[3].strokeDashArray = [2,2]
    pc.slices[3].labelRadius = 1.75
    pc.slices[3].fontColor = colors.red
    pc.slices[0].fillColor = colors.darkcyan
```

```
                    pc.slices[1].fillColor = colors.blueviolet
                    pc.slices[2].fillColor = colors.blue
                    pc.slices[3].fillColor = colors.cyan
                    pc.slices[4].fillColor = colors.aquamarine
                    pc.slices[5].fillColor = colors.cadetblue
                    pc.slices[6].fillColor = colors.lightcoral
                    self.slices[1].visible = 0
                    self.slices[3].visible = 1
                    self.slices[4].visible = 1
                    self.slices[5].visible = 1
                    self.slices[6].visible = 0

                    d.add(pc)
                    return d
```

*Properties of Example Widget*

```
        checkLabelOverlap = 0
        data = [12.5, 20.100000000000001, 2.0, 22.0, 5.0, 18.0, 13.0]
        direction = 'clockwise'
        height = 200
        labels = None
        simpleLabels = 1
        slices = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8e2a3ec>
        startAngle = 90
        width = 300
        x = 0
        y = 0
```

## WedgeLabel(Label)

## Public Attributes

**angle** None

**bottomPadding** padding at bottom of box

**boxAnchor** None

**boxFillColor** None

**boxStrokeColor** None

**boxStrokeWidth** None

**boxTarget** None

**dx** None

**dy** None

**fillColor** None

**fontName** None

**fontSize** None

**height** None

**leading** None

**leftPadding** padding at left of box

**maxWidth** None

**rightPadding** padding at right of box

**strokeColor** None

**strokeWidth** None

**text** None

**textAnchor** None

**topPadding** padding at top of box

**visible** True if the label is to be drawn

**width** None

**x** None

**y** None

*Example*

```
def demo(self):
    """This shows a label positioned with its top right corner
    at the top centre of the drawing, and rotated 45 degrees."""

    d = Drawing(200, 100)

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

    lab = Label()
    lab.setOrigin(100,90)
    lab.boxAnchor = 'ne'
    lab.angle = 45
    lab.dx = 0
    lab.dy = -20
    lab.boxStrokeColor = colors.green
```

```
                    lab.setText('Another\nMulti-Line\nString')
                    d.add(lab)

                    return d
```

*Properties of Example Widget*

```
        angle = 0
        bottomPadding = 0
        boxAnchor = 'c'
        boxFillColor = None
        boxStrokeColor = None
        boxStrokeWidth = 0.5
        boxTarget = 'normal'
        dx = 0
        dy = 0
        fillColor = Color(0,0,0)
        fontName = 'Times-Roman'
        fontSize = 10
        height = None
        leading = None
        leftPadding = 0
        maxWidth = None
        rightPadding = 0
        strokeColor = None
        strokeWidth = 0.10000000000000001
        textAnchor = 'start'
        topPadding = 0
        visible = 1
        width = None
        x = 0
        y = 0
```

## Functions

### sample0a( ... )

Make a degenerated pie chart with only one slice.

*Example*

```
def sample0a():
    "Make a degenerated pie chart with only one slice."

    d = Drawing(400, 200)

    pc = Pie()
    pc.x = 150
    pc.y = 50
    pc.data = [10]
    pc.labels = ['a']
    pc.slices.strokeWidth=1#0.5

    d.add(pc)

    return d
```



a
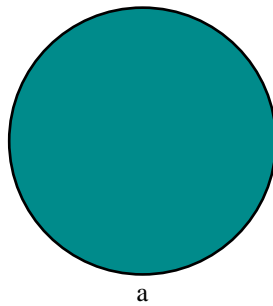
## **sample0b( ... )**

Make a degenerated pie chart with only one slice.

*Example*

```
def sample0b():
    "Make a degenerated pie chart with only one slice."

    d = Drawing(400, 200)

    pc = Pie()
    pc.x = 150
    pc.y = 50
    pc.width = 120
    pc.height = 100
    pc.data = [10]
    pc.labels = ['a']
    pc.slices.strokeWidth=1#0.5

    d.add(pc)

    return d
```

a
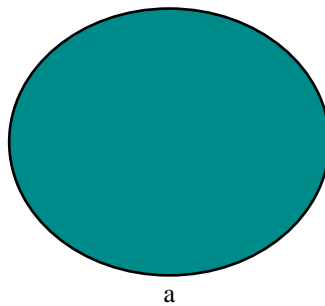
## sample1( ... )

Make a typical pie chart with with one slice treated in a special way.

*Example*

```
def sample1():
    "Make a typical pie chart with with one slice treated in a special way."

    d = Drawing(400, 200)

    pc = Pie()
    pc.x = 150
    pc.y = 50
    pc.data = [10, 20, 30, 40, 50, 60]
    pc.labels = ['a', 'b', 'c', 'd', 'e', 'f']

    pc.slices.strokeWidth=1#0.5
    pc.slices[3].popout = 20
    pc.slices[3].strokeWidth = 2
    pc.slices[3].strokeDashArray = [2,2]
    pc.slices[3].labelRadius = 1.75
    pc.slices[3].fontColor = colors.red

    d.add(pc)

    return d
```
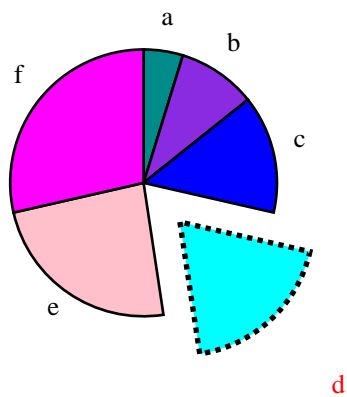
## sample2( ... )

Make a pie chart with nine slices.

*Example*

```
def sample2():
    "Make a pie chart with nine slices."

    d = Drawing(400, 200)

    pc = Pie()
    pc.x = 125
    pc.y = 25
    pc.data = [0.31, 0.148, 0.108,
               0.076, 0.033, 0.03,
               0.019, 0.126, 0.15]
    pc.labels = ['1', '2', '3', '4', '5', '6', '7', '8', 'X']

    pc.width = 150
    pc.height = 150
    pc.slices.strokeWidth=1#0.5

    pc.slices[0].fillColor = colors.steelblue
    pc.slices[1].fillColor = colors.thistle
    pc.slices[2].fillColor = colors.cornflower
    pc.slices[3].fillColor = colors.lightsteelblue
    pc.slices[4].fillColor = colors.aquamarine
    pc.slices[5].fillColor = colors.cadetblue
    pc.slices[6].fillColor = colors.lightcoral
    pc.slices[7].fillColor = colors.tan
    pc.slices[8].fillColor = colors.darkseagreen

    d.add(pc)

    return d
```
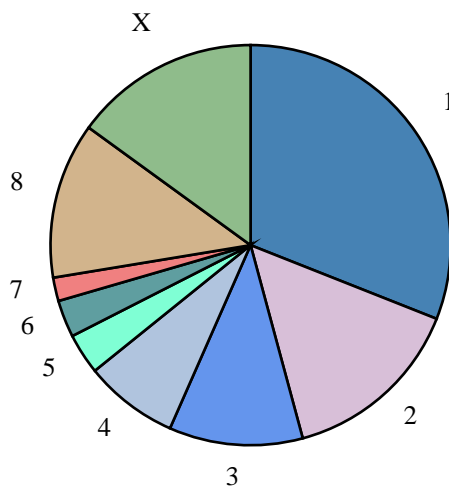
## **sample3( ... )**

Make a pie chart with a very slim slice.

*Example*

```
def sample3():
    "Make a pie chart with a very slim slice."

    d = Drawing(400, 200)

    pc = Pie()
    pc.x = 125
    pc.y = 25

    pc.data = [74, 1, 25]

    pc.width = 150
    pc.height = 150
    pc.slices.strokeWidth=1#0.5
    pc.slices[0].fillColor = colors.steelblue
    pc.slices[1].fillColor = colors.thistle
    pc.slices[2].fillColor = colors.cornflower

    d.add(pc)

    return d
```
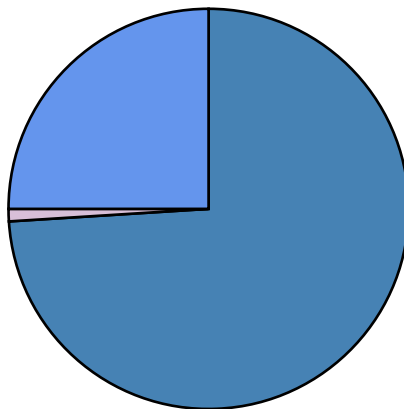
## sample4( ... )

Make a pie chart with several very slim slices.

*Example*

```
def sample4():
    "Make a pie chart with several very slim slices."

    d = Drawing(400, 200)

    pc = Pie()
    pc.x = 125
    pc.y = 25

    pc.data = [74, 1, 1, 1, 1, 22]

    pc.width = 150
    pc.height = 150
    pc.slices.strokeWidth=1#0.5
    pc.slices[0].fillColor = colors.steelblue
    pc.slices[1].fillColor = colors.thistle
    pc.slices[2].fillColor = colors.cornflower
    pc.slices[3].fillColor = colors.lightsteelblue
    pc.slices[4].fillColor = colors.aquamarine
    pc.slices[5].fillColor = colors.cadetblue

    d.add(pc)

    return d
```
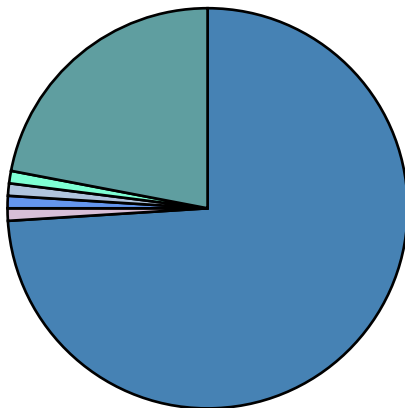
## **slidebox**

## Classes

### **SlideBox(Widget)**

Returns a slidebox widget

## Public Attributes

**background** Colour of the background to the drawing (if any)

**bottomPadding** Padding at bottom of drawing

**boxHeight** Height of the boxes

**boxOutlineColor** Colour used to outline the boxes (if any)

**boxOutlineWidth** Width of the box outline (if any)

**boxSpacing** Space between the boxes

**boxWidth** Width of the boxes

**endColor** Color of last box

**labelFillColor** Colour for number insides

**labelFontName** Name of font used for the labels

**labelFontSize** Size of font used for the labels

**labelStrokeColor** Colour for for number outlines

**leftPadding** Padding on left of drawing

**numberOfBoxes** How many boxes there are

**rightPadding** Padding on right of drawing

**sourceLabelFillColor** Colour ink for the 'source' label (bottom right)

**sourceLabelFontName** Name of font used for the 'source' label

**sourceLabelFontSize** Font size for the 'source' label

**sourceLabelOffset** Padding at bottom of drawing

**sourceLabelText** Text used for the 'source' label (can be empty)

**startColor** Color of first box

**topPadding** Padding at top of drawing

**triangleFillColor** Colour of indicator triangles

**triangleHeight** Height of indicator triangles

**trianglePosition** Which box is highlighted by the triangles

**triangleStrokeColor** Colour of indicator triangle outline

**triangleStrokeWidth** Colour of indicator triangle outline

**triangleWidth** Width of indicator triangles


*Example*
```
def demo(self,drawing=None):
```

```
from reportlab.lib import colors
if not drawing:
    tx,ty=self._getDrawingDimensions()
    drawing = Drawing(tx,ty)
drawing.add(self.draw())
return drawing
```



*Source: ReportLab*

*Properties of Example Widget*

