

---

Documentation for package "reportlab.graphics"  
Generated by: graphdocpy.py version 0.8  
Date generated: 2002-05-28 16:05  
Format: PDF

---

|                                  |    |
|----------------------------------|----|
| reportlab.graphics               | 7  |
| axes                             | 7  |
| Classes                          | 7  |
| AdjYValueAxis(YValueAxis)        | 7  |
| Public Attributes                | 7  |
| CategoryAxis(Widget)             | 10 |
| Public Attributes                | 10 |
| NormalDateXValueAxis(XValueAxis) | 10 |
| Public Attributes                | 10 |
| ValueAxis(Widget)                | 13 |
| Public Attributes                | 13 |
| XCategoryAxis(CategoryAxis)      | 13 |
| Public Attributes                | 13 |
| XValueAxis(ValueAxis)            | 14 |
| Public Attributes                | 14 |
| YCategoryAxis(CategoryAxis)      | 16 |
| Public Attributes                | 16 |
| YValueAxis(ValueAxis)            | 16 |
| Public Attributes                | 16 |
| Functions                        | 19 |
| sample0a( ... )                  | 19 |
| sample0b( ... )                  | 20 |
| sample1( ... )                   | 21 |
| sample4a( ... )                  | 22 |
| sample4b( ... )                  | 23 |
| sample4c( ... )                  | 24 |
| sample4c1( ... )                 | 25 |
| sample4d( ... )                  | 26 |
| sample5a( ... )                  | 27 |
| sample5b( ... )                  | 28 |
| sample5c( ... )                  | 29 |
| sample5d( ... )                  | 30 |
| sample6a( ... )                  | 31 |
| sample6b( ... )                  | 32 |
| sample6c( ... )                  | 33 |
| sample6d( ... )                  | 34 |
| sample7a( ... )                  | 35 |
| sample7b( ... )                  | 36 |
| sample7c( ... )                  | 37 |
| sample7d( ... )                  | 38 |

---

|                              |    |
|------------------------------|----|
| barcharts                    | 39 |
| Classes                      | 39 |
| BarChart(Widget)             | 39 |
| Public Attributes            | 39 |
| HorizontalBarChart(BarChart) | 39 |
| Public Attributes            | 39 |
| SampleH5c4(Drawing)          | 42 |
| VerticalBarChart(BarChart)   | 42 |
| Public Attributes            | 42 |
| Functions                    | 45 |
| sampleH0a( ... )             | 45 |
| sampleH0b( ... )             | 46 |
| sampleH0c( ... )             | 47 |
| sampleH1( ... )              | 48 |
| sampleH2a( ... )             | 49 |
| sampleH2b( ... )             | 50 |
| sampleH2c( ... )             | 51 |
| sampleH3( ... )              | 52 |
| sampleH4a( ... )             | 53 |
| sampleH4b( ... )             | 54 |
| sampleH4c( ... )             | 55 |
| sampleH4d( ... )             | 56 |
| sampleH5a( ... )             | 57 |
| sampleH5b( ... )             | 58 |
| sampleH5c1( ... )            | 59 |
| sampleH5c2( ... )            | 60 |
| sampleH5c3( ... )            | 61 |
| sampleH5c4( ... )            | 62 |
| sampleStacked1( ... )        | 63 |
| sampleSymbol1( ... )         | 64 |
| sampleV0a( ... )             | 65 |
| sampleV0b( ... )             | 66 |
| sampleV0c( ... )             | 67 |
| sampleV1( ... )              | 68 |
| sampleV2a( ... )             | 69 |
| sampleV2b( ... )             | 70 |
| sampleV2c( ... )             | 71 |
| sampleV3( ... )              | 72 |
| sampleV4a( ... )             | 73 |
| sampleV4b( ... )             | 74 |

---

|  |     |
|--|-----|
| sampleV4c( ... )                               | 75  |
| sampleV4d( ... )                               | 76  |
| sampleV5a( ... )                               | 77  |
| sampleV5b( ... )                               | 78  |
| sampleV5c1( ... )                              | 79  |
| sampleV5c2( ... )                              | 80  |
| sampleV5c3( ... )                              | 81  |
| sampleV5c4( ... )                              | 82  |
| legends  | 83  |
| Classes  | 83  |
| Legend(Widget)                                 | 83  |
| Public Attributes                              | 83  |
| Functions                                      | 85  |
| sample1c( ... )                                | 85  |
| sample2c( ... )                                | 86  |
| linecharts                                     | 87  |
| Classes  | 87  |
| HorizontalLineChart(LineChart)                 | 87  |
| Public Attributes                              | 87  |
| LineChart(Widget)                              | 90  |
| Public Attributes                              | 90  |
| SampleHorizontalLineChart(HorizontalLineChart) | 91  |
| Public Attributes                              | 91  |
| VerticalLineChart(LineChart)                   | 94  |
| Public Attributes                              | 94  |
| Functions                                      | 95  |
| sample1( ... )                                 | 95  |
| sample1a( ... )                                | 96  |
| sample2( ... )                                 | 97  |
| sample3( ... )                                 | 98  |
| lineplots                                      | 99  |
| Classes  | 99  |
| GridLinePlot(LinePlot)                         | 99  |
| Public Attributes                              | 99  |
| LinePlot(Widget)                               | 103 |
| Public Attributes                              | 103 |
| ScatterPlot(LinePlot)                          | 106 |
| Public Attributes                              | 106 |
| Functions                                      | 109 |
| sample1a( ... )                                | 109 |

---

|                         |     |
|-------------------------|-----|
| sample1b( ... )         | 110 |
| sample1c( ... )         | 111 |
| sample2( ... )          | 112 |
| piecharts               | 114 |
| Classes                 | 114 |
| LegendedPie(Pie)        | 114 |
| Public Attributes       | 114 |
| Pie(Widget)             | 116 |
| Public Attributes       | 116 |
| Functions               | 118 |
| sample0a( ... )         | 118 |
| sample0b( ... )         | 119 |
| sample1( ... )          | 120 |
| sample2( ... )          | 121 |
| sample3( ... )          | 122 |
| sample4( ... )          | 123 |
| textlabels              | 124 |
| Classes                 | 124 |
| BarChartLabel(Label)    | 124 |
| Public Attributes       | 124 |
| Label(Widget)           | 126 |
| Public Attributes       | 126 |
| NA_Label(BarChartLabel) | 128 |
| Public Attributes       | 128 |
| flags                   | 130 |
| Classes                 | 130 |
| Flag(_Symbol)           | 130 |
| Public Attributes       | 130 |
| Star(_Symbol)           | 131 |
| Public Attributes       | 131 |
| signsandsymbols         | 131 |
| Classes                 | 131 |
| ArrowOne(_Symbol)       | 131 |
| Public Attributes       | 132 |
| ArrowTwo(ArrowOne)      | 132 |
| Public Attributes       | 132 |
| Crossbox(_Symbol)       | 132 |
| Public Attributes       | 132 |
| DangerSign(_Symbol)     | 133 |
| Public Attributes       | 133 |

---

|                       |     |
|-----------------------|-----|
| ETriangle(_Symbol)    | 133 |
| Public Attributes     | 133 |
| FloppyDisk(_Symbol)   | 133 |
| Public Attributes     | 134 |
| NoEntry(_Symbol)      | 134 |
| Public Attributes     | 134 |
| NoSmoking(NotAllowed) | 134 |
| Public Attributes     | 134 |
| NotAllowed(_Symbol)   | 135 |
| Public Attributes     | 135 |
| Octagon(_Symbol)      | 135 |
| Public Attributes     | 135 |
| RTriangle(_Symbol)    | 135 |
| Public Attributes     | 136 |
| SmileyFace(_Symbol)   | 136 |
| Public Attributes     | 136 |
| StopSign(_Symbol)     | 136 |
| Public Attributes     | 136 |
| Tickbox(_Symbol)      | 137 |
| Public Attributes     | 137 |
| YesNo(_Symbol)        | 137 |
| Public Attributes     | 137 |
| _Symbol(Widget)       | 138 |
| Public Attributes     | 138 |
| grids                 | 138 |
| Classes               | 138 |
| DoubleGrid(Widget)    | 138 |
| Public Attributes     | 138 |
| Grid(Widget)          | 140 |
| Public Attributes     | 140 |
| ShadedRect(Widget)    | 142 |
| Public Attributes     | 142 |

---

## reportlab.graphics

### 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.

**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 lowest tick label when true.

**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.

**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.

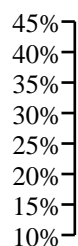
**visibleAxis** Display axis line, if true.

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

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

### *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 = ['10%', '15%', '20%', '25%', '30%', '35%', '40%', '45%']
labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x849324c>
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
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
visibleTicks = 1
```

## CategoryAxis(Widget)

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

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

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

**visibleGrid** Display axis grid, 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.

**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.

**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.

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

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

#### *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 = '%d'
labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x84b11cc>
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
tickDown = 5
tickUp = 0
valueMax = None
valueMin = None
valueStep = None
valueSteps = None
visible = 1
visibleAxis = 1
visibleGrid = 0
visibleTicks = 1
xLabelFormat = '{mm}/{yy}'
```

## **ValueAxis(Widget)**

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.

**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.

**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.

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

## **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.

**tickUp** Tick length up the axis.

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

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

**visibleGrid** Display axis grid, 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.

**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.

**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.

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

#### *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 = '%d'
labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x84d264c>
maximumTicks = 7
minimumTickSpacing = 10
rangeRound = 'none'
strokeColor = Color(0,0,0)
strokeDashArray = None
strokeWidth = 1
tickDown = 5
tickUp = 0
valueMax = None
valueMin = None
valueStep = None
visible = 1
visibleAxis = 1
visibleGrid = 0
visibleTicks = 1
```

## **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.

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

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

**visibleGrid** Display axis grid, 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.

**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.

**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.

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

### *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 = '%d'
labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x84eaacc>
maximumTicks = 7
minimumTickSpacing = 10
rangeRound = 'none'
strokeColor = Color(0,0,0)
strokeDashArray = None
```

```
strokeWidth = 1
tickLeft = 5
tickRight = 0
valueMax = None
valueMin = None
valueStep = None
visible = 1
visibleAxis = 1
visibleGrid = 0
visibleTicks = 1
```

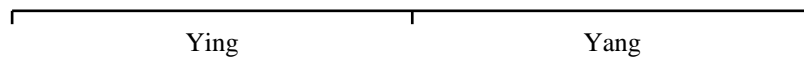
## Functions

### **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
```

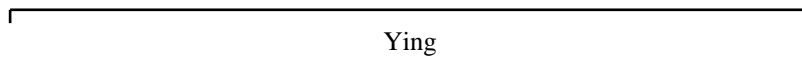


**sample0b( ... )**

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
```

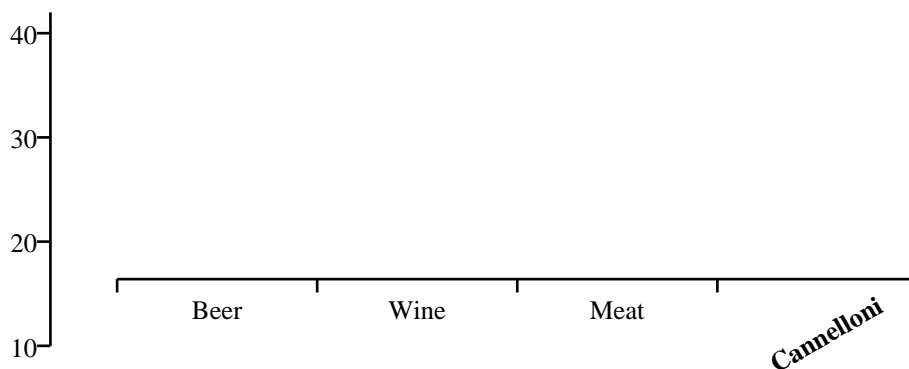


**sample1( ... )**

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
```

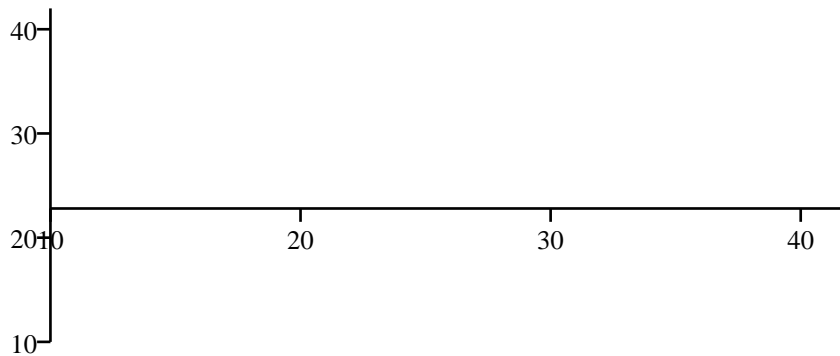


**sample4a( ... )**

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
```

