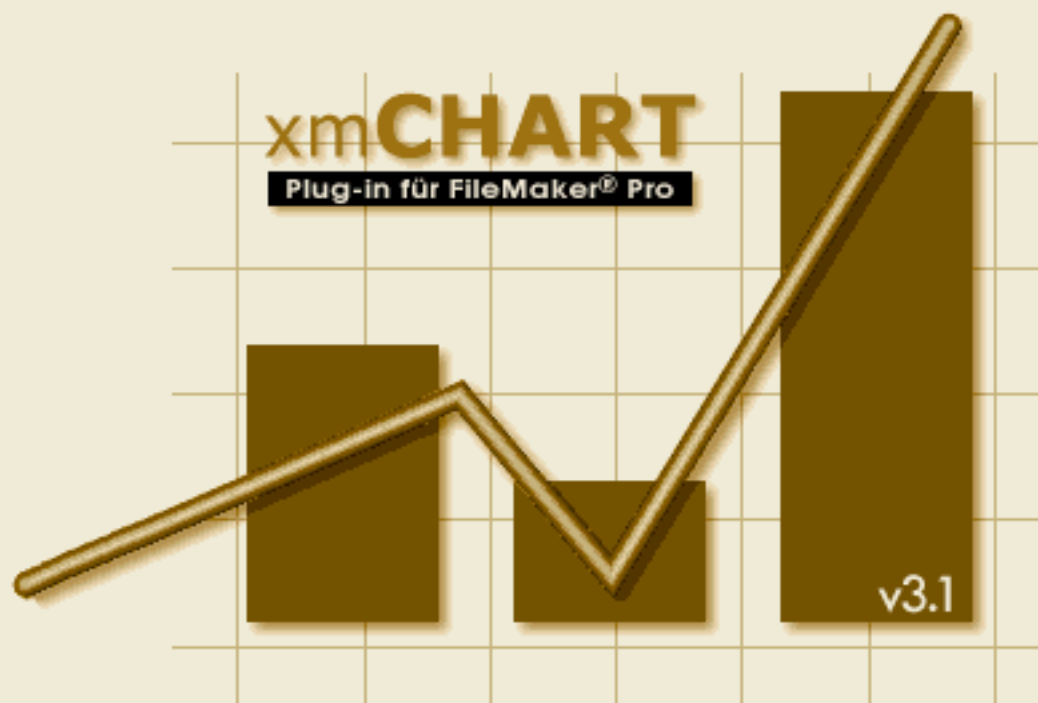


# Tutorial



*X2max*  
www.x2max.com

FileMaker Pro is a registered trademark of FileMaker, Inc.  
© 1997-2006 by X2max Software. All rights reserved.

# Inhaltsverzeichnis

## Einführung

<b>Voraussetzungen</b>	<b>8</b>
Betriebssystem	8
FileMaker Pro Version	8
FileMaker Pro Kenntnisse	8
<b>Installation</b>	<b>9</b>
Installation	9
Support	10
<b>Grundlagen</b>	<b>11</b>
Was ist xmCHART	11
Wie funktioniert xmCHART	11
Ein erstes Beispiel	13

## Komponenten

<b>Übersicht</b>	<b>18</b>
<b>Layout</b>	<b>19</b>
OpenDrawing()	19
CloseDrawing()	21
OpenView()	21
CloseView()	22
OpenChart()	22
CloseChart()	26

<b>Daten</b>	<b>27</b>
ChartData()	27
ChartDataOptions()	28
ChartDataUpperLimits()	30
ChartDataLowerLimits()	30
ChartDataRead()	32
ChartDataWrite()	33
Eingabe von Datum und Uhrzeit	35
<b>Diagramme</b>	<b>37</b>
ScatterChart()	37
ScatterChart2D()	40
LineChart()	42
LineChart2D()	45
AreaChart()	47
AreaChartOptions()	52
AreaChart2D()	54
BarChart()	58
BarChartOptions()	68
GanttChart()	70
BubbleChart()	75
BubbleChartOptions()	78
BubbleChart 2D()	78
PieChart()	80
PieChartExplodes()	85
PieChartExplodeDepths()	87
PieChartAuxLines()	88
PieChartLabelOptions()	89
PieChartInnerLabelTexts()	90
PieChartInnerLabelStyle()	90
PieChartInnerLabelBackground()	90
PieChartCenterLabelText()	91
PieChartCenterLabelStyle()	91
PieChartCenterLabelBackground()	91
PolarChart()	92
PolarChartOptions()	96
RadarChart()	100
RadarChartOptions()	104
HighLowChart()	108
CandlestickChart()	115
Histogram()	118
HistogramRange()	118
HistogramOptions()	121
BoxPlot()	126
BoxPlotOptions()	127

<b>Stile</b>	<b>134</b>
FillStyle()	134
PictureStyle()	136
BorderStyle()	138
LineStyle()	140
SymbolStyle()	142
ShadowStyle()	144
ArrowStyle()	146
LabelTexts()	148
LabelStyle()	151
LabelBackground()	156
LabelOptions()	157
 <b>Hintergrund</b>	 <b>169</b>
Background()	169
BackgroundPict()	173
ChartBackground()	177
ChartBackgroundPict()	179
 <b>Raster</b>	 <b>182</b>
MajorGridLineWidths()	182
MajorGridLineColors()	182
MajorGridLinePatterns()	182
MinorGridLineWidths()	184
MinorGridLineColors()	184
MinorGridLinePatterns()	184
MajorGridStripeColors()	187
MajorGridStripePatterns()	187
MinorGridStripeColors()	187
MinorGridStripePatterns()	187
GridFrame()	190
GridLocation()	191
 <b>Achsen</b>	 <b>193</b>
Scaling()	193
ScalingOptions()	202
AxisLine()	205
AxisMajorTicks()	205
AxisMinorTicks()	205
AxisOptions()	207
AxisLabelText()	209
AxisLabelStyle()	212
AxisLabelBackground()	213
AxisLabelOptions()	215
AxisMajorTickLabelTexts()	217

AxisMajorTickLabelStyle()	219
AxisMajorTickLabelBackground()	220
AxisMajorTickLabelOptions()	221
AxisMinorTickLabelTexts()	224
AxisMinorTickLabelStyle()	224
AxisMinorTickLabelBackground()	224
AxisMinorTickLabelOptions()	224
<b>Legende</b>	<b>228</b>
LegendTexts()	228
LegendStyle()	229
LegendBackground()	230
LegendOptions()	231
<b>Titel</b>	<b>239</b>
TitleText()	239
TitleStyle()	240
TitleSubStyle()	240
TitleBackground()	241
TitleOptions()	242
<b>Hilfslinien</b>	<b>245</b>
DropLineStyle()	245
DropLineReferencePoint()	247
DropLineReferenceLine()	249
DropLineReferenceSeries()	250
<b>Gleitende Durchschnitte</b>	<b>253</b>
MovingAverage()	253
MovingAverageLineStyle()	255
MovingAverageOptions()	257
<b>Kurvenanpassung</b>	<b>266</b>
CurveFitting()	267
CurveFittingLineStyle()	269
CurveFittingOptions()	271
<b>Fehlerbalken</b>	<b>275</b>
ErrorBars()	275
ErrorBarData()	277
ErrorBarStyle()	278
ErrorBarStyle2D()	280

<b>Grafische Basisfunktionen</b>	<b>283</b>
AddText()	284
AddLine()	287
AddFrame()	288
AddRect()	289
AddRoundFrame()	290
AddRoundRect()	291
AddEllipse()	292
AddOval()	293
AddArc()	294
AddSlice()	295
AddPolyline()	296
AddPolygon()	297
AddSmoothPolyline()	298
AddSmoothPolygon()	299
AddPath()	300
AddSymbol()	304
AddArrow()	306
AddPicture()	307
AddClipRect()	308
AddClipRoundRect()	308
AddClipOval()	308
AddClipSlice()	308
AddClipPolygon()	308
AddClipSmoothPolygon()	308
AddClipReset()	309
 <b>Ausgabe</b>	 <b>310</b>
SendToClipboard()	310
SaveAsPDFFile()	311
SaveAsEMFFile()	312
SaveAsPICTFile()	312
SaveAsGIFFile()	313
SaveAsJPGFile()	313
SaveAsPNGFile()	314
SaveAsBMPFile()	314
SaveAsTIFFFile()	314
 <b>Diverses</b>	 <b>315</b>
DateTimeOptions()	315
SetDecimalPoint()	316
SetThousandsSep()	316
 <b>Index</b>	 <b>318</b>

# Einführung

# Voraussetzungen

**Betriebssystem:**

- Windows XP oder höher
- Mac OS X 10.3 oder höher

**FileMaker Pro Version:**

Version: 7.0/8.0/8.5

(Für FileMaker Pro 4/5/5.5/6 wird xmCHART 2.2 benötigt.)

**FileMaker Pro Kenntnisse:**

Um mit xmCHART erfolgreich arbeiten zu können, sind folgende FileMaker Pro-Kenntnisse notwendig:

- Definieren von FileMaker Pro Feldern, im speziellen, Variablen.
- Erstellen und Bearbeiten von Layouts.
- Handhabung von externen Funktionen. Über externe Funktionen werden Plug-ins in FileMaker Pro Datenbanken eingebunden.
- Kenntnisse in FileMaker Pro Scripting. Mittels Scripts kann die Erstellung von Diagrammen vollkommen automatisiert werden.

Darauf aufbauend wird in den nachfolgenden Abschnitten, auf sämtliche xmCHART betreffenden Kenntnisse und Besonderheiten detailliert eingegangen. Zusätzliche Ideen, Anregungen und Hilfestellungen finden sich in *xmBeispiele* inkl. mitgelieferter FileMaker Pro Beispieldateien.

# Installation

**Installation:**

- Falls notwendig, beenden Sie FileMaker Pro.
- Kopieren Sie xmCHART (Plug-in) in den Ordner "Extensions", welcher sich normalerweise im "FileMaker Pro"-Ordner befindet.
- Starten Sie FileMaker Pro.

Weiters muß sichergestellt sein, dass xmCHART aktiviert ist. Dies kann unter dem Menüpunkt *"Einstellungen...> Plugins"* überprüft werden.

xmCHART kann auch problemlos in FileMaker Pro Runtime Lösungen eingebunden werden. Zu diesem Zweck erstellen Sie innerhalb des Ordners, welcher die FileMaker Pro Dateien für Ihre Runtime Lösung enthält, einen Ordner mit dem Namen "Extensions" und kopieren xmCHART (Plug-in) in diesen Ordner.

**Wichtig:**

**Um Diagramme in einem FileMaker Pro Medienfeld verzerrungsfrei in der richtigen Größe darzustellen, sind die folgenden zwei Punkte zu beachten:**

- (1) Die Größe des Medienfeldes und die Größe des Diagramms müssen identisch sein. Die Breite und die Höhe des Diagramms werden durch die Funktion `OpenDrawing(breite;höhe)` festgelegt (in Pixel). Die Breite und Höhe des Medienfeldes können im Layout Modus unter dem Menüpunkt *Ansicht>Objektgröße* kontrolliert werden. Durch Mausklick auf eine der Dimensionsangaben [cm, px, in] am rechten Rand des "Größe"-Fensters kann zwischen Zentimeter, Pixel und Inch hin- und hergeschaltet werden.
- (2) Im Layout Modus ist unter dem Menüpunkt *Format>Grafik...*, "Bild verkleinern/vergrößern" einzustellen, und "Proportionen beibehalten" zu deaktivieren. Falls "Proportionen beibehalten" aktiviert ist, verschwinden Diagramme oder erscheinen spiegelverkehrt.

**Support:**

Sollten Probleme bei der Installation auftreten, bitte senden Sie eine entsprechende e-mail mit den folgenden Informationen:

- Plattform und Betriebssystemversion
- FileMaker Pro Version
- Fehlerbeschreibung

an: <support@x2max.com>. Wir bemühen uns, möglichst rasch mit Ihnen Kontakt aufzunehmen und das Problem zu lösen.

# Grundlagen

## Was ist xmCHART

xmCHART ist ein sehr mächtiges FileMaker Pro Plug-in zur grafischen Darstellung numerischer Daten. Dabei werden alle wichtigen Diagrammarten, wie Balken-, Punkte-, Linien-, Flächen-, HochTief- und Kuchen-diagramme unterstützt. Zusätzlich werden auch statistische und finanztechnische Diagramme sowie Polar-, Radar- und Gantt-diagramme zur Verfügung gestellt. Mittels eines umfangreichen Funktionensatzes können alle wichtigen Diagrammkomponenten wie etwa Skalierungen, Achsen, Raster oder Beschriftungen vom Benutzer variiert werden. Des weiteren können zur Vervollständigung eines Diagramms, Titel, Legende, Hintergrundbild, ergänzende Texte oder Bildinformationen, wie etwa ein Logo, einfach hinzugefügt werden.

xmCHART erschließt somit, sowohl dem FileMaker Pro Entwickler als auch dem Anwender eine Vielzahl neuer Möglichkeiten zur Auswertung und Darstellung numerischer Daten.

## Wie funktioniert xmCHART

Die xmCHART zugrundeliegende Idee besteht darin, dem Anwender einen umfangreichen Satz an Funktionen zur Verfügung zu stellen, mit welchem das Erscheinungsbild eines Diagramms detailliert kontrolliert werden kann. Dabei reichen bereits drei Funktionen aus, um eine Vielzahl unterschiedlicher Diagramme zu erstellen; mit der ersten Funktion `OpenDrawing()` wird die Größe des Diagramms festgelegt, die zweite Funktion `ChartData()` enthält die darzustellenden Daten und die dritte Funktion legt die Art des Diagramms fest. Zum Beispiel:

```
OpenDrawing(400;300) // Breite & Höhe  
ChartData(23 45 -2,76 12.9 5; 24 13 8.5 -3)  
BarChart(shadow+horizontal)
```

Tatsächlich stehen aber derzeit über 150 Funktionen zur Verfügung, mit welchen beinahe alle Diagrammteile variiert werden können. Sämtliche Funktionsaufrufe werden entweder manuell oder automatisch mittels eines FileMaker Pro Scripts, zeilenweise, das heißt eine Funktion pro Zeile, in ein FileMaker Pro Textfeld eingetragen. Die Vorteile dieses Vorgangsweise sind:

- **Übersichtlichkeit**

Alle zur Erstellung eines Diagramms benötigten Funktionsaufrufe sind übersichtlich, in einem einzigen Textfeld zusammengefaßt und können noch zusätzlich durch erklärende Kommentare ergänzt werden.

- **Einfaches Entwickeln und Testen**

Änderungen können unmittelbar — ohne Aufruf des Script Editors — ausprobiert werden. Eventuell auftretende Fehler, werden durch Angabe der Zeilennummer, des Funktionsnamens und des Argumentindex genau spezifiziert.

TIPP: *Nicht benötigte Funktionen können im Entwicklungs- und Teststadium einfach deaktiviert werden — entweder durch Vorsetzen eines Kommentarzeichens "//" oder bei mehrzeiligen Scriptpassagen durch umschließen mit /\* ... \*/.*

- **Portierbarkeit und Support**

Funktionsaufrufe können einfach mittels der üblichen Editierbefehle *Kopieren* und *Einsetzen* in andere FileMaker Pro Datenbanken übernommen werden oder auch per e-mail versendet werden, zum Beispiel bei Fragen an den technischen Support:

`<support@x2max.com>`

Soll nun ein Diagramm erstellt werden, wird der Inhalt dieses Textfeldes, welches im allgemeinen aus einer Anzahl von Funktionsaufrufen und optionalen Kommentarzeilen besteht, mittels eines kurzen FileMaker Pro Scripts an xmCHART übergeben. xmCHART überprüft anschließend diesen Textstring auf mögliche Tippfehler, fehlende Klammern, ungültige Werte, etc. Im Falle eines Fehlers wird abgebrochen und von xmCHART an FileMaker Pro eine Fehlermitteilung geliefert. Um dem Anwender ein rasches und einfaches Korrigieren zu ermöglichen, liefert xmCHART genaue Fehlermeldungen, mit Zeilennummer, und wenn möglich, mit Funktionsnamen und Argumentindex.

Werden keine Fehler gefunden, beginnt xmCHART mit dem Aufbau des Diagramms; dies geschieht für den Anwender unsichtbar im Hintergrund. Die Reihenfolge der Funktionsaufrufe hat in der Regel weder einen Einfluss auf die Geschwindigkeit noch auf das Aussehen des Diagramms.

Eventuell im Zuge des Diagrammaufbaus auftretende Fehler, wie etwa eine zu klein vorgegebene Diagrammgröße, führen ebenfalls zum Abbruch und zu einer entsprechenden Fehlermeldung. Tritt kein Fehler auf, wird abschließend das aufgebaute Diagramm von xmCHART in ein FileMaker Pro Medienfeld kopiert.

Die Funktionsweise von xmCHART läßt sich also zusammengefaßt in folgende drei Bereiche gliedern:

- Überprüfung der eingegebenen Funktionsaufrufe.
- Aufbau des Diagramms *offscreen*, d.h. für den Anwender unsichtbar im Hintergrund.
- Kopieren des erstellten Diagramms in ein FileMaker Pro Medienfeld.

## Ein erstes Beispiel

Im folgenden, einfachen Beispiel werden fünf Zahlenwerte durch ein Balkendiagramm grafisch dargestellt. Dabei sollen die Balken beschriftet und mit Schatten ausgeführt werden, weiters soll ein Diagrammtitel hinzugefügt werden. Die dafür notwendigen Schritte können in drei Aufgabenbereiche gegliedert werden.

### • FileMaker Pro Felder definieren

Im allgemeinen benötigt xmCHART drei FileMaker Pro Felder; zweckmäßigerweise sind das globale Felder, welche im folgenden durch das Präfix "g" (global) gekennzeichnet werden.

Erstens, ein Textfeld, welches alle Funktionsaufrufe und optionale Kommentarzeilen enthält, im folgenden als *gFunktionen* bezeichnet. Zweitens, ein Medienfeld, welches das von xmCHART erstellte Diagramm übernimmt, im folgenden als *gDiagramm* bezeichnet. Und drittens, ein weiteres Textfeld *gFehler*, in welches eventuell auftretende Fehlermeldungen eingetragen werden. (Abb. 1)

<u>Feldname</u>	<u>Feldtyp</u>	<u>Optionen</u>
gFunktionen	Text	Global
gDiagramm	Medien	Global
gFehler	Text	Global

Abb. 1

**TIPP:** *Anstelle eines Textfeldes und Medienfeldes kann auch ein Formelfeld verwendet werden. Als Ergebnis liefert das Formelfeld dann das Diagramm. Der Vorteil eines*

*Formelfeldes besteht darin, dass zur Erstellung des Diagramms kein FileMaker Pro Script aufgerufen werden muss, oder anders gesagt, das Diagramm wird automatisch aktualisiert sobald sich Werte im Formelfeld ändern, was sich in vielen Fällen als äusserst vorteilhaft erweisen kann.*

#### • FileMaker Pro Scripts erstellen

In der Regel ist ein kurzes FileMaker Pro Script notwendig, welches den Inhalt von *gFunktionen* mittels der externen Funktion `xmCH_DrawChart()` an `xmCHART` übergibt. (Abb. 2)

```
Feldwert setzen [ xmCHART::gDiagramm;  
                    xmCH_DrawChart(xmCHART::gFunktionen) ]  
Feldwert setzen [ xmCHART::gFehler;xmCH_GetErrorMessage("103") ]
```

Abb. 2

Kein FileMaker Pro Script ist notwendig wenn anstelle eines Textfeldes und Medienfeldes ein Formelfeld verwendet wird.

Im Falle eines Fehlers wird in *gFehler* eine entsprechende Mitteilung gespeichert; tritt kein Fehler auf, so wird *gFehler* ein leerer Text ("" ) zugewiesen.

Weitere Scripts sind notwendig, wenn `xmCHART`-Funktionen dynamisch aufgebaut werden sollen, d.h. der Inhalt des Textfeldes *gFunktionen* wird mittels Scriptbefehle automatisch erstellt. Dies ist üblicherweise zur Erstellung der Diagrammdaten notwendig. Auf diese äußerst vielseitige und mächtige Handhabung von `xmCHART` wird in *xmBeispiele* ausführlich eingegangen.

#### • xmCHART Funktionen

Die zum Aufbau und zur Gestaltung eines Diagramms benötigten Funktionen werden in das Textfeld *gFunktionen* eingetragen. Dabei sind drei Funktionsaufrufe unbedingt erforderlich, nämlich erstens die Funktion `OpenDrawing()`, welche die Größe des Diagramms festlegt, zweitens `ChartData()`, welche alle zur Darstellung verwendeten numerischen Werte enthält und drittens eine Funktion, welche die Art des Diagramms festlegt, z.B. `BarChart()`. Zur Zeit stehen folgenden Diagrammarten zur Verfügung: Balken-, Punkte-, Linien-, Flächen-, HochTief- und Kuchendiagramme, sowie statistische Diagramme (Histogramme, Box-Plots) und finanztechnische Diagramme (Candlestick-Diagramme, High-Low-

Close-Open-Diagramme), weiters Polar-, Radar- und Gantt-diagramme.