```
background = None
bottomPadding = 5
boxHeight = 15.590551181102363
boxOutlineColor = Color(0,0,0)
boxOutlineWidth = 0.57999999999999996
boxSpacing = 2.1259842519685037
boxWidth = 20.69291338582677
endColor = Color(.098039,.301961,.529412)
labelFillColor = Color(1,1,1)
labelFontName = 'Helvetica-Bold'
labelFontSize = 10
labelStrokeColor = Color(0,0,0)
leftPadding = 5
numberOfBoxes = 7
rightPadding = 5
sourceLabelFillColor = Color(0,0,0)
sourceLabelFontName = 'Helvetica-Oblique'
sourceLabelFontSize = 6
sourceLabelOffset = 5.6692913385826778
sourceLabelText = 'Source: ReportLab'
startColor = Color(.909804,.878431,.466667)
topPadding = 5
triangleFillColor = Color(1,1,1)
triangleHeight = 3.401574803149606
trianglePosition = 7
triangleStrokeColor = Color(0,0,0)
triangleStrokeWidth = 0.57999999999999996
triangleWidth = 10.771653543307087
```

## spider

Spider Chart

Normal use shows variation of 5-10 parameters against some 'norm' or target.
When there is more than one series, place the series with the largest
numbers first, as it will be overdrawn by each successive one.

## Classes

## SpiderChart(PlotArea)

## Public Attributes

**background** Handle to background object.

**data** Data to be plotted, list of (lists of) numbers.

**debug** Used only for debugging.

**direction** 'clockwise' or 'anticlockwise'

**fillColor** Color of the plot area interior.

**height** Height of the chart.

**labels** optional list of labels to use for each data point

**spokeLabels** collection of spoke label descriptor objects

**spokes** collection of spoke descriptor objects

**startAngle** angle of first slice; like the compass, 0 is due North

**strandLabels** collection of strand label descriptor objects

**strands** collection of strand descriptor objects

**strokeColor** Color of the plot area border.

**strokeWidth** Width plot area border.

**width** Width of the chart.

**x** X position of the lower-left corner of the chart.

**y** Y position of the lower-left corner of the chart.


*Example*
```
def demo(self):
    d = Drawing(200, 200)
    d.add(SpiderChart())
    return d
```
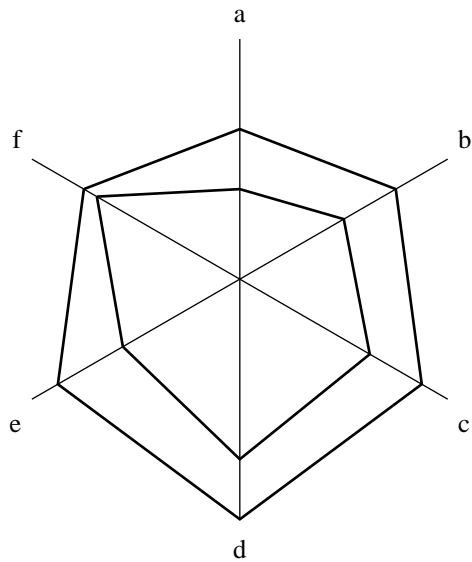
*Properties of Example Widget*

```
background = None
data = [[10, 12, 14, 16, 14, 12], [6, 8, 10, 12, 9, 11]]
debug = 0
direction = 'clockwise'
fillColor = None
height = 180
labels = ['a', 'b', 'c', 'd', 'e', 'f']
spokeLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8eca6cc>
spokes = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8eca46c>
startAngle = 90
strandLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8eca5ac>
strands = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8eca68c>
strokeColor = None
strokeWidth = 1
width = 180
x = 10
y = 10
```

## `SpokeLabel(WedgeLabel)`

## Public Attributes

**angle** None

**bottomPadding** padding at bottom of box

**boxAnchor** None

**boxFillColor** None

**boxStrokeColor** None

**boxStrokeWidth** None

**boxTarget** None

**dx** None

**dy** None

**fillColor** None

**fontName** None

**fontSize** None

**height** None

**leading** None

**leftPadding** padding at left of box

**maxWidth** None

**rightPadding** padding at right of box

**strokeColor** None

**strokeWidth** None

**text** None

**textAnchor** None

**topPadding** padding at top of box

**visible** True if the label is to be drawn

**width** None

**x** None

**y** None

*Example*

```
def demo(self):
    """This shows a label positioned with its top right corner
    at the top centre of the drawing, and rotated 45 degrees."""

    d = Drawing(200, 100)

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

    lab = Label()
    lab.setOrigin(100,90)
    lab.boxAnchor = 'ne'
    lab.angle = 45
    lab.dx = 0
    lab.dy = -20
    lab.boxStrokeColor = colors.green
```

```
        lab.setText('Another\nMulti-Line\nString')
        d.add(lab)

        return d
```

*Properties of Example Widget*

```
angle = 0
bottomPadding = 0
boxAnchor = 'c'
boxFillColor = None
boxStrokeColor = None
boxStrokeWidth = 0.5
boxTarget = 'normal'
dx = 0
dy = 0
fillColor = Color(0,0,0)
fontName = 'Times-Roman'
fontSize = 10
height = None
leading = None
leftPadding = 0
maxWidth = None
rightPadding = 0
strokeColor = None
strokeWidth = 0.10000000000000001
textAnchor = 'start'
topPadding = 0
visible = 1
width = None
x = 0
y = 0
```

## **StrandLabel(SpokeLabel)**

## Public Attributes

**angle** None

**bottomPadding** padding at bottom of box

**boxAnchor** None

**boxFillColor** None

**boxStrokeColor** None

**boxStrokeWidth** None

**boxTarget** None

**dR** radial shift for label

**dx** None

**dy** None

**fillColor** None

**fontName** None

**fontSize** None

**format** Format for the label

**height** None

**leading** None

**leftPadding** padding at left of box

**maxWidth** None

**rightPadding** padding at right of box

**strokeColor** None

**strokeWidth** None

**text** None

**textAnchor** None

**topPadding** padding at top of box

**visible** True if the label is to be drawn

**width** None

**x** None

**y** None

*Example*

```
def demo(self):
    """This shows a label positioned with its top right corner
    at the top centre of the drawing, and rotated 45 degrees."""

    d = Drawing(200, 100)

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

    lab = Label()
    lab.setOrigin(100,90)
    lab.boxAnchor = 'ne'
```

```
                        lab.angle = 45
                        lab.dx = 0
                        lab.dy = -20
                        lab.boxStrokeColor = colors.green
                        lab.setText('Another\nMulti-Line\nString')
                        d.add(lab)

                        return d
```

*Properties of Example Widget*

```
                angle = 0
                bottomPadding = 0
                boxAnchor = 'c'
                boxFillColor = None
                boxStrokeColor = None
                boxStrokeWidth = 0.5
                boxTarget = 'normal'
                dR = 0
                dx = 0
                dy = 0
                fillColor = Color(0,0,0)
                fontName = 'Times-Roman'
                fontSize = 10
                format = ''
                height = None
                leading = None
                leftPadding = 0
                maxWidth = None
                rightPadding = 0
                strokeColor = None
                strokeWidth = 0.10000000000000001
                textAnchor = 'start'
                topPadding = 0
                visible = 1
                width = None
                x = 0
                y = 0
```

## Functions

### sample1( ... )

Make a simple spider chart

*Example*

```
def sample1():
    "Make a simple spider chart"
    d = Drawing(400, 400)
    sp = SpiderChart()
    sp.x = 50
    sp.y = 50
    sp.width = 300
    sp.height = 300
    sp.data = [[10,12,14,16,14,12], [6,8,10,12,9,15],[7,8,17,4,12,8]]
    sp.labels = ['a','b','c','d','e','f']
    sp.strands[0].strokeColor = colors.cornsilk
    sp.strands[1].strokeColor = colors.cyan
    sp.strands[2].strokeColor = colors.palegreen
    sp.strands[0].fillColor = colors.cornsilk
    sp.strands[1].fillColor = colors.cyan
    sp.strands[2].fillColor = colors.palegreen
    sp.spokes.strokeDashArray = (2,2)
    d.add(sp)
    return d
```
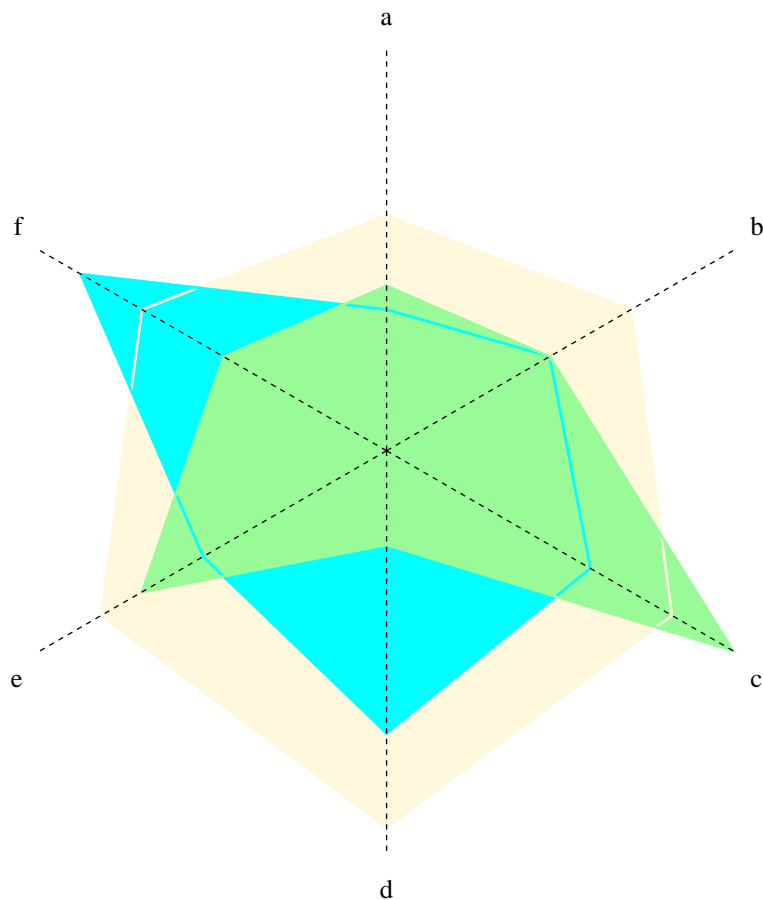
## **sample2( ... )**

Make a spider chart with markers, but no fill

*Example*