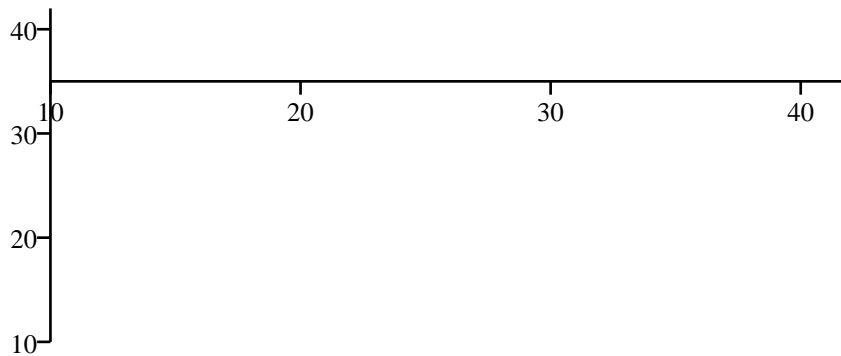


**sample4b( ... )**

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
```

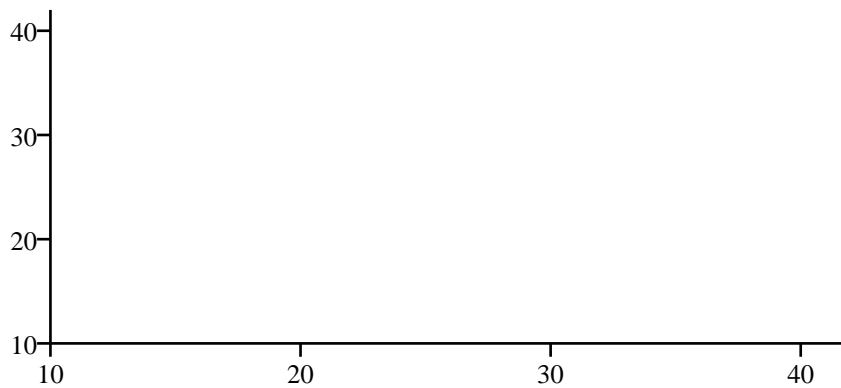


**sample4c( ... )**

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
```

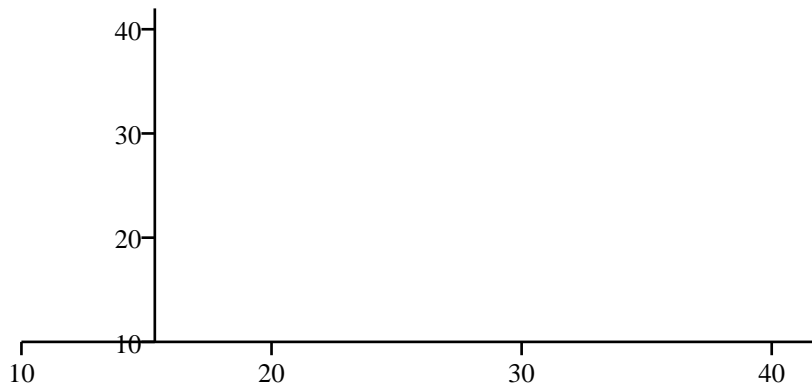


**sample5a( ... )**

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
```

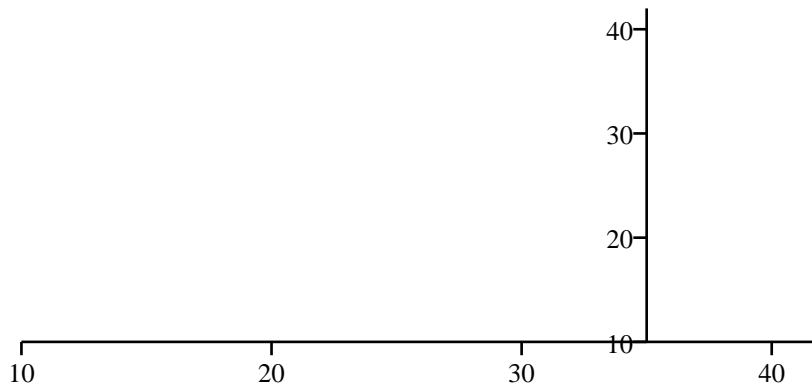


**sample5b( ... )**

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
```

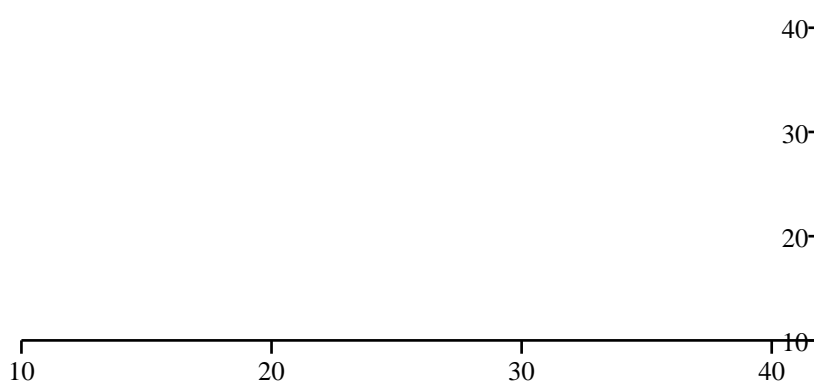


**sample5c( ... )**

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
```

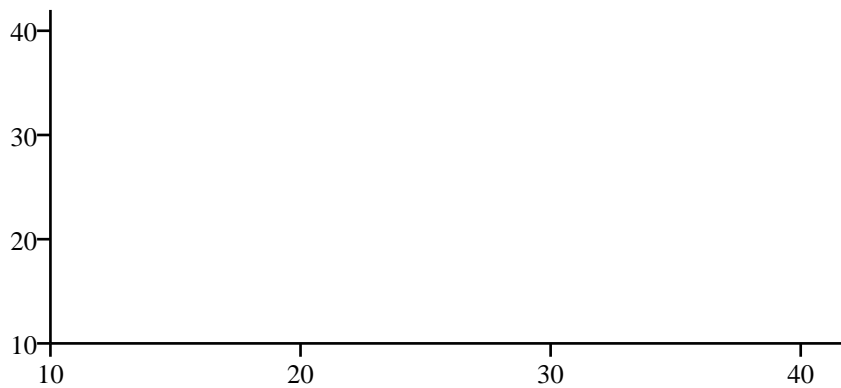


**sample5d( ... )**

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
```

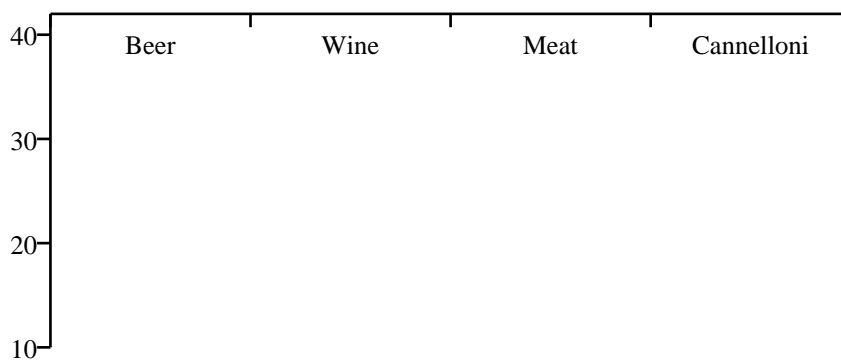


**sample6a( ... )**

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
```

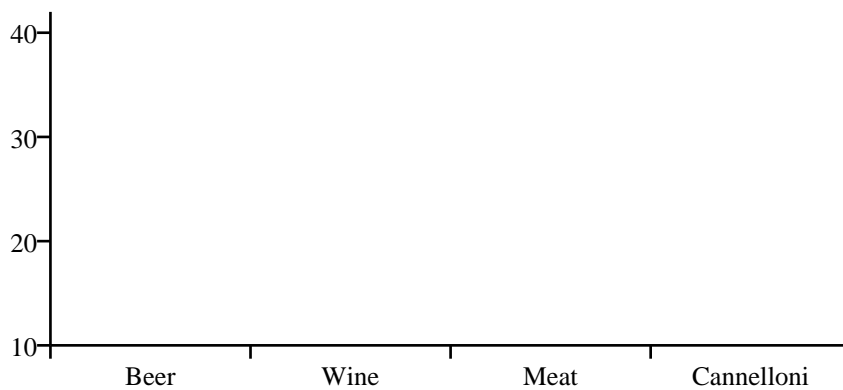


**sample6b( ... )**

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
```



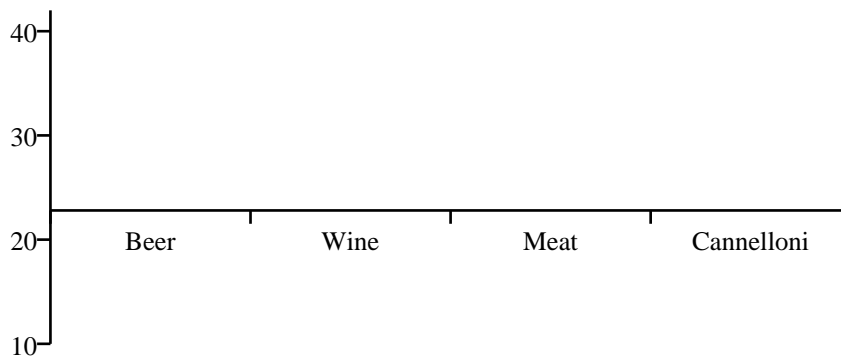


**sample6c( ... )**

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
```

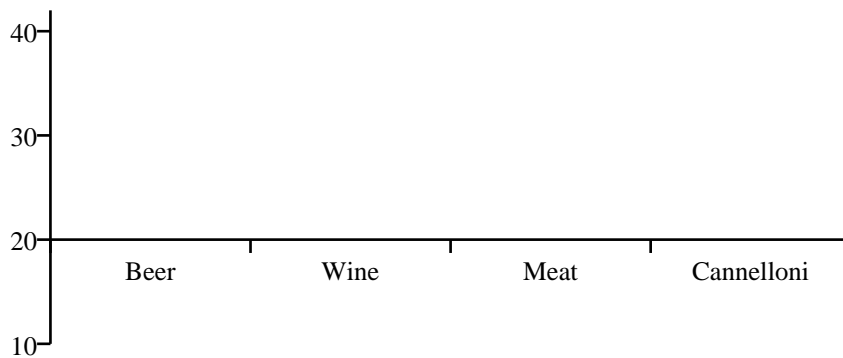


**sample6d( ... )**

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
```

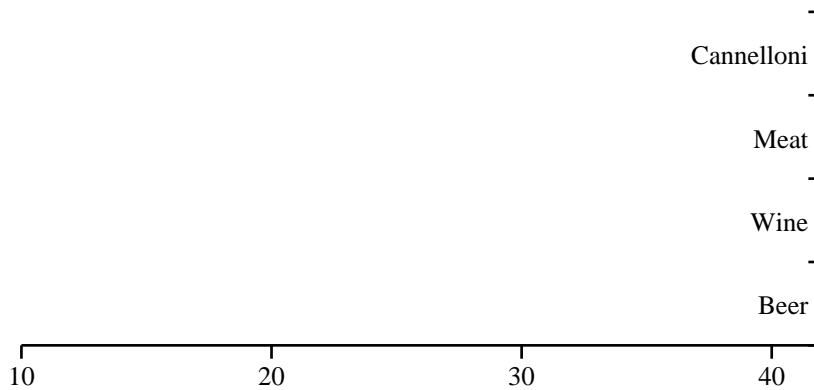


**sample7a( ... )**

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
```

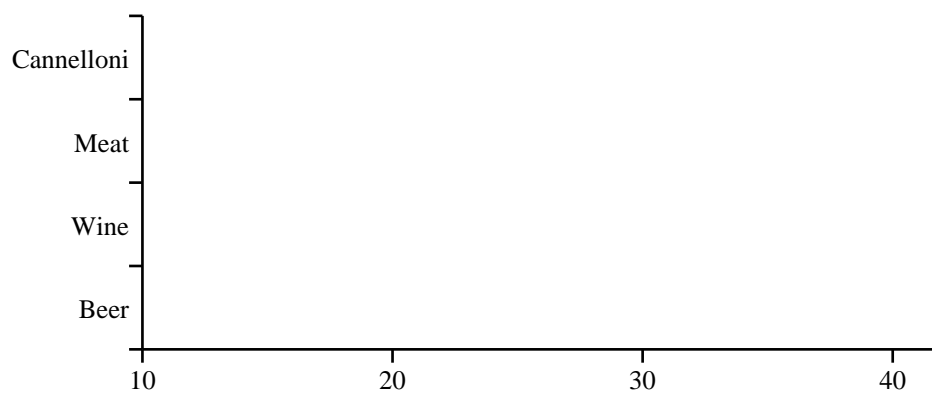


**sample7b( ... )**

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
```

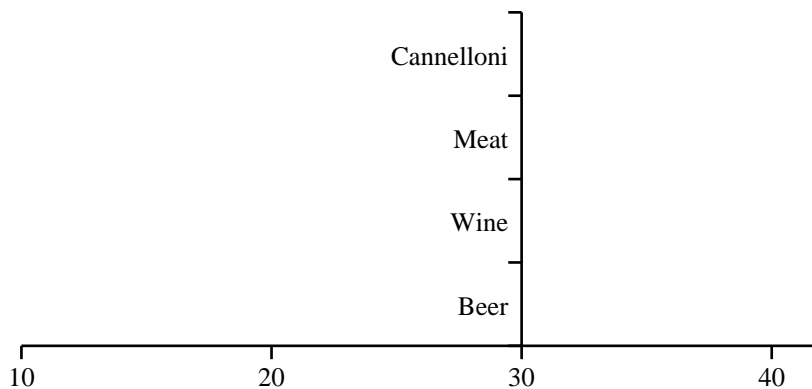


**sample7c( ... )**

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
```

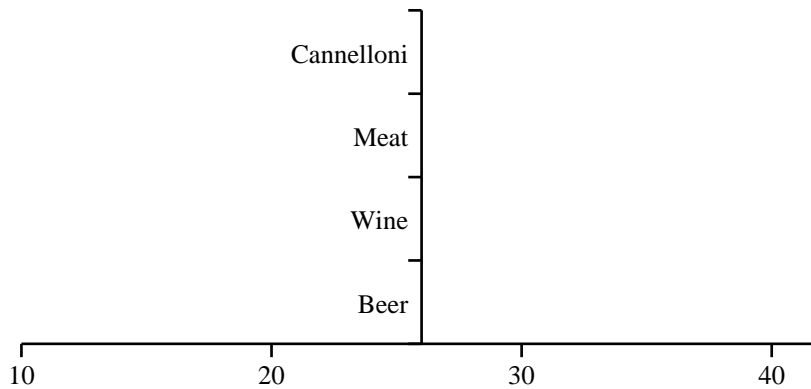


**sample7d( ... )**

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 (Widget )**

Abstract base class, unusable by itself.

### **Public Attributes**

**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.

### **HorizontalBarChart (BarChart )**

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

### **Public Attributes**

**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.

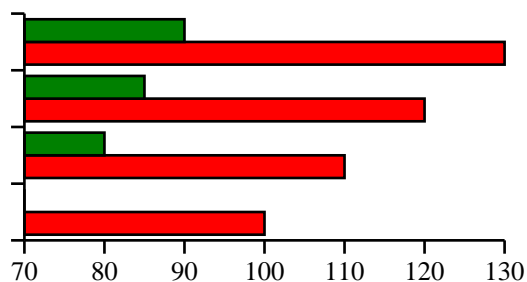
**width** Width of the chart.

**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