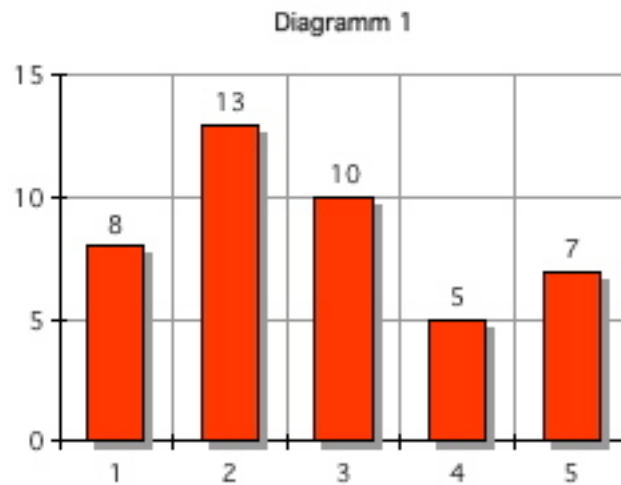
Bei der Eingabe der Funktionen sind folgende Regeln zu beachten: *(Eine ausführlichere Darstellung findet sich in xmReferenz, Abschnitt "Syntax".)*

- Pro Zeile nur ein Funktionsaufruf; Leerzeilen sind erlaubt.
- Kommentare werden entweder durch Voranstellen zweier Schrägstriche "//" oder durch umschließen mit /\* ... \*/ gekennzeichnet.
- Besitzt eine Funktion mehrere Argumente, so werden diese durch Strichpunkte (",") getrennt.
- Optionale, d.h. nicht unbedingt erforderliche Argumente, können bei der Eingabe übersprungen werden. In diesem Fall werden die in xmCHART gespeicherten Standardwerte verwendet. Diese können aus *xmReferenz* entnommen werden. Zum Beispiel:  
`LegendBackground(white;;2;;;3)`
- Texte und Namen von Schriften, z.B. "Times" sind unter Hochkomma (") zu setzen. Soll ein Hochkomma ausgegeben werden, so ist dieses *doppelt* einzugeben. Zum Beispiel:  
`TitleText(" "A"BC" " ")` bewirkt die Ausgabe: "A"BC"

**ACHTUNG:** *Hochkommas (") sind nicht mit typografischen Anführungszeichen (") zu verwechseln. Typografische Anführungszeichen können unter dem Menü "Ablage>Dateioptionen...>Text" aktiviert bzw. deaktiviert werden.*

Zur Erstellung des eingangs beschriebenen Diagramms sind die folgenden vier Funktionen erforderlichen: (Abb. 3)

```
OpenDrawing(250;200)
ChartData(8 13 10 5 7)
BarChart(label+shadow)
TitleText("Diagramm 1")
```



Dazu noch folgende Anmerkungen:

- `OpenDrawing(breite;höhe)`  
breite: gesamte Breite des Diagramms inkl. Achsen, Legende, Titel, etc. (in Pixels)  
höhe: Die gesamte Höhe des Diagramms inkl. Achsen, Legende, Titel, etc. (in Pixels)
- `ChartData()`  
Die zur Darstellung kommenden numerischen Werte werden mittels der Funktion `ChartData()` an `xmCHART` übergeben. Die einzelnen Zahlenwerte einer Serie werden durch Leerzeichen getrennt. Bei der Übergabe mehrerer Serien sind diese durch Strichpunkte zu trennen; dabei kann die Anzahl der Werte pro Serie unterschiedlich sein. Zum Beispiel:  
`ChartData(78 -12; 45 7 -23; 0; 12 -34 78 23,5)`
- `BarChart()`  
Durch Eingabe sog. *Gestaltungskonstanten* kann das Aussehen eines Diagramms variiert werden. Die Gestaltungskonstanten können durch ein Pluszeichen "+" kombiniert werden. Welche Konstanten für welche Diagrammart zur Verfügung stehen, kann *xmReferenz* entnommen werden.

# Komponenten

# Komponenten

## Übersicht

Ein mit xmCHART erstelltes Diagramm besteht in der Regel aus einer Vielzahl von Einzelkomponenten, wie z.B. Achsen, Raster, Legende, Titel oder dem Diagramm selbst. Nachfolgend werden alle in xmCHART zur Verfügung stehenden Funktionen, gruppiert nach Komponenten, zusammen mit zahlreichen Beispielen erklärt:

- Layout
- Daten
- Diagramme
- Stile
- Hintergrund
- Raster
- Achsen
- Legende
- Titel
- Hilfslinien
- Gleitende Durchschnitte
- Kurvenanpassung
- Fehlerbalken
- Grafische Basisfunktionen
- Ausgabe
- Diverses

## Layout

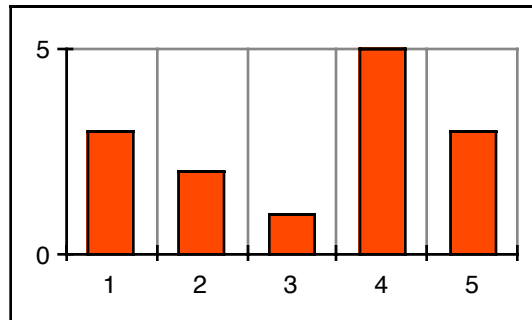
Insgesamt stehen sechs Layout-Funktionen zur Verfügung. Alle Funktionen treten paarweise in Form von *Open*- und *Close*-Funktionen auf. Sie dienen zum:

- Festlegen der Zeichnungsgröße.
- Präzises Positionieren von Diagrammen.
- Überlagern von Diagrammen.
- Strukturieren komplexer Zeichnungen und Diagramme.

### **OpenDrawing(breite;höhe;typ;glättung)**

Die wichtigste Layout-Funktion ist `OpenDrawing()`. Diese definiert die Breite und Höhe einer Zeichnung (in Pixels) und ist stets als erster Funktionsaufruf anzuführen. Zum Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 2 1 5 3)
  BarChart()
CloseDrawing()
```



Optional kann als 3. Argument noch festgelegt werden, ob eine Zeichnung im Vektorformat (*typ=0*) oder im Bitmapformat (*typ=1*) erstellt werden soll. Fehlt das 3. Argument, so wird standardmäßig die Zeichnung im Vektorformat erstellt. Das heißt, unter Mac OS X im PDF-Format und unter Windows im EMF-Format. Das Vektorformat garantiert hohe Druckqualität, das Bitmapformat erweist sich zweckmäßig als plattformübergreifendes Format.

Mit dem optionalen 4. Argument kann die Glättung (antialiasing) von Schriften und geometrischen Objekten kontrolliert werden. Dazu stehen die folgenden vier Möglichkeiten zur Verfügung:

- 0: keine Glättung
- 1: geometrischen Objekte glätten
- 2: Schriften glätten
- 3: geometrischen Objekte und Schriften glätten (default)

Anmerkungen:

- Um Zeichnungen in einem FileMaker Pro Medienfeld richtig darzustellen, ist es notwendig, im Layout Modus, Menüpunkt *Format>Grafik...*, "Bild verkleinern/vergrößern" einzustellen, und "Proportionen beibehalten" zu deaktivieren.
- Zuzufolge eines Fehlers in FileMaker Pro 7 funktioniert die Glättung von Texten und geometrischen Objekten unter Mac OS X nur im Bitmapformat (*typ=1*) zuverlässig. Dieser Fehler wurde ab FileMaker Pro 8 korrigiert.

Beispiel für Vektorformat:

```
OpenDrawing(200;100;0) // Vektorformat (default)
  ChartData(3 2 1 5 3)
  BarChart()
CloseDrawing()
```

Beispiel für Bitmapformat:

```
OpenDrawing(200;100;1) // Bitmapformat
  ChartData(3 2 1 5 3)
  BarChart()
CloseDrawing()
```

Der Unterschied zwischen Vektor- und Bitmapformat kann durch Vergrößern der Zeichnung überprüft werden. Dies kann einfach durch den bei FileMaker Pro links unten platzierten Zoom-Button erfolgen.



Beispiele:

```
OpenDrawing(200;100;0;2) // nur Schriften glätten
  ChartData(3 2 1 5 3)
  BarChart()
CloseDrawing()
```

```
OpenDrawing(200;100;1;3) // Bitmap + Antialiasing
  ChartData(3 2 1 5 3)
  BarChart()
CloseDrawing()
```

**CloseDrawing()**

CloseDrawing() besitzt keine Argumente und kann optional als letzter Funktionsaufruf angeführt werden.

Ausgabefunktionen, wie zum Beispiel SaveAsJPGFile() können auch außerhalb der Funktionen OpenDrawing() und CloseDrawing() angeführt werden.

**OpenView(links;oben;breite;höhe)**

Views ermöglichen die Unterteilung einer Zeichnung in mehrere Teilbereiche. So kann z.B. eine Serie von Grafikbefehlen, wie Linien, Flächen oder Texte innerhalb eines Views zusammengefaßt werden; durch einfaches Verändern der Position des Views, werden alle Grafikkomponenten innerhalb des Views automatisch mitverschoben. Andernfalls müßten die Koordinaten jedes einzelnen Grafikbefehls geändert werden. Das heißt, alle Koordinatenangaben innerhalb eines Views beziehen sich immer auf den View-Ursprung Links/Oben. Zum Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenView(80;40;100;70)
    AddFrame(0;0;100;70)
    ChartData(3 2 1 5 3)
    BarChart()
  CloseView()
CloseDrawing()
```

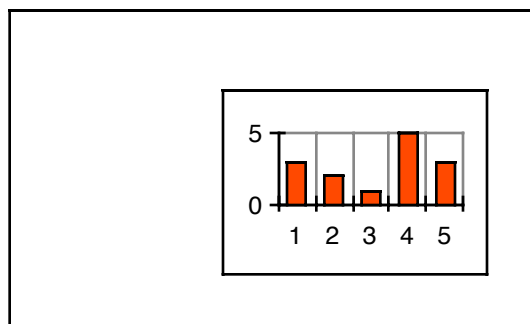


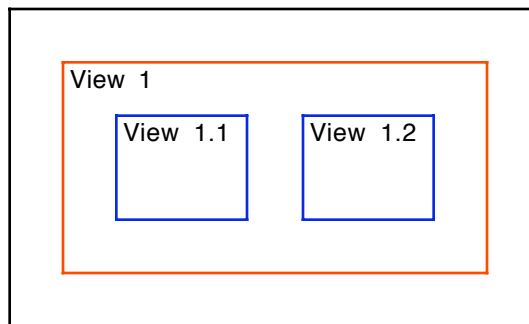
Diagramme und Grafikelemente, welche teilweise oder ganz außerhalb eines Views platziert sind, werden abgeschnitten (*geclippt*). Verschachtelte Views, d.h. Views innerhalb eines Views sind ebenfalls möglich.

Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenView(20;20;160;80)
    AddFrame(0;0;160;80;1;red)
    AddText(3;10;"View 1")
    OpenView(20;20;50;40)
      AddFrame(0;0;50;40;1;blue)
      AddText(3;10;"View 1.1")
    CloseView()
    OpenView(90;20;50;40)
      AddFrame(0;0;50;40;1;blue)
      AddText(3;10;"View 1.2")
    CloseView()
  CloseView()
CloseDrawing()

```



### **CloseView()**

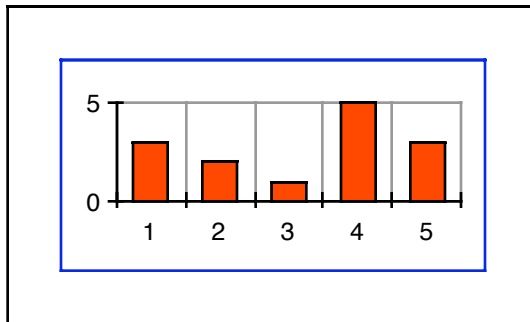
`CloseView()` besitzt keine Argumente und schließt die Befehlsfolge innerhalb eines Views ab. Wird nur ein einziger View definiert, kann auf `CloseView()` verzichtet werden.

### **OpenChart(links;oben;breite;höhe;istDiagrammfläche)**

`OpenChart()` ermöglicht das präzise Positionieren eines Diagramms innerhalb einer Zeichnung oder eines Views. Dabei kann optional durch das fünfte Argument *istDiagrammfläche* entweder die gesamte — alle Diagrammkomponenten inkl. Achsenbeschriftungen, Titel und Legende umfassende — Fläche festgelegt werden (*istDiagrammfläche=off*) oder nur die eigentliche Diagrammfläche, d.h. die vom Raster umschlossene Fläche (*istDiagrammfläche=on*).

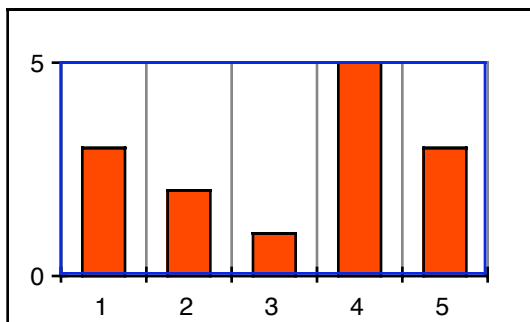
Beispiel für *istDiagrammfläche=off*: (default)

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
OpenChart(20;20;160;80;off)
  ChartData(3 2 1 5 3)
  BarChart()
  AddFrame(20;20;160;80;1;blue)
CloseChart()
CloseDrawing()
```



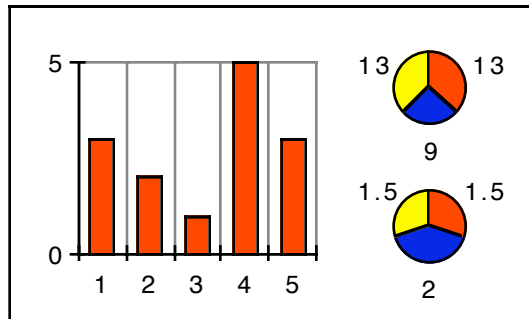
Beispiel für *istDiagrammfläche=on*:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
OpenChart(20;20;160;80;on)
  ChartData(3 2 1 5 3)
  BarChart()
  AddFrame(20;20;160;80;1;blue)
CloseChart()
CloseDrawing()
```



Das Argument *istDiagrammfläche=off* kann zur Platzierung mehrerer Diagramme innerhalb einer Zeichnung verwendet werden. Beispiel:

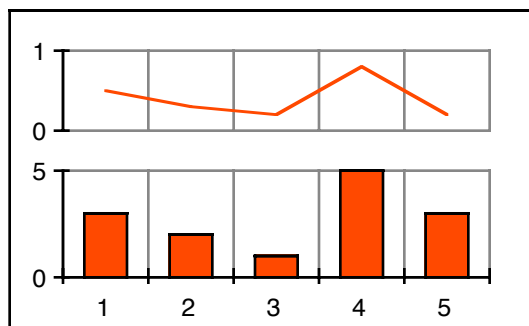
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenChart(5;5;120;115;off)
    ChartData(3 2 1 5 3)
    BarChart()
  CloseChart()
  OpenChart(120;0;75;75;off)
    ChartData(13 9 13)
    PieChart(label)
  CloseChart()
  OpenChart(120;50;75;75;off)
    ChartData(1.5 2 1.5)
    PieChart(label)
  CloseChart()
CloseDrawing()
```



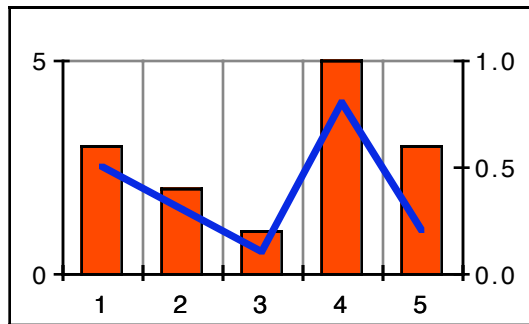
Das Argument *istDiagrammfläche=on* kann zum präzisen Aneinanderreihen oder Überlagern mehrerer Diagramme verwendet werden.

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenChart(20;15;160;30;on)
    ChartData(0.5 0.3 0.2 0.8 0.2)
    LineChart(;on)
    AxisOptions(x;none) // x-Achse ausblenden
  CloseChart()
  OpenChart(20;60;160;40;on)
    ChartData(3 2 1 5 3)
    BarChart()
  CloseChart()
CloseDrawing()
```



```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenChart(20;20;150;80;on)
    ChartData(3 2 1 5 3)
    BarChart()
  CloseChart()
  OpenChart(20;20;150;80;on)
    ChartData(0.5 0.3 0.1 0.8 0.2)
    LineChart(;on)
    LineStyle(1;poly;2;blue)
    AxisOptions(y;;on) // y-Achse rechts
    GridLocation(xy;none) // kein Raster
  CloseChart()
CloseDrawing()
```

**CloseChart()**

CloseChart() besitzt keine Argumente und schließt die Befehlsfolge innerhalb eines Diagramms ab. Falls nur ein einziges Diagramm definiert wird, kann auf CloseChart() verzichtet werden.

## Daten

Zur Eingabe und Verwaltung der Diagrammdaten stehen sechs Funktionen zur Verfügung. Sie dienen:

- zur Eingabe der Diagrammdaten
- zum Import und Export der Diagrammdaten
- zur Behandlung ungültiger oder nicht vorhandener Werte

### **ChartData(*serie1*;*serie2*...)**

Die Funktion `ChartData()` dient zur Übergabe der Diagrammwerte an `xmCHART`. Die Zahlen werden üblicherweise mittels FileMaker Pro Scripts aus den Datenbankfeldern in die Funktion `ChartData()` kopiert. In der Beispielsammlung *xmBeispiele* wird darauf detailliert eingegangen. Anstelle numerischer Daten können auch Datumswerte und Uhrzeiten an `xmCHART` übergeben werden. Dies wird nachfolgend im Kapitel *Eingabe von Datum und Uhrzeit* erläutert.

Jede Serie besteht aus einer Folge von Zahlen, welche durch Leerzeichen, Tabs oder Zeilenumbrüche getrennt sind. Zum Beispiel:

```
ChartData(78 -12; 45 7 -23; 0; 12 -34 78 23)
```

Dezimalzahlen können sowohl durch Dezimalpunkt als auch Dezimalkomma dargestellt werden. Beispiel:

```
ChartData(-10 2.31 0 -,69 0.01 -8,00 .0)
```

Zahlen können auch im E-Format eingegeben werden. Zum Beispiel:

```
ChartData(-1.2e04 0.2E04 .2e-3)
```

Fehlende Werte können durch NULL gekennzeichnet werden. Beispiel:

```
ChartData(12 98.3 null 8 Null NULL 7.23 -0.67)
```

Tausender-Trennzeichen sind nicht erlaubt. Beispiel:

```
ChartData(1,234.56 12.345,67) // nicht erlaubt!
```

Werden mehrere Werteserien an `ChartData()` übergeben, so sind diese durch einen Strichpunkt ";" zu trennen. Die Bedeutung der einzelnen Werteserien in `ChartData()` ist abhängig vom darzustellenden Diagramm. Bei den eindimensionalen Diagrammfunktionen `AreaChart()`, `BarChart()`, `BoxPlot()`, `Histogram()`, `LineChart()`, `RadarChart()` und `ScatterChart()` entspricht eine Werteserie in `ChartData()` genau einer Diagrammserie. Bei den zweidimensionalen Diagrammfunktionen `LineChart2D()`, `ScatterChart2D()`, `Area-`

Chart2D() und PolarChart() enthält die erste Werteserie in ChartData() die x-Werte, die zweite Werteserie die y-Werte der 1. Diagrammserie, die 3. und 4. Werteserie die x- und y-Werte der 2. Diagrammserie usw. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 19 10 15 13 6 2; // 1.Serie
            18 12 19 24 15 15 6) // 2.Serie
  LineChart(symbol)
  SymbolStyle(all;bullet;4)
CloseDrawing()

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 19 10 15 13 6 2; // x-Werte
            18 12 19 24 15 15 6) // y-Werte
  LineChart2D(symbol)
  SymbolStyle(all;bullet;4)
CloseDrawing()
```

Spezielle Zuordnungen zwischen Werteserien und Diagrammserien, wie zum Beispiel bei Gantt-, Hoch/Tief- oder Blasendiagrammen, werden im Abschnitt *Diagramme* erläutert.