```
def sample2():
    "Make a spider chart with markers, but no fill"
    d = Drawing(400, 400)
    sp = SpiderChart()
    sp.x = 50
    sp.y = 50
    sp.width = 300
    sp.height = 300
    sp.data = [[10,12,14,16,14,12], [6,8,10,12,9,15],[7,8,17,4,12,8]]
    sp.labels = ['U','V','W','X','Y','Z']
    sp.strands.strokeWidth = 1
    sp.strands[0].fillColor = colors.pink
    sp.strands[1].fillColor = colors.lightblue
    sp.strands[2].fillColor = colors.palegreen
    sp.strands[0].strokeColor = colors.red
    sp.strands[1].strokeColor = colors.blue
    sp.strands[2].strokeColor = colors.green
    sp.strands.symbol = "FilledDiamond"
    sp.strands[1].symbol = makeMarker("Circle")
    sp.strands[1].symbol.strokeWidth = 0.5
    sp.strands[1].symbol.fillColor = colors.yellow
    sp.strands.symbolSize = 6
    sp.strandLabels[0,3]._text = 'special'
    sp.strandLabels[0,1]._text = 'one'
    sp.strandLabels[0,0]._text = 'zero'
    sp.strandLabels[1,0]._text = 'Earth'
    sp.strandLabels[2,2]._text = 'Mars'
    sp.strandLabels.format = 'values'
    sp.strandLabels.dR = -5
    d.add(sp)
    return d
```
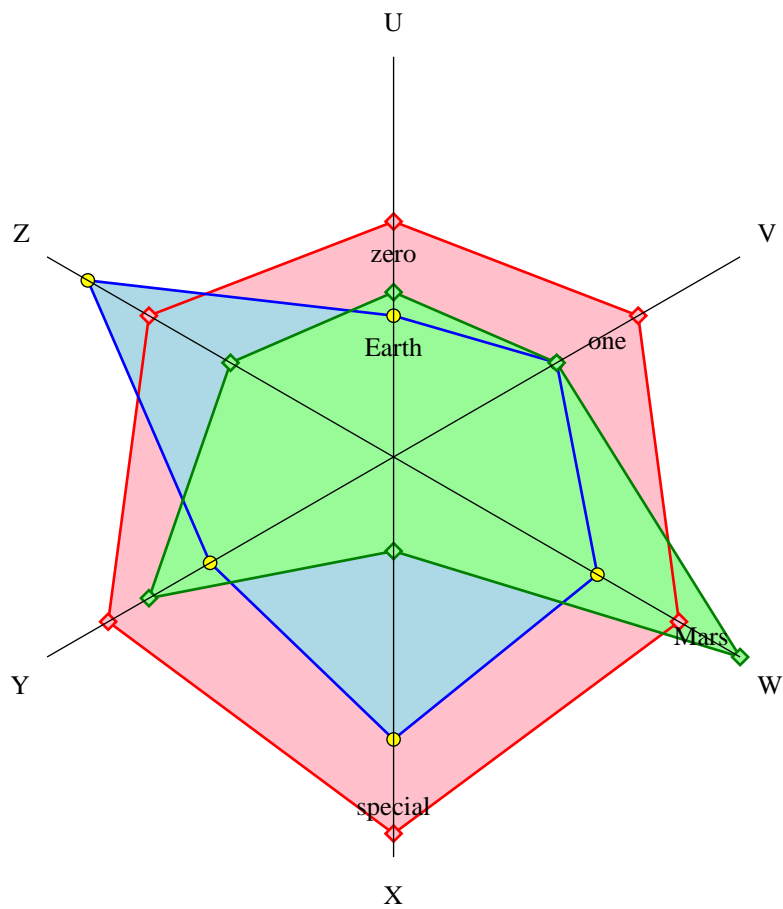
## `textlabels`

#Copyright ReportLab Europe Ltd. 2000-2004
#see license.txt for license details
#history http://www.reportlab.co.uk/cgi-bin/viewcvs.cgi/public/reportlab/trunk/reportlab/graphics/charts/textlabels.py

## Classes

### `BarChartLabel(Label)`

An extended Label allowing for nudging, lines visibility etc

## Public Attributes

**angle** None

**bottomPadding** padding at bottom of box

**boxAnchor** None

**boxFillColor** None

**boxStrokeColor** None

**boxStrokeWidth** None

**boxTarget** None

**dx** None

**dy** None

**fillColor** None

**fixedEnd** None or fixed draw ends +/-

**fixedStart** None or fixed draw starts +/-

**fontName** None

**fontSize** None

**height** None

**leading** None

**leftPadding** padding at left of box

**lineStrokeColor** Color for a drawn line

**lineStrokeWidth** Non-zero for a drawn line

**maxWidth** None

**nudge** Non-zero sign dependent nudge

**rightPadding** padding at right of box

**strokeColor** None

**strokeWidth** None

**text** None

**textAnchor** None

**topPadding** padding at top of box

**visible** True if the label is to be drawn

**width** None

**x** None

**y** None

*Example*

```
def demo(self):
    """This shows a label positioned with its top right corner
    at the top centre of the drawing, and rotated 45 degrees."""

    d = Drawing(200, 100)

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

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

    return d
```

*Properties of Example Widget*

```
angle = 0
bottomPadding = 0
boxAnchor = 'c'
boxFillColor = None
boxStrokeColor = None
boxStrokeWidth = 0.5
boxTarget = 'normal'
dx = 0
dy = 0
fillColor = Color(0,0,0)
fixedEnd = None
fixedStart = None
fontName = 'Times-Roman'
fontSize = 10
height = None
leading = None
leftPadding = 0
lineStrokeColor = None
lineStrokeWidth = 0
maxWidth = None
nudge = 0
rightPadding = 0
strokeColor = None
strokeWidth = 0.10000000000000001
textAnchor = 'start'
topPadding = 0
visible = 1
width = None
x = 0
y = 0
```

## `Label(Widget)`

A text label to attach to something else, such as a chart axis.

This allows you to specify an offset, angle and many anchor properties relative to the label's origin.  It allows, for example, angled multiline axis labels.

# Public Attributes

**angle** None

**bottomPadding** padding at bottom of box

**boxAnchor** None

**boxFillColor** None

**boxStrokeColor** None

**boxStrokeWidth** None

**boxTarget** None

**dx** None

**dy** None

**fillColor** None

**fontName** None

**fontSize** None

**height** None

**leading** None

**leftPadding** padding at left of box

**maxWidth** None

**rightPadding** padding at right of box

**strokeColor** None

**strokeWidth** None

**text** None

**textAnchor** None

**topPadding** padding at top of box

**visible** True if the label is to be drawn

**width** None

**x** None

**y** None

*Example*

```
def demo(self):
    """This shows a label positioned with its top right corner
    at the top centre of the drawing, and rotated 45 degrees."""

    d = Drawing(200, 100)

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

```
lab = Label()
lab.setOrigin(100,90)
lab.boxAnchor = 'ne'
lab.angle = 45
lab.dx = 0
lab.dy = -20
lab.boxStrokeColor = colors.green
lab.setText('Another\nMulti-Line\nString')
d.add(lab)

return d
```

*Properties of Example Widget*

```
angle = 0
bottomPadding = 0
boxAnchor = 'c'
boxFillColor = None
boxStrokeColor = None
boxStrokeWidth = 0.5
boxTarget = 'normal'
dx = 0
dy = 0
fillColor = Color(0,0,0)
fontName = 'Times-Roman'
fontSize = 10
height = None
leading = None
leftPadding = 0
maxWidth = None
rightPadding = 0
strokeColor = None
strokeWidth = 0.10000000000000001
textAnchor = 'start'
topPadding = 0
visible = 1
width = None
x = 0
y = 0
```

## `NA_Label(BarChartLabel)`

An extended Label allowing for nudging, lines visibility etc

## Public Attributes

**angle** None

**bottomPadding** padding at bottom of box

**boxAnchor** None

**boxFillColor** None

**boxStrokeColor** None

**boxStrokeWidth** None

**boxTarget** None

**dx** None

**dy** None

**fillColor** None

**fixedEnd** None or fixed draw ends +/-

**fixedStart** None or fixed draw starts +/-

**fontName** None

**fontSize** None

**height** None

**leading** None

**leftPadding** padding at left of box

**lineStrokeColor** Color for a drawn line

**lineStrokeWidth** Non-zero for a drawn line

**maxWidth** None

**nudge** Non-zero sign dependent nudge

**rightPadding** padding at right of box

**strokeColor** None

**strokeWidth** None

**text** Text to be used for N/A values

**textAnchor** None

**topPadding** padding at top of box

**visible** True if the label is to be drawn

**width** None

**x** None

**y** None

*Example*
```
        def demo(self):
            """This shows a label positioned with its top right corner
            at the top centre of the drawing, and rotated 45 degrees."""
```

```
d = Drawing(200, 100)

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

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

return d
```

*Properties of Example Widget*

```
angle = 0
bottomPadding = 0
boxAnchor = 'c'
boxFillColor = None
boxStrokeColor = None
boxStrokeWidth = 0.5
boxTarget = 'normal'
dx = 0
dy = 0
fillColor = Color(0,0,0)
fixedEnd = None
fixedStart = None
fontName = 'Times-Roman'
fontSize = 10
height = None
leading = None
leftPadding = 0
lineStrokeColor = None
lineStrokeWidth = 0
maxWidth = None
nudge = 0
rightPadding = 0
strokeColor = None
strokeWidth = 0.10000000000000001
text = 'n/a'
textAnchor = 'start'
topPadding = 0
visible = 1
width = None
x = 0
y = 0
```

## **bubble**

#Autogenerated by ReportLab guiedit do not edit

## **Classes**

## **Bubble(_DrawingEditorMixin, Drawing)**

*Example*

```
def __init__(self,width=200,height=150,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    self._add(self,ScatterPlot(),name='chart',validate=None,desc="The main chart")
    self.chart.width      = 115
    self.chart.height     = 80
    self.chart.x          = 30
    self.chart.y          = 40
    self.chart.lines[0].strokeColor = color01
    self.chart.lines[1].strokeColor = color02
    self.chart.lines[2].strokeColor = color03
    self.chart.lines[3].strokeColor = color04
    self.chart.lines[4].strokeColor = color05
    self.chart.lines[5].strokeColor = color06
    self.chart.lines[6].strokeColor = color07
    self.chart.lines[7].strokeColor = color08
    self.chart.lines[8].strokeColor = color09
    self.chart.lines[9].strokeColor = color10
    self.chart.lines.symbol.kind        ='Circle'
    self.chart.lines.symbol.size        = 15
    self.chart.fillColor          = backgroundGrey
    self.chart.lineLabels.fontName            = 'Helvetica'
    self.chart.xValueAxis.labels.fontName     = 'Helvetica'
    self.chart.xValueAxis.labels.fontSize     = 7
    self.chart.xValueAxis.forceZero           = 0
    self.chart.data               = [((100,100), (200,200), (250,210), (300,300), (350,450))]
    self.chart.xValueAxis.avoidBoundFrac      = 1
    self.chart.xValueAxis.gridEnd             = 115
    self.chart.xValueAxis.tickDown            = 3
    self.chart.xValueAxis.visibleGrid         = 1
    self.chart.yValueAxis.tickLeft       = 3
    self.chart.yValueAxis.labels.fontName     = 'Helvetica'
    self.chart.yValueAxis.labels.fontSize     = 7
    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
    self.Title.fontName   = 'Helvetica-Bold'
    self.Title.fontSize   = 7
    self.Title.x          = 100
    self.Title.y          = 135
    self.Title._text      = 'Chart Title'
    self.Title.maxWidth   = 180
    self.Title.height     = 20
    self.Title.textAnchor ='middle'
    self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
    self.Legend.colorNamePairs = [(color01, 'Widgets')]
    self.Legend.fontName     = 'Helvetica'
    self.Legend.fontSize     = 7
    self.Legend.x            = 153
    self.Legend.y            = 85
    self.Legend.dxTextSpace  = 5
    self.Legend.dy           = 5
    self.Legend.dx           = 5
    self.Legend.deltay       = 5
    self.Legend.alignment    ='right'
    self.chart.lineLabelFormat  = None
    self.chart.xLabel            = 'X Axis'
    self.chart.y                 = 30
    self.chart.yLabel            = 'Y Axis'
    self.chart.yValueAxis.labelTextFormat     = '%d'
    self.chart.yValueAxis.forceZero           = 1
    self.chart.xValueAxis.forceZero           = 1


    self._add(self,0,name='preview',validate=None,desc=None)
```