```
barLabelFormat = None
barLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8587dcc>
barSpacing = 0
barWidth = 10
bars = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8587e0c>
categoryAxis.categoryNames = None
categoryAxis.gridEnd = 0
```



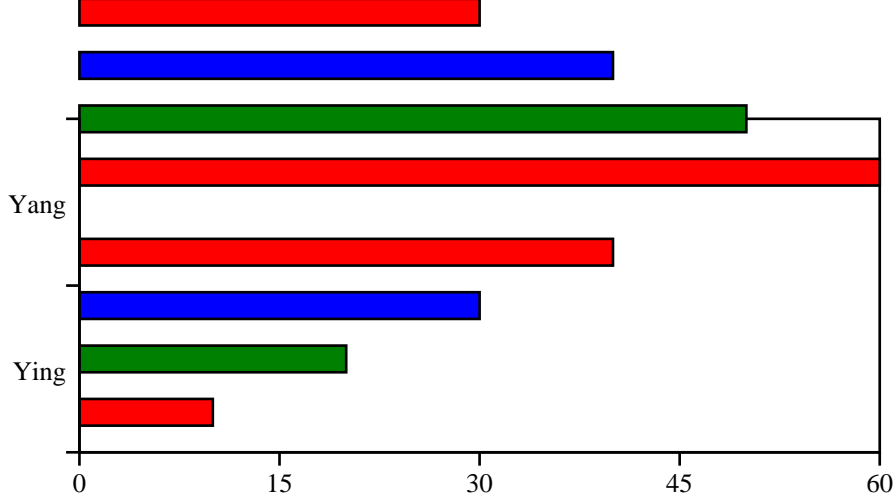
```
categoryAxis.gridStart = 0
categoryAxis.gridStrokeColor = Color(0,0,0)
categoryAxis.gridStrokeDashArray = None
categoryAxis.gridStrokeWidth = 0.25
categoryAxis.joinAxis = None
categoryAxis.joinAxisMode = None
categoryAxis.joinAxisPos = None
categoryAxis.labelAxisMode = 'axis'
categoryAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8587cec>
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.visible = 1
categoryAxis.visibleAxis = 1
categoryAxis.visibleGrid = 0
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 = '%d'
valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x8587d6c>
valueAxis.maximumTicks = 7
valueAxis.minimumTickSpacing = 10
valueAxis.rangeRound = 'none'
valueAxis.strokeColor = Color(0,0,0)
valueAxis.strokeDashArray = None
valueAxis.strokeWidth = 1
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.visibleTicks = 1
width = 180
x = 20
y = 10
```

## 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

**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.

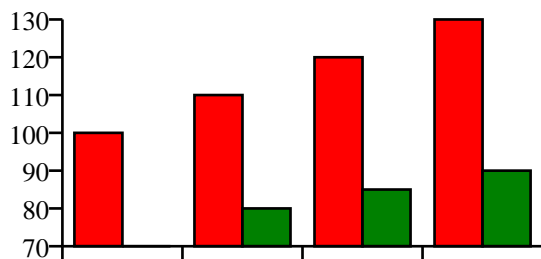
**width** Width of the chart.

**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*

```
barLabelFormat = None
barLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x859e72c>
barSpacing = 0
barWidth = 10
bars = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x859e76c>
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 0x859e62c>
categoryAxis.reverseDirection = 0
categoryAxis.strokeColor = Color(0,0,0)
categoryAxis.strokeDashArray = None
categoryAxis.strokeWidth = 1
```

```
categoryAxis.style = 'parallel'
categoryAxis.tickDown = 5
categoryAxis.tickUp = 0
categoryAxis.visible = 1
categoryAxis.visibleAxis = 1
categoryAxis.visibleGrid = 0
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 = '%d'
valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x859e6ac>
valueAxis.maximumTicks = 7
valueAxis.minimumTickSpacing = 10
valueAxis.rangeRound = 'none'
valueAxis.strokeColor = Color(0,0,0)
valueAxis.strokeDashArray = None
valueAxis.strokeWidth = 1
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.visibleTicks = 1
width = 180
x = 20
y = 10
```

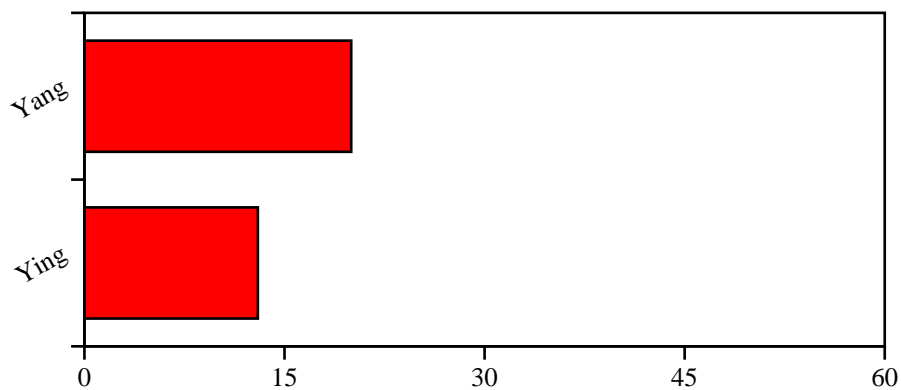
## Functions

### `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
```

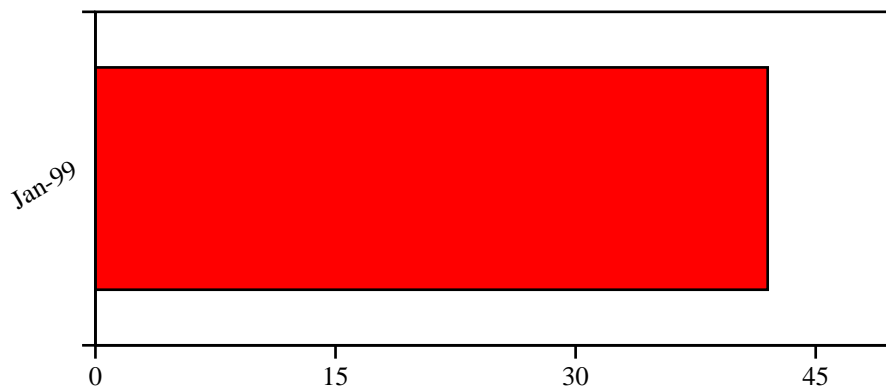


**sampleH0b( ... )**

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
```

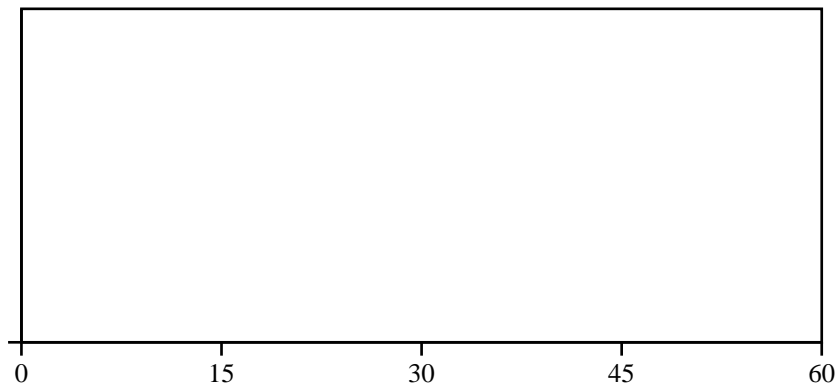


**sampleH0c( ... )**

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
```

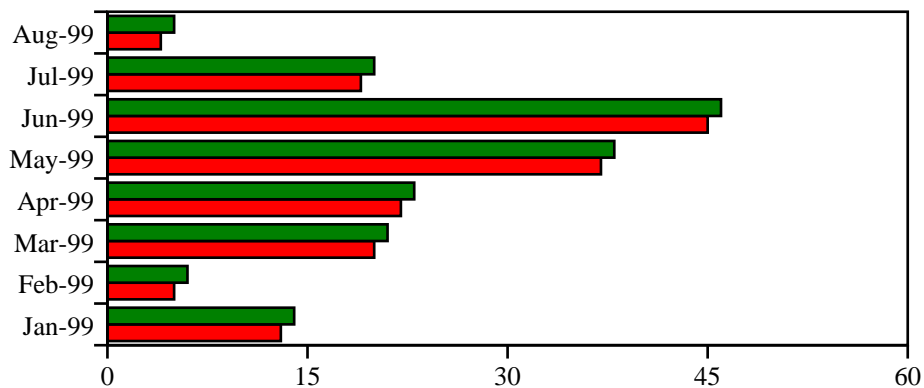


**sampleH1( ... )**

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)
    return drawing
```



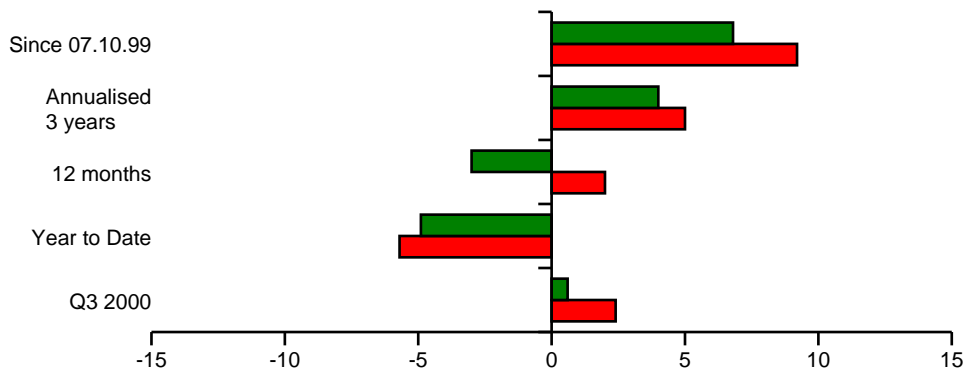


**sampleH2a( ... )**

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
```

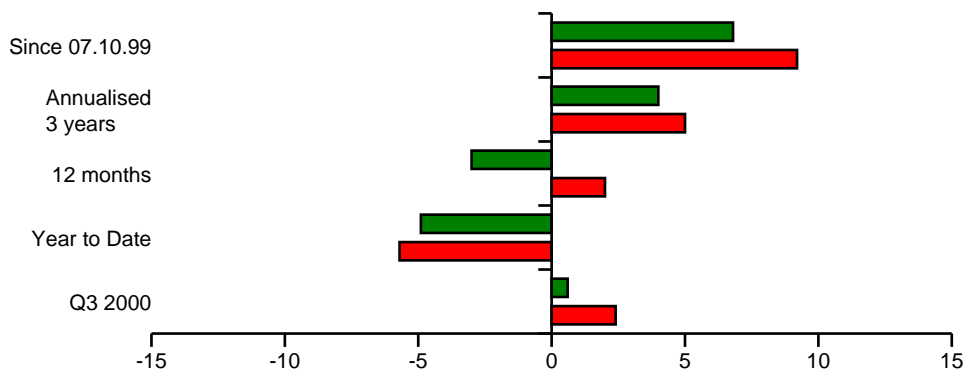


**sampleH2b( ... )**

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
```

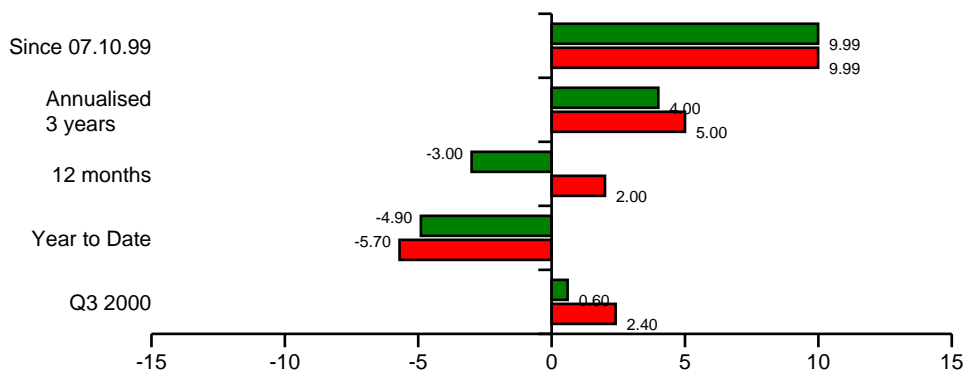


**sampleH2c( ... )**

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
```