### ChartDataOptions(koordReihenfolge)

Standardmäßig werden die Werte einer Diagrammserie hintereinander eingegeben (*koordReihenfolge=xyxy*). Beispiel:

```
ChartData(2 3 -10 5 2; // Serie 1
          3 9 12 -3 11) // Serie 2
```

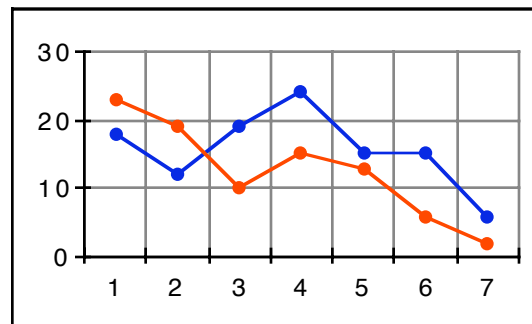
Bisweilen erweist es sich jedoch vorteilhaft, die Daten in transponierter Form einzugeben, d.h. die Zeilen und Spalten sind vertauscht angeordnet (*koordReihenfolge=xyxy*). Beispiel:

```
ChartData(2 3;
          3 9;
          -10 12;
          5 -3;
          2 11)
```

So können, zum Beispiel in einem FileMaker Pro Script, durch eine einzige Schleife gleichzeitig mehrere Werteserien aus einer Folge von Datensätzen in transponierter Form an ChartData() übergeben werden. Andernfalls, nicht transponiert, wird für jede Werteserie eine eigene Schleife über die erforderlichen FileMaker Pro Datensätze benötigt. Zu

beachten ist, dass die Funktion `ChartDataOptions()` vor der Funktion `ChartData()` anzuführen ist. Beispiele:

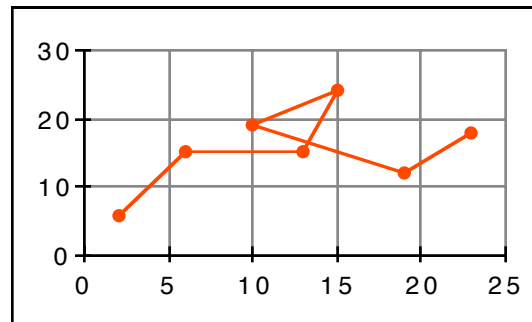
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartDataOptions(xyxy) // vor ChartData()!
  ChartData(23 18;
            19 12;
            10 19;
            15 24;
            13 15;
            6 15;
            2 6)
  LineChart(symbol;on)
  SymbolStyle(1;bullet;4)
  SymbolStyle(2;bullet;4)
CloseDrawing()
```



Die folgenden zwei Scripts führen zum gleichen Ergebnis:

```
// Eingabedaten nicht transponiert (default)
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 19 10 15 13 6 2; // x-Werte
            18 12 19 24 15 15 6) // y-Werte
  LineChart2D(symbol)
  SymbolStyle(1;bullet;4)
CloseDrawing()
```

```
// Eingabedaten transponiert
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartDataOptions(xyxy) // vor ChartData(!)
ChartData(23 18;
          19 12;
          10 19;
          15 24;
          13 15;
          6 15;
          2 6)
LineChart2D(symbol)
SymbolStyle(1;bullet;4)
CloseDrawing()
```



**ChartDataUpperLimits(maxWert1;maxWert2...)**

**ChartDataLowerLimits(minWert1;minWert2...)**

Die beiden Funktionen ChartDataUpperLimits() und ChartDataLowerLimits() ermöglichen zu jeder Werteserie einen oberen und unteren Grenzwert zu definieren, d.h. alle Werte, welche kleiner als *minWert* bzw. größer als *maxWert* sind, werden nicht dargestellt. Eine mögliche Anwendung ist die Darstellung unvollständiger Werteserien. Fehlende Daten, wie z.B. fehlende Meßwerte, werden durch "ungültige" Werte ergänzt, d.h. durch Werte, welche kleiner als *minWert* oder größer als *maxWert* sind.

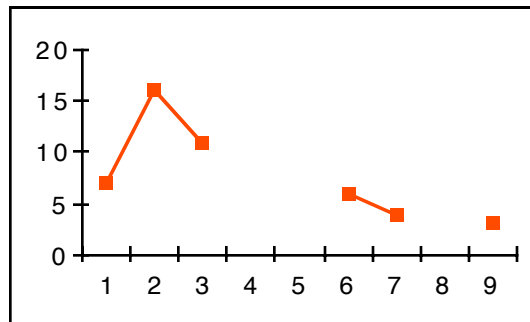
Zu beachten ist, dass die Funktionen ChartDataUpperLimits() und ChartDataLowerLimits() *nach* der Funktion ChartData() anzuführen sind.

Eine andere und meist einfachere Methode zur Übergabe unvollständiger Werteserien besteht darin, die fehlenden Werte in ChartData() durch NULL zu ersetzen. Zum Beispiel:

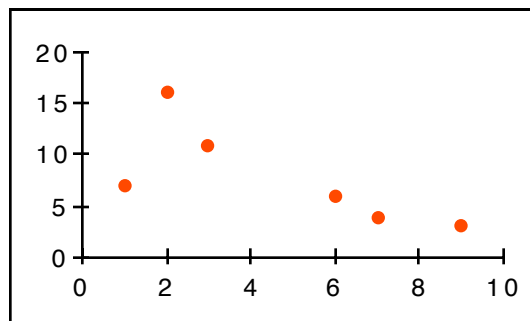
```
ChartData(12 98.3 null 8 Null NULL 7.23 -0.67)
```

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  // -1..."fehlende Werte"
  ChartData(7 16 11 -1 -1 6 4 -1 3)
  ChartDataLowerLimits(0.0) // nach ChartData()!
  LineChart(symbol;on)
  SymbolStyle(1;square;4;;red)
  GridLocation(;none)
CloseDrawing()
```



```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  // 9999..."fehlende Werte"
  ChartData(1 2 3 4 5 6 7 8 9;
            7 16 11 9999 9999 6 4 9999 3)
  ChartDataUpperLimits(1000) // nach ChartData()!
  ScatterChart2D()
  SymbolStyle(1;bullet;4)
  GridLocation(;none)
CloseDrawing()
```



**ChartDataRead(dateiname;istTransponiert;  
serientrennzeichen;elementtrennzeichen)**

Anstelle die Diagrammwerte via ChartData() direkt aus FileMaker Pro zu übernehmen, können diese auch aus einer Textdatei importiert werden. Mit dem ersten Argument *dateiname* wird der Name der Datei inklusive optionalem Dateipfad festgelegt; Details dazu finden sich im Abschnitt *Ausgabe*. Standardmäßig werden die Daten zeilenweise eingelesen. Falls das 2. Argument *istTransponiert=on* gesetzt wird, werden die Daten spaltenweise eingelesen, d.h. eine Datenserie entspricht einer Spalte in der Datei. Durch die Argumente *serientrennzeichen* und *elementtrennzeichen* können benutzerdefinierte Trennzeichen zwischen den Serien und zwischen den einzelnen Werten festgelegt werden. Wenn nicht anders vorgegeben, sind Serien durch einen Zeilenumbruch "\n" zu trennen und die einzelne Werte durch ein Tabulatorzeichen "\t". Dezimalzahlen können sowohl mit Dezimalpunkt als auch mit Dezimalkomma angeführt werden; Tausender-Trennzeichen sind jedoch nicht erlaubt. Die Funktion ChartDataOptions() wird in Verbindung mit ChartDataRead() ignoriert. Beispiele:

```
ChartDataRead("Macintosh HD/Daten/Plotdaten.dat")  
ChartDataRead("C:/Daten/plotdata.txt";";";" ")
```

Die Funktion ChartDataRead() kann beliebig platziert werden, jedoch immer innerhalb der Funktionen OpenDrawing() und CloseDrawing(). ChartDataRead() kann auch mehrfach aufgerufen werden.

Beispiele:

```
OpenDrawing(200;120)  
  AddFrame(0;0;200;120)  
  LineChart(symbol;on)  
  SymbolStyle(1;bullet;4)  
  SymbolStyle(2;bullet;4)  
  ChartDataRead("Daten")  
CloseDrawing()
```

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenChart(5;5;120;115;off)
    ChartDataRead("Daten_1.txt")
    BarChart()
  CloseChart()
  OpenChart(120;0;75;75;off)
    ChartDataRead("Daten_2.txt")
    PieChart(label)
  CloseChart()
  OpenChart(120;50;75;75;off)
    ChartDataRead("Daten_3.txt")
    PieChart(label)
  CloseChart()
CloseDrawing()

```

**ChartDataWrite(dateiname;dateiflag;dateityp;  
istTransponiert;serientrennzeichen;  
elementtrennzeichen;format1;format2...)**

Mit der Funktion `ChartDataWrite()` ist es möglich, die Diagrammwerte in eine Textdatei zu sichern. Die ersten drei Argumente *dateiname*, *dateiflag* und *dateityp* werden detailliert im Abschnitt *Ausgabe* erklärt. Standardmäßig werden die Daten zeilenweise weggeschrieben, d.h. einer Werteserie entspricht eine Ausgabezeile. Falls das 4. Argument *istTransponiert=on* gesetzt wird, werden die Daten spaltenweise ausgegeben, d.h. einer Werteserie entspricht eine Ausgabespalte. Weiters können durch die Argumente *serientrennzeichen* und *elementtrennzeichen* benutzerdefinierte Trennzeichen festgelegt werden. Wenn nicht anders vorgegeben, werden Serien durch einen Zeilenumbruch "\n" getrennt und die einzelne Werte durch ein Tabulatorzeichen "\t". Optional können den Werteserien individuelle Formatanweisungen zugeordnet werden. Die Formatanweisungen werden periodisch wiederholt, falls weniger Formatanweisungen als Werteserien vorhanden sind. Wenn keine Formatanweisungen vorgegeben werden, wird das Defaultformat "|u|" verwendet. Sämtliche Formatanweisungen inkl. zahlreicher Beispiele finden sich in *xmReferenz*.

Alle Zahlen werden ohne Tausender-Trennzeichen ausgegeben, alle Dezimalzahlen stets mit einem Dezimalpunkt "." und nicht mit einem Dezimalkomma ",".

Beispiele:

```
ChartDataWrite("Daten/dl.txt";;"ttx";;" " " " | i0 | " | f2 | ")
ChartDataWrite("Macintosh HD/Plots/ExportDaten";replace)
ChartDataWrite("C:/Programme/Daten/export.txt";;"on")
```

Die Funktion `ChartDataWrite()` kann sowohl vor als auch nach der Funktion `ChartData()` angeführt werden, jedoch immer innerhalb der Funktionen `OpenDrawing()` und `CloseDrawing()`.

`ChartDataWrite()` kann auch mehrfach aufgerufen werden.

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartDataWrite("Daten";replace;;on;;" | f1 | " | f2 | ")
  ChartData(23 19 10 15 13 6 2; // x-Werte
            18 12 19 24 15 15 6) // y-Werte
  LineChart2D(symbol)
  SymbolStyle(1;bullet;4)
CloseDrawing()
```

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenChart(5;5;120;115;off)
    ChartData(3 2 1 5 3)
    ChartDataWrite("Daten_1.txt";replace)
    BarChart()
  CloseChart()
  OpenChart(120;0;75;75;off)
    ChartData(13 9 13)
    ChartDataWrite("Daten_2.txt";replace)
    PieChart(label)
  CloseChart()
  OpenChart(120;50;75;75;off)
    ChartData(1.5 2 1,5)
    ChartDataWrite("Daten_3.txt";replace)
    PieChart(label)
  CloseChart()
CloseDrawing()
```

## Eingabe von Datum und Uhrzeit

Anstelle numerischer Daten können an `ChartData()` auch Datumswerte und Uhrzeiten übergeben werden. Unterstützt werden sowohl reine Datumseingaben, wie zum Beispiel:

```
ChartData(1.1.2006 17.3.2006 18.3.2006 22.10.2007)
```

als auch reine Uhrzeiteingaben. Zum Beispiel:

```
ChartData(1:00 8:32 14:01:38 23:22:11)
```

Aber auch kombinierte Datums- und Uhrzeitwerte (sog. timestamps) sind erlaubt. Datum und Uhrzeit werden entweder durch ein Konkatenationszeichen "&" zusammengehängt oder unter Hochkomma gestellt. Zum Beispiel:

```
ChartData(1.1.2006&11:22 "17.3.2006 18:33:00")
```

Als Trennzeichen für die Eingabe von Datumswerten sind Punkt, Schrägstrich und Bindestrich erlaubt. Zum Beispiel:

```
ChartData(1.1.2006 13/4/2006 22-05-2006)
```

Zur Eingabe von Uhrzeiten sind Doppelpunkt und Apostroph erlaubt. Typografische Apostrophe (') sind nicht erlaubt. Zum Beispiel:

```
ChartData(0:22 11:22:33 3'13 14'05'22)
```

Durch das 1. Argument *datumsfolge* der Funktion `DateTimeOptions()` kann die Reihenfolge von Tag, Monat und Jahr bei der Eingabe von Datumswerten vorgegeben werden. Die folgenden drei Reihenfolgen sind zulässig:

Reihenfolge	Konstante	Wert	Beispiel
Jahr, Monat, Tag	ymd	1	2006-08-04
Monat, Tag, Jahr	mdy	2 (default)	4/8/2006
Tag, Monat, Jahr	dmy	3	4.8.2006

Um Mehrdeutigkeiten bei den eingegebenen Datumswerten auszuschließen — wie zum Beispiel 4/5/06 oder 4/8/2006 (europäisch: 4. August 2006 oder amerikanisch: 8. April 2006) — sind folgende Regeln zu beachten.

- Jahresangaben erfolgen zweckmäßigerweise immer inklusive des Jahrhunderts, z.B. 1998 und nicht 98. Wird das Jahrhundert weggelassen, dann wird von xmCHART die zweistellige Jahreszahl automatisch in eine vierstellige Jahreszahl umgewandelt. Die Umrechnung erfolgt so, dass die vierstellige Jahreszahl, ausgehend vom aktuellen Jahr, entweder innerhalb der vorherigen 70 Jahre oder innerhalb der nächsten 30 Jahre liegt.

Zum Beispiel, ist das aktuelle Jahr 2006, dann werden die Jahre 00, 01, 02 bis 36 in die Jahre 2000, 2001, 2002 bis 2036 umgerechnet, die Jahre 37 bis 99 in 1937 bis 1999. Ist das aktuelle Jahr 2007, dann werden die Jahre 00, 01, 02 bis 37 in die Jahre 2000, 2001, 2002 bis 2037 umgerechnet, die Jahre 38 bis 99 in 1938 bis 1999.

- Erfolgt die Eingabe des Datums entweder in der Reihenfolge Tag-Monat-Jahr oder Monat-Tag-Jahr dann ist es immer zweckmäßig die Datumsreihenfolge explizit durch die Funktion `DateTimeOptions()` vorzugeben. Dabei ist es wichtig, dass die Funktion `DateTimeOptions()` immer *vor* der Funktion `ChartData()` angeführt wird. Bei der Reihenfolge Jahr-Monat-Tag ist das nicht unbedingt erforderlich, da diese Reihenfolge eindeutig ist — vorausgesetzt die Jahreszahl wird inklusive Jahrhundert eingegeben.

#### Beispiele:

```
ChartData(2006-02-01 2006-04-12) // 1.Feb., 12.Apr.  
                                // (eindeutig)
```

```
DateTimeOptions(dmy) // immer vor ChartData() anführen!  
ChartData(1/2/2006 12/4/2006) // 1.Feb., 12.Apr.
```

```
DateTimeOptions(mdy) // amerikanische Datumsschreibweise  
ChartData(1/2/2006 12/4/2006) // 2.Jan., 4.Dez.
```

## Diagramme

xmCHART stellt eine Vielzahl von Diagrammarten zur Verfügung, welche grob in folgende Kategorien eingeteilt werden können:

- Punktediagramme (1- und 2-dimensional)
- Liniendiagramme (1- und 2-dimensional)
- Flächendiagramme (1- und 2-dimensional)
- Balkendiagramme (2- und 3-dimensional)
- Blasendiagramme (1- und 2-dimensional)
- Kuchendiagramme (2- und 3-dimensional)
- Polar- und Radardiagramme
- Börsendiagramme
- Statistische Diagramme

Alle Diagramme können durch zahlreiche Optionen gestaltet und variiert werden. Nachfolgend werden sämtliche Diagrammarten ausführlich erläutert und durch eine Vielzahl anschaulicher Beispiele ergänzt.

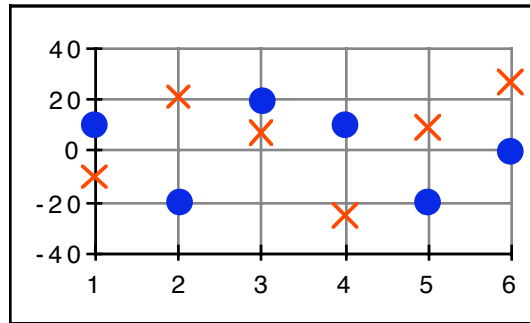
### **Punktediagramme:**

Zur Darstellung von Punktediagrammen stehen zwei Funktionen zur Verfügung:

#### **ScatterChart(darstellung;intervalleVerschieben)**

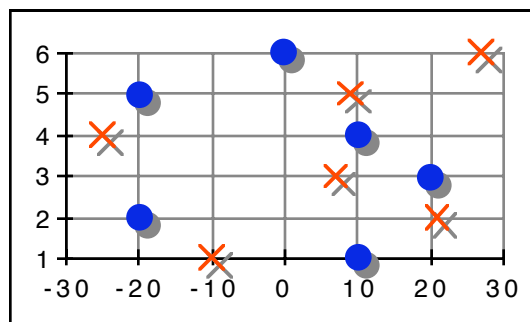
Die Funktion ScatterChart( ) dient zur Darstellung von 1-dimensionalen Punktediagrammen. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(-10 21 7 -25 9 27; // 1.Serie
            10 -20 20 10 -20 0) // 2.Serie
  ScatterChart()
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Symbole mit Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

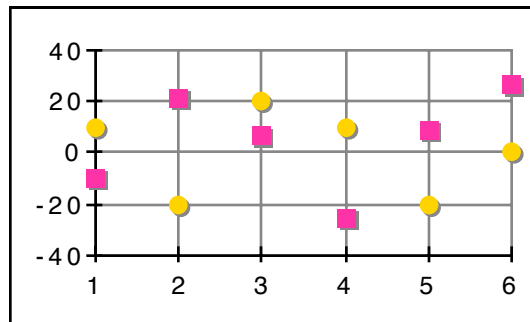
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(-10 21 7 -25 9 27; // 1.Serie
            10 -20 20 10 -20 0) // 2.Serie
  ScatterChart(shadow+horizontal)
CloseDrawing()
```



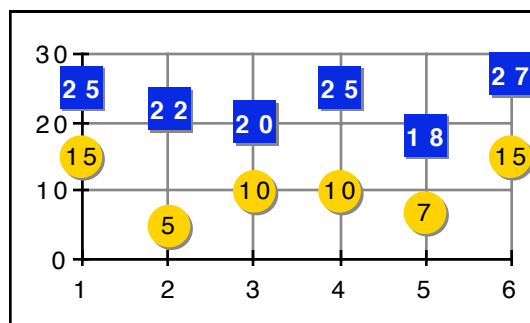
Die Darstellung der Symbole und Schatten kann durch die Stilfunktionen *SymbolStyle()* und *ShadowStyle()* variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen *LabelTexts()*, *LabelStyle()*, *LabelBackground()* und *LabelOptions()*. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert.

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(-10 21 7 -25 9 27; // 1.Serie
            10 -20 20 10 -20 0) // 2.Serie
  ScatterChart(shadow)
  SymbolStyle(1;square;6;;purple)
  SymbolStyle(2;bulet;6;;darkYellow)
  ShadowStyle(all;1;gray)
CloseDrawing()
```

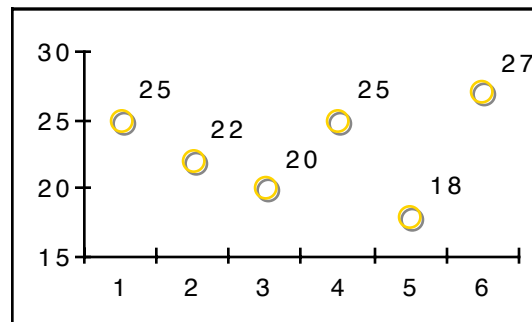


```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(25 22 20 25 18 27; // 1.Serie
            15 5 10 10 7 15) // 2.Serie
  ScatterChart(shadow+label)
  SymbolStyle(1;square;15;;blue)
  SymbolStyle(2;bulet;15;;darkYellow)
  ShadowStyle(all;1;gray)
  LabelOptions(all;centerCenter)
  LabelStyle(1;;;bold;white)
CloseDrawing()
```



Bei eindimensionalen Diagrammen erweist es sich bisweilen als vorteilhaft, die Symbole um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen liegen, sondern in der Mitte der Intervalle. Dies geschieht durch Aktivieren des 2. Arguments *intervalleVerschieben=on*. Beispiel:

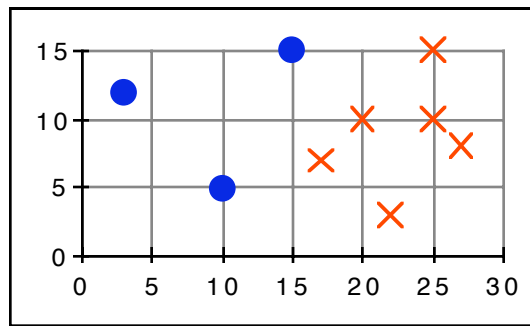
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(25 22 20 25 18 27)
  ScatterChart(shadow+label;on)
  SymbolStyle(1;circle;8;;darkYellow)
  ShadowStyle(all;1;gray)
  GridLocation(all;none)
CloseDrawing()
```



### ScatterChart2D(darstellung)

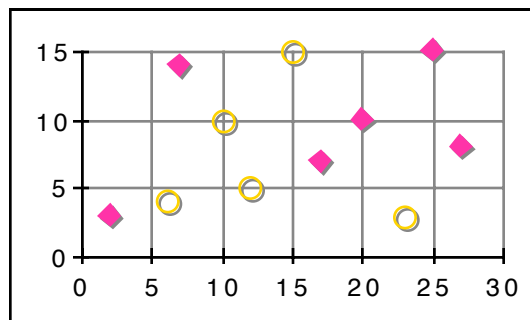
Die Funktion ScatterChart2D() ermöglicht die Darstellung von zweidimensionalen Punktediagrammen. Dabei liefert die 1. Werteserie in der Funktion ChartData() die x-Werte, die 2. Werteserie die y-Werte zur ersten Symbolserie, die 3. und 4. Werteserie in ChartData() die x- und y-Werte zur zweiten Symbolserie usw. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(25 22 20 25 17 27; // x-Werte 1.Serie
            15 3 10 10 7 8; // y-Werte 1.Serie
            3 10 15; // x-Werte 2.Serie
            12 5 15) // y-Werte 2.Serie
  ScatterChart2D()
CloseDrawing()
```



Wie bei den eindimensionalen Punktediagrammen können auch hier durch das Argument *darstellung* die Symbole mit Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Die Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Eine gedrehte Darstellung (*darstellung=horizontal*) wird bei zweidimensionalen Diagrammen nicht unterstützt, da dies einfach durch Vertauschen der Werteserien in `ChartData()` möglich ist. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(25 2 20 7 17 27; // x-Werte 1.Serie
          15 3 10 14 7 8; // y-Werte 1.Serie
          23 10 15 12 6; // x-Werte 2.Serie
          3 10 15 5 4) // y-Werte 2.Serie
ScatterChart2D(shadow)
SymbolStyle(1;diamond;7;;purple)
SymbolStyle(2;circle;8;;darkYellow)
ShadowStyle(all;1;gray)
CloseDrawing()
```