**Chart Title**



## clustered_bar

#Autogenerated by ReportLab guiedit do not edit

## Classes

## ClusteredBar(_DrawingEditorMixin, Drawing)

*Example*

```
def __init__(self,width=200,height=150,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    self._add(self,HorizontalBarChart(),name='chart',validate=None,desc="The main chart")
    self.chart.width       = 115
    self.chart.height      = 80
    self.chart.x           = 30
    self.chart.y           = 40
    self.chart.bars[0].fillColor = color01
    self.chart.bars[1].fillColor = color02
    self.chart.bars[2].fillColor = color03
    self.chart.bars[3].fillColor = color04
    self.chart.bars[4].fillColor = color05
    self.chart.bars[5].fillColor = color06
    self.chart.bars[6].fillColor = color07
    self.chart.bars[7].fillColor = color08
    self.chart.bars[8].fillColor = color09
    self.chart.bars[9].fillColor = color10
    self.chart.fillColor         = backgroundGrey
    self.chart.barLabels.fontName            = 'Helvetica'
    self.chart.valueAxis.labels.fontName     = 'Helvetica'
    self.chart.valueAxis.labels.fontSize     = 6
    self.chart.valueAxis.forceZero           = 1
    self.chart.data            = [(100, 150, 180), (125, 180, 200)]
    self.chart.groupSpacing    = 15
    self.chart.valueAxis.avoidBoundFrac      = 1
    self.chart.valueAxis.gridEnd             = 80
    self.chart.valueAxis.tickDown            = 3
    self.chart.valueAxis.visibleGrid         = 1
    self.chart.categoryAxis.categoryNames    = ['North', 'South', 'Central']
    self.chart.categoryAxis.tickLeft         = 3
    self.chart.categoryAxis.labels.fontName  = 'Helvetica'
    self.chart.categoryAxis.labels.fontSize  = 6
    self.chart.categoryAxis.labels.dx        = -3
    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
    self.Title.fontName   = 'Helvetica-Bold'
    self.Title.fontSize   = 7
    self.Title.x          = 100
    self.Title.y          = 135
    self.Title._text      = 'Chart Title'
    self.Title.maxWidth   = 180
    self.Title.height     = 20
    self.Title.textAnchor ='middle'
```

```
self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
self.Legend.fontName        = 'Helvetica'
self.Legend.fontSize        = 7
self.Legend.x               = 153
self.Legend.y               = 85
self.Legend.dxTextSpace     = 5
self.Legend.dy              = 5
self.Legend.dx              = 5
self.Legend.deltay          = 5
self.Legend.alignment       ='right'
self._add(self,Label(),name='XLabel',validate=None,desc="The label on the horizontal axis")
self.XLabel.fontName        = 'Helvetica'
self.XLabel.fontSize        = 7
self.XLabel.x               = 85
self.XLabel.y               = 10
self.XLabel.textAnchor      ='middle'
self.XLabel.maxWidth        = 100
self.XLabel.height          = 20
self.XLabel._text           = "X Axis"
self._add(self,Label(),name='YLabel',validate=None,desc="The label on the vertical axis")
self.YLabel.fontName        = 'Helvetica'
self.YLabel.fontSize        = 7
self.YLabel.x               = 12
self.YLabel.y               = 80
self.YLabel.angle           = 90
self.YLabel.textAnchor      ='middle'
self.YLabel.maxWidth        = 100
self.YLabel.height          = 20
self.YLabel._text           = "Y Axis"
self._add(self,0,name='preview',validate=None,desc=None)
```



## clustered_column

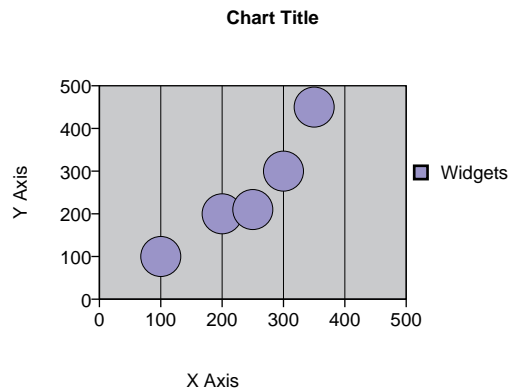#Autogenerated by ReportLab guiedit do not edit

## Classes

## ClusteredColumn(_DrawingEditorMixin, Drawing)

*Example*
```
def __init__(self,width=200,height=150,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    self._add(self,VerticalBarChart(),name='chart',validate=None,desc="The main chart")
    self.chart.width       = 115
    self.chart.height      = 80
    self.chart.x           = 30
    self.chart.y           = 40
    self.chart.bars[0].fillColor = color01
    self.chart.bars[1].fillColor = color02
```
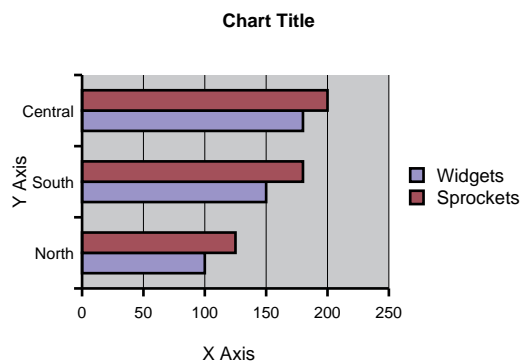
```
self.chart.bars[2].fillColor = color03
self.chart.bars[3].fillColor = color04
self.chart.bars[4].fillColor = color05
self.chart.bars[5].fillColor = color06
self.chart.bars[6].fillColor = color07
self.chart.bars[7].fillColor = color08
self.chart.bars[8].fillColor = color09
self.chart.bars[9].fillColor = color10
self.chart.fillColor          = backgroundGrey
self.chart.barLabels.fontName              = 'Helvetica'
self.chart.valueAxis.labels.fontName       = 'Helvetica'
self.chart.valueAxis.labels.fontSize       = 7
self.chart.valueAxis.forceZero             = 1
self.chart.data              = [(100, 150, 180), (125, 180, 200)]
self.chart.groupSpacing      = 15
self.chart.valueAxis.avoidBoundFrac        = 1
self.chart.valueAxis.gridEnd               = 115
self.chart.valueAxis.tickLeft              = 3
self.chart.valueAxis.visibleGrid           = 1
self.chart.categoryAxis.categoryNames      = ['North', 'South', 'Central']
self.chart.categoryAxis.tickDown           = 3
self.chart.categoryAxis.labels.fontName    = 'Helvetica'
self.chart.categoryAxis.labels.fontSize    = 7
self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
self.Title.fontName   = 'Helvetica-Bold'
self.Title.fontSize   = 7
self.Title.x          = 100
self.Title.y          = 135
self.Title._text      = 'Chart Title'
self.Title.maxWidth  = 180
self.Title.height    = 20
self.Title.textAnchor ='middle'
self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
self.Legend.fontName      = 'Helvetica'
self.Legend.fontSize      = 7
self.Legend.x             = 153
self.Legend.y             = 85
self.Legend.dxTextSpace   = 5
self.Legend.dy            = 5
self.Legend.dx            = 5
self.Legend.deltay        = 5
self.Legend.alignment     ='right'
self._add(self,Label(),name='XLabel',validate=None,desc="The label on the horizontal axis")
self.XLabel.fontName      = 'Helvetica'
self.XLabel.fontSize      = 7
self.XLabel.x             = 85
self.XLabel.y             = 10
self.XLabel.textAnchor    ='middle'
self.XLabel.maxWidth      = 100
self.XLabel.height        = 20
self.XLabel._text          = "X Axis"
self._add(self,Label(),name='YLabel',validate=None,desc="The label on the vertical axis")
self.YLabel.fontName      = 'Helvetica'
self.YLabel.fontSize      = 7
self.YLabel.x             = 12
self.YLabel.y             = 80
self.YLabel.angle         = 90
self.YLabel.textAnchor    ='middle'
self.YLabel.maxWidth      = 100
self.YLabel.height        = 20
self.YLabel._text          = "Y Axis"
self._add(self,0,name='preview',validate=None,desc=None)
```

**Chart Title**



## exploded_pie

#Autogenerated by ReportLab guiedit do not edit
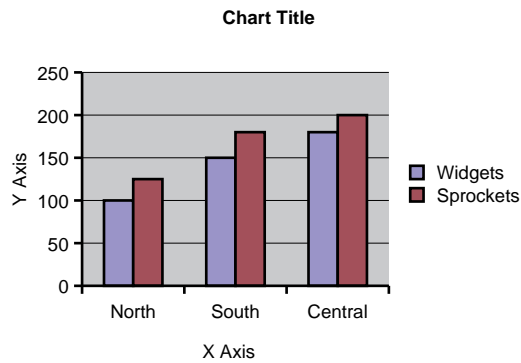
## Classes

## ExplodedPie(_DrawingEditorMixin, Drawing)

*Example*

```
def __init__(self,width=200,height=150,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    self._add(self,Pie(),name='chart',validate=None,desc="The main chart")
    self.chart.width        = 100
    self.chart.height       = 100
    self.chart.x            = 25
    self.chart.y            = 25
    self.chart.slices[0].fillColor = color01
    self.chart.slices[1].fillColor = color02
    self.chart.slices[2].fillColor = color03
    self.chart.slices[3].fillColor = color04
    self.chart.slices[4].fillColor = color05
    self.chart.slices[5].fillColor = color06
    self.chart.slices[6].fillColor = color07
    self.chart.slices[7].fillColor = color08
    self.chart.slices[8].fillColor = color09
    self.chart.slices[9].fillColor = color10
    self.chart.data                = (100, 150, 180)
    self.chart.startAngle          = -90
    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
    self.Title.fontName    = 'Helvetica-Bold'
    self.Title.fontSize    = 7
    self.Title.x           = 100
    self.Title.y           = 135
    self.Title._text       = 'Chart Title'
    self.Title.maxWidth    = 180
    self.Title.height      = 20
    self.Title.textAnchor  ='middle'
    self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
    self.Legend.colorNamePairs = [(color01, 'North'), (color02, 'South'), (color03, 'Central')]
    self.Legend.fontName       = 'Helvetica'
    self.Legend.fontSize       = 7
    self.Legend.x              = 160
    self.Legend.y              = 85
    self.Legend.dxTextSpace    = 5
    self.Legend.dy             = 5
    self.Legend.dx             = 5
    self.Legend.deltay         = 5
    self.Legend.alignment      ='right'
    self.Legend.columnMaximum  = 10
    self.chart.slices.strokeWidth  = 1
    self.chart.slices.fontName     = 'Helvetica'
    self.background                = ShadedRect()
```

```
                self.background.fillColorStart = backgroundGrey
                self.background.fillColorEnd   = backgroundGrey
                self.background.numShades       = 1
                self.background.strokeWidth     = 0.5
                self.background.x               = 20
                self.background.y               = 20
                self.chart.slices.popout        = 5
                self.background.height          = 110
                self.background.width           = 110
                self._add(self,0,name='preview',validate=None,desc=None)
```

**Chart Title**



```
North
South
Central
```

## filled_radar

#Autogenerated by ReportLab guiedit do not edit

## Classes

## FilledRadarChart(_DrawingEditorMixin, Drawing)

*Example*

```
        def __init__(self,width=200,height=150,*args,**kw):
            apply(Drawing.__init__,(self,width,height)+args,kw)
            self._add(self,SpiderChart(),name='chart',validate=None,desc="The main chart")
            self.chart.width        = 90
            self.chart.height       = 90
            self.chart.x            = 45
            self.chart.y            = 25
            self.chart.strands[0].fillColor  = color01
            self.chart.strands[1].fillColor  = color02
            self.chart.strands[2].fillColor  = color03
            self.chart.strands[3].fillColor  = color04
            self.chart.strands[4].fillColor  = color05
            self.chart.strands[5].fillColor  = color06
            self.chart.strands[6].fillColor  = color07
            self.chart.strands[7].fillColor  = color08
            self.chart.strands[8].fillColor  = color09
            self.chart.strands[9].fillColor  = color10
            self.chart.strandLabels.fontName = 'Helvetica'
            self.chart.strandLabels.fontSize = 6
            self.chart.fillColor             = backgroundGrey
            self.chart.data                  = [(125, 180, 200), (100, 150, 180)]
            self.chart.labels                = ['North', 'South', 'Central']
            self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
            self.Title.fontName   = 'Helvetica-Bold'
            self.Title.fontSize    = 7
            self.Title.x           = 100
            self.Title.y           = 135
            self.Title._text       = 'Chart Title'
            self.Title.maxWidth    = 180
            self.Title.height      = 20
```

```
                    self.Title.textAnchor ='middle'
                    self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
                    self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
                    self.Legend.fontName       = 'Helvetica'
                    self.Legend.fontSize       = 7
                    self.Legend.x              = 153
                    self.Legend.y              = 85
                    self.Legend.dxTextSpace    = 5
                    self.Legend.dy             = 5
                    self.Legend.dx             = 5
                    self.Legend.deltay         = 5
                    self.Legend.alignment      ='right'
                    self._add(self,0,name='preview',validate=None,desc=None)
```