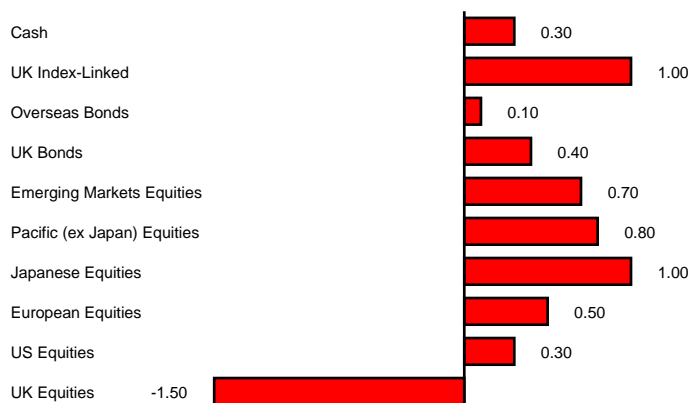
**sampleH3( ... )**

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
```

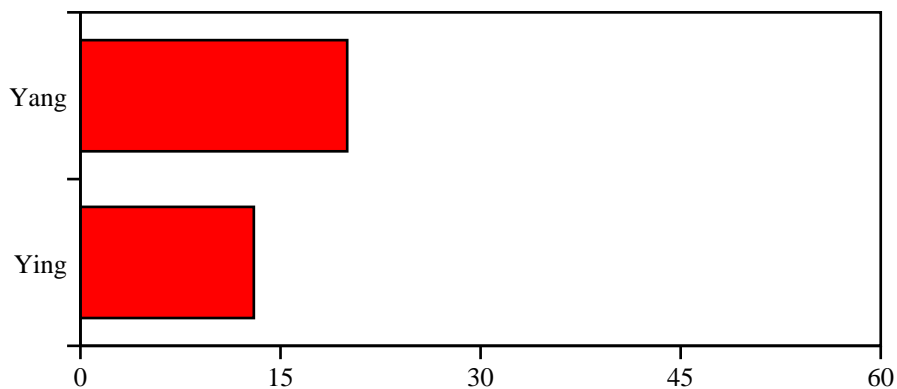


**sampleH4a( ... )**

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
```

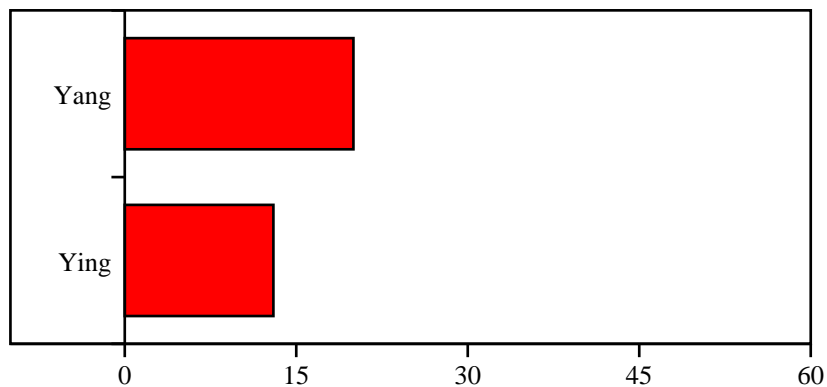


**sampleH4b( ... )**

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
```

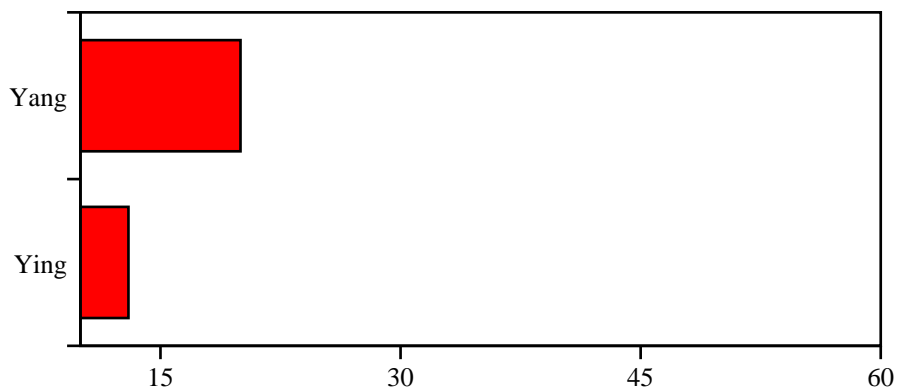


**sampleH4c( ... )**

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
```

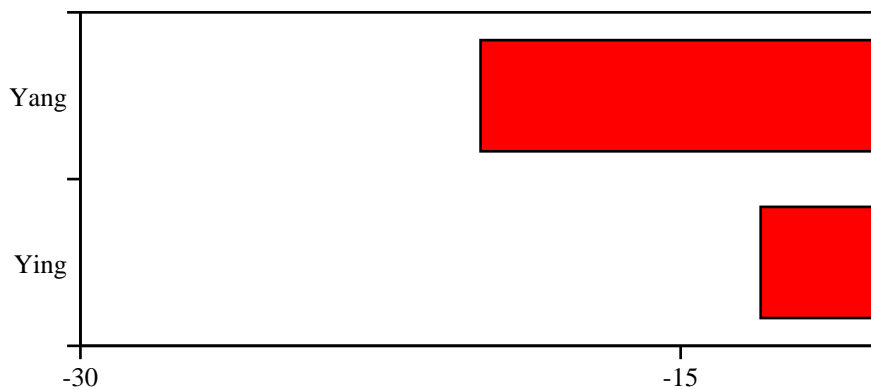


**sampleH4d( ... )**

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
```



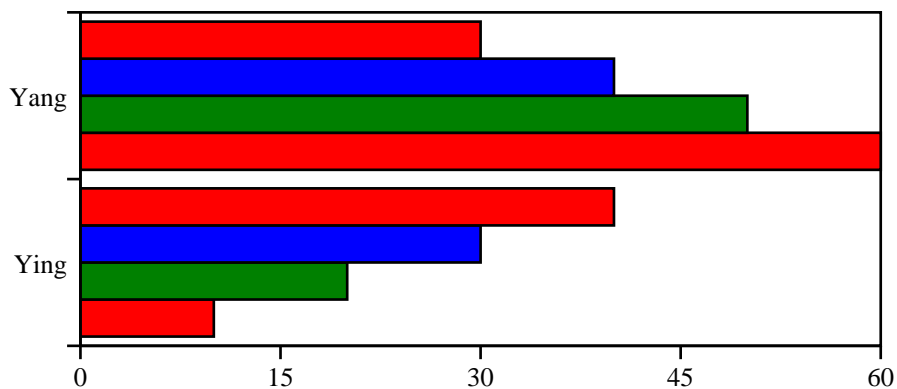


**sampleH5a( ... )**

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
```

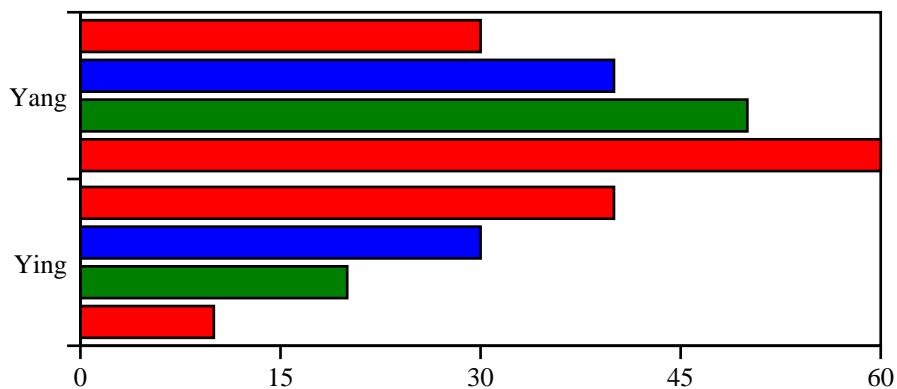


**sampleH5b( ... )**

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
```

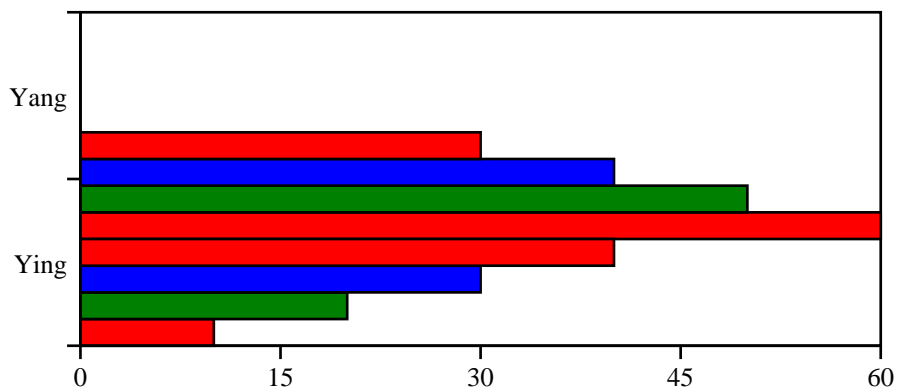


**sampleH5c1( ... )**

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
```

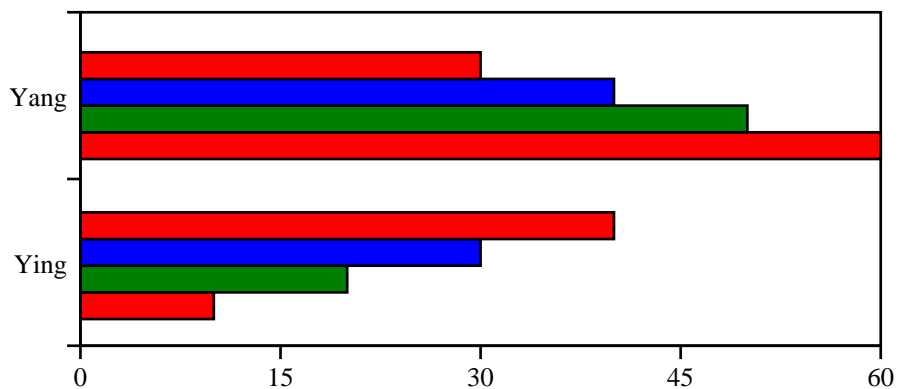


**sampleH5c2( ... )**

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
```

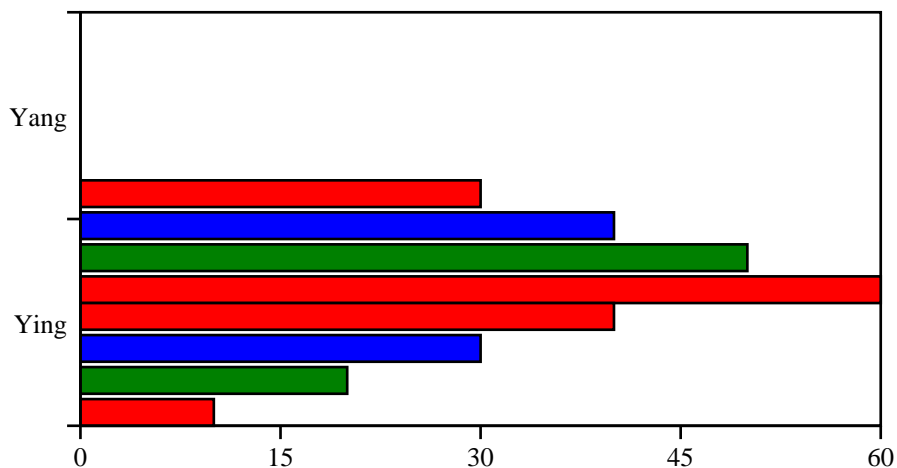


**sampleH5c3( ... )**

Simple bar chart with absolute spacing.

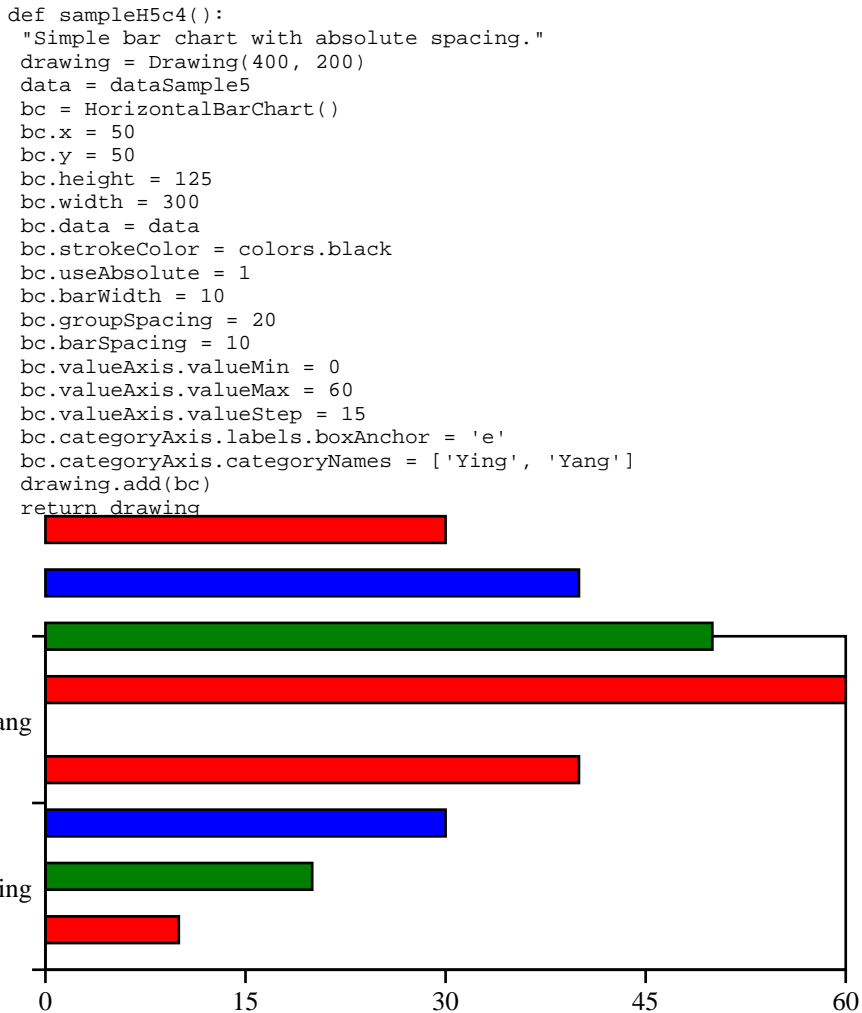
*Example*

```
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.