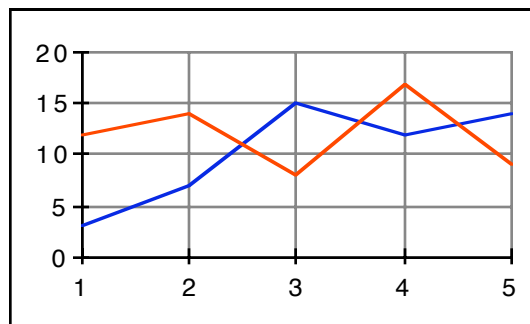
**Liniendiagramme:**

Zur Darstellung von Liniendiagrammen stehen zwei Funktionen zur Verfügung:

**LineChart(darstellung;intervalleVerschieben)**

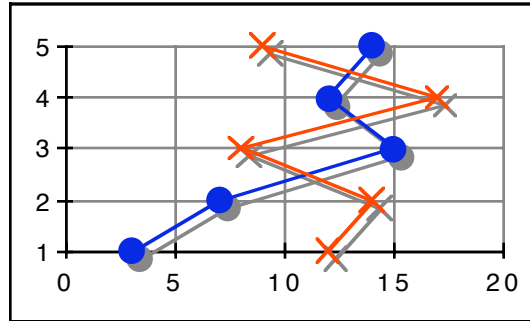
Die Funktion LineChart() dient zur Darstellung von eindimensionalen Liniendiagrammen. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(12 14 8 17 9;
            3 7 15 12 14)
  LineChart()
CloseDrawing()
```



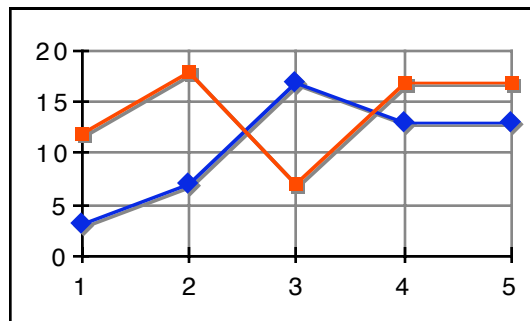
Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Linien mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(12 14 8 17 9; 3 7 15 12 14)
  LineChart(shadow+horizontal+symbol)
CloseDrawing()
```

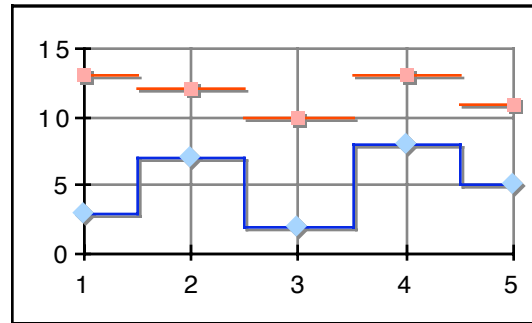


Die Darstellung der Linien, Symbole und Schatten kann durch die Stilfunktionen `LineStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(12 18 7 17 17; 3 7 17 13 13)
  LineChart(shadow+symbol)
  SymbolStyle(1;square;4;;red)
  SymbolStyle(2;diamond;6;;blue)
  ShadowStyle(all;1;gray)
CloseDrawing()
```



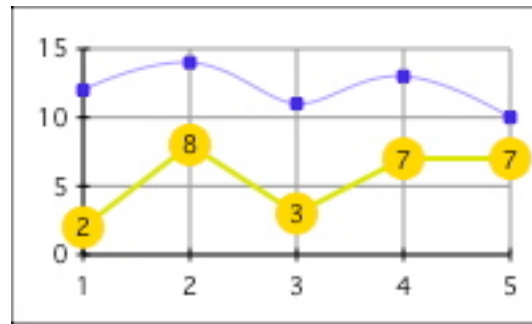
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(13 12 10 13 11; 3 7 2 8 5)
  LineChart(shadow+symbol)
  LineStyle(1;jump)
  LineStyle(2;step)
  SymbolStyle(1;square;4;;lightRed)
  SymbolStyle(2;diamond;6;;lightBlue)
  ShadowStyle(all;1;gray)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(12 14 11 13 10; 2 8 3 7 7)
  LineChart(symbol+label)
  LineStyle(1;smooth;1;150 150 255)
  LineStyle(2;poly;2;220 220 0)
  SymbolStyle(1;square;4;;blue)
  SymbolStyle(2;bullet;15;;darkYellow)
  LabelTexts(1;" ") // keine Beschriftung
  LabelOptions(2;centerCenter)
CloseDrawing()

```

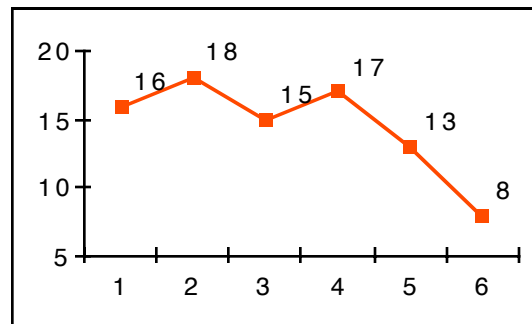


Bei eindimensionalen Liniendiagrammen besteht die Möglichkeit, die Polygonpunkte um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen liegen, sondern in der Mitte der Intervalle. Dies wird durch Aktivieren des 2. Arguments *intervalleVerschieben=on* ermöglicht. Beispiel:

```

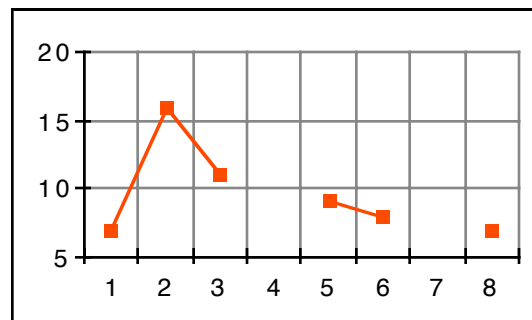
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 15 17 13 8)
  LineChart(symbol+label;on)
  SymbolStyle(1;square;4;;red)
  GridLocation(all;none)
CloseDrawing()

```



Zusätzlich besteht die Möglichkeit unterbrochene Linienzüge darzustellen. Dazu werden die fehlenden Werte in der Funktion `ChartData()` durch `NULL` ersetzt. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(7 16 11 null 9 8 null 7)
LineChart(symbol;on)
SymbolStyle(1;square;4;;red)
CloseDrawing()
```



### LineChart2D(darstellung)

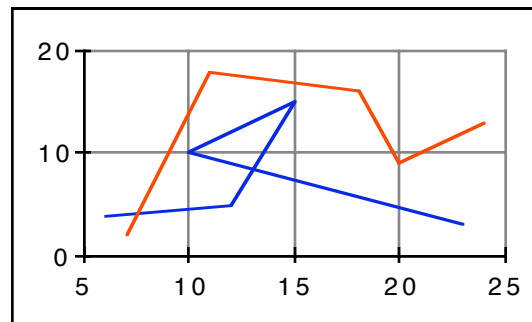
Die Funktion `LineChart2D()` ermöglicht die Darstellung von zwei-dimensionalen Liniendiagrammen. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die x-Werte, die 2. Werteserie die y-Werte des ersten Linienzugs, die 3. und 4. Werteserie in `ChartData()` die x- und y-Werte des zweiten Linienzugs usw.

Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 7 11 18 20 24; // x-Werte 1.Serie
            2 18 16  9 13; // y-Werte 1.Serie
            23 10 15 12  6; // x-Werte 2.Serie
            3 10 15  5  4) // y-Werte 2.Serie
  LineChart2D()
CloseDrawing()

```

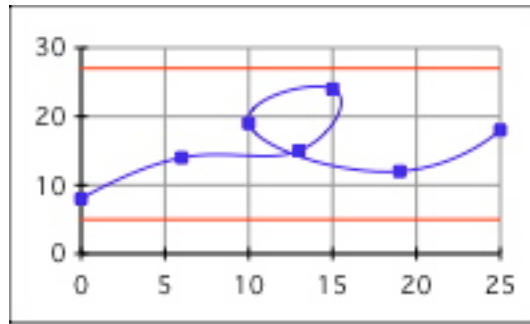


Wie bei den eindimensionalen Liniendiagrammen können auch hier durch das Argument *darstellung* die Linien mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Die Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Eine gedrehte Darstellung (*darstellung=horizontal*) wird bei zweidimensionalen Diagrammen nicht unterstützt, da dies einfach durch Vertauschen der Werteserien in `ChartData()` möglich ist. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(25 19 10 15 13  6 0; // x-Werte 1.Serie
            18 12 19 24 15 14 8; // y-Werte 1.Serie
            0 25; 27 27; // obere Begrenzungslinie
            0 25;  5  5) // untere Begrenzungslinie
  LineChart2D(symbol)
  LineStyle(all;poly;1;red)
  LineStyle(1;smooth;1;blue)
  SymbolStyle(all;none)
  SymbolStyle(1;square;4;;blue)
CloseDrawing()

```



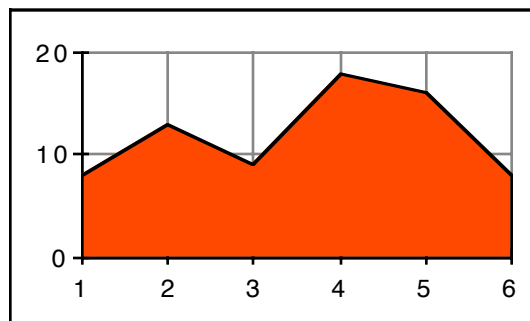
### Flächendiagramme:

Flächendiagramme können sowohl ein- als auch zweidimensional dargestellt werden.

#### **AreaChart(darstellung;intervalleVerschieben)**

Die Funktion `AreaChart()` dient zur Darstellung von eindimensionalen Flächendiagrammen. Beispiel:

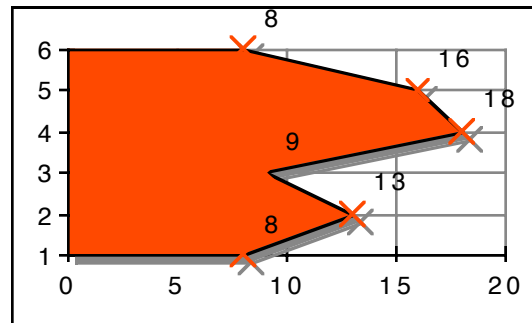
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(8 13 9 18 16 8)
AreaChart()
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren.

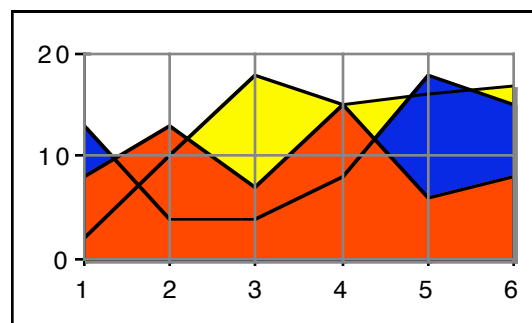
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(8 13 9 18 16 8)
  AreaChart(symbol+label+horizontal+shadow)
CloseDrawing()
```



Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert. Beispiele:

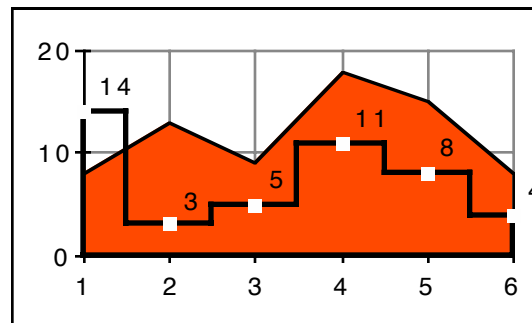
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 8 13 7 15 6 8;
            13 4 4 8 18 15;
            2 10 18 15 16 17)
  AreaChart(shadow)
  ShadowStyle(all;1;gray)
  GridLocation(all;front)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(8 13 9 18 15 8; 14 3 5 11 8 4)
  AreaChart(symbol+label)
  SymbolStyle(1;none)
  LabelTexts(1;" ") // keine Beschriftung
  BorderStyle(2;step;2)
  FillStyle(2;;transparent)
  SymbolStyle(2;square;4;;white)
CloseDrawing()

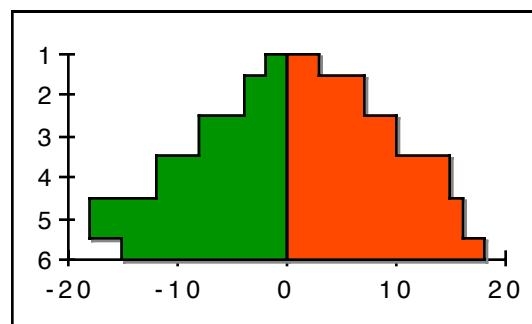
```



```

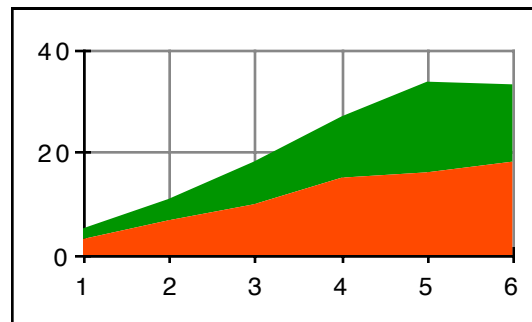
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 3 7 10 15 16 18;
            -2 -4 -8 -12 -18 -15)
  AreaChart(shadow+horizontal)
  BorderStyle(all;step)
  FillStyle(2;green)
  ShadowStyle(all;1;gray)
  GridLocation(all;none)
  ScalingOptions(y;on) // y-Achse von oben nach unten
CloseDrawing()

```

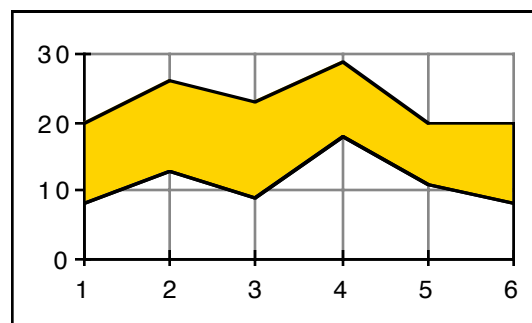


Weiters können eindimensionale Flächendiagramme gestapelt (*darstellung=stacked*) und proportional (*darstellung=proportional*) dargestellt werden. Proportionale Diagramme werden automatisch gestapelt. Die Beschriftung gestapelter und proportionaler Diagramme wird ausführlich anhand zahlreicher Beispiele im Abschnitt *Stile* erläutert. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 7 10 15 16 18;
            2 4 8 12 18 15)
  AreaChart(stacked)
  BorderStyle(all;none)
  FillStyle(2;green)
CloseDrawing()
```



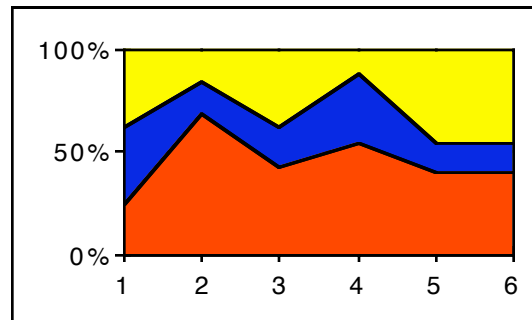
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 8 13 9 18 11 8;
            12 13 14 11 9 12)
  AreaChart(stacked)
  FillStyle(1;;transparent)
  FillStyle(2;darkYellow)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 8 13 9 18 14 8;
            12 3 4 11 5 3;
            12 3 8 4 16 9)
  AreaChart(stacked+proportional)
  Scaling(y;percent)
CloseDrawing()

```

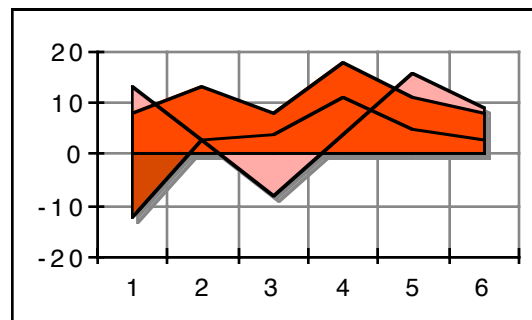


Bei eindimensionalen Diagrammen besteht optional die Möglichkeit, die Polygonpunkte um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen, sondern in der Mitte der Intervalle liegen. Dies geschieht durch Aktivieren des 2. Arguments *intervalleVerschieben=on*. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 8 13 8 18 11 8;
            -12 3 4 11 5 3;
            13 3 -8 4 16 9)
  AreaChart(shadow;on)
  FillStyle(2;darkRed)
  FillStyle(3;lightRed)
  ShadowStyle(all;2;gray)
CloseDrawing()

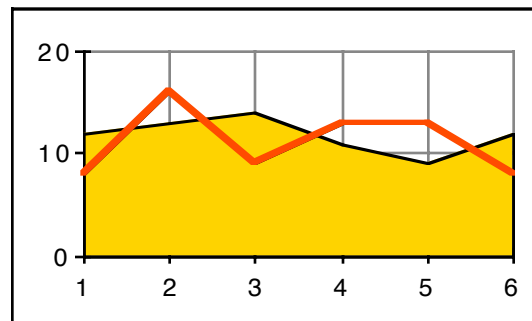
```



### **AreaChartOptions(linienstilVerwenden;referenzWert; posNegGetrenntStapeln)**

Optional kann durch Aktivieren des ersten Arguments *linienstilVerwenden=on* die Berandung durch einen Linienzug ergänzt werden. Im Gegensatz zur standardmäßigen Berandung, welche die gesamte Fläche umschließt, d.h. inklusive der Seitenränder und der Grundlinie, bezieht sich der Linienzug nur auf den Bereich zwischen den Diagrammwerten. Das Aussehen des Polygonzugs kann durch die Funktion `LineStyle()` gestaltet werden. Dabei wird in der Funktion `LineStyle()` das zweite Argument *linienführung*, welches üblicherweise zur Kontrolle des Kurvenverlaufs dient, in diesem Zusammenhang ignoriert. Eine mögliche Anwendung ist die gleichzeitige Darstellung eines Flächen- und Liniendiagramms; dazu wird die Füllung einer Serie *transparent* gesetzt. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 8 16 9 13 13 8; // Linie
            12 13 14 11 9 12) // Fläche
  AreaChart()
  AreaChartOptions(on) // nach AreaChart()!
  LineStyle(1;;2;red)
  LineStyle(2;;0)
  FillStyle(1;;transparent) // keine Füllung
  FillStyle(2;darkYellow)
CloseDrawing()
```



Das zweite Argument *referenzwert* ermöglicht die Vorgabe einer benutzerdefinierten Grundlinie. Standardmäßig beginnen alle Flächen bei Null, d.h. *referenzwert=0*.

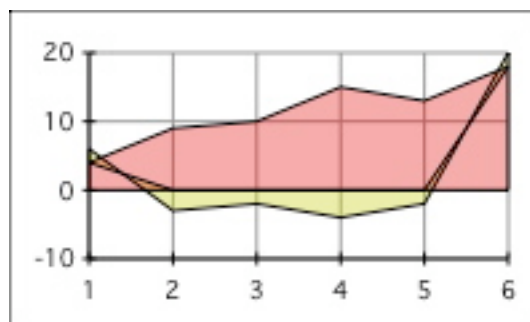
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 13 8 18 11 0)
  AreaChart()
  AreaChartOptions(;10) // nach AreaChart()!
  FillStyle(1;50 0 255 50)
CloseDrawing()
```



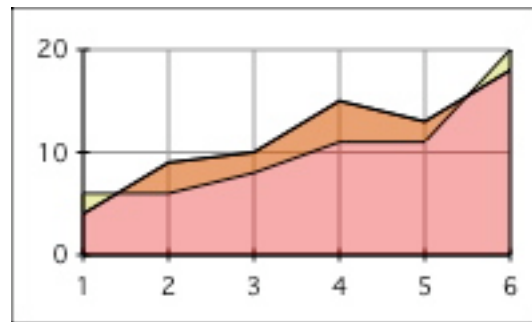
Mit dem 3. Argument *posNegGetrenntStapeln* kann die Stapelung negativer Werte kontrolliert werden. Standardmäßig werden positive und negative Werte getrennt gestapelt (*posNegGetrenntStapeln=on*), d.h. positive Werte werden "oberhalb" der Referenzlinie, negative Werte "unterhalb" aufsummiert. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 9 10 15 13 18;
            2 -3 -2 -4 -2 2)
  AreaChart(stacked)
  AreaChartOptions(;;on) // nach AreaChart()!
  FillStyle(1;200 0 0 100)
  FillStyle(2;200 200 0 100)
CloseDrawing()
```



Wird *posNegGetrenntStapeln=off* gesetzt, werden positive und negative Werte nicht getrennt gestapelt. Dieser Darstellung erweist sich bisweilen zur Darstellung von Differenzwerten als vorteilhaft, wie zum Beispiel Soll-Ist-Abweichungen. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 4  9  10 15 13 18;
           2 -3 -2 -4 -2  2)
AreaChart(stacked)
AreaChartOptions(;;off) // nach AreaChart()!
FillStyle(1;200  0 0 100)
FillStyle(2;200 200 0 100)
CloseDrawing()
```



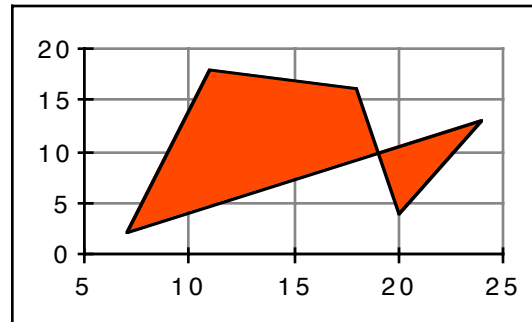
Zu beachten ist, dass die Funktion `AreaChartOptions()` nach der Funktion `AreaChart()` anzuführen ist. Diese Regel gilt ganz allgemein: *Diagrammoptionen sind stets nach der eigentlichen Diagrammfunktion anzuführen.*

### **AreaChart2D(darstellung;bisZurAchseFüllen)**

Die Funktion `AreaChart2D()` ermöglicht die Darstellung von zweidimensionalen Flächendiagrammen. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die x-Werte, die 2. Werteserie die y-Werte der ersten Fläche, die 3. und 4. Werteserie in `ChartData()` die x- und y-Werte der zweiten Fläche, usw.