## line_chart

#Autogenerated by ReportLab guiedit do not edit

### Classes

### LineChart(_DrawingEditorMixin, Drawing)

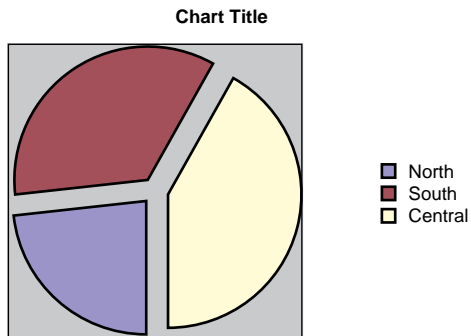*Example*

```
          def __init__(self,width=200,height=150,*args,**kw):
              apply(Drawing.__init__,(self,width,height)+args,kw)
              self._add(self,LinePlot(),name='chart',validate=None,desc="The main chart")
              self.chart.width      = 115
              self.chart.height     = 80
              self.chart.x          = 30
              self.chart.y          = 40
              self.chart.lines[0].strokeColor = color01
              self.chart.lines[1].strokeColor = color02
              self.chart.lines[2].strokeColor = color03
              self.chart.lines[3].strokeColor = color04
              self.chart.lines[4].strokeColor = color05
              self.chart.lines[5].strokeColor = color06
              self.chart.lines[6].strokeColor = color07
              self.chart.lines[7].strokeColor = color08
              self.chart.lines[8].strokeColor = color09
              self.chart.lines[9].strokeColor = color10
              self.chart.fillColor            = backgroundGrey
              self.chart.lineLabels.fontName            = 'Helvetica'
              self.chart.xValueAxis.labels.fontName     = 'Helvetica'
              self.chart.xValueAxis.labels.fontSize     = 7
              self.chart.xValueAxis.forceZero           = 0
              self.chart.data              = [((0, 50), (100,100), (200,200), (250,210), (300,300), (400,500
              self.chart.xValueAxis.avoidBoundFrac      = 1
              self.chart.xValueAxis.gridEnd             = 115
              self.chart.xValueAxis.tickDown            = 3
              self.chart.xValueAxis.visibleGrid         = 1
```

```
                    self.chart.yValueAxis.tickLeft              = 3
                    self.chart.yValueAxis.labels.fontName       = 'Helvetica'
                    self.chart.yValueAxis.labels.fontSize       = 7
                    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
                    self.Title.fontName   = 'Helvetica-Bold'
                    self.Title.fontSize   = 7
                    self.Title.x          = 100
                    self.Title.y          = 135
                    self.Title._text      = 'Chart Title'
                    self.Title.maxWidth   = 180
                    self.Title.height     = 20
                    self.Title.textAnchor ='middle'
                    self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
                    self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
                    self.Legend.fontName       = 'Helvetica'
                    self.Legend.fontSize        = 7
                    self.Legend.x               = 153
                    self.Legend.y               = 85
                    self.Legend.dxTextSpace     = 5
                    self.Legend.dy              = 5
                    self.Legend.dx              = 5
                    self.Legend.deltay          = 5
                    self.Legend.alignment       ='right'
                    self._add(self,Label(),name='XLabel',validate=None,desc="The label on the horizontal axis")
                    self.XLabel.fontName        = 'Helvetica'
                    self.XLabel.fontSize        = 7
                    self.XLabel.x               = 85
                    self.XLabel.y               = 10
                    self.XLabel.textAnchor      ='middle'
                    self.XLabel.maxWidth        = 100
                    self.XLabel.height          = 20
                    self.XLabel._text           = "X Axis"
                    self._add(self,Label(),name='YLabel',validate=None,desc="The label on the vertical axis")
                    self.YLabel.fontName        = 'Helvetica'
                    self.YLabel.fontSize        = 7
                    self.YLabel.x               = 12
                    self.YLabel.y               = 80
                    self.YLabel.angle           = 90
                    self.YLabel.textAnchor      ='middle'
                    self.YLabel.maxWidth        = 100
                    self.YLabel.height          = 20
                    self.YLabel._text           = "Y Axis"
                    self.chart.yValueAxis.forceZero           = 1
                    self.chart.xValueAxis.forceZero           = 1
                    self._add(self,0,name='preview',validate=None,desc=None)
```



## `linechart_with_markers`

#Autogenerated by ReportLab guiedit do not edit

## Classes

---

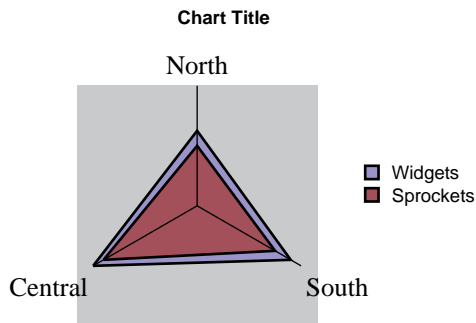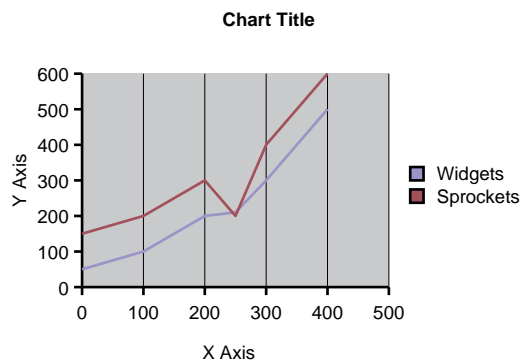## LineChartWithMarkers(_DrawingEditorMixin, Drawing)

*Example*

```
def __init__(self,width=200,height=150,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    self._add(self,LinePlot(),name='chart',validate=None,desc="The main chart")
    self.chart.width       = 115
    self.chart.height      = 80
    self.chart.x           = 30
    self.chart.y           = 40
    self.chart.lines[0].strokeColor = color01
    self.chart.lines[1].strokeColor = color02
    self.chart.lines[2].strokeColor = color03
    self.chart.lines[3].strokeColor = color04
    self.chart.lines[4].strokeColor = color05
    self.chart.lines[5].strokeColor = color06
    self.chart.lines[6].strokeColor = color07
    self.chart.lines[7].strokeColor = color08
    self.chart.lines[8].strokeColor = color09
    self.chart.lines[9].strokeColor = color10
    self.chart.lines[0].symbol = makeMarker('FilledSquare')
    self.chart.lines[1].symbol = makeMarker('FilledDiamond')
    self.chart.lines[2].symbol = makeMarker('FilledStarFive')
    self.chart.lines[3].symbol = makeMarker('FilledTriangle')
    self.chart.lines[4].symbol = makeMarker('FilledCircle')
    self.chart.lines[5].symbol = makeMarker('FilledPentagon')
    self.chart.lines[6].symbol = makeMarker('FilledStarSix')
    self.chart.lines[7].symbol = makeMarker('FilledHeptagon')
    self.chart.lines[8].symbol = makeMarker('FilledOctagon')
    self.chart.lines[9].symbol = makeMarker('FilledCross')
    self.chart.fillColor        = backgroundGrey
    self.chart.lineLabels.fontName              = 'Helvetica'
    self.chart.xValueAxis.labels.fontName       = 'Helvetica'
    self.chart.xValueAxis.labels.fontSize       = 7
    self.chart.xValueAxis.forceZero             = 0
    self.chart.data                 = [((0, 50), (100,100), (200,200), (250,210), (300,300), (400,500
    self.chart.xValueAxis.avoidBoundFrac        = 1
    self.chart.xValueAxis.gridEnd               = 115
    self.chart.xValueAxis.tickDown              = 3
    self.chart.xValueAxis.visibleGrid           = 1
    self.chart.yValueAxis.tickLeft              = 3
    self.chart.yValueAxis.labels.fontName       = 'Helvetica'
    self.chart.yValueAxis.labels.fontSize       = 7
    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
    self.Title.fontName   = 'Helvetica-Bold'
    self.Title.fontSize   = 7
    self.Title.x          = 100
    self.Title.y          = 135
    self.Title._text      = 'Chart Title'
    self.Title.maxWidth   = 180
    self.Title.height     = 20
    self.Title.textAnchor ='middle'
    self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
    self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
    self.Legend.fontName      = 'Helvetica'
    self.Legend.fontSize      = 7
    self.Legend.x             = 153
    self.Legend.y             = 85
    self.Legend.dxTextSpace   = 5
    self.Legend.dy            = 5
    self.Legend.dx            = 5
    self.Legend.deltay        = 5
    self.Legend.alignment     ='right'
    self._add(self,Label(),name='XLabel',validate=None,desc="The label on the horizontal axis")
    self.XLabel.fontName      = 'Helvetica'
    self.XLabel.fontSize      = 7
    self.XLabel.x             = 85
    self.XLabel.y             = 10
    self.XLabel.textAnchor    ='middle'
    self.XLabel.maxWidth      = 100
    self.XLabel.height        = 20
    self.XLabel._text         = "X Axis"
    self._add(self,Label(),name='YLabel',validate=None,desc="The label on the vertical axis")
```

```
            self.YLabel.fontName      = 'Helvetica'
            self.YLabel.fontSize      = 7
            self.YLabel.x             = 12
            self.YLabel.y             = 80
            self.YLabel.angle         = 90
            self.YLabel.textAnchor    ='middle'
            self.YLabel.maxWidth      = 100
            self.YLabel.height        = 20
            self.YLabel._text         = "Y Axis"
            self.chart.yValueAxis.forceZero        = 1
            self.chart.xValueAxis.forceZero        = 1
            self._add(self,0,name='preview',validate=None,desc=None)
```



## **radar**

#Autogenerated by ReportLab guiedit do not edit

## Classes

## **RadarChart(_DrawingEditorMixin, Drawing)**
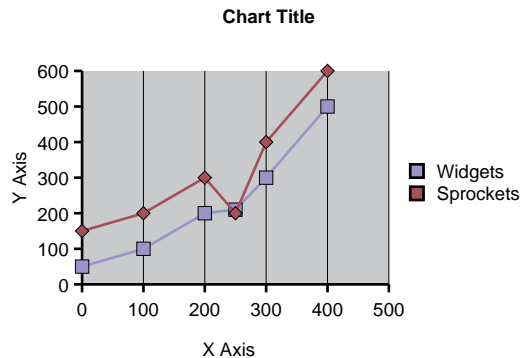
*Example*

```
        def __init__(self,width=200,height=150,*args,**kw):
            apply(Drawing.__init__,(self,width,height)+args,kw)
            self._add(self,SpiderChart(),name='chart',validate=None,desc="The main chart")
            self.chart.width      = 90
            self.chart.height     = 90
            self.chart.x          = 45
            self.chart.y          = 25
            self.chart.strands[0].strokeColor= color01
            self.chart.strands[1].strokeColor= color02
            self.chart.strands[2].strokeColor= color03
            self.chart.strands[3].strokeColor= color04
            self.chart.strands[4].strokeColor= color05
            self.chart.strands[5].strokeColor= color06
            self.chart.strands[6].strokeColor= color07
            self.chart.strands[7].strokeColor= color08
            self.chart.strands[8].strokeColor= color09
            self.chart.strands[9].strokeColor= color10
            self.chart.strands[0].fillColor  = None
            self.chart.strands[1].fillColor  = None
            self.chart.strands[2].fillColor  = None
            self.chart.strands[3].fillColor  = None
            self.chart.strands[4].fillColor  = None
            self.chart.strands[5].fillColor  = None
            self.chart.strands[6].fillColor  = None
            self.chart.strands[7].fillColor  = None
            self.chart.strands[8].fillColor  = None
            self.chart.strands[9].fillColor  = None
            self.chart.strands.strokeWidth   = 1
```

```
            self.chart.strandLabels.fontName = 'Helvetica'
            self.chart.strandLabels.fontSize = 6
            self.chart.fillColor              = backgroundGrey
            self.chart.data                   = [(125, 180, 200), (100, 150, 180)]
            self.chart.labels                 = ['North', 'South', 'Central']
            self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
            self.Title.fontName  = 'Helvetica-Bold'
            self.Title.fontSize  = 7
            self.Title.x         = 100
            self.Title.y         = 135
            self.Title._text     = 'Chart Title'
            self.Title.maxWidth  = 180
            self.Title.height    = 20
            self.Title.textAnchor ='middle'
            self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
            self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
            self.Legend.fontName       = 'Helvetica'
            self.Legend.fontSize       = 7
            self.Legend.x              = 153
            self.Legend.y              = 85
            self.Legend.dxTextSpace    = 5
            self.Legend.dy             = 5
            self.Legend.dx             = 5
            self.Legend.deltay         = 5
            self.Legend.alignment      ='right'
            self.chart.strands.strokeWidth     = 1
            self._add(self,0,name='preview',validate=None,desc=None)
```
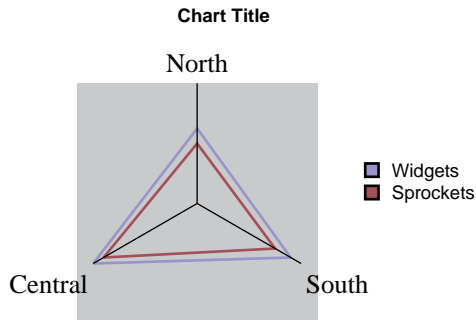