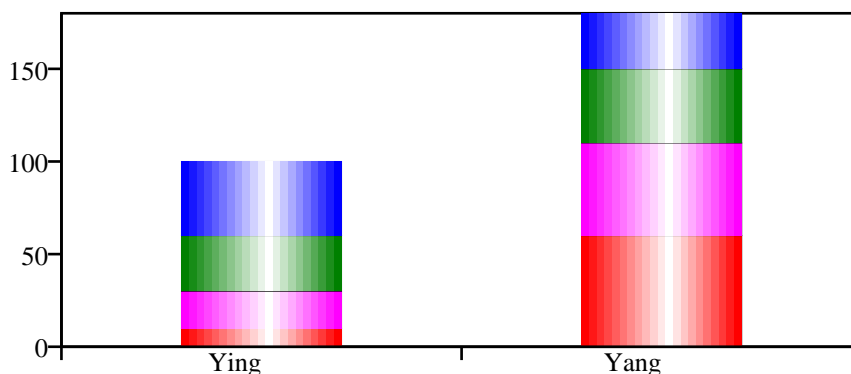
*Example*

**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']
    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
```

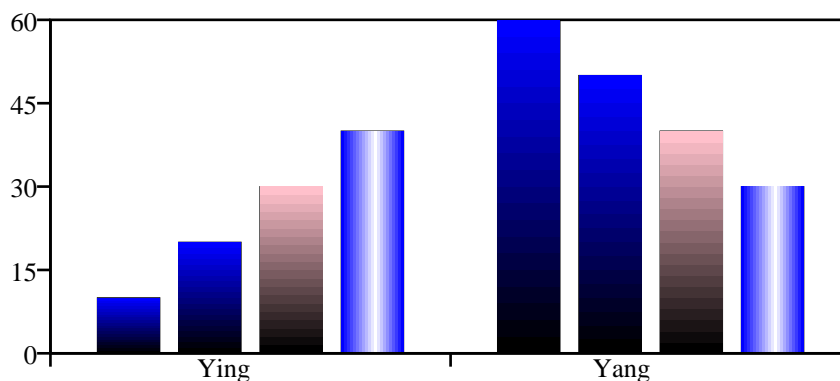


**sampleSymbol1( ... )**

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']
    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
```



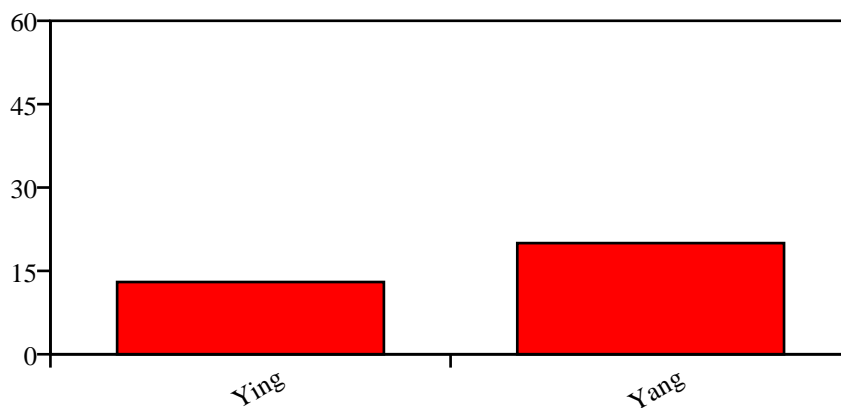


**sampleV0a( ... )**

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
```

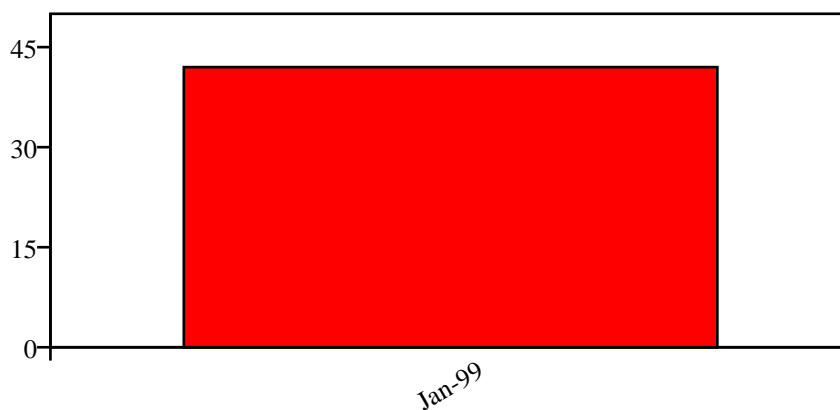


**sampleV0b( ... )**

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
```

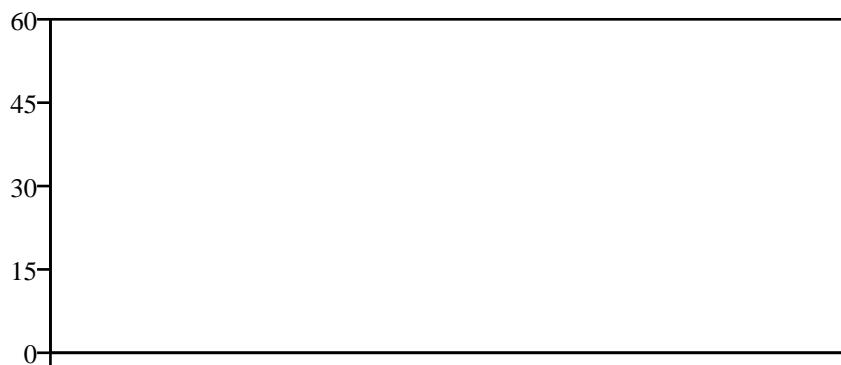


**sampleV0c( ... )**

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
```

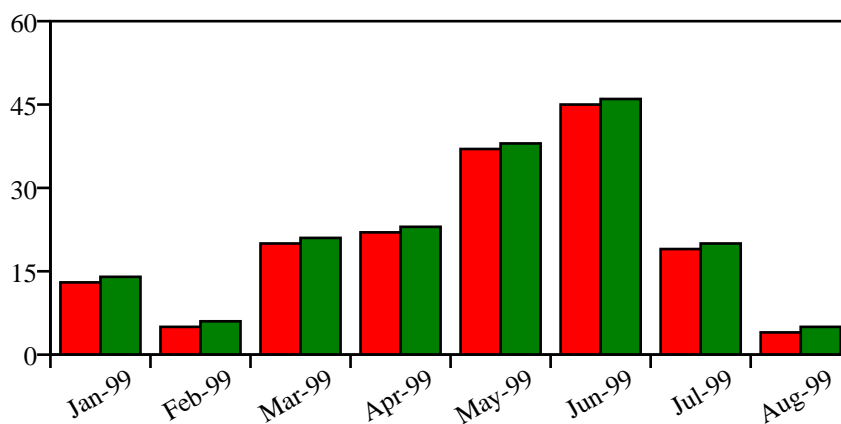


**sampleV1( ... )**

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
```

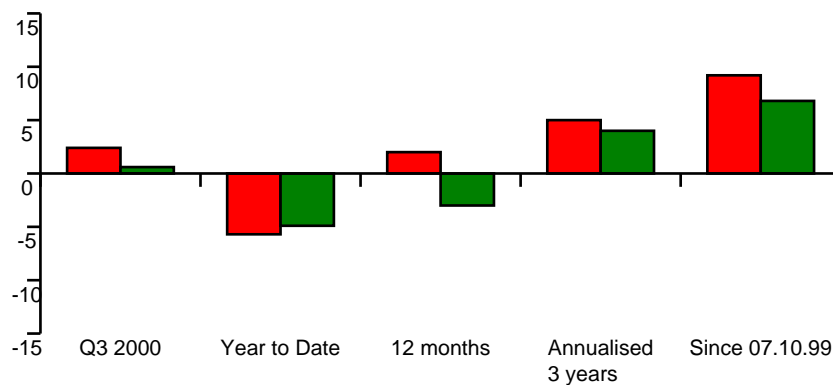


**sampleV2a( ... )**

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
```

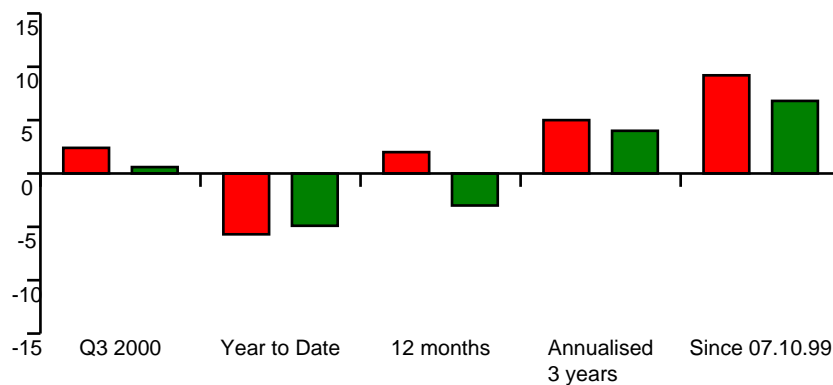


**sampleV2b( ... )**

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
```

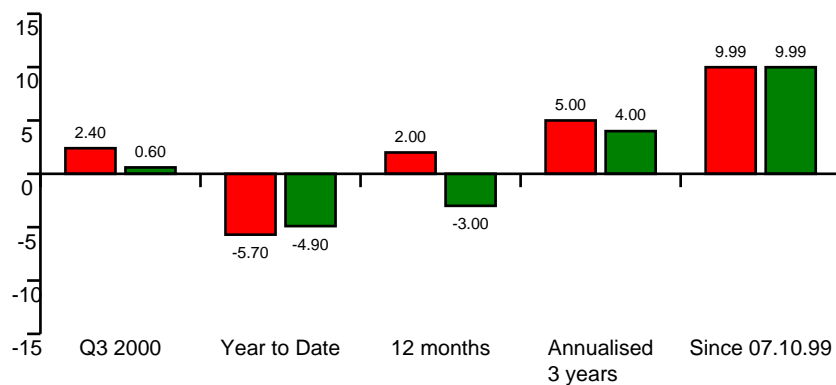


**sampleV2c( ... )**

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*

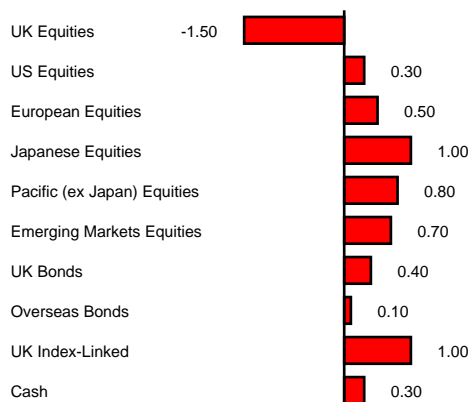
```
def sampleV3():
    "Faked horizontal bar chart using a vertical real one (deprecated)."
```

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



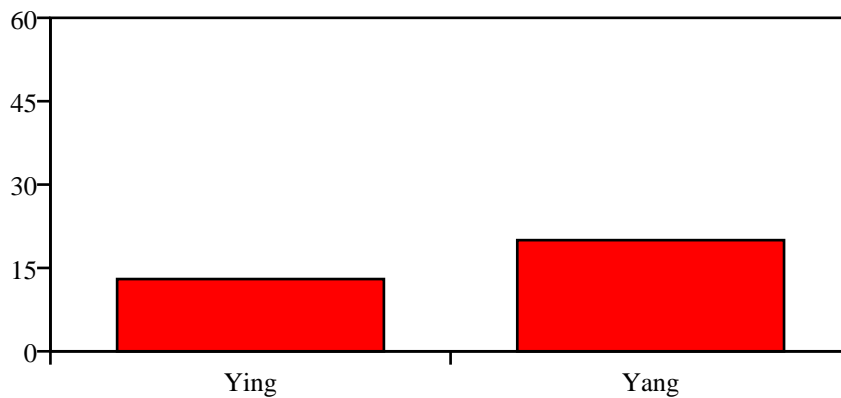


**sampleV4a( ... )**

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
```

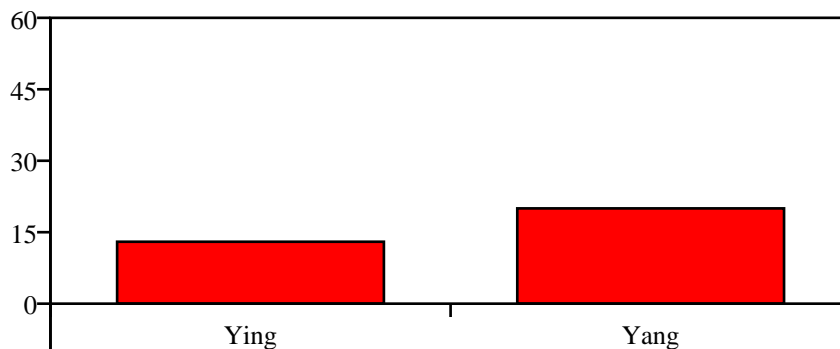


**sampleV4b( ... )**

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
```