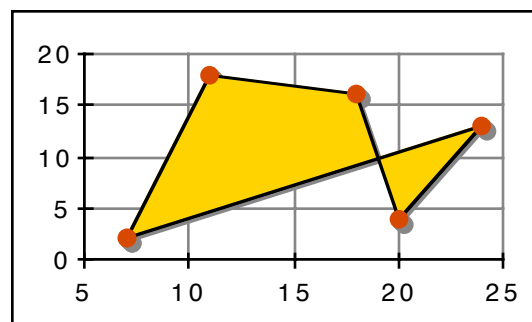
Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 7 11 18 20 24 7; // x-Werte
           2 18 16 4 13 2) // y-Werte
AreaChart2D()
CloseDrawing()
```



Wie bei den eindimensionalen Flächendiagrammen können auch hier durch das Argument *darstellung* die Flächen mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Die Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Eine gedrehte Darstellung (*darstellung=horizontal*) wird bei zweidimensionalen Diagrammen nicht unterstützt, da dies einfach durch Vertauschen der Werteserien in `ChartData()` möglich ist. Beispiele:

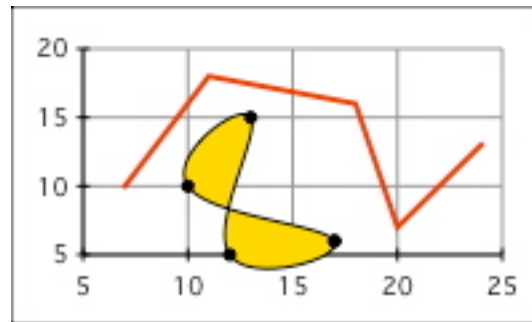
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 7 11 18 20 24 7; // x-Werte
           2 18 16 4 13 2) // y-Werte
AreaChart2D(symbol+shadow)
FillStyle(1;darkYellow)
ShadowStyle(1;1)
SymbolStyle(1;bullet;6;;darkRed)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 7 11 18 20 24; // x-Werte 1.Serie
            10 18 16 7 13; // y-Werte 1.Serie
            17 10 13 12 17; // x-Werte 2.Serie
            6 10 15 5 6) // y-Werte 2.Serie
  AreaChart2D(symbol)
  FillStyle(1;;transparent)
  FillStyle(2;darkYellow)
  BorderStyle(1;poly;2;darkRed)
  BorderStyle(2;smooth)
  SymbolStyle(1;none)
  SymbolStyle(2;bullet;4;;black)
CloseDrawing()

```



Durch das 2. Argument *bisZurAchseFüllen* ist es möglich die Füllung bis zur x-Achse bzw. y-Achse hin auszudehnen. Folgende drei Möglichkeiten stehen zur Verfügung:

*bisZurAchseFüllen=0*: keine Füllung bis zur Achse (default)

*bisZurAchseFüllen=1*: Füllung bis zur x-Achse

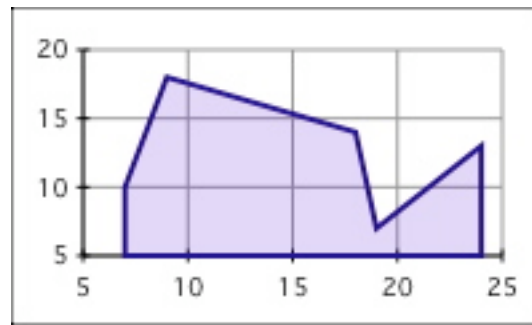
*bisZurAchseFüllen=2*: Füllung bis zur y-Achse

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 7 9 18 19 24; // x-Werte
            10 18 14 7 13) // y-Werte
  AreaChart2D(;1)
  FillStyle(1;50 0 200 50)
  BorderStyle(1;poly;2;darkBlue)
CloseDrawing()

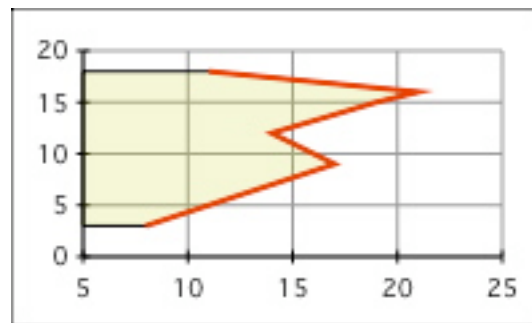
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(8 17 14 19 21 11; // x-Werte
          3  9 12 15 16 18) // y-Werte
AreaChart2D(;2)
AreaChartOptions(on)
FillStyle(1;200 200 0 50)
BorderStyle(1;3;1;)
LineStyle(1;;2;darkRed)
CloseDrawing()

```

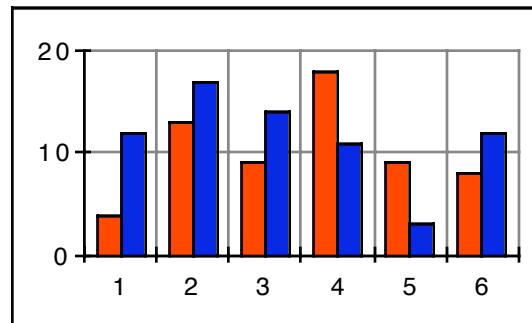


**Balkendiagramme:**

**BarChart(darstellung;kategorienabstand;serienabstand;  
balkentiefe)**

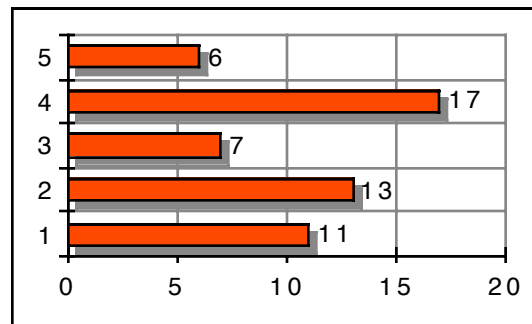
Die Funktion BarChart() dient zur Darstellung von Balkendiagrammen.  
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9 8; // "rote" Serie
            12 17 14 11 3 12) // "blaue" Serie
  BarChart()
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Balken mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren.  
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(11 13 7 17 6)
  BarChart(label+shadow+horizontal)
CloseDrawing()
```



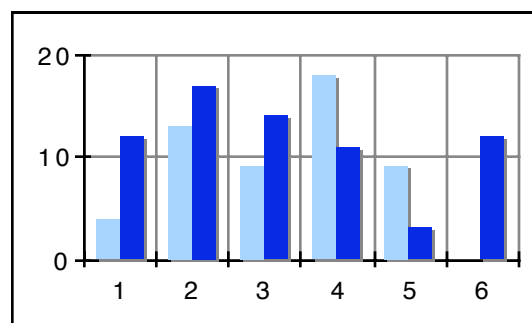
Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9;
             12 17 14 11 3 12)
  BarChart(shadow)
  FillStyle(1;lightBlue)
  BorderStyle(all;none)
  ShadowStyle(all;1;gray)
CloseDrawing()

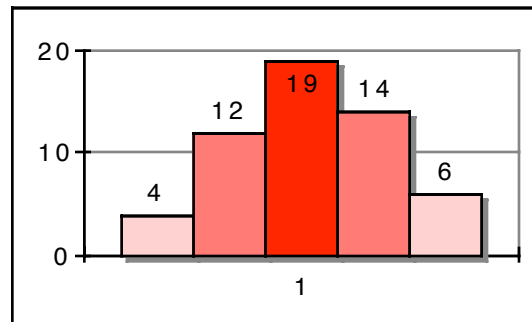
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4; 12; 19; 14; 6)
  BarChart(shadow+label)
  FillStyle(1;255 200 200)
  FillStyle(2;255 100 100)
  FillStyle(3;255 0 0)
  FillStyle(4;255 100 100)
  FillStyle(5;255 200 200)
  ShadowStyle(all;2;gray)
CloseDrawing()

```

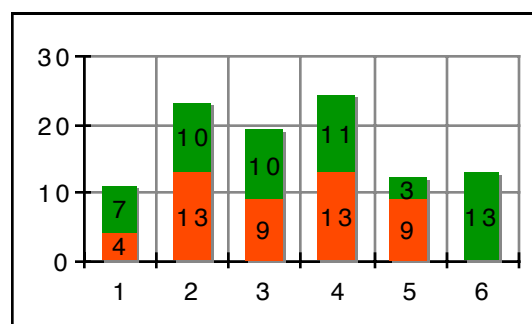


Weiters können Balkendiagramme gestapelt (*darstellung=stacked*) und proportional (*darstellung=proportional*) dargestellt werden. Proportionale Diagramme werden automatisch gestapelt. Die Beschriftung gestapelter und proportionaler Diagramme wird ausführlich anhand zahlreicher Beispiele im Abschnitt *Stile* erläutert. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(6 13 9 13 9;
            7 10 10 11 5 13)
  BarChart(shadow+stacked+label)
  FillStyle(2;green)
  BorderStyle(all;none)
  ShadowStyle(all;1;gray)
CloseDrawing()

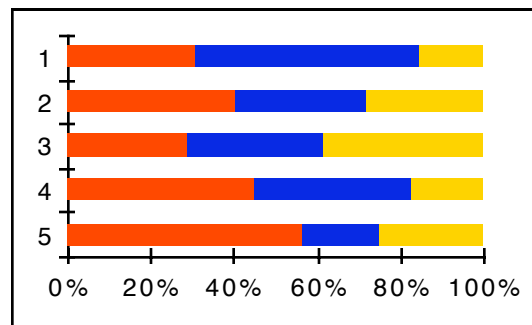
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 13 9 13 9;
            7 10 10 11 3;
            2 9 12 5 4)
  BarChart(horizontal+proportional)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridLocation(all;none)
  Scaling(x;percent)
  ScalingOptions(y;on) // y-Achse von oben nach unten
CloseDrawing()

```



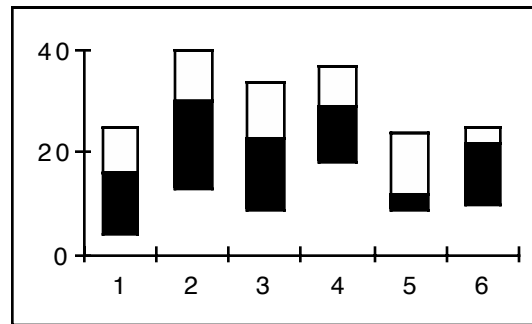
Zusätzlich besteht die Möglichkeit, durch Transparent-Setzen der ersten Datenserie, gleitende Balkendiagramme zu erstellen.

Beispiel:

```

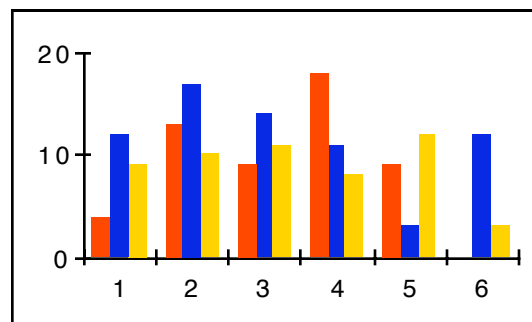
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9 10;
            12 17 14 11 3 12;
            9 10 11 8 12 3)
  BarChart(stacked)
  BorderStyle(1;none)
  FillStyle(1;;transparent)
  FillStyle(2;black)
  FillStyle(3;white)
  GridLocation(all;none)
CloseDrawing()

```



Durch die beiden Argumente *kategorienabstand* und *serienabstand* können die Abstände zwischen den Balken kontrolliert werden. Beide Argumente sind in Prozent der Balkenbreite einzugeben. Standardmäßig ist der Abstand zwischen zwei Balkengruppen (Kategorien) genau eine Balkenbreite, d.h. *kategorienabstand*=100. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9;
             12 17 14 11 3 12;
             9 10 11 8 12 3)
  BarChart(;100) // Default-Abstand
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridLocation(all;none)
CloseDrawing()
```



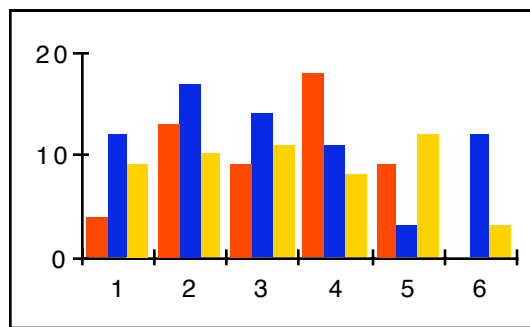
Der Abstand zwischen den Kategorien wird z.B. auf die Hälfte verkleinert durch *kategorienabstand*=50, oder komplett unterdrückt durch *kategorienabstand*=0.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9;
             12 17 14 11 3 12;
             9 10 11 8 12 3)
  BarChart(;50)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridLocation(all;none)
CloseDrawing()

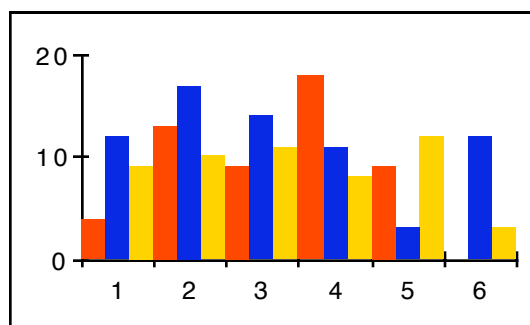
```



```

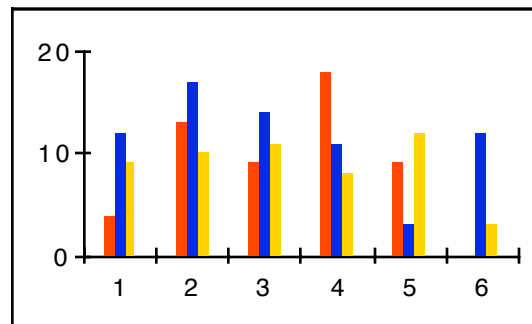
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9;
             12 17 14 11 3 12;
             9 10 11 8 12 3)
  BarChart(;0)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridLocation(all;none)
CloseDrawing()

```



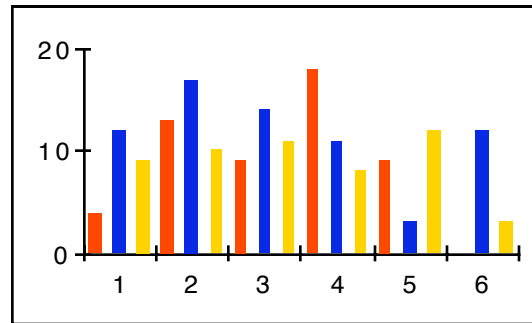
Durch z.B. *kategorienabstand=400* wird der Abstand zwischen zwei Balkengruppen auf das 4-fache der Balkenbreite vergrößert. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9;
            12 17 14 11 3 12;
            9 10 11 8 12 3)
  BarChart(;400)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridLocation(all;none)
CloseDrawing()
```



Bei nicht gestapelten Balken sind standardmäßig keine Lücken zwischen den Serien vorgesehen (*serienabstand=0*). Durch z.B. *serienabstand=100* wird zwischen den einzelnen Serien eine Lücke eingefügt. Die Größe der Lücke entspricht bei einem *serienabstand=100* der Balkenbreite. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9;
            12 17 14 11 3 12;
            9 10 11 8 12 3)
  BarChart(;;100)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridLocation(all;none)
CloseDrawing()
```

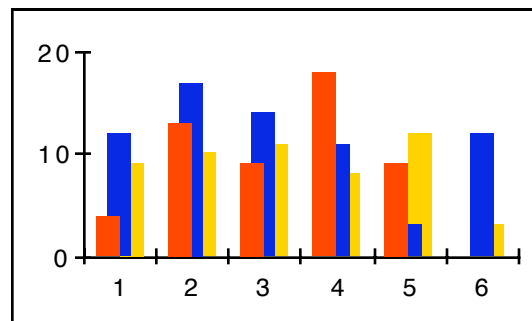


Ein negativer *serienabstand* führt bei nicht gestapelten Balken zu einer teilweisen oder vollständigen Überlappung der Balken. Beispiel:

```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 4 13 9 18 9;
           12 17 14 11 3 12;
           9 10 11 8 12 3)
BarChart(;;-50)
FillStyle(3;darkYellow)
BorderStyle(all;none)
GridLocation(all;none)
CloseDrawing()

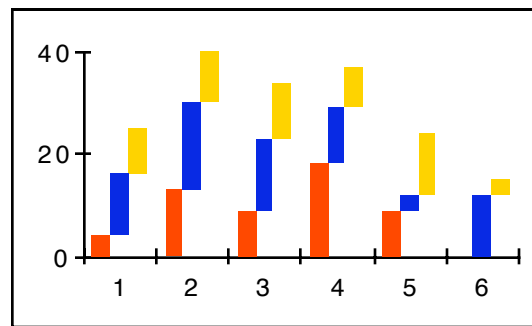
```



Da gestapelte Balken nicht nebeneinander, sondern übereinander angeordnet sind, ist der *serienabstand* standardmäßig -100. Bei einem *serienabstand* ungleich -100, zum Beispiel *serienabstand*=0, werden die gestapelten Balken versetzt dargestellt.