## scatter

#Autogenerated by ReportLab guiedit do not edit

## Classes

## Scatter(_DrawingEditorMixin, Drawing)

*Example*
```
        def __init__(self,width=200,height=150,*args,**kw):
            apply(Drawing.__init__,(self,width,height)+args,kw)
            self._add(self,ScatterPlot(),name='chart',validate=None,desc="The main chart")
            self.chart.width       = 115
            self.chart.height      = 80
            self.chart.x           = 30
            self.chart.y           = 40
            self.chart.lines[0].strokeColor = color01
            self.chart.lines[1].strokeColor = color02
            self.chart.lines[2].strokeColor = color03
            self.chart.lines[3].strokeColor = color04
            self.chart.lines[4].strokeColor = color05
            self.chart.lines[5].strokeColor = color06
```

```
                        self.chart.lines[6].strokeColor = color07
                        self.chart.lines[7].strokeColor = color08
                        self.chart.lines[8].strokeColor = color09
                        self.chart.lines[9].strokeColor = color10
                        self.chart.fillColor       = backgroundGrey
                        self.chart.lineLabels.fontName          = 'Helvetica'
                        self.chart.xValueAxis.labels.fontName   = 'Helvetica'
                        self.chart.xValueAxis.labels.fontSize   = 7
                        self.chart.xValueAxis.forceZero         = 0
                        self.chart.data               = [((100,100), (200,200), (250,210), (300,300), (400,500)), ((100
                        self.chart.xValueAxis.avoidBoundFrac      = 1
                        self.chart.xValueAxis.gridEnd             = 115
                        self.chart.xValueAxis.tickDown            = 3
                        self.chart.xValueAxis.visibleGrid         = 1
                        self.chart.yValueAxis.tickLeft         = 3
                        self.chart.yValueAxis.labels.fontName     = 'Helvetica'
                        self.chart.yValueAxis.labels.fontSize     = 7
                        self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
                        self.Title.fontName   = 'Helvetica-Bold'
                        self.Title.fontSize   = 7
                        self.Title.x          = 100
                        self.Title.y          = 135
                        self.Title._text      = 'Chart Title'
                        self.Title.maxWidth   = 180
                        self.Title.height     = 20
                        self.Title.textAnchor ='middle'
                        self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
                        self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
                        self.Legend.fontName      = 'Helvetica'
                        self.Legend.fontSize      = 7
                        self.Legend.x             = 153
                        self.Legend.y             = 85
                        self.Legend.dxTextSpace   = 5
                        self.Legend.dy            = 5
                        self.Legend.dx            = 5
                        self.Legend.deltay        = 5
                        self.Legend.alignment     ='right'
                        self.chart.lineLabelFormat  = None
                        self.chart.xLabel         = 'X Axis'
                        self.chart.y              = 30
                        self.chart.yLabel         = 'Y Axis'
                        self.chart.yValueAxis.labelTextFormat    = '%d'
                        self.chart.yValueAxis.forceZero          = 1
                        self.chart.xValueAxis.forceZero          = 1


                        self._add(self,0,name='preview',validate=None,desc=None)
```
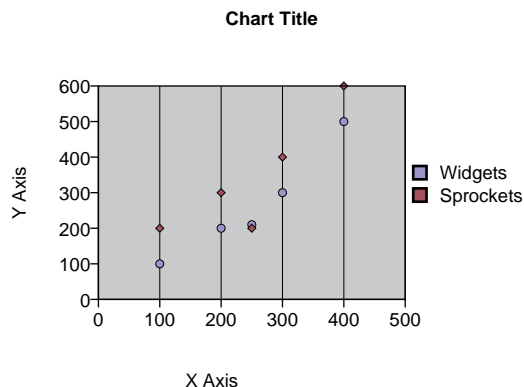


## scatter_lines
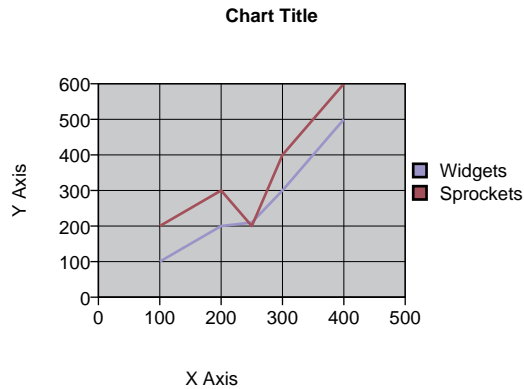
#Autogenerated by ReportLab guiedit do not edit

### Classes

## ScatterLines(_DrawingEditorMixin, Drawing)

*Example*

```
def __init__(self,width=200,height=150,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    self._add(self,ScatterPlot(),name='chart',validate=None,desc="The main chart")
    self.chart.width      = 115
    self.chart.height     = 80
    self.chart.x          = 30
    self.chart.y          = 40
    self.chart.lines[0].strokeColor = color01
    self.chart.lines[1].strokeColor = color02
    self.chart.lines[2].strokeColor = color03
    self.chart.lines[3].strokeColor = color04
    self.chart.lines[4].strokeColor = color05
    self.chart.lines[5].strokeColor = color06
    self.chart.lines[6].strokeColor = color07
    self.chart.lines[7].strokeColor = color08
    self.chart.lines[8].strokeColor = color09
    self.chart.lines[9].strokeColor = color10
    self.chart.lines[0].symbol = None
    self.chart.lines[1].symbol = None
    self.chart.lines[2].symbol = None
    self.chart.lines[3].symbol = None
    self.chart.lines[4].symbol = None
    self.chart.lines[5].symbol = None
    self.chart.lines[6].symbol = None
    self.chart.lines[7].symbol = None
    self.chart.lines[8].symbol = None
    self.chart.lines[9].symbol = None
    self.chart.fillColor         = backgroundGrey
    self.chart.lineLabels.fontName              = 'Helvetica'
    self.chart.xValueAxis.labels.fontName       = 'Helvetica'
    self.chart.xValueAxis.labels.fontSize       = 7
    self.chart.xValueAxis.forceZero             = 0
    self.chart.data              = [((100,100), (200,200), (250,210), (300,300), (400,500)), ((100
    self.chart.xValueAxis.avoidBoundFrac        = 1
    self.chart.xValueAxis.gridEnd               = 115
    self.chart.xValueAxis.tickDown              = 3
    self.chart.xValueAxis.visibleGrid           = 1
    self.chart.yValueAxis.tickLeft              = 3
    self.chart.yValueAxis.labels.fontName       = 'Helvetica'
    self.chart.yValueAxis.labels.fontSize       = 7
    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
    self.Title.fontName   = 'Helvetica-Bold'
    self.Title.fontSize   = 7
    self.Title.x          = 100
    self.Title.y          = 135
    self.Title._text      = 'Chart Title'
    self.Title.maxWidth   = 180
    self.Title.height     = 20
    self.Title.textAnchor ='middle'
    self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
    self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
    self.Legend.fontName        = 'Helvetica'
    self.Legend.fontSize        = 7
    self.Legend.x               = 153
    self.Legend.y               = 85
    self.Legend.dxTextSpace     = 5
    self.Legend.dy              = 5
    self.Legend.dx              = 5
    self.Legend.deltay          = 5
    self.Legend.alignment       ='right'
    self.chart.lineLabelFormat   = None
    self.chart.xLabel            = 'X Axis'
    self.chart.y                 = 30
    self.chart.yLabel            = 'Y Axis'
    self.chart.yValueAxis.gridEnd              = 115
    self.chart.yValueAxis.visibleGrid          = 1
    self.chart.yValueAxis.labelTextFormat      = '%d'
    self.chart.yValueAxis.forceZero            = 1
    self.chart.xValueAxis.forceZero            = 1
    self.chart.joinedLines                     = 1
    self._add(self,0,name='preview',validate=None,desc=None)
```

## scatter_lines_markers

#Autogenerated by ReportLab guiedit do not edit

## Classes

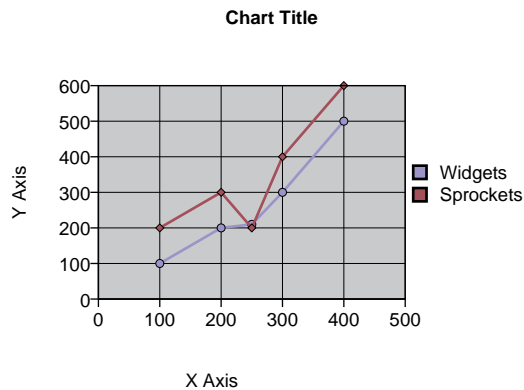## ScatterLinesMarkers(_DrawingEditorMixin, Drawing)

*Example*

```
def __init__(self,width=200,height=150,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    self._add(self,ScatterPlot(),name='chart',validate=None,desc="The main chart")
    self.chart.width       = 115
    self.chart.height      = 80
    self.chart.x           = 30
    self.chart.y           = 40
    self.chart.lines[0].strokeColor = color01
    self.chart.lines[1].strokeColor = color02
    self.chart.lines[2].strokeColor = color03
    self.chart.lines[3].strokeColor = color04
    self.chart.lines[4].strokeColor = color05
    self.chart.lines[5].strokeColor = color06
    self.chart.lines[6].strokeColor = color07
    self.chart.lines[7].strokeColor = color08
    self.chart.lines[8].strokeColor = color09
    self.chart.lines[9].strokeColor = color10
    self.chart.fillColor         = backgroundGrey
    self.chart.lineLabels.fontName              = 'Helvetica'
    self.chart.xValueAxis.labels.fontName       = 'Helvetica'
    self.chart.xValueAxis.labels.fontSize       = 7
    self.chart.xValueAxis.forceZero             = 0
    self.chart.data          = [((100,100), (200,200), (250,210), (300,300), (400,500)), ((100
    self.chart.xValueAxis.avoidBoundFrac        = 1
    self.chart.xValueAxis.gridEnd               = 115
    self.chart.xValueAxis.tickDown              = 3
    self.chart.xValueAxis.visibleGrid           = 1
    self.chart.yValueAxis.tickLeft              = 3
    self.chart.yValueAxis.labels.fontName       = 'Helvetica'
    self.chart.yValueAxis.labels.fontSize       = 7
    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
    self.Title.fontName   = 'Helvetica-Bold'
    self.Title.fontSize   = 7
    self.Title.x          = 100
    self.Title.y          = 135
    self.Title._text      = 'Chart Title'
    self.Title.maxWidth   = 180
    self.Title.height     = 20
    self.Title.textAnchor ='middle'
    self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
    self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
    self.Legend.fontName       = 'Helvetica'
```

```
self.Legend.fontSize       = 7
self.Legend.x              = 153
self.Legend.y              = 85
self.Legend.dxTextSpace    = 5
self.Legend.dy             = 5
self.Legend.dx             = 5
self.Legend.deltay         = 5
self.Legend.alignment      ='right'
self.chart.lineLabelFormat  = None
self.chart.xLabel           = 'X Axis'
self.chart.y                = 30
self.chart.yLabel           = 'Y Axis'
self.chart.yValueAxis.gridEnd         = 115
self.chart.yValueAxis.visibleGrid     = 1
self.chart.yValueAxis.labelTextFormat = '%d'
self.chart.yValueAxis.forceZero       = 1
self.chart.xValueAxis.forceZero       = 1
self.chart.joinedLines                = 1
self._add(self,0,name='preview',validate=None,desc=None)
```