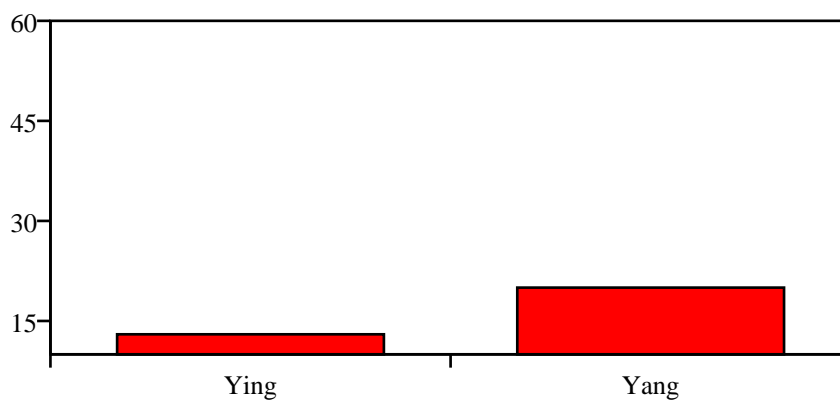


**sampleV4c( ... )**

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

*Example*

```
def sampleV4c():
    "A bar chart showing value axis region starting 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
```

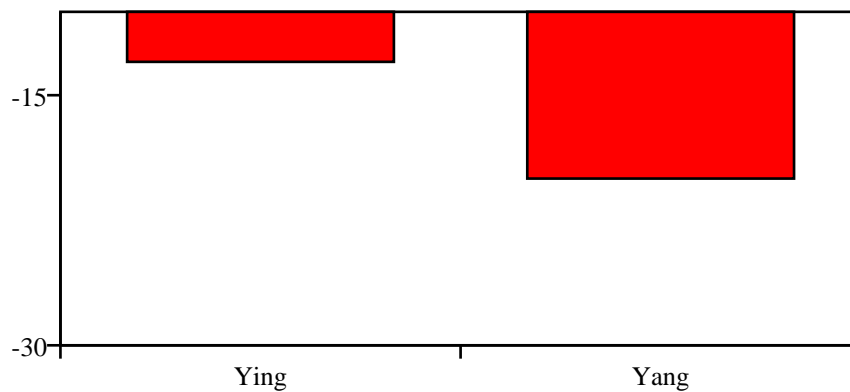


**sampleV4d( ... )**

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
```

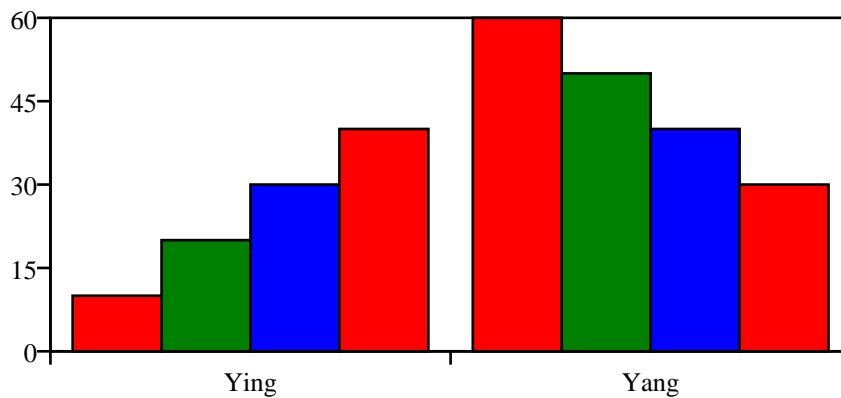


**sampleV5a( ... )**

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
```

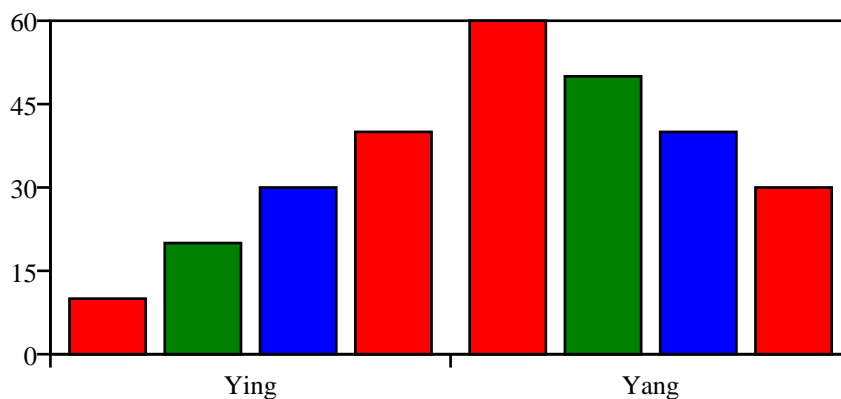


**sampleV5b( ... )**

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
```

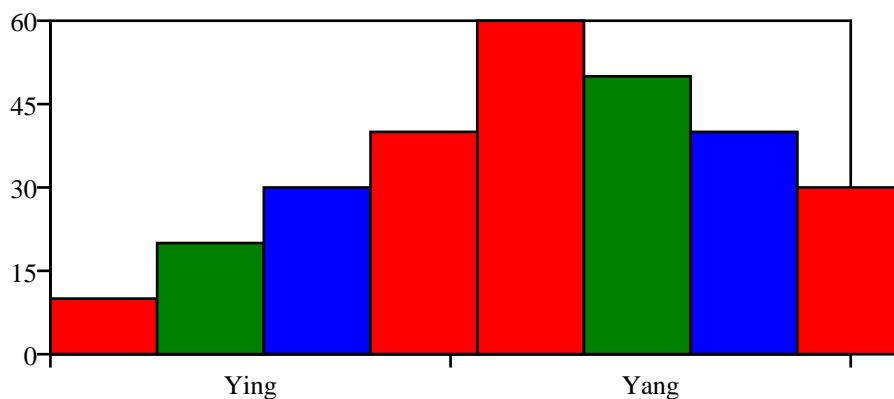


**sampleV5c1( ... )**

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
```

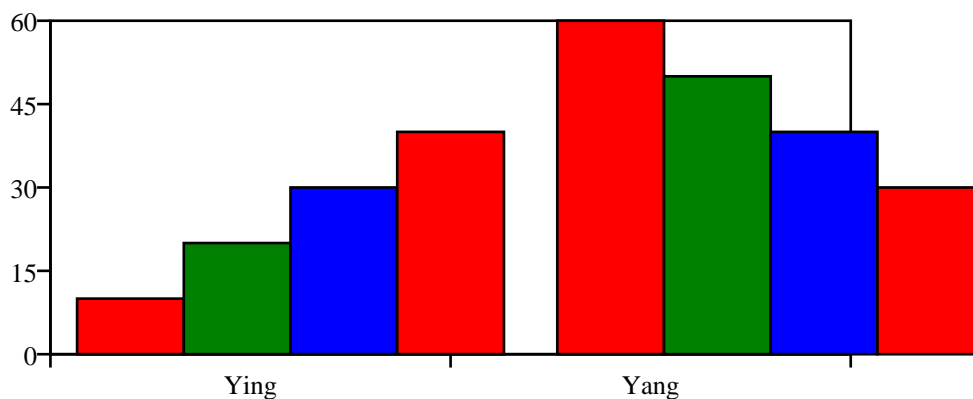


**sampleV5c2( ... )**

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
```



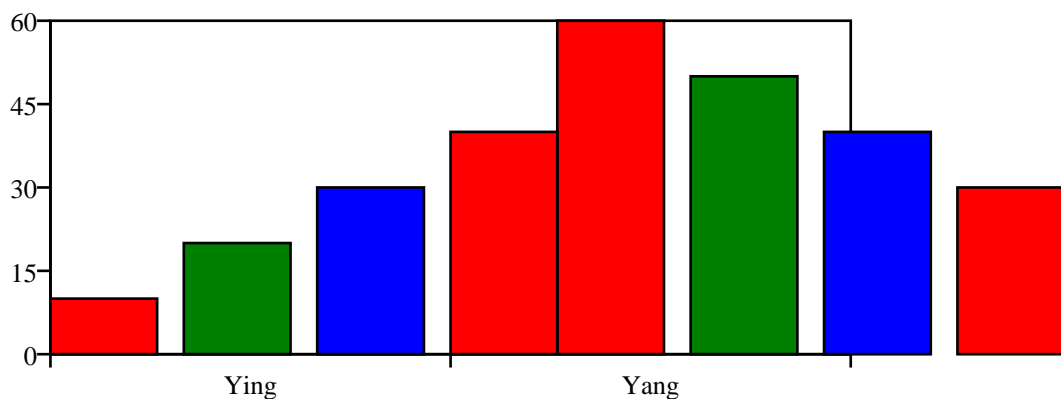


**sampleV5c3( ... )**

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
```

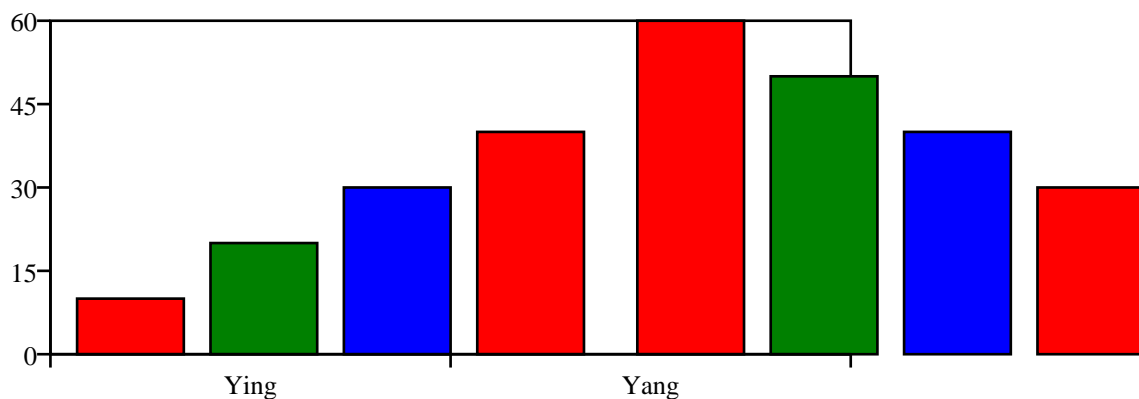


**sampleV5c4( ... )**

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
```



## 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 deltax=None

**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

**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

**strokeColor** Border color of the swatches

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








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

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

### 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 = string.split('red green blue yellow pink black white', ' ')
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  |  |       |   |

### *Properties of Example Widget*

```
alignment = 'left'
autoXPadding = 5
autoYPadding = 2
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
dx = 10
dxTextSpace = 10
dy = 10
fillColor = Color(0,0,0)
fontName = 'Times-Roman'
fontSize = 10
strokeColor = Color(0,0,0)
strokeWidth = 1
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 = string.split('red green blue yellow pink black white', ' ')
    items = map(lambda i:(getattr(colors, i), i), items)
    legend.colorNamePairs = items
    d.add(legend, 'legend')
    return d
```

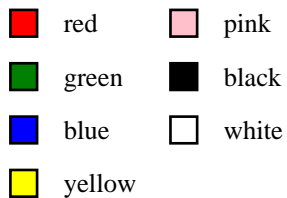
|   |  |   |
|---|--|---|
|  red   |  yellow |  white |
|  green |  pink   |   |
|  blue  |  black  |   |

**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 = string.split('red green blue yellow pink black white', ' ')
    items = map(lambda i:(getattr(colors, i), i), items)
    legend.colorNamePairs = items
    d.add(legend, 'legend')
    return d
```



## linecharts

This module defines a very preliminary Line Chart example.

### Classes

#### 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

**categoryAxis** Handle of the category axis.

**categoryNames** List of category names.

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

**fillColor** Color used for background interior of plot area.

**groupSpacing** ? - Likely to disappear.

**height** Height of the chart.

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

**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.

**strokeColor** Color used for background border of plot area.

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

**valueAxis** Handle of the value axis.

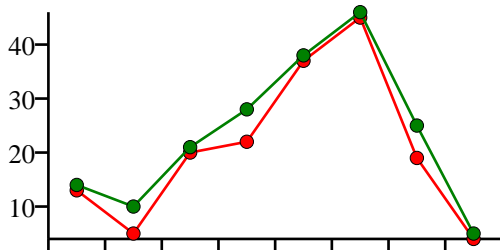
**width** Width of the chart.

**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

```
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 0x86788ac>
categoryAxis.reverseDirection = 0
categoryAxis.strokeColor = Color(0,0,0)
categoryAxis.strokeDashArray = None
categoryAxis.strokeWidth = 1
categoryAxis.style = 'parallel'
categoryAxis.tickDown = 5
categoryAxis.tickUp = 0
categoryAxis.visible = 1
categoryAxis.visibleAxis = 1
categoryAxis.visibleGrid = 0
categoryAxis.visibleTicks = 1
```



```
categoryNames = ('North', 'South', 'East', 'West')
data = [(100, 110, 120, 130), (70, 80, 80, 90)]
fillColor = None
groupSpacing = 1
height = 100
joinedLines = 1
lineLabelFormat = None
lineLabelNudge = 10
lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86789ac>
lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x867896c>
strokeColor = None
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 = '%d'
valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x867890c>
valueAxis.maximumTicks = 7
valueAxis.minimumTickSpacing = 10
valueAxis.rangeRound = 'none'
valueAxis.strokeColor = Color(0,0,0)
valueAxis.strokeDashArray = None
valueAxis.strokeWidth = 1
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.visibleTicks = 1
width = 200
x = 0
y = 0
```

## **LineChart(Widget)**

### **Public Attributes**

#### *Example*

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

#### *Properties of Example Widget*

## SampleHorizontalLineChart(HorizontalLineChart)

Sample class overwriting one method to draw additional horizontal lines.