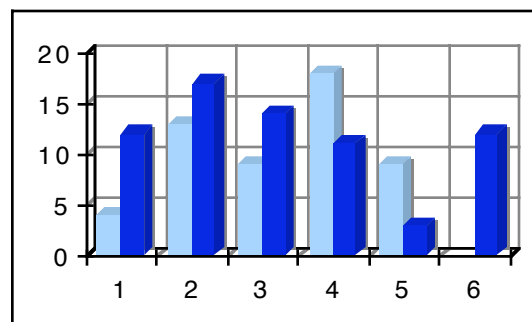
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9;
             12 17 14 11 3 12;
             9 10 11 8 12 3)
  BarChart(stacked;;0)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridLocation(all;none)
CloseDrawing()
```



Mit dem 4. Argument *balkentiefe* kann die dreidimensionale Darstellung der Balken kontrolliert werden. Die Balkentiefe ist dabei in Prozent der Balkenbreite einzugeben. Beispiele:

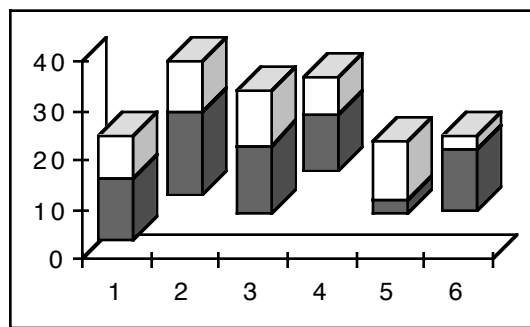
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9;
             12 17 14 11 3 12)
  BarChart(shadow;;;50)
  FillStyle(1;lightBlue)
  BorderStyle(all;none)
  ShadowStyle(all;1;gray)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9 10;
            12 17 14 11 3 12;
            9 10 11 8 12 3)
  BarChart(stacked;;;100)
  BorderStyle(1;none)
  FillStyle(1;;transparent)
  FillStyle(2;darkGray)
  FillStyle(3;white)
  GridLocation(all;none)
CloseDrawing()

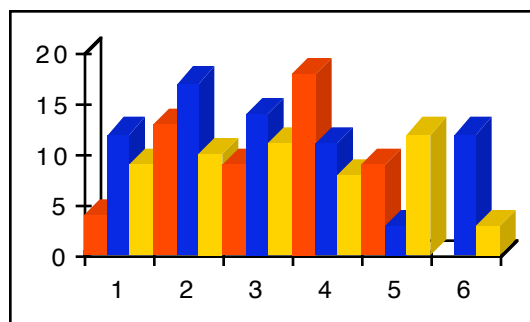
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9 18 9;
            12 17 14 11 3 12;
            9 10 11 8 12 3)
  BarChart(;0;;100)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridLocation(all;none)
CloseDrawing()

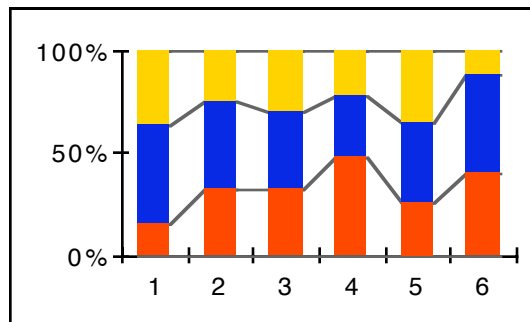
```



### **BarChartOptions(verbindungslinien;referenzwert; farbteilung)**

Durch Aktivieren des ersten Arguments *verbindungslinien=on* werden die — in diesem Fall üblicherweise gestapelten — Balken durch Linien verbunden. Die Gestaltung der Linien erfolgt durch die Funktion `LineStyle()`. Die Funktion `LineStyle()` wird ausführlich im Abschnitt *Stile* erläutert. Zu beachten ist, dass die Funktion `BarChartOptions()` nach der Funktion `BarChart()` anzuführen ist. Diese Regel gilt ganz allgemein: *Diagrammoptionen sind stets nach der eigentlichen Diagrammfunktion anzuführen*. Beispiel:

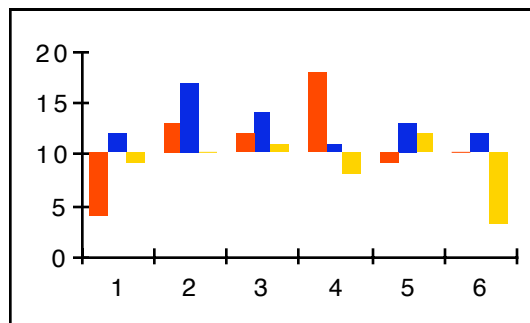
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 12 18  9 10;
            12 17 14 11 13 12;
            9 10 11  8 12  3)
  BarChart(proportional)
  BarChartOptions(on)
  FillStyle(3;darkYellow)
  LineStyle(all;poly;1;darkGray)
  BorderStyle(all;none)
  GridLocation(all;none)
  Scaling(y;percent)
CloseDrawing()
```



Das zweite Argument *referenzwert* ermöglicht die Vorgabe einer benutzerdefinierten Grundlinie. Standardmäßig beginnen alle Balken bei Null, d.h. *referenzwert=0*.

Beispiel:

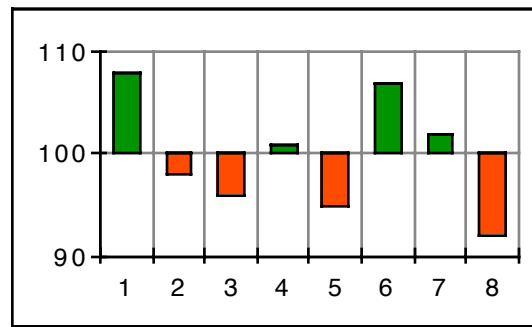
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 12 18  9 10;
            12 17 14 11 13 12;
            9 10 11  8 12  3)
  BarChart()
  BarChartOptions(;10) // Referenzwert = 10
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridLocation(all;none)
CloseDrawing()
```



Durch Aktivieren des dritten Arguments *farbteilung=on* werden die Werte, welche größer bzw. kleiner als der Referenzwert sind, durch Balken unterschiedlicher Farbe dargestellt. Das Aussehen der "positiven" Balken, d.h. die Diagrammwerte sind größer als der Referenzwert, wird durch Stilfunktionen mit ungeraden Serienindizes festgelegt, das Aussehen der "negativen" Balken durch Stilfunktionen mit geradzahligen Serienindizes.

Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(108 98 96 101 95 107 102 92)
  BarChart()
  BarChartOptions(;100;on) // Referenzwert = 100
  FillStyle(1;green) // "positiv", ungerader Serienind.
  FillStyle(2;red)   // "negativ", gerader Serienindex
CloseDrawing()
```



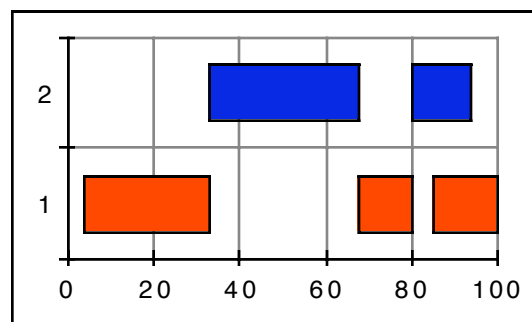
Zu beachten ist, dass die Farbteilung bei der Legende nicht unterstützt wird.

### Gantt-Diagramme:

#### **GanttChart(darstellung;kategorienabstand;balkentiefe)**

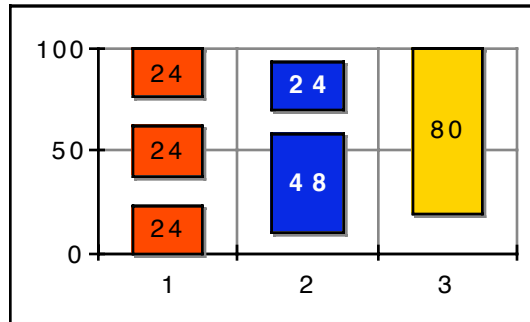
Die Funktion `GanttChart()` dient zur grafischen Aufbereitung von zeitabhängigen Aufgaben. Diese Art der Darstellung wurde von Henry L. Gantt 1917 erstmals verwendet. Jede Aufgabe wird durch die Eingabe von mindestens zwei Werten in `ChartData()` festgelegt; der erste Wert definiert den Beginn einer Aufgabe, der zweite Wert das Ende. Durch die Vorgabe von mehr als zwei Werten pro Serie kann eine Aufgabe in mehrere Zeitabschnitte unterteilt werden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 33 68 80 85 100; // Proj 1 (3 Teilber.)
            33 68 80 94)        // Proj 2 (2 Teilber.)
  GanttChart()
CloseDrawing()
```



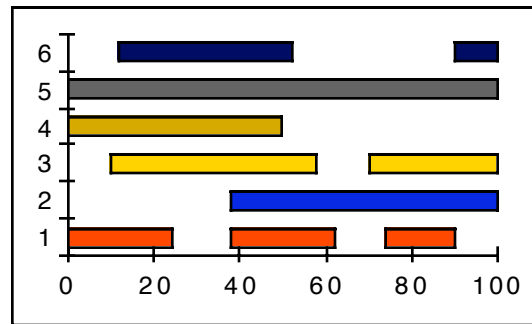
Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Balken durch Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 0 24 38 62 76 100;
           10 58 70 94;
           20 100)
GanttChart(label+shadow+horizontal)
FillStyle(3;darkYellow)
LabelStyle(2;;;bold;white)
ShadowStyle(all;1;gray)
CloseDrawing()
```



Die Darstellung der Füllungen, Berandungen und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert. Beispiele:

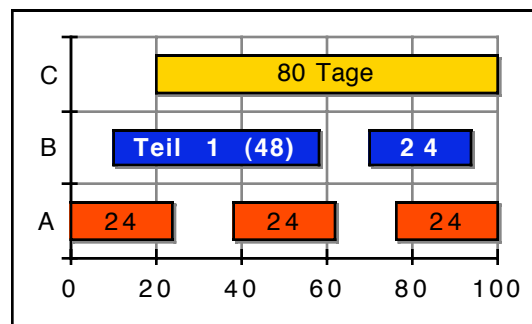
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 0 24 38 62 74 90;
           38 100;
           10 58 70 100;
           0 50;
           0 100;
           12 52 90 100)
GanttChart()
FillStyle(3;darkYellow)
GridLocation(all;none)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 0 24 38 62 76 100;
            10 58 70 94;
            20 100)
  GanttChart(label+shadow)
  FillStyle(3;darkYellow)
  LabelStyle(2;;;bold;white)
  LabelTexts(2;"Teil 1 (|u|)";"|u|")
  LabelTexts(3;"|u| Tage")
  ShadowStyle(all;1;gray)
  AxisMajorTickLabelTexts(y;"A";"B";"C")
CloseDrawing()

```



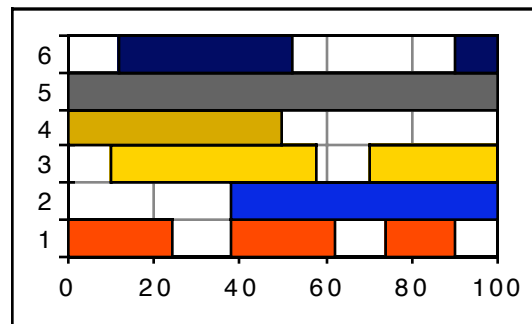
Durch das 2. Argument *kategorienabstand* kann der Abstand zwischen den Balken kontrolliert werden. Das Argument *kategorienabstand* ist in Prozent der Balkenbreite einzugeben. Standardmäßig ist der Abstand zwischen zwei Balken (Kategorien) genau eine Balkenbreite, d.h. *kategorienabstand*=100. Bei einem *kategorienabstand* kleiner 100 wird der Balkenabstand kleiner und die Balken breiter, bei einem *kategorienabstand* größer 100 wird der Abstand größer und die Balken schmaler.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 0  24  38 62  74 90;
             38 100;
             10 58 70 100;
             0  50;
             0 100;
             12 52 90 100)
  GanttChart(;0)
  FillStyle(3;darkYellow)
  GridFrame()
CloseDrawing()

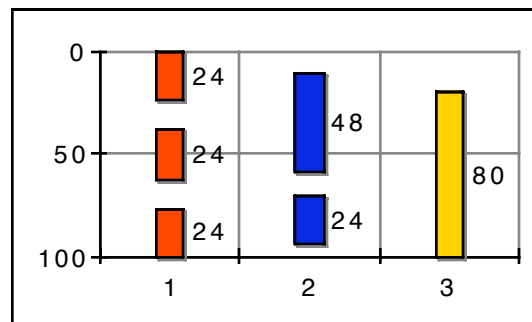
```



```

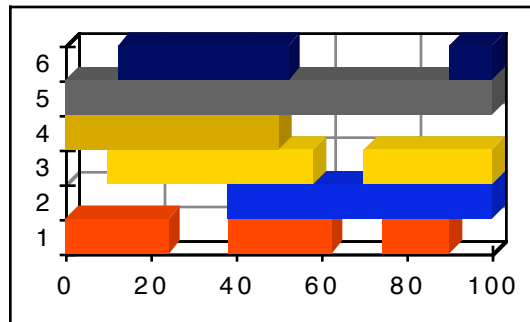
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 0  24  38 62  76 100;
             10 58 70 94;
             20 100)
  GanttChart(label+shadow+horizontal;400)
  FillStyle(3;darkYellow)
  ShadowStyle(all;1;gray)
  ScalingOptions(y;on) // y-Achse von oben nach unten
CloseDrawing()

```

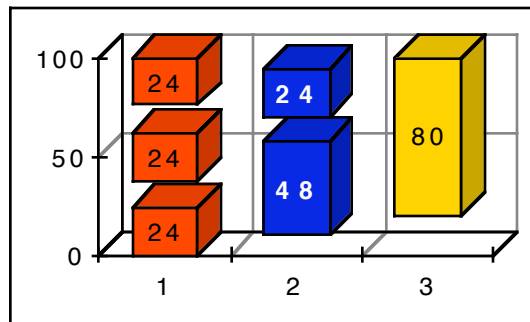


Mit dem 3. Argument *balkentiefe* kann die dreidimensionale Darstellung der Balken kontrolliert werden. Die Balkentiefe ist dabei in Prozent der Balkenbreite einzugeben. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 0 24 38 62 74 90;
             38 100; 10 58 70 100;
             0 50; 0 100;
             12 52 90 100)
  GanttChart(;0;50)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
  GridFrame()
CloseDrawing()
```



```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 0 24 38 62 76 100;
             10 58 70 94; 20 100)
  GanttChart(label+horizontal;;50)
  FillStyle(3;darkYellow)
  LabelStyle(2;;;bold;white)
  ShadowStyle(all;1;gray)
CloseDrawing()
```



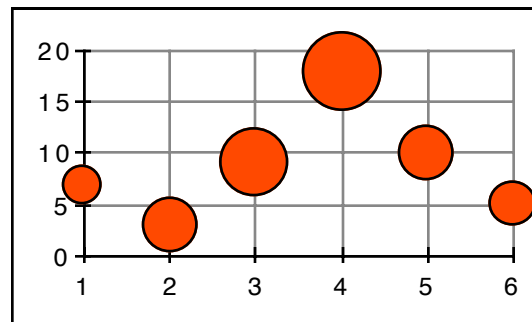
**Blasendiagramme:**

Blasendiagramme (bubble charts) können sowohl ein- als auch zwei-dimensional dargestellt werden.

**BubbleChart(*darstellung*;intervalleVerschieben)**

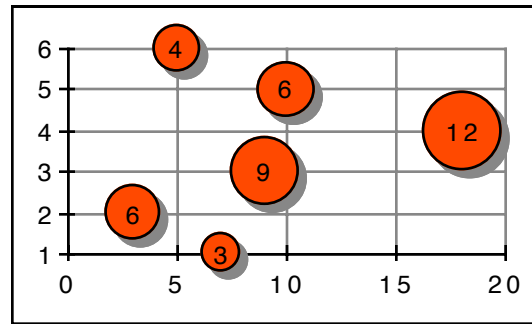
Die Funktion `BubbleChart()` dient zur Darstellung von eindimensionalen Blasendiagrammen. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die Positionen, die 2. Werteserie die Durchmesser zur ersten Serie, die 3. und 4. Werteserie in `ChartData()` die Positionen und Durchmesser zur zweiten Serie usw. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(7 3 9 18 10 5; // Positionen
          3 6 9 12 6 4) // Durchmesser
BubbleChart()
CloseDrawing()
```



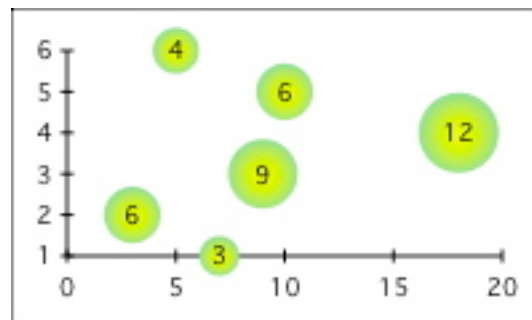
Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung der Blasen mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(7 3 9 18 10 5; // Positionen
          3 6 9 12 6 4) // Durchmesser
BubbleChart(horizontal+shadow+label)
CloseDrawing()
```



Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert. Beispiel:

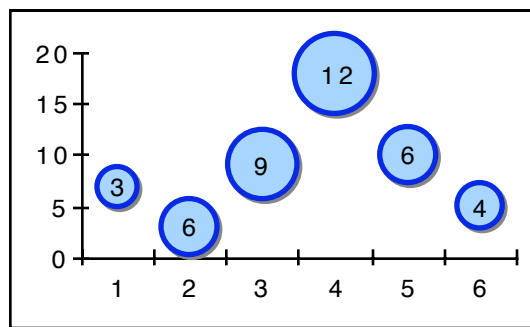
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(7 3 9 18 10 5; // Positionen
            3 6 9 12 6 4) // Durchmesser
  BubbleChart(label+horizontal)
  PictureStyle(1;resource;"39")
  BorderStyle(1;none)
  GridLocation(all;none)
CloseDrawing()
```



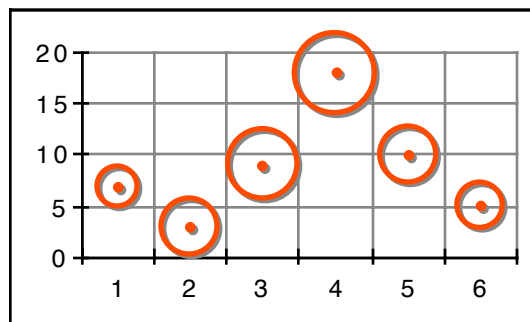
Bei eindimensionalen Blasendiagrammen besteht optional die Möglichkeit, alle Blasen um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen liegen, sondern in der Mitte der Intervalle. Dies wird durch Aktivieren des 2. Arguments *intervalleVerschieben=on* ermöglicht.

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(7 3 9 18 10 5; // Positionen
            3 6 9 12 6 4) // Durchmesser
  BubbleChart(shadow+label;on)
  FillStyle(1;lightBlue)
  BorderStyle(1;;2;blue)
  ShadowStyle(all;1;gray)
  GridLocation(all;none)
CloseDrawing()
```



```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(7 3 9 18 10 5; // Positionen
            3 6 9 12 6 4) // Durchmesser
  BubbleChart(shadow+symbol;on)
  FillStyle(1;;transparent)
  BorderStyle(1;;2;red)
  SymbolStyle(1;bullet;3;;red)
  ShadowStyle(all;1;gray)
CloseDrawing()
```

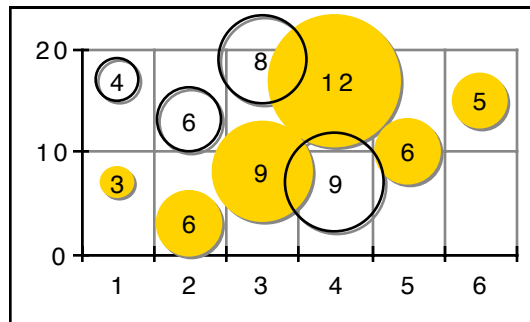


**BubbleChartOptions(maxDurchmesser;typ)**

Mit dem ersten Argument *maxDurchmesser* kann der maximale Blasen-durchmesser kontrolliert werden. Wird kein Wert vorgegeben, so ist standardmäßig der maximale Blasendurchmesser gleich 30 Pixel. Zusätzlich kann noch das Größenverhältnis der Blasen zueinander gesteuert werden, d.h. bei *typ=areaProp* verhalten sich die Flächen der Blasen zueinander proportional, bei *typ=diameterProp* verhalten sich die Durchmesser zueinander proportional. Standardmäßig ist *typ=areaProp*.

Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(17 13 19 7;          // Positionen 1.Serie
          4 6 8 9;            // Durchmesser 1.Serie
          7 3 8 17 10 15;     // Positionen 2.Serie
          3 6 9 12 6 5)      // Durchmesser 2.Serie
BubbleChart(shadow+label;on)
BubbleChartOptions(50;diameterProp)
FillStyle(1;;transparent)
FillStyle(2;darkYellow)
BorderStyle(2;none)
ShadowStyle(all;1;gray)
CloseDrawing()
```



Zu beachten ist, dass die Funktion `BubbleChartOptions()` nach der Funktion `BubbleChart()` bzw. `BubbleChart2D()` anzuführen ist.

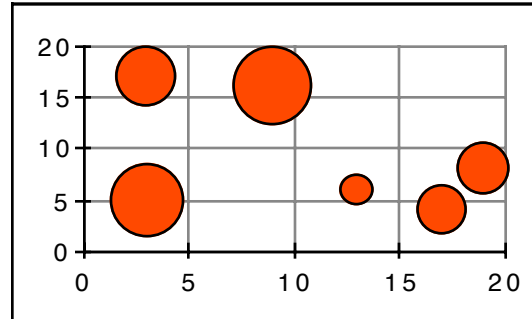
**BubbleChart2D(darstellung)**

Die Funktion `BubbleChart2D()` ermöglicht die Darstellung von zwei-dimensionalen Blasendiagrammen. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die x-Werte, die 2. Werteserie die y-Werte und die 3. Werteserie die Durchmesser zur ersten Serie, die 4. und 5. Werteserie in `ChartData()` die x- und y-Werte zur zweiten Serie, die 6. Werteserie die Durchmesser zur zweiten Serie usw. Beispiel:

```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(17 13 19 9 3 3; // x-Positionen
          4 6 8 16 17 5; // y-Positionen
          7 3 8 17 10 15) // Durchmesser
BubbleChart2D()
CloseDrawing()

```

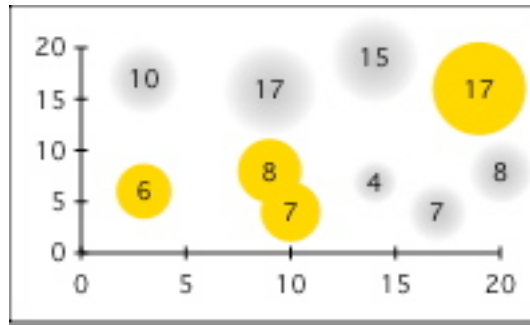


Wie bei den eindimensionalen Blasendiagrammen können auch hier durch das Argument *darstellung* die Blasen mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Die Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Eine gedrehte Darstellung (*darstellung=horizontal*) wird bei zweidimensionalen Diagrammen nicht unterstützt, da dies einfach durch Vertauschen der Werteserien in `ChartData()` möglich ist. Beispiel:

```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(17 14 20 9 3 14; // x-Positionen 1.Serie
          4 7 8 16 17 19; // y-Positionen 1.Serie
          7 4 8 17 10 15; // Durchmesser 1.Serie
          10 3 9 19;      // x-Positionen 2.Serie
          4 6 8 16;       // y-Positionen 2.Serie
          7 6 8 17)       // Durchmesser 2.Serie
BubbleChart2D(label)
BubbleChartOptions(35;areaProp)
PictureStyle(1;resource;"33")
FillStyle(2;darkYellow)
BorderStyle(all;none)
GridLocation(all;none)
CloseDrawing()

```

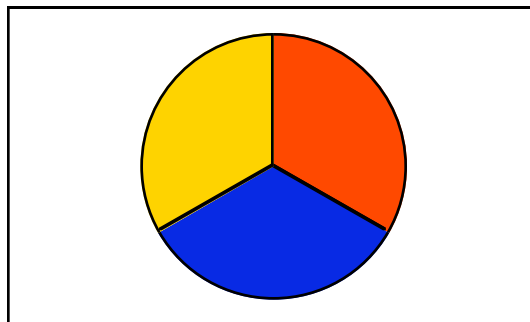


### Kuchendiagramme:

**PieChart(darstellung;3DHöhe;innenradius;startwinkel;  
öffnungswinkel)**

Die Funktion `PieChart()` ermöglicht die Darstellung einer Vielzahl unterschiedlicher Tortendiagramme. Beispiel:

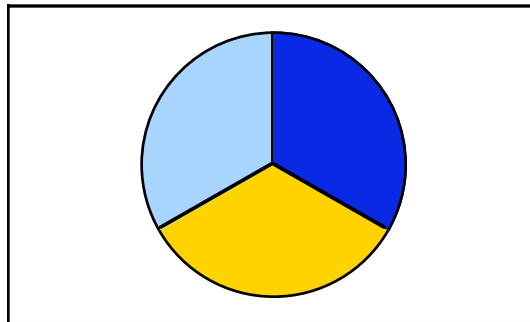
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(135 135 135)
  PieChart()
  FillStyle(3;darkYellow)
CloseDrawing()
```



Dargestellt wird nur die erste Werteserie in `ChartData()`, weitere Werteserien werden ignoriert. Das gleiche gilt für Werte kleiner gleich Null.