## simple_pie

#Autogenerated by ReportLab guiedit do not edit

## Classes

## SimplePie(_DrawingEditorMixin, Drawing)

*Example*

```
def __init__(self,width=200,height=150,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    self._add(self,Pie(),name='chart',validate=None,desc="The main chart")
    self.chart.width      = 100
    self.chart.height     = 100
    self.chart.x          = 25
    self.chart.y          = 25
    self.chart.slices[0].fillColor = color01
    self.chart.slices[1].fillColor = color02
    self.chart.slices[2].fillColor = color03
    self.chart.slices[3].fillColor = color04
    self.chart.slices[4].fillColor = color05
    self.chart.slices[5].fillColor = color06
    self.chart.slices[6].fillColor = color07
    self.chart.slices[7].fillColor = color08
    self.chart.slices[8].fillColor = color09
    self.chart.slices[9].fillColor = color10
    self.chart.data              = (100, 150, 180)
    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
    self.Title.fontName   = 'Helvetica-Bold'
    self.Title.fontSize   = 7
```

```
            self.Title.x          = 100
            self.Title.y          = 135
            self.Title._text      = 'Chart Title'
            self.Title.maxWidth   = 180
            self.Title.height     = 20
            self.Title.textAnchor ='middle'
            self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
            self.Legend.colorNamePairs = [(color01, 'North'), (color02, 'South'),(color03, 'Central')]
            self.Legend.fontName       = 'Helvetica'
            self.Legend.fontSize       = 7
            self.Legend.x              = 160
            self.Legend.y              = 85
            self.Legend.dxTextSpace    = 5
            self.Legend.dy             = 5
            self.Legend.dx             = 5
            self.Legend.deltay         = 5
            self.Legend.alignment      ='right'
            self.chart.slices.strokeWidth  = 1
            self.chart.slices.fontName     = 'Helvetica'
            self.background                = ShadedRect()
            self.background.fillColorStart = backgroundGrey
            self.background.fillColorEnd   = backgroundGrey
            self.background.numShades      = 1
            self.background.strokeWidth    = 0.5
            self.background.x              = 25
            self.background.y              = 25
            self.Legend.columnMaximum  = 10
            self._add(self,0,name='preview',validate=None,desc=None)
```



## stacked_bar

#Autogenerated by ReportLab guiedit do not edit

## Classes

## StackedBar(_DrawingEditorMixin, Drawing)

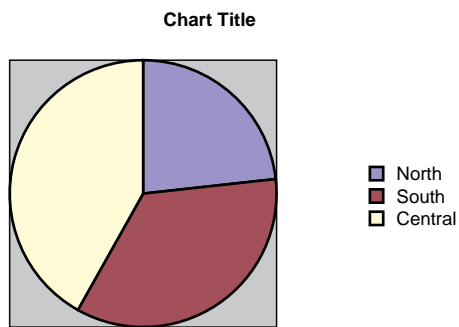*Example*
```
        def __init__(self,width=200,height=150,*args,**kw):
            apply(Drawing.__init__,(self,width,height)+args,kw)
            self._add(self,HorizontalBarChart(),name='chart',validate=None,desc="The main chart")
            self.chart.width    = 115
            self.chart.height   = 80
            self.chart.x        = 30
            self.chart.y        = 40
            self.chart.bars[0].fillColor = color01
            self.chart.bars[1].fillColor = color02
            self.chart.bars[2].fillColor = color03
            self.chart.bars[3].fillColor = color04
            self.chart.bars[4].fillColor = color05
```

```
                    self.chart.bars[5].fillColor = color06
                    self.chart.bars[6].fillColor = color07
                    self.chart.bars[7].fillColor = color08
                    self.chart.bars[8].fillColor = color09
                    self.chart.bars[9].fillColor = color10
                    self.chart.fillColor         = backgroundGrey
                    self.chart.barLabels.fontName              = 'Helvetica'
                    self.chart.valueAxis.labels.fontName       = 'Helvetica'
                    self.chart.valueAxis.labels.fontSize       = 6
                    self.chart.valueAxis.forceZero             = 1
                    self.chart.data              = [(100, 150, 180), (125, 180, 200)]
                    self.chart.groupSpacing      = 15
                    self.chart.valueAxis.avoidBoundFrac        = 1
                    self.chart.valueAxis.gridEnd               = 80
                    self.chart.valueAxis.tickDown              = 3
                    self.chart.valueAxis.visibleGrid           = 1
                    self.chart.categoryAxis.categoryNames      = ['North', 'South', 'Central']
                    self.chart.categoryAxis.tickLeft           = 3
                    self.chart.categoryAxis.labels.fontName    = 'Helvetica'
                    self.chart.categoryAxis.labels.fontSize    = 6
                    self.chart.categoryAxis.labels.dx          = -3
                    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
                    self.Title.fontName   = 'Helvetica-Bold'
                    self.Title.fontSize   = 7
                    self.Title.x          = 100
                    self.Title.y          = 135
                    self.Title._text      = 'Chart Title'
                    self.Title.maxWidth   = 180
                    self.Title.height     = 20
                    self.Title.textAnchor ='middle'
                    self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
                    self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
                    self.Legend.fontName       = 'Helvetica'
                    self.Legend.fontSize       = 7
                    self.Legend.x              = 153
                    self.Legend.y              = 85
                    self.Legend.dxTextSpace    = 5
                    self.Legend.dy             = 5
                    self.Legend.dx             = 5
                    self.Legend.deltay         = 5
                    self.Legend.alignment      ='right'
                    self._add(self,Label(),name='XLabel',validate=None,desc="The label on the horizontal axis")
                    self.XLabel.fontName       = 'Helvetica'
                    self.XLabel.fontSize       = 7
                    self.XLabel.x              = 85
                    self.XLabel.y              = 10
                    self.XLabel.textAnchor     ='middle'
                    self.XLabel.maxWidth       = 100
                    self.XLabel.height         = 20
                    self.XLabel._text          = "X Axis"
                    self._add(self,Label(),name='YLabel',validate=None,desc="The label on the vertical axis")
                    self.YLabel.fontName       = 'Helvetica'
                    self.YLabel.fontSize       = 7
                    self.YLabel.x              = 12
                    self.YLabel.y              = 80
                    self.YLabel.angle          = 90
                    self.YLabel.textAnchor     ='middle'
                    self.YLabel.maxWidth       = 100
                    self.YLabel.height         = 20
                    self.YLabel._text          = "Y Axis"
                    self.chart.categoryAxis.style='stacked'
                    self._add(self,0,name='preview',validate=None,desc=None)
```

## stacked_column

#Autogenerated by ReportLab guiedit do not edit

## Classes
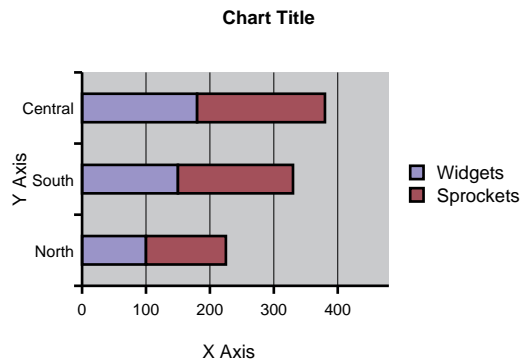
## StackedColumn(_DrawingEditorMixin, Drawing)
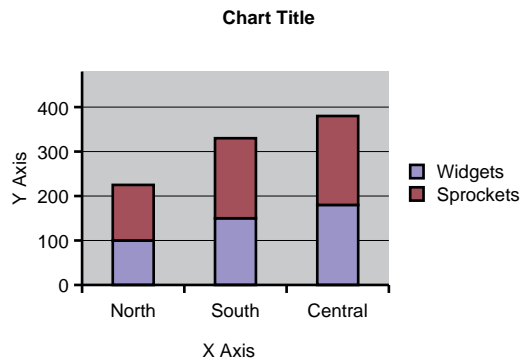
*Example*

```
def __init__(self,width=200,height=150,*args,**kw):
    apply(Drawing.__init__,(self,width,height)+args,kw)
    self._add(self,VerticalBarChart(),name='chart',validate=None,desc="The main chart")
    self.chart.width      = 115
    self.chart.height     = 80
    self.chart.x          = 30
    self.chart.y          = 40
    self.chart.bars[0].fillColor = color01
    self.chart.bars[1].fillColor = color02
    self.chart.bars[2].fillColor = color03
    self.chart.bars[3].fillColor = color04
    self.chart.bars[4].fillColor = color05
    self.chart.bars[5].fillColor = color06
    self.chart.bars[6].fillColor = color07
    self.chart.bars[7].fillColor = color08
    self.chart.bars[8].fillColor = color09
    self.chart.bars[9].fillColor = color10
    self.chart.fillColor          = backgroundGrey
    self.chart.barLabels.fontName             = 'Helvetica'
    self.chart.valueAxis.labels.fontName      = 'Helvetica'
    self.chart.valueAxis.labels.fontSize      = 7
    self.chart.valueAxis.forceZero            = 1
    self.chart.data               = [(100, 150, 180), (125, 180, 200)]
    self.chart.groupSpacing       = 15
    self.chart.valueAxis.avoidBoundFrac       = 1
    self.chart.valueAxis.gridEnd              = 115
    self.chart.valueAxis.tickLeft             = 3
    self.chart.valueAxis.visibleGrid          = 1
    self.chart.categoryAxis.categoryNames     = ['North', 'South', 'Central']
    self.chart.categoryAxis.tickDown          = 3
    self.chart.categoryAxis.labels.fontName   = 'Helvetica'
    self.chart.categoryAxis.labels.fontSize   = 7
    self._add(self,Label(),name='Title',validate=None,desc="The title at the top of the chart")
    self.Title.fontName   = 'Helvetica-Bold'
    self.Title.fontSize   = 7
    self.Title.x          = 100
    self.Title.y          = 135
    self.Title._text      = 'Chart Title'
    self.Title.maxWidth   = 180
    self.Title.height     = 20
    self.Title.textAnchor ='middle'
    self._add(self,Legend(),name='Legend',validate=None,desc="The legend or key for the chart")
    self.Legend.colorNamePairs = [(color01, 'Widgets'), (color02, 'Sprockets')]
```

```
self.Legend.fontName       = 'Helvetica'
self.Legend.fontSize       = 7
self.Legend.x              = 153
self.Legend.y              = 85
self.Legend.dxTextSpace    = 5
self.Legend.dy             = 5
self.Legend.dx             = 5
self.Legend.deltay         = 5
self.Legend.alignment      ='right'
self._add(self,Label(),name='XLabel',validate=None,desc="The label on the horizontal axis")
self.XLabel.fontName       = 'Helvetica'
self.XLabel.fontSize       = 7
self.XLabel.x              = 85
self.XLabel.y              = 10
self.XLabel.textAnchor     ='middle'
self.XLabel.maxWidth       = 100
self.XLabel.height         = 20
self.XLabel._text          = "X Axis"
self._add(self,Label(),name='YLabel',validate=None,desc="The label on the vertical axis")
self.YLabel.fontName       = 'Helvetica'
self.YLabel.fontSize       = 7
self.YLabel.x              = 12
self.YLabel.y              = 80
self.YLabel.angle          = 90
self.YLabel.textAnchor     ='middle'
self.YLabel.maxWidth       = 100
self.YLabel.height         = 20
self.YLabel._text          = "Y Axis"
self.chart.categoryAxis.style='stacked'
self._add(self,0,name='preview',validate=None,desc=None)
```