### Public Attributes

**categoryAxis** Handle of the category axis.

**categoryNames** List of category names.

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

**fillColor** Color used for background interior of plot area.

**groupSpacing** ? - Likely to disappear.

**height** Height of the chart.

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

**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.

**strokeColor** Color used for background border of plot area.

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

**valueAxis** Handle of the value axis.

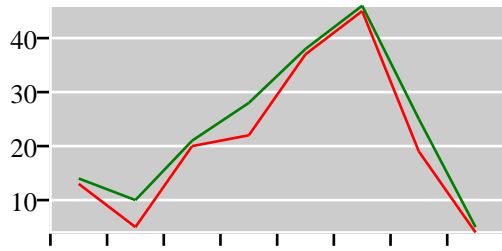
**width** Width of the chart.

**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

```
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 0x8690fcc>
categoryAxis.reverseDirection = 0
categoryAxis.strokeColor = Color(0,0,0)
categoryAxis.strokeDashArray = None
categoryAxis.strokeWidth = 1
categoryAxis.style = 'parallel'
categoryAxis.tickDown = 5
categoryAxis.tickUp = 0
categoryAxis.visible = 1
categoryAxis.visibleAxis = 1
categoryAxis.visibleGrid = 0
categoryAxis.visibleTicks = 1
categoryNames = ('North', 'South', 'East', 'West')
data = [(100, 110, 120, 130), (70, 80, 80, 90)]
fillColor = None
groupSpacing = 1
height = 100
joinedLines = 1
lineLabelFormat = None
lineLabelNudge = 10
lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x869b10c>
lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x869b0cc>
strokeColor = None
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 = '%d'
valueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x869b06c>
valueAxis.maximumTicks = 7
valueAxis.minimumTickSpacing = 10
valueAxis.rangeRound = 'none'
valueAxis.strokeColor = Color(0,0,0)
valueAxis.strokeDashArray = None
valueAxis.strokeWidth = 1
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.visibleTicks = 1
width = 200
```

x = 0  
y = 0

## **VerticalLineChart (LineChart)**

### **Public Attributes**

#### *Example*

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

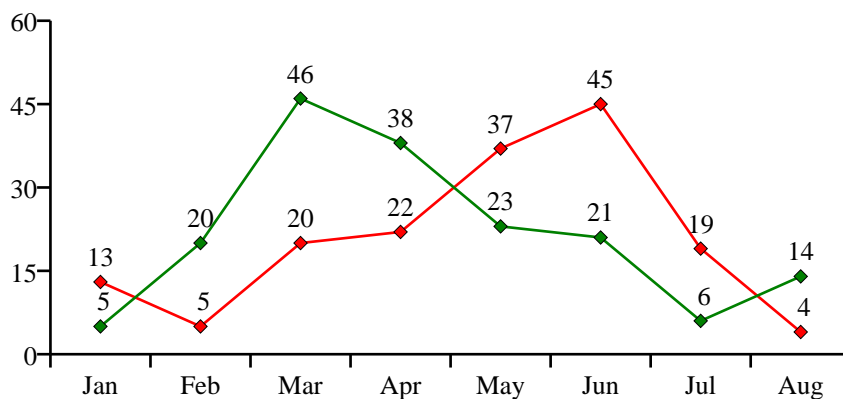
#### *Properties of Example Widget*

## Functions

**sample1( ... )**

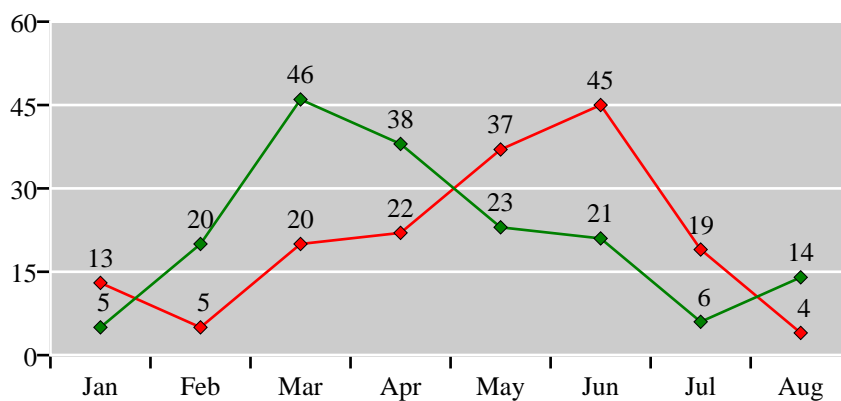
### *Example*

```
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*

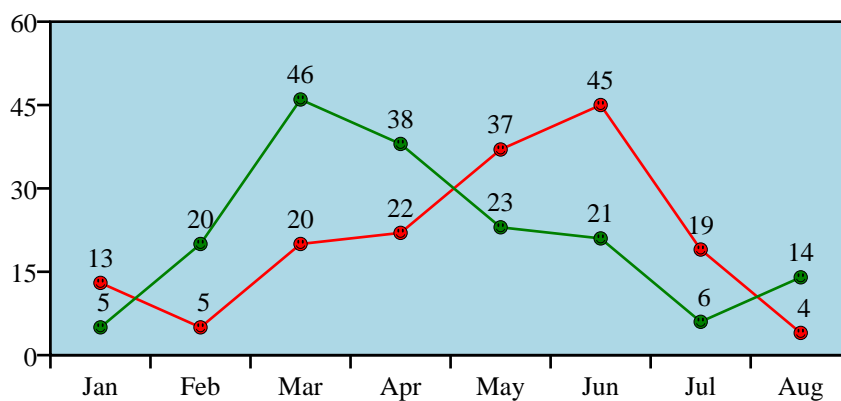
```
def sample1a():
    drawing = Drawing(400, 200)
    data = [
        (13, 5, 20, 22, 37, 45, 19, 4),
        (5, 20, 46, 38, 23, 21, 6, 14)
    ]
    lc = SampleHorizontalLineChart()
    lc.x = 50
    lc.y = 50
    lc.height = 125
    lc.width = 300
    lc.data = data
    lc.joinedLines = 1
    lc.strokeColor = 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
```





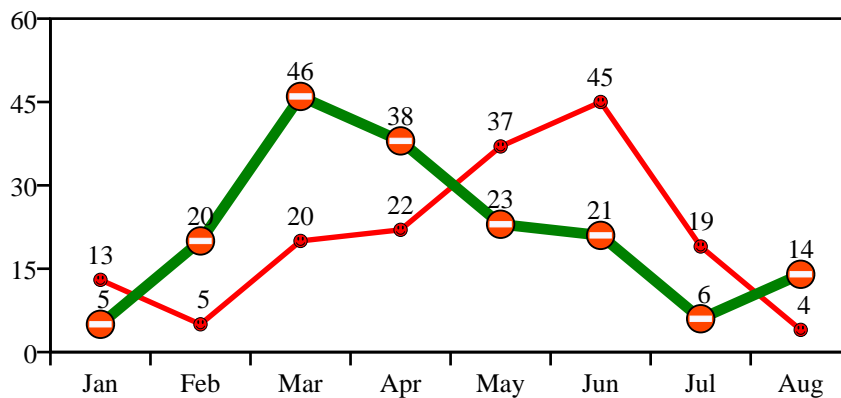
**sample2( ... )***Example*

```
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

#### 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 background 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

**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.

**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.

**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 = 100
joinedLines = 1
lineLabelFormat = None
lineLabelNudge = 10
lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86d4bcc>
lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86d4b8c>
reversePlotOrder = 0
scaleFactor = None
strokeColor = None
width = 200
x = 0
xValueAxis.avoidBoundFrac = None
xValueAxis.bottomAxisLabelSlack = 0.10000000000000001
xValueAxis.dailyFreq = 0
xValueAxis.dayOfWeekName = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
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 = '%d'
xValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86d4c6c>
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.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.visibleTicks = 1
xValueAxis.xLabelFormat = '{mm}/{yy}'
y = 0
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 = '%d'
yValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86d4a6c>
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.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.visibleTicks = 1
```

## LinePlot(Widget)

Line plot with multiple lines.

Both x- and y-axis are value axis (so there are no separate X and Y versions of this class).

## Public Attributes

**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.

**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.

**width** Width of the chart.

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

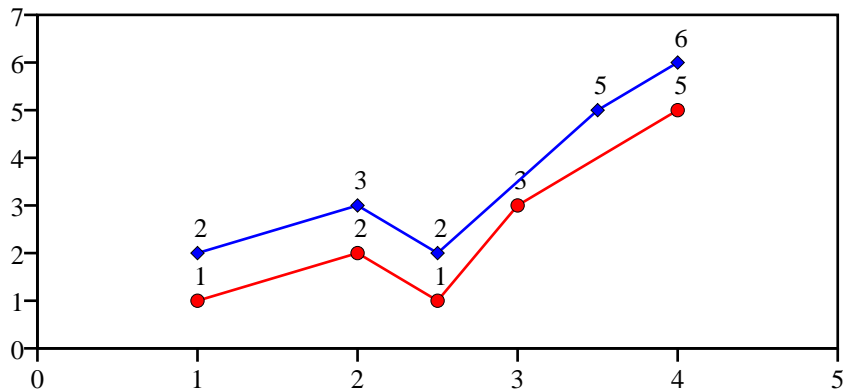
**xValueAxis** Handle of the x axis.

**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*

```
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 = 100
joinedLines = 1
lineLabelFormat = None
lineLabelNudge = 10
lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86d8dac>
lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86d8d6c>
reversePlotOrder = 0
strokeColor = None
width = 200
x = 0
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 = '%d'
xValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86d8cac>
xValueAxis.maximumTicks = 7
xValueAxis.minimumTickSpacing = 10
xValueAxis.rangeRound = 'none'
xValueAxis.strokeColor = Color(0,0,0)
xValueAxis.strokeDashArray = None
xValueAxis.strokeWidth = 1
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.visibleTicks = 1
y = 0
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 = '%d'
yValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86d8d0c>
yValueAxis.maximumTicks = 7
yValueAxis.minimumTickSpacing = 10
yValueAxis.rangeRound = 'none'
yValueAxis.strokeColor = Color(0,0,0)
yValueAxis.strokeDashArray = None
yValueAxis.strokeWidth = 1
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.visibleTicks = 1
```

## ScatterPlot(LinePlot)

A scatter plot widget

### Public Attributes

**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

**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.

**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
lineLabelFormat = '%.2f'
lineLabelNudge = 0
lineLabels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86e56ac>
lines = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86e566c>
outerBorderColor = Color(0,0,0)
outerBorderOn = 1
reversePlotOrder = 0
rightPadding = 10
strokeColor = None
topPadding = 5
width = 142
x = 25.996000000000002
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 = '%d'
xValueAxis.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86ae4ec>
xValueAxis.maximumTicks = 7
xValueAxis.minimumTickSpacing = 10
xValueAxis.rangeRound = 'both'
xValueAxis.strokeColor = Color(0,0,0)
xValueAxis.strokeDashArray = None
xValueAxis.strokeWidth = 0.5
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.visibleTicks = 1
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.labels = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x86e55ec>
yValueAxis.maximumTicks = 7
yValueAxis.minimumTickSpacing = 10
yValueAxis.rangeRound = 'both'
yValueAxis.strokeColor = Color(0,0,0)
```

```
yValueAxis.strokeDashArray = None
yValueAxis.strokeWidth = 0.5
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.visibleTicks = 1
```

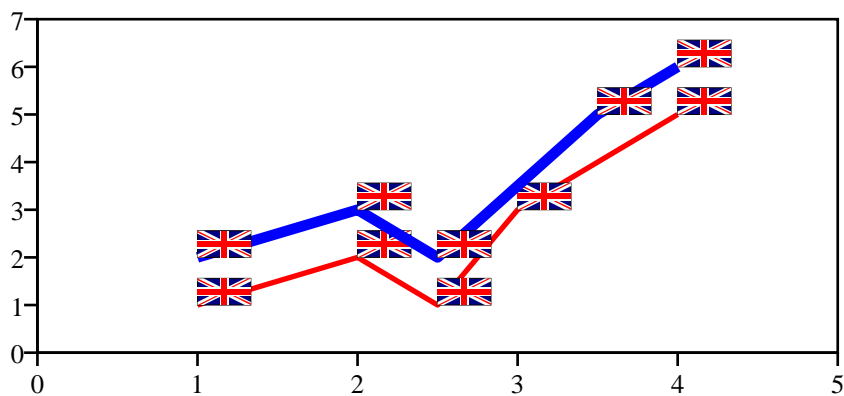
## Functions

### `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
```

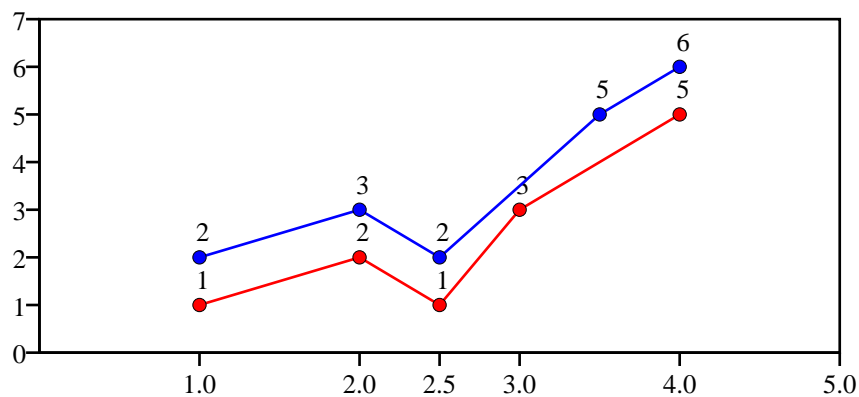


**sample1b( ... )**

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
```