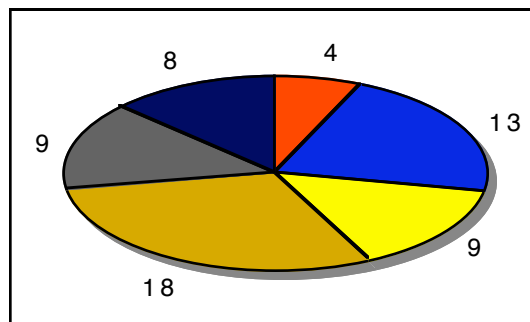
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  // Werte kleiner gleich Null werden übersprungen.
  ChartData(-100 135 135 0 135;
            100 200 300) // Weitere Serien ignoriert.
  PieChart()
  FillStyle(3;darkYellow)
  FillStyle(5;lightBlue)
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht u.a. die Wahl zwischen einer elliptischen Form (*darstellung=oval*), d.h. die gesamte zur Verfügung stehende Fläche kann ausgenutzt werden, oder einer kreisförmigen Darstellung. Standardmäßig werden alle Tortendiagramme kreisförmig dargestellt. Zusätzlich können Diagramme mit Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren. Beispiel:

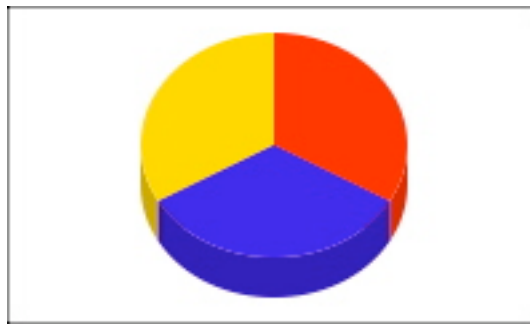
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 13 9 18 9 8)
  PieChart(oval+label+shadow)
CloseDrawing()
```



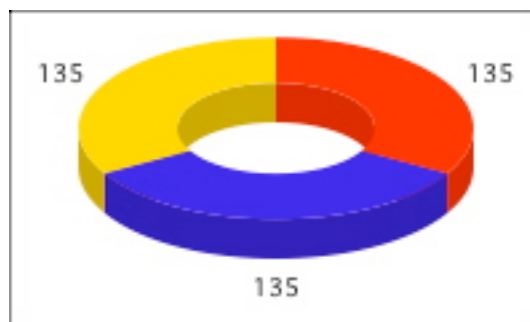
Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden; die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden im Abschnitt *Stile* erläutert.

Das 2. Argument *3DHöhe* ermöglicht die Darstellung dreidimensionaler Tortendiagramme. Die 3DHöhe ist dabei in Prozent der Segmentlänge einzugeben. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(135 135 135)
  PieChart(;30)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```

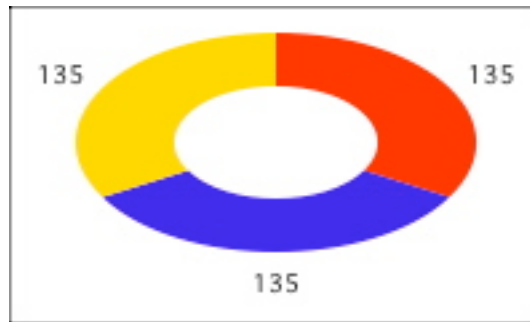


```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(135 135 135)
  PieChart(label+oval;20;50)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```



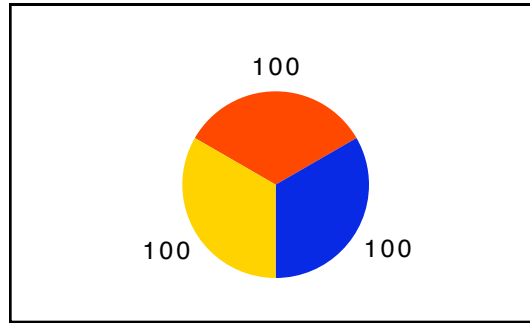
Das 3. Argument *innenradius* dient zur Darstellung von Ringdiagrammen. Der Innenradius ist dabei in Prozent der Segmentlänge einzugeben. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(135 135 135)
  PieChart(label+oval;;50)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```



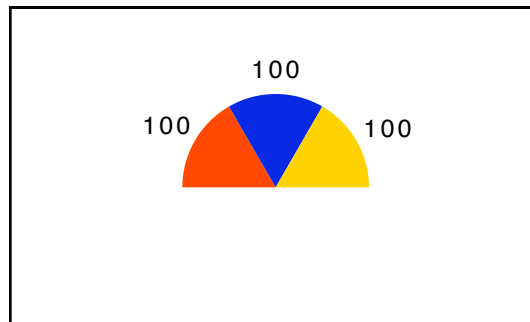
Standardmäßig werden die Segmente eines Tortendiagramms oben beginnend (12 Uhr), im Uhrzeigersinn über einen Bereich von 360 Grad (Vollkreis) aufgetragen. Mit den Argumenten *startwinkel* und *öffnungswinkel* können sowohl die Position des ersten Segments als auch der Richtungssinn und der Öffnungswinkel des Kreissektors festgelegt werden. Durch Vorgabe eines Startwinkels größer Null verschieben sich alle Segmente im Uhrzeigersinn, durch einen negativen Startwinkel gegen den Uhrzeigersinn. Alle Winkel sind im Bereich von  $\pm 360$  Grad anzugeben, dabei ist 0 Grad oben (12 Uhr), 90 Grad ist rechts (3 Uhr) und -90 Grad links (9 Uhr). Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(100 100 100)
  PieChart(label;;;-60)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```

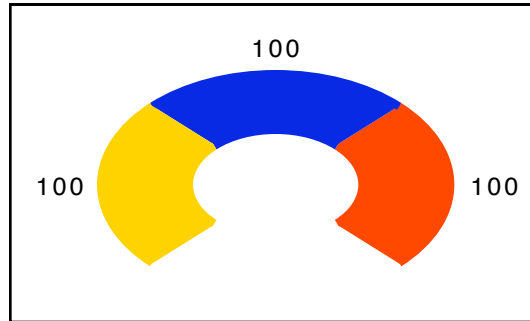


Bei einem Öffnungswinkel ungleich  $\pm 360$  Grad werden die Segmente in Form eines Kreissektors angeordnet. Ein negativer Öffnungswinkel bewirkt, dass die Segmente gegen den Uhrzeigersinn aufgetragen werden. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(100 100 100)
  PieChart(label;;;-90;180)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```



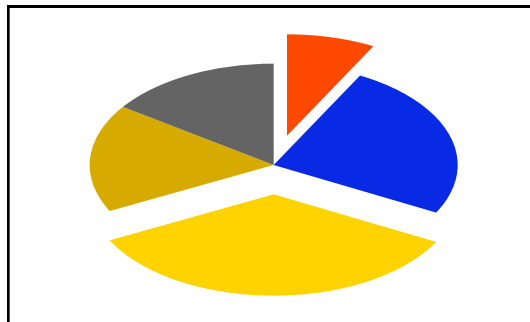
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(100 100 100)
  PieChart(oval+label;;50;135;-270)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```



### **PieChartExplodes(versatz;segment1;segment2...)**

Durch die Funktion `PieChartExplodes()` können Segmente eines Tortendiagramms radial nach aussen versetzt (explodierend) dargestellt werden. Das erste Argument *versatz* definiert die Größe des Versatzes, welcher in Prozent der Segmentlänge anzugeben ist. Falls kein Versatzwert vorgegeben wird, werden die Segmente um 20% der Segmentlänge versetzt (*versatz=20*). Die weiteren Argumente in `PieChartExplodes()` legen fest, welche Segmente versetzt dargestellt werden sollen. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 13 18 9 8)
  PieChart(oval)
  PieChartExplodes(30;1;3)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```

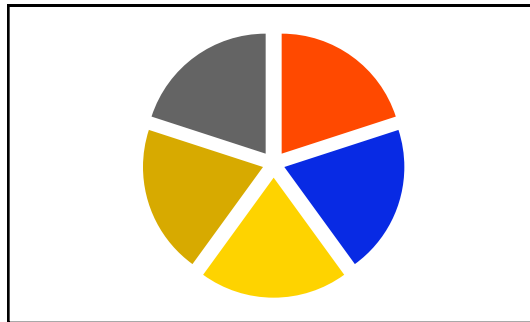


Zusätzlich stehen noch die folgenden Segmentkonstanten zur Verfügung:

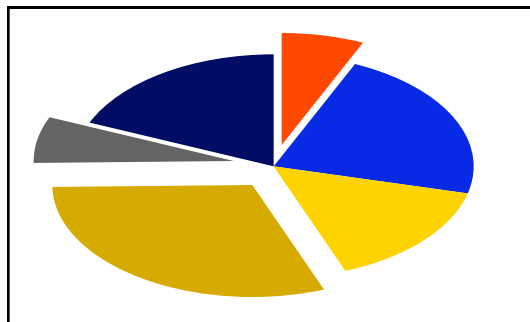
<i>Konstante</i>	<i>Versetzt darstellen</i>
none	kein Segment
all	alle Segmente
max	größtes Segment
min	kleinstes Segment

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(72 72 72 72 72)
  PieChart()
  PieChartExplodes(10;all)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```



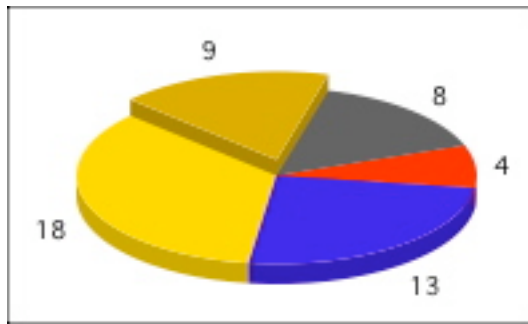
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 13 9 18 4 11)
  PieChart(oval)
  PieChartExplodes(;max;min)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```



**PieChartExplodeDepths(versatz;segment1;segment2...)**

Durch die Funktion `PieChartExplodeDepths()` können Segmente eines 3D-Tortendiagramms vertikal nach oben verschoben werden. Bei 2D-Tortendiagrammen wird die Funktion `PieChartExplodeDepths()` ignoriert. Das erste Argument *versatz* definiert die Größe des Versatzes, welcher in Prozent der Segmenthöhe anzugeben ist. Falls kein Versatzwert vorgegeben wird, werden die Segmente um 20% der Segmenthöhe versetzt (*versatz=20*). Die weiteren Argumente in `PieChartExplodeDepths()` legen fest, welche Segmente vertikal versetzt dargestellt werden sollen. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 13 18 9 8)
  PieChart(label+oval;10;;70)
  PieChartExplodeDepths(80;4)
  FillStyle(3;darkYellow)
  BorderStyle(all;none)
CloseDrawing()
```



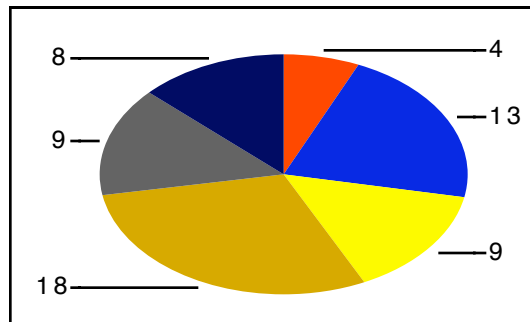
Die beiden Funktionen `PieChartExplodes()` und `PieChartExplodeDepths()` können sowohl mehrfach angeführt als auch kombiniert werden. Dadurch ergibt sich die Möglichkeit, Segmente ganz individuell radial nach aussen und/oder vertikal nach oben zu verschieben.

Zu beachten ist, dass die Funktionen `PieChartExplodes()` und `PieChartExplodeDepths()` nach der Funktion `PieChart()` anzuführen sind. Diese Regel gilt ganz allgemein: *Diagrammoptionen sind stets nach der eigentlichen Diagrammfunktion anzuführen.*

**PieChartAuxLines(horizontaleLänge;extensionLänge;  
vAusrichtung;strichstärke;farbe;muster)**

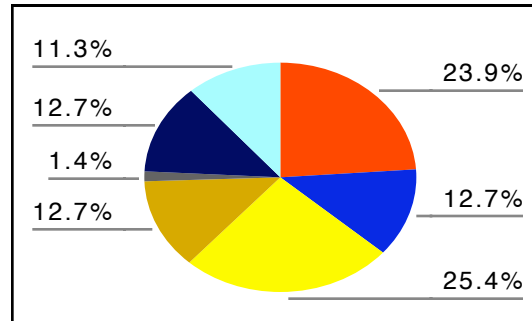
Die Funktion `PieChartAuxLines()` kann zur besseren Platzierung der außenliegenden Beschriftungen verwendet werden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 13 9 18 9 8)
  PieChart(label+oval)
  PieChartAuxLines()
  BorderStyle(all;none)
CloseDrawing()
```



Die horizontale Länge der Hilfslinien kann mit dem Argument *horizontaleLänge* kontrolliert werden. Diese ist in Prozent der Segmentlänge anzugeben. Das Argument *extensionLänge* wird zur Zeit nicht genutzt und wird ignoriert. Mit dem Argument *vAusrichtung* kann die Position der Hilfslinie relativ zur Beschriftung festgelegt werden. Bei *vAusrichtung=bottom* wird die Hilfslinie unterhalb der Beschriftung platziert, bei *vAusrichtung=top* oberhalb, bei *vAusrichtung=center* in der Mitte (default). Die Strichstärke, die Farbe und das Muster können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Beispiele:

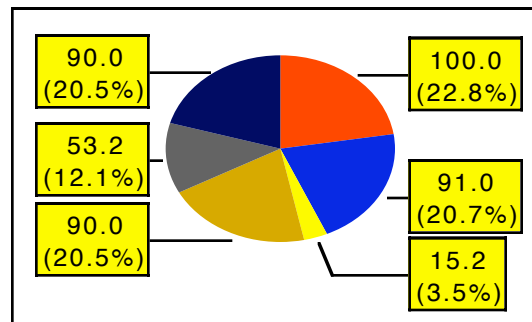
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(17 9 18 9 1 9 8)
  PieChart(label+oval)
  PieChartAuxLines(;;bottom;1;gray)
  LabelTexts("||f1|")
  BorderStyle(all;none)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(100 91 15.2 90 53.2 89.97)
  PieChart(label+oval)
  PieChartAuxLines()
  LabelTexts(";|f1|\n(|f1|%)")
  LabelStyle(;;;;center)
  LabelBackground(1;yellow)
  BorderStyle(all;none)
CloseDrawing()

```



Die formatierte Ausgabe der Werte (absolut und/oder relativ) wird ausführlich inkl. Beispiele im Abschnitt *Stile* erläutert.

### **PieChartLabelOptions(relativeLimitsVerwenden; textversatzAußen;textversatzInnen)**

Das 1. Argument *relativeLimitsVerwenden* ermöglicht anstelle absoluter Grenzwerte, relative Grenzwerte zu definieren: Diagrammwerte, welche nicht innerhalb dieser Limits liegen, werden nicht beschriftet. Die Grenzwerte, d.h. das untere und obere Limit werden in der Funktion `LabelOptions()` festgelegt.

Mit den Argumenten *textversatzAußen* und *textversatzInnen* kann der Abstand zwischen der Berandung und der äußeren bzw. inneren Beschriftung kontrolliert werden. Weitere Informationen dazu im Abschnitt *Stile*.

```

PieChartInnerLabelTexts(text1;text2...)
PieChartInnerLabelStyle(schrift;größe;stil;farbe;
    ausrichtung;neigungswinkel;maxBreite;
    maxHöhe;auslassungsPosition)
PieChartInnerLabelBackground(füllfarbe;füllmuster;
    rahmenbreite;rahmenfarbe;rahmenmuster;
    schattenabstand;schattenfarbe;schattenmuster)

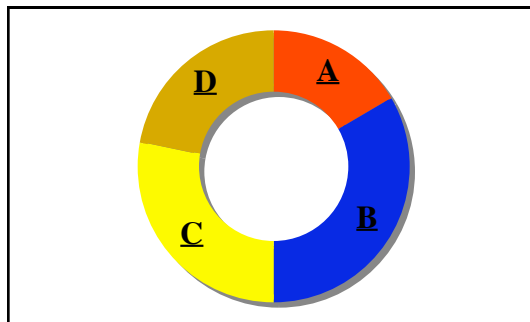
```

Standardmäßig wird die Beschriftung außerhalb des Tortendiagramms platziert (*darstellung=label*). Durch die Funktionen `PieChartInnerLabelTexts()`, `PieChartInnerLabelStyle()` und `PieChartInnerLabelBackground()` können zusätzlich auch innerhalb des Diagramms Beschriftungen angeordnet und gestaltet werden. Der Aufbau und die Wirkungsweise der drei Funktionen ist gleich der "Standard"-Funktionen `LabelTexts()`, `LabelStyle()` und `LabelBackground()`. Letztere werden detailliert im Abschnitt *Stile* behandelt. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(45 90 75 60)
  PieChart(shadow;;60)
  PieChartInnerLabelTexts("A";"B";"C";"D")
  PieChartInnerLabelStyle("Times";12;bold+underline)
  BorderStyle(all;none)
  ShadowStyle(all;2;gray)
CloseDrawing()

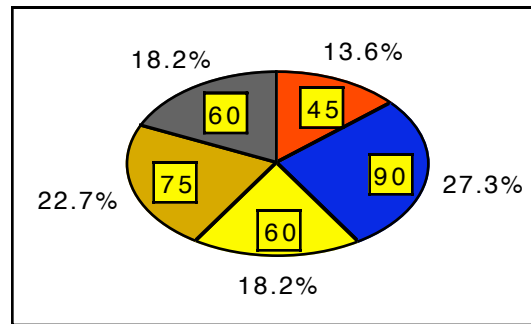
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(45 90 60 75 60)
  PieChart(label+oval)
  LabelTexts(1;"||f1|%" // relative Werte
  PieChartInnerLabelTexts("|u|") // absolute Werte
  PieChartInnerLabelBackground(yellow)
CloseDrawing()

```



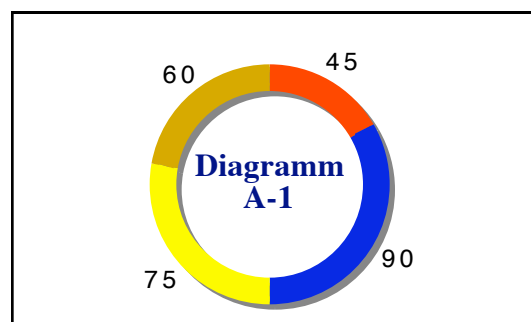
**PieChartCenterLabelText(text)**

**PieChartCenterLabelStyle(schrift;größe;stil;farbe;  
ausrichtung;neigungswinkel;maxBreite;  
maxHöhe;auslassungsPosition)**

**PieChartCenterLabelBackground(füllfarbe;füllmuster;  
rahmenbreite;rahmenfarbe;rahmenmuster;  
schattenabstand;schattenfarbe;schattenmuster)**