## eventcal

This file is a

## Classes

## EventCalendar(Widget)

## Public Attributes

*Example*
```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

```
data = []
day = 0
endTime = None
height = 150
startTime = None
timeColWidth = None
trackNames = None
trackRowHeight = 20
width = 300
x = 0
y = 0
```

## flags

This file is a collection of flag graphics as widgets.

All flags are represented at the ratio of 1:2, even where the official ratio for the flag is something else
(such as 3:5 for the German national flag). The only exceptions are for where this would look _very_ wrong,
such as the Danish flag whose (ratio is 28:37), or the Swiss flag (which is square).

Unless otherwise stated, these flags are all the 'national flags' of the countries, rather than their
state flags, naval flags, ensigns or any other variants. (National flags are the flag flown by civilians
of a country and the ones usually used to represent a country abroad. State flags are the variants used by
the government and by diplomatic missions overseas).

To check on how close these are to the 'official' representations of flags, check the World Flag Database at
http://www.flags.ndirect.co.uk/

The flags this file contains are:

EU Members:
United Kingdom, Austria, Belgium, Denmark, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg,
Holland (The Netherlands), Spain, Sweden

Others:
USA, Czech Republic, European Union, Switzerland, Turkey, Brazil

(Brazilian flag contributed by Publio da Costa Melo [publio@planetarium.com.br]).

## Classes

## Flag(_Symbol)

This is a generic flag class that all the flags in this file use as a basis.

This class basically provides edges and a tidy-up routine to hide any bits of
line that overlap the 'outside' of the flag

possible attributes:
'x', 'y', 'size', 'fillColor'

## Public Attributes

**border** Whether a background is drawn

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** Background color

**kind** Which flag

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## `Star(_Symbol)`

This draws a 5-pointed star.

possible attributes:
'x', 'y', 'size', 'fillColor', 'strokeColor'

## Public Attributes

**angle** angle in degrees

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## `grids`

#Copyright ReportLab Europe Ltd. 2000-2004
#see license.txt for license details
#history http://www.reportlab.co.uk/cgi-bin/viewcvs.cgi/public/reportlab/trunk/reportlab/graphics/widgets/grids.py

## Classes

## `DoubleGrid(Widget)`

This combines two ordinary Grid objects orthogonal to each other.

## Public Attributes

**grid0** The first grid component.

**grid1** The second grid component.

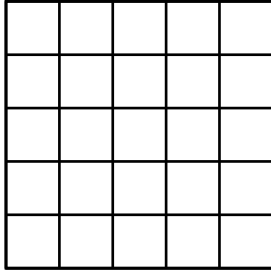**height** The grid's height.

**width** The grid's width.

**x** The grid's lower-left x position.

**y** The grid's lower-left y position.

*Example*
```
def demo(self):
    D = Drawing(100, 100)
    g = DoubleGrid()
    D.add(g)
    return D
```

*Properties of Example Widget*

```
grid0.delta = 20
grid0.delta0 = 0
grid0.deltaSteps = []
grid0.fillColor = Color(1,1,1)
grid0.height = 100
grid0.orientation = 'vertical'
grid0.stripeColors = [Color(1,0,0), Color(0,.501961,0), Color(0,0,1)]
grid0.strokeColor = Color(0,0,0)
grid0.strokeWidth = 1
grid0.useLines = 1
grid0.useRects = 0
grid0.width = 100
grid0.x = 0
grid0.y = 0
grid1.delta = 20
grid1.delta0 = 0
grid1.deltaSteps = []
grid1.fillColor = Color(1,1,1)
grid1.height = 100
grid1.orientation = 'horizontal'
grid1.stripeColors = [Color(1,0,0), Color(0,.501961,0), Color(0,0,1)]
grid1.strokeColor = Color(0,0,0)
grid1.strokeWidth = 1
grid1.useLines = 1
grid1.useRects = 0
grid1.width = 100
grid1.x = 0
grid1.y = 0
height = 100
width = 100
x = 0
y = 0
```

## `Grid(Widget)`

This makes a rectangular grid of equidistant stripes.

The grid contains an outer border rectangle, and stripes
inside which can be drawn with lines and/or as solid tiles.
The drawing order is: outer rectangle, then lines and tiles.

The stripes' width is indicated as 'delta'. The sequence of
stripes can have an offset named 'delta0'. Both values need
to be positive!

## Public Attributes

**delta** Determines the width/height of the stripes.

**delta0** Determines the stripes initial width/height offset.

**deltaSteps** List of deltas to be used cyclically.

**fillColor** Background color for entire rectangle.

**height** The grid's height.

**orientation** Determines if stripes are vertical or horizontal.

**rectStrokeColor** Color for outer rect stroke.

**rectStrokeWidth** Width for outer rect stroke.

**stripeColors** Colors applied cyclically in the right or upper direction.

**strokeColor** Color used for lines.

**strokeWidth** Width used for lines.

**useLines** Determines if stripes are drawn with lines.

**useRects** Determines if stripes are drawn with solid rectangles.

**width** The grid's width.

**x** The grid's lower-left x position.

**y** The grid's lower-left y position.

*Example*

```
def demo(self):
    D = Drawing(100, 100)

    g = Grid()
    D.add(g)

    return D
```

*Properties of Example Widget*

```
delta = 20
delta0 = 0
deltaSteps = []
fillColor = Color(1,1,1)
height = 100
orientation = 'vertical'
stripeColors = [Color(1,0,0), Color(0,.501961,0), Color(0,0,1)]
strokeColor = Color(0,0,0)
strokeWidth = 2
useLines = 0
```

```
useRects = 1
width = 100
x = 0
y = 0
```

## ShadedPolygon(Widget, LineShape)

## Public Attributes

**angle** Shading angle

**cylinderMode** True if shading reverses in middle.

**fillColorEnd** None

**fillColorStart** None

**numShades** The number of interpolating colors.

**points** None

**strokeColor** None

**strokeDashArray** None

**strokeLineCap** None

**strokeLineJoin** None

**strokeMiterLimit** None

**strokeWidth** None

*Example*

```
def demo(self):
    msg = "demo() must be implemented for each Widget!"
    raise shapes.NotImplementedError, msg
```

*Properties of Example Widget*

```
angle = 90
cylinderMode = 0
fillColorEnd = Color(0,.501961,0)
fillColorStart = Color(1,0,0)
numShades = 50
points = [-1, -1, 2, 2, 3, -1]
strokeColor = Color(0,0,0)
strokeDashArray = None
strokeLineCap = 0
strokeLineJoin = 0
strokeMiterLimit = 0
strokeWidth = 1
```

## ShadedRect(Widget)

This makes a rectangle with shaded colors between two colors.

Colors are interpolated linearly between 'fillColorStart'
and 'fillColorEnd', both of which appear at the margins.
If 'numShades' is set to one, though, only 'fillColorStart'
is used.

## Public Attributes

**cylinderMode** True if shading reverses in middle.

**fillColorEnd** End value of the color shade.

**fillColorStart** Start value of the color shade.

**height** The grid's height.

**numShades** The number of interpolating colors.

**orientation** Determines if stripes are vertical or horizontal.

**strokeColor** Color used for border line.

**strokeWidth** Width used for lines.

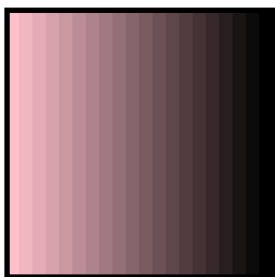**width** The grid's width.

**x** The grid's lower-left x position.

**y** The grid's lower-left y position.

*Example*

```
def demo(self):
    D = Drawing(100, 100)
    g = ShadedRect()
    D.add(g)

    return D
```



*Properties of Example Widget*

```
cylinderMode = 0
fillColorEnd = Color(0,0,0)
fillColorStart = Color(1,.752941,.796078)
height = 100
numShades = 20
orientation = 'vertical'
strokeColor = Color(0,0,0)
strokeWidth = 2
width = 100
x = 0
y = 0
```

## `signsandsymbols`

This file is a collection of widgets to produce some common signs and symbols.

Widgets include:
- ETriangle (an equilateral triangle),
- RTriangle (a right angled triangle),
- Octagon,
- Crossbox,
- Tickbox,
- SmileyFace,
- StopSign,
- NoEntry,
- NotAllowed (the red roundel from 'no smoking' signs),
- NoSmoking,
- DangerSign (a black exclamation point in a yellow triangle),
- YesNo (returns a tickbox or a crossbox depending on a testvalue),
- FloppyDisk,
- ArrowOne, and
- ArrowTwo

## Classes

### `ArrowOne(_Symbol)`

This widget draws an arrow (style one).

possible attributes:
'x', 'y', 'size', 'fillColor'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

### `ArrowTwo(ArrowOne)`

This widget draws an arrow (style two).

possible attributes:
'x', 'y', 'size', 'fillColor'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## `Crossbox(_Symbol)`

This draws a black box with a red cross in it - a 'checkbox'.

possible attributes:
'x', 'y', 'size', 'crossColor', 'strokeColor', 'crosswidth'

# Public Attributes

**crossColor** None

**crosswidth** None

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## `DangerSign(_Symbol)`

This draws a 'danger' sign: a yellow box with a black exclamation point.

possible attributes:
'x', 'y', 'size', 'strokeColor', 'fillColor', 'strokeWidth'

# Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

### ETriangle(_Symbol)

This draws an equilateral triangle.

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

### FloppyDisk(_Symbol)

This widget draws an icon of a floppy disk.

possible attributes:
'x', 'y', 'size', 'diskcolor'

## Public Attributes

**diskColor** None

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

### NoEntry(_Symbol)

This draws a (British) No Entry sign - a red circle with a white line on it.

possible attributes:
'x', 'y', 'size'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**innerBarColor** color of the inner bar

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## `NoSmoking(NotAllowed)`

This draws a no-smoking sign.

possible attributes:
'x', 'y', 'size'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## `NotAllowed(_Symbol)`

This draws a 'forbidden' roundel (as used in the no-smoking sign).

possible attributes:
'x', 'y', 'size'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## `Octagon(_Symbol)`

This widget draws an Octagon.

possible attributes:
'x', 'y', 'size', 'fillColor', 'strokeColor'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

### `RTriangle(_Symbol)`

This draws a right-angled triangle.

possible attributes:
'x', 'y', 'size', 'fillColor', 'strokeColor'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

### `SmileyFace(_Symbol)`

This draws a classic smiley face.

possible attributes:
'x', 'y', 'size', 'fillColor'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## StopSign(_Symbol)

This draws a (British) stop sign.

possible attributes:
'x', 'y', 'size'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**stopColor** color of the word stop

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## Tickbox(_Symbol)

This draws a black box with a red tick in it - another 'checkbox'.

possible attributes:
'x', 'y', 'size', 'tickColor', 'strokeColor', 'tickwidth'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**tickColor** None

**tickwidth** None

**x** symbol x coordinate

**y** symbol y coordinate

## YesNo(_Symbol)

This widget draw a tickbox or crossbox depending on 'testValue'.

If this widget is supplied with a 'True' or 1 as a value for testValue, it will use the tickbox widget. Otherwise, it will produce a crossbox.

possible attributes:
'x', 'y', 'size', 'tickcolor', 'crosscolor', 'testValue'

## Public Attributes

**crosscolor** None

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**testValue** None

**tickcolor** None

**x** symbol x coordinate

**y** symbol y coordinate

### _Symbol(Widget)

Abstract base widget
possible attributes:
'x', 'y', 'size', 'fillColor', 'strokeColor'

## Public Attributes

**dx** symbol x coordinate adjustment

**dy** symbol x coordinate adjustment

**fillColor** None

**size** None

**strokeColor** None

**strokeWidth** None

**x** symbol x coordinate

**y** symbol y coordinate