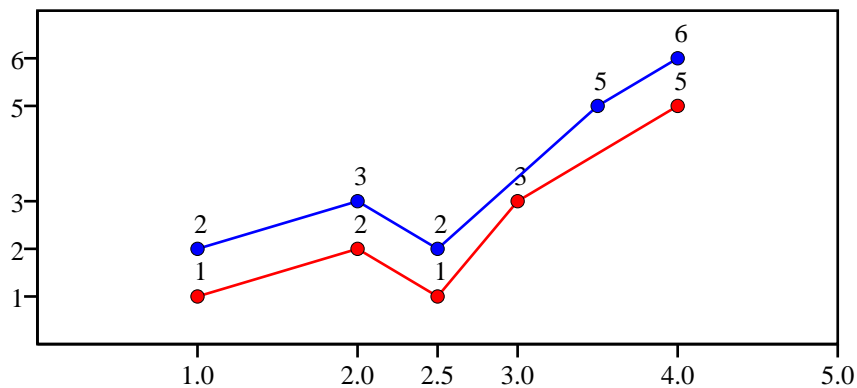


**sample1c( ... )**

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
```



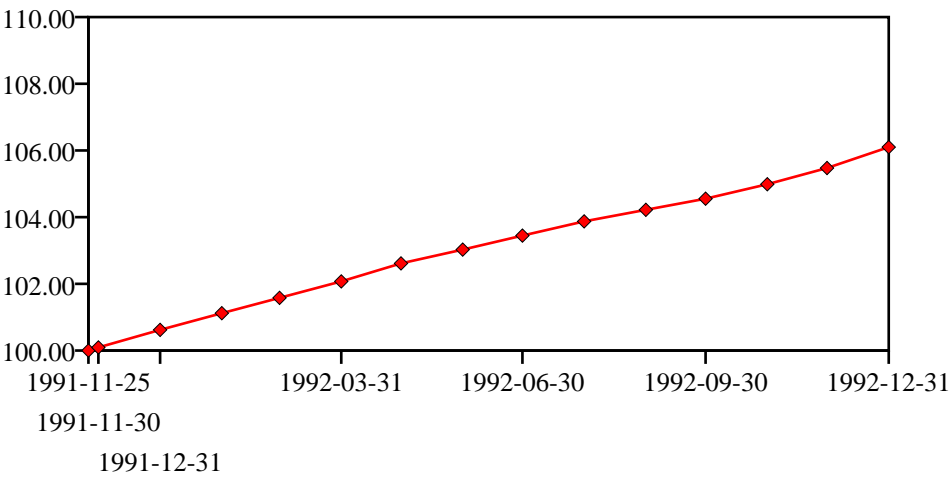
**sample2( ... )**

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

### LegendedPie(Pie)

Pie with a two part legend (one editable with swatches, one hidden without swatches).

## Public Attributes

**bottomPadding** Padding at bottom of drawing

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

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

**drawLegend** If true then create and draw legend

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

**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)

**pieAndLegend\_colors** Colours used for both swatches and pie

**rightPadding** Padding on right of drawing

**slices** collection of wedge descriptor objects

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

**topPadding** Padding at top of drawing

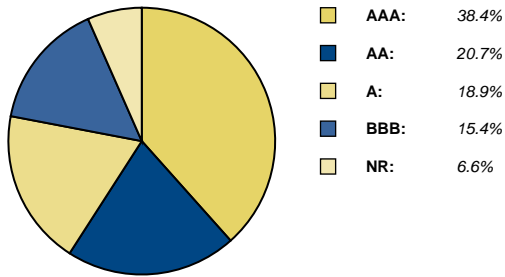
**width** width of pie bounding box. Need not be same as width.

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

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

### Example

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



### Properties of Example Widget

```

bottomPadding = 5
data = [38.399999999999999,
        20.699999999999999,
        18.899999999999999,
        15.4,
        6.5999999999999996]
direction = 'clockwise'
drawLegend = 1
height = 100
labels = None
leftPadding = 5
legend1.alignment = 'right'
legend1.autoXPadding = 5
legend1.autoYPadding = 2
legend1.colorNamePairs = [(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.dx = 5.6699999999999999
legend1.dxTextSpace = 11.3900000000000001
legend1.dy = 5.6699999999999999
legend1.fillColor = Color(0,0,0)
legend1.fontName = 'Helvetica-Bold'
legend1.fontSize = 6
legend1.strokeColor = Color(0,0,0)
legend1.strokeWidth = 0.5
legend1.x = 117
legend1.y = 100
legendNumberFormat = '%.1f%%'
legendNumberOffset = 51
legend_data = [38.399999999999999,
               20.699999999999999,
               18.899999999999999,
               15.4,
               6.5999999999999996]
legend_names = ['AAA:', 'AA:', 'A:', 'BBB:', 'NR:']
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)]
rightPadding = 5
slices = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x870530c>
startAngle = 90
topPadding = 5
width = 100
x = 0
y = 0

```

## Pie(Widget)

### Public Attributes

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

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

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

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

**slices** collection of wedge descriptor objects

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

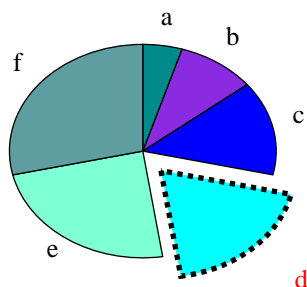
**width** width of pie bounding box. Need not be same as width.

**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)
    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

```
data = [1]
direction = 'clockwise'
height = 100
labels = None
slices = <reportlab.graphics.widgetbase.TypedPropertyCollection instance at 0x871c68c>
startAngle = 90
```

```
width = 100  
x = 0  
y = 0
```

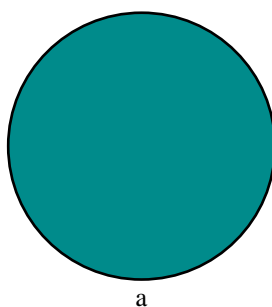
## Functions

### `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
```

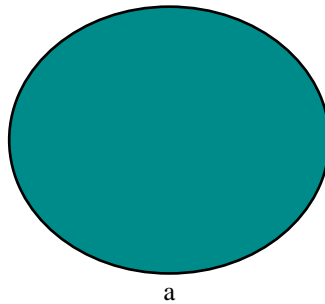


**sample0b( ... )**

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
```

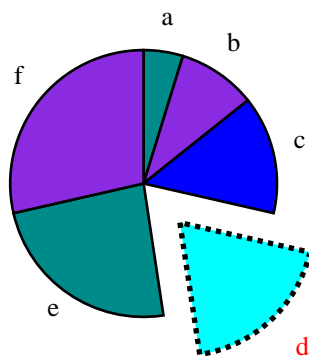


**sample1( ... )**

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
```



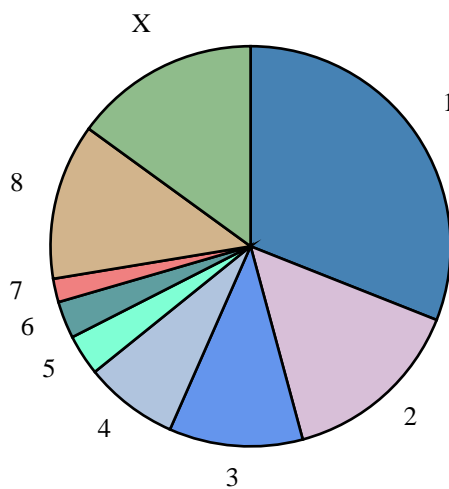


**sample2( ... )**

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
```

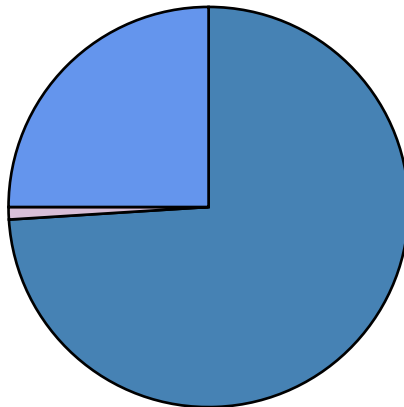


**sample3( ... )**

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
```

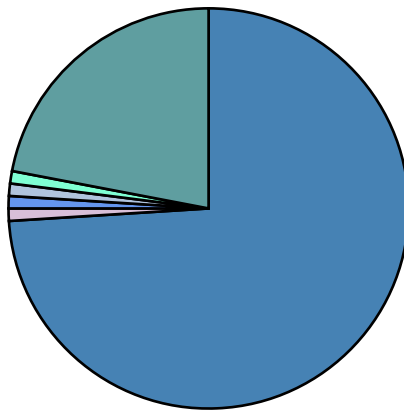


**sample4( ... )**

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
```



## textlabels

#copyright ReportLab Inc. 2000-2001

#see license.txt for license details

#history <http://cvs.sourceforge.net/cgi-bin/cvsweb.cgi/reportlab/graphics/charts/textlabels.py?cvsroot=reportlab>

#\$Header: /cvsroot/reportlab/reportlab/graphics/charts/textlabels.py,v 1.24 2002/03/26 11:49:10 rgbecker Exp \$

## Classes

### BarChartLabel (Label)

An extended Label allowing for nudging, lines visibility etc

## Public Attributes

**angle** None

**boxAnchor** None

**boxFillColor** None

**boxStrokeColor** None

**boxStrokeWidth** 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

**lineStrokeColor** Color for a drawn line

**lineStrokeWidth** Non-zero for a drawn line

**maxWidth** None

**nudge** Non-zero sign dependent nudge

**strokeColor** None

**strokeWidth** None

**text** None

**textAnchor** None

**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)
    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
boxAnchor = 'c'
boxFillColor = None
boxStrokeColor = None
boxStrokeWidth = 0.5
dx = 0
dy = 0
fillColor = Color(0,0,0)
fixedEnd = None
fixedStart = None
fontName = 'Times-Roman'
fontSize = 10
height = None
leading = None
lineStrokeColor = None
lineStrokeWidth = 0
maxWidth = None
nudge = 0
strokeColor = None
strokeWidth = 0.10000000000000001
textAnchor = 'start'
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

**boxAnchor** None

**boxFillColor** None

**boxStrokeColor** None

**boxStrokeWidth** None

**dx** None

**dy** None

**fillColor** None

**fontName** None

**fontSize** None

**height** None

**leading** None

**maxWidth** None

**strokeColor** None

**strokeWidth** None

**text** None

**textAnchor** None

**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)
    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
boxAnchor = 'c'
boxFillColor = None
boxStrokeColor = None
boxStrokeWidth = 0.5
dx = 0
dy = 0
fillColor = Color(0,0,0)
fontName = 'Times-Roman'
fontSize = 10
height = None
leading = None
maxWidth = None
strokeColor = None
strokeWidth = 0.10000000000000001
textAnchor = 'start'
visible = 1
width = None
x = 0
y = 0
```

## NA\_Label (BarChartLabel)

An extended Label allowing for nudging, lines visibility etc

### Public Attributes

**angle** None

**boxAnchor** None

**boxFillColor** None

**boxStrokeColor** None

**boxStrokeWidth** 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

**lineStrokeColor** Color for a drawn line

**lineStrokeWidth** Non-zero for a drawn line

**maxWidth** None

**nudge** Non-zero sign dependent nudge

**strokeColor** None

**strokeWidth** None

**text** Text to be used for N/A values

**textAnchor** None

**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)
    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
boxAnchor = 'c'
boxFillColor = None
boxStrokeColor = None
boxStrokeWidth = 0.5
dx = 0
dy = 0
fillColor = Color(0,0,0)
fixedEnd = None
fixedStart = None
fontName = 'Times-Roman'
fontSize = 10
height = None
leading = None
lineStrokeColor = None
lineStrokeWidth = 0
maxWidth = None
nudge = 0
strokeColor = None
strokeWidth = 0.10000000000000001
text = 'n/a'
textAnchor = 'start'
visible = 1
width = None
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

## 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

## **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

## **grids**

#copyright ReportLab Inc. 2000-2001  
#see license.txt for license details  
#history <http://cvs.sourceforge.net/cgi-bin/cvsweb.cgi/reportlab/graphics/widgets/grids.py?cvsroot=reportlab>  
#\$Header: /cvsroot/reportlab/reportlab/graphics/widgets/grids.py,v 1.22 2002/03/26 11:17:18 rgbecker Exp \$

## **Classes**

### **`DoubleGrid(Widget)`**

This combines two ordinary Grid objects orthogonal to each other.

## **Public Attributes**

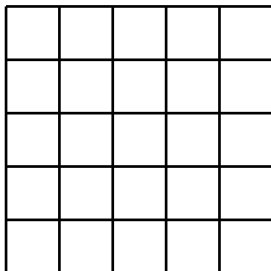
**grid0** The first grid component.  
**grid1** The second grid component.  
**height** The grid's height.  
**width** The grid's width.

**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.

**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.

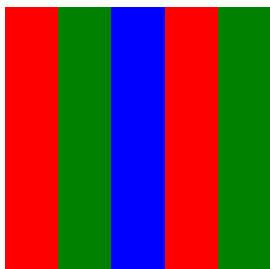
**width** The grid's width.

**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
```

## 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.

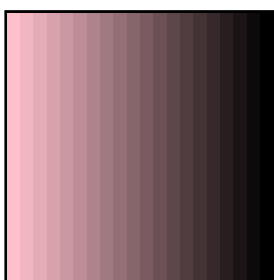
**width** The grid's width.

**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
```