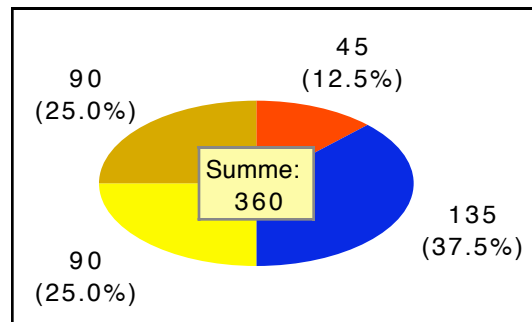
Zusätzlich kann durch die Funktionen `PieChartCenterLabelText()`, `PieChartCenterLabelStyle()` und `PieChartCenterLabelBackground()` ein Text in der Mitte des Diagramms angeordnet und gestaltet werden. Der Aufbau und die Wirkungsweise der drei Funktionen ist gleich der "Standard"-Funktionen `LabelTexts()`, `LabelStyle()` und `LabelBackground()`, siehe Abschnitt *Stile*. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(45 90 75 60)
  PieChart(shadow+label;;85)
  PieChartCenterLabelText("Diagramm\nA-1")
  PieChartCenterLabelStyle("Times";12;bold;darkBlue)
  BorderStyle(all;none)
  ShadowStyle(all;2;gray)
CloseDrawing()
```



Enthält der Text in der Funktion `PieChartCenterLabelText()` eine Formatanweisung, z.B. "`|u|`", so wird die Summe aller positiven Werte ausgegeben. Eine detaillierte Erklärung aller Formatanweisungen inkl. zahlreicher Beispiele ist in *xmReferenz* zu finden. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(45 135 90 90)
PieChart(oval+label)
LabelTexts(1;"|u|\n(|f1|%)")
LabelStyle(;;;;center)
PieChartCenterLabelText("Summe: \n|u|")
PieChartCenterLabelBackground(lightYellow;;1;gray)
BorderStyle(all;none)
CloseDrawing()
```

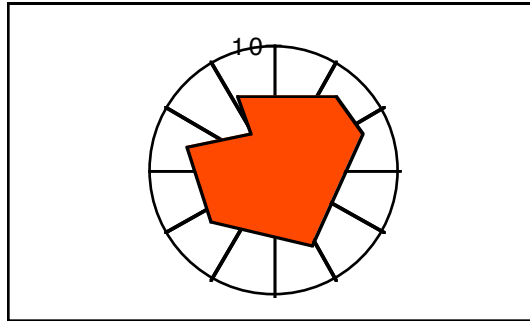


### Polardiagramme:

#### **PolarChart(darstellung;startwinkel;öffnungswinkel)**

Die Funktion `PolarChart()` ermöglicht die Darstellung von zwei-dimensionalen Werten in Polarkoordinaten. Dabei liefert die 1. Werteserie in der Funktion `ChartData()` die x-Werte, die 2. Werteserie die y-Werte zur ersten Serie, die 3. und 4. Werteserie in `ChartData()` die x- und y-Werte zur zweiten Serie usw. Beispiel:

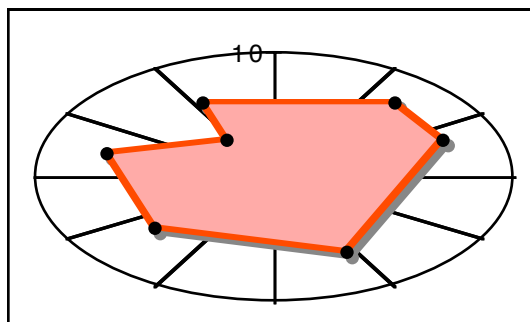
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(5 7 3 -5 -7 -2 -3; // x-Werte
          6 3 -6 -4 2 3 6) // y-Werte
PolarChart()
MajorGridLineColors(all;all;black)
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht u.a. die Wahl zwischen einer elliptischen Form (*darstellung=oval*), d.h. die gesamte zur Verfügung stehende Fläche kann ausgenutzt werden, oder einer kreisförmigen Darstellung. Standardmäßig werden Polardiagramme kreisförmig dargestellt. Zusätzlich können Diagramme mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren.

Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden im Abschnitt *Stile* erläutert. Beispiele:

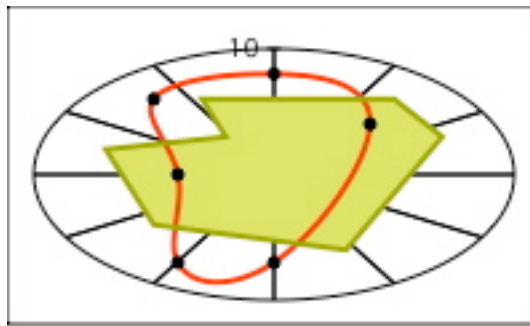
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 7 3 -5 -7 -2 -3; // x-Werte
            6 3 -6 -4 2 3 6) // y-Werte
  PolarChart(oval+shadow+symbol)
  FillStyle(1;lightRed)
  BorderStyle(1;poly;2;red)
  SymbolStyle(1;bullet;4;1;black)
  ShadowStyle(1;2;gray)
  MajorGridLineColors(all;all;black)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 7 3 -5 -7 -2 -3; // x-Werte 1.Serie
            6 3 -6 -4 2 3 6; // y-Werte 1.Serie
            0 4 0 -4 -4 -5; // x-Werte 2.Serie
            8 4 -7 -7 0 6) // y-Werte 2.Serie
  PolarChart(oval+symbol)
  // 1. Serie
  FillStyle(1;220 220 100)
  BorderStyle(1;poly;2;150 150 0)
  SymbolStyle(1;none)
  // 2. Serie
  FillStyle(2;;transparent)
  BorderStyle(2;smooth;2;red)
  SymbolStyle(2;bullet;4;1;black)
  MajorGridLineColors(all;all;black)
CloseDrawing()

```

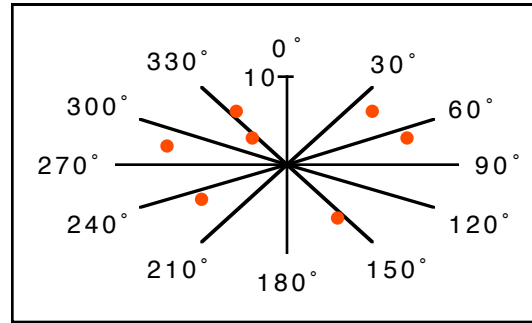


Standardmäßig werden die Punkte einer Serie durch eine polygonal begrenzte Fläche dargestellt. Durch Ausblenden sowohl der Füllung als auch der Berandung mittels der Funktionen `FillStyle(;;transparent)` und `BorderStyle(;none)` ist es möglich, die Diagrammwerte ausschließlich durch Symbole darzustellen. Beispiel:

```

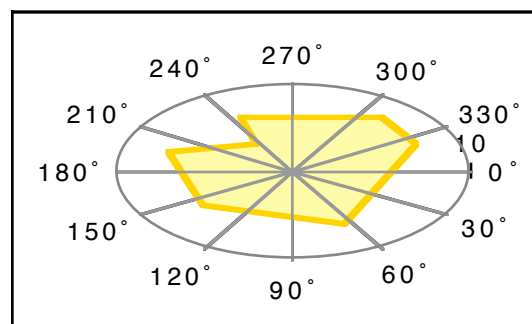
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 7 3 -5 -7 -2 -3; // x-Werte
            6 3 -6 -4 2 3 6) // y-Werte
  PolarChart(oval+symbol)
  FillStyle(;;transparent)
  BorderStyle(;none)
  SymbolStyle(1;bullet;4;1;red)
  AxisLabelText(all;"|u|°")
  GridLocation(all;none) // kein Raster
CloseDrawing()

```



Die Texte der Achsenbeschriftungen werden mit der Funktion `AxisLabelText()` definiert; Details dazu finden sich im Abschnitt *Achsen*. Standardmäßig beginnt die Achsenbeschriftung oben (12 Uhr) und verläuft im Uhrzeigersinn über einen Bereich von 360 Grad. Mit den Argumenten *startwinkel* und *öffnungswinkel* können sowohl die Position der "Nulllinie" als auch der Richtungssinn und der Öffnungswinkel des Rasters festgelegt werden. Durch Vorgabe eines Startwinkels größer Null verschieben sich alle Achsen im Uhrzeigersinn, durch einen negativen Startwinkel gegen den Uhrzeigersinn. Alle Winkel sind im Bereich von  $\pm 360$  Grad anzugeben, dabei ist 0 Grad oben (12 Uhr), 90 Grad ist rechts (3 Uhr) und -90 Grad links (9 Uhr). Beispiele:

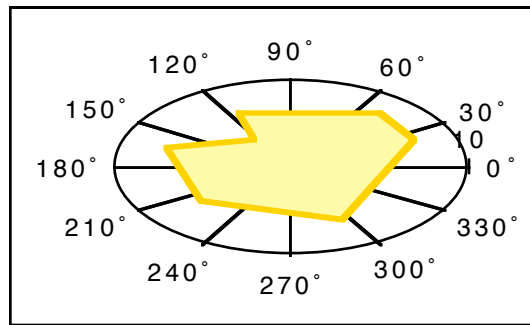
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 7 3 -5 -7 -2 -3; // x-Werte
            6 3 -6 -4 2 3 6) // y-Werte
  // 90...0-Grad-Linie liegt "rechts"
  PolarChart(oval;90)
  FillStyle(1;lightYellow)
  BorderStyle(1;poly;2;darkYellow)
  GridLocation(all;front)
  AxisLabelText(all;"|u|°")
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 7 3 -5 -7 -2 -3; // x-Werte
            6 3 -6 -4 2 3 6) // y-Werte
  // 90...0-Grad-Linie liegt "rechts"
  // -360...Skalierung gegen den Uhrzeigersinn
  PolarChart(oval;90;-360)
  FillStyle(1;lightYellow)
  BorderStyle(1;poly;2;darkYellow)
  MajorGridLineColors(all;all;black)
  AxisLabelText(all;"|u|°")
CloseDrawing()

```



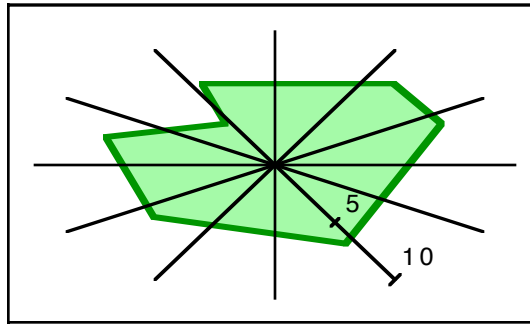
**PolarChartOptions(skalenachse;rasterform;  
pfeileHinzufügen;polygonNichtSchließen;achsenanzahl)**

Polardiagramme können durch die Funktion `PolarChartOptions()` auf vielfältige Weise variiert werden. Zu beachten ist, dass die Funktion `PolarChartOptions()` nach der Funktion `PolarChart()` anzuführen ist. Mit dem ersten Argument *skalenachse* kann festgelegt werden, entlang welcher Achse die Skalierung aufgetragen wird. Standardmäßig wird die Skalierung der ersten Achse zugeordnet (*skalenachse=1*), dies ist die Achse bei Null Grad. Bei *skalenachse=0* wird keine Skala ausgegeben. Beispiel:

```

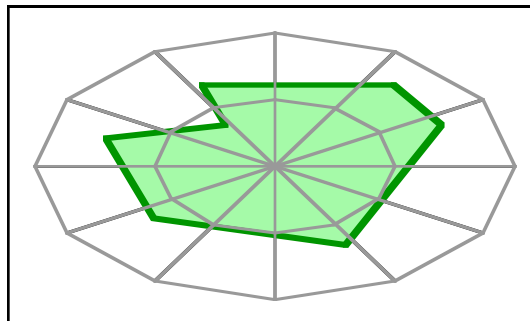
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 7 3 -5 -7 -2 -3; // x-Werte
            6 3 -6 -4 2 3 6) // y-Werte
  PolarChart(oval)
  PolarChartOptions(6) // nach PolarChart(!)
  AxisOptions(all;front)
  FillStyle(1;lightGreen)
  BorderStyle(1;poly;2;green)
  GridLocation(all;none) // kein Raster
CloseDrawing()

```



Mit dem zweiten Argument *rasterform* kann zwischen dem standardmäßigen ovalen Raster (*rasterform=oval*), einem polygonalen Raster (*rasterform=poly*) und einem ausgeblendeten Raster (*rasterform=none*) gewählt werden. Das Raster kann auch durch die Funktion `GridLocation(;none)` ausgeblendet werden. Beispiel:

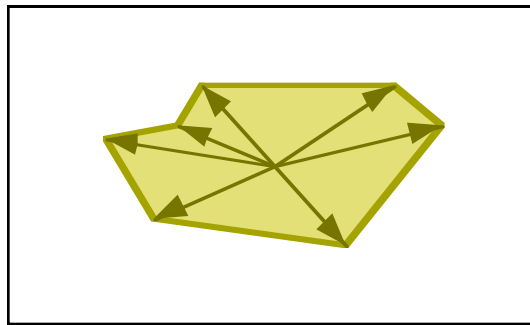
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 7 3 -5 -7 -2 -3; // x-Werte
            6 3 -6 -4 2 3 6) // y-Werte
  PolarChart(oval)
  PolarChartOptions(0;poly) // nach PolarChart()!
  FillStyle(1;lightGreen)
  BorderStyle(1;poly;2;green)
  GridLocation(all;front)
CloseDrawing()
```



Optional können Polardiagrammen durch vom Zentrum ausgehende Pfeile ergänzt werden (*pfeileHinzufügen=on*). Die Gestaltung der Pfeile erfolgt durch die Funktion `ArrowStyle()` und wird im Abschnitt *Stile* behandelt.

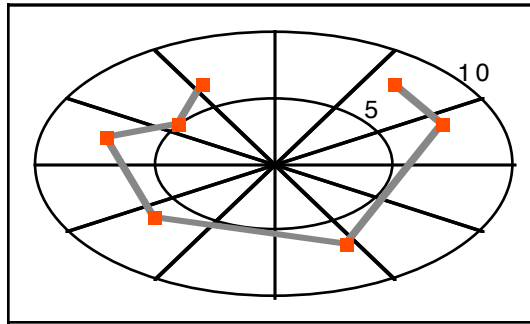
Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(5 7 3 -5 -7 -4 -3; // x-Werte
          6 3 -6 -4 2 3 6) // y-Werte
PolarChart(oval)
PolarChartOptions(0;none;on) // nach PolarChart(!
FillStyle(1;220 220 100)
BorderStyle(1;poly;2;150 150 0)
ArrowStyle(1;1;100 100 0;;;12;8)
AxisOptions(all;none) // keine Achsen
CloseDrawing()
```



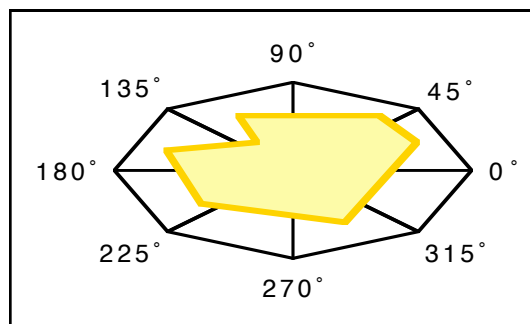
Durch die Funktion `FillStyle(;;transparent)` kann die Füllung ausgeblendet werden; die so verbleibende Berandung verbindet die Punkte in Form eines geschlossenen Linienzuges. Durch Aktivieren des 4. Argumentes *polygonNichtSchließen=on* kann die Schlusslinie zwischen dem letzten und ersten Punkt ausgeblendet werden. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(5 7 3 -5 -7 -4 -3; // x-Werte
          6 3 -6 -4 2 3 6) // y-Werte
PolarChart(oval+symbol)
PolarChartOptions(3;oval;off;on) // nach PolarChart()
FillStyle(1;;transparent)
BorderStyle(1;poly;2;gray)
SymbolStyle(1;square;4;1;red)
AxisMajorTicks(1;0)
MajorGridLineColors(all;all;black)
CloseDrawing()
```



Die Anzahl der radialen Rasterlinien (Achsen) kann mit dem 5. Argument *achsenanzahl* kontrolliert werden. Standardmäßig werden Polardiagramme mit 12 radialen Rasterlinien dargestellt. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 7 3 -5 -7 -2 -3; // x-Werte
            6 3 -6 -4 2 3 6) // y-Werte
  // 90...0-Grad-Linie liegt "rechts"
  // -360...Skalierung gegen den Uhrzeigersinn
  PolarChart(oval;90;-360)
  PolarChartOptions(0;poly;off;;8) // nach PolarChart()
  FillStyle(1;lightYellow)
  BorderStyle(1;poly;2;darkYellow)
  MajorGridLineColors(all;all;black)
  AxisLabelText(all;"|u|°")
CloseDrawing()
```

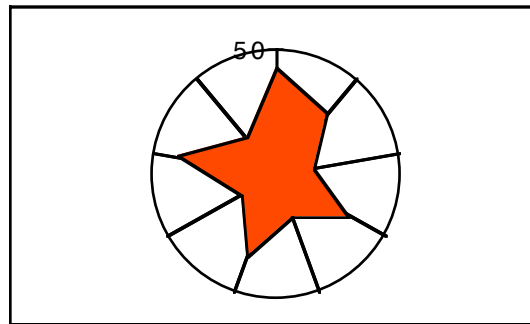


**Radardiagramme:**

**RadarChart(darstellung;startwinkel;öffnungswinkel;  
intervalleVerschieben)**

Bei der Funktion RadarChart() werden die Diagrammwerte entlang strahlenförmig ausgerichteter Achsen angeordnet. Die Anzahl der Achsen wird durch die Anzahl der Werte bestimmt. Die einzelnen Werte einer Serie werden standardmäßig zu einer polygonal begrenzten Fläche verbunden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(43 32 15 33 18 35 16 40 19)
  RadarChart()
  MajorGridLineColors(all;all;black)
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht u.a. die Wahl zwischen einer elliptischen Form (*darstellung=oval*), d.h. die gesamte zur Verfügung stehende Fläche kann ausgenutzt werden, oder einer kreisförmigen Darstellung. Standardmäßig werden Radardiagramme kreisförmig dargestellt. Zusätzlich können Diagramme mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*) ergänzt werden. Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren.

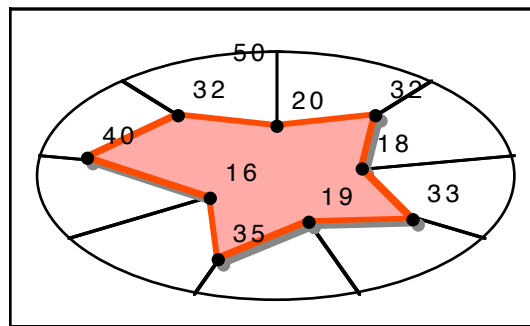
Die Darstellung der Füllungen, Berandungen, Symbole und Schatten kann durch die Stilfunktionen FillStyle(), PictureStyle(), BorderStyle(), SymbolStyle() und ShadowStyle() variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen LabelTexts(), LabelStyle(), LabelBackground() und LabelOptions(). Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* erläutert.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(20 32 18 33 19 35 16 40 32)
  RadarChart(oval+symbol+label+shadow)
  FillStyle(1;lightRed)
  BorderStyle(1;poly;2;red)
  ShadowStyle(1;2;gray)
  SymbolStyle(1;bullet;4;;black)
  MajorGridLineColors(all;all;black)
CloseDrawing()

```



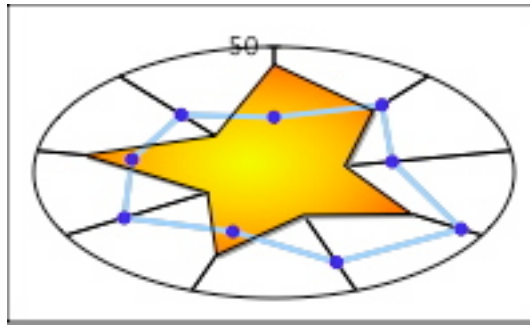
```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(43 32 15 33 18 35 16 40 19; // 1.Serie
            22 35 25 45 38 25 36 30 30) // 2.Serie
  RadarChart(oval+symbol+shadow)
  // 1. Serie
  PictureStyle(1;resource;"41")
  SymbolStyle(1;none)
  ShadowStyle(1;1;gray)

  // 2. Serie
  FillStyle(2;;transparent)
  BorderStyle(2;poly;2;lightBlue)
  SymbolStyle(2;bullet;4;1;)
  ShadowStyle(2;0)

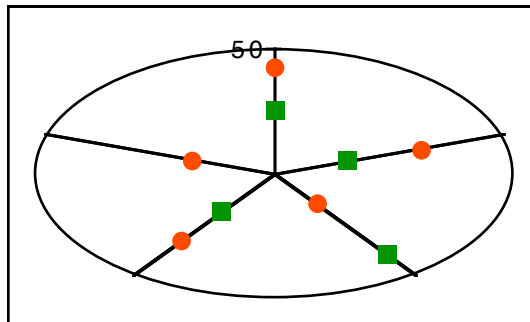
  MajorGridLineColors(all;all;black)
CloseDrawing()

```



Durch Ausblenden der Füllung und der Berandung ist es möglich, die Diagrammwerte ausschließlich durch Symbole darzustellen. Beispiel:

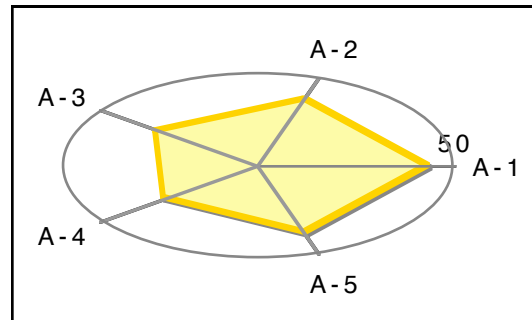
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(43 32 15 33 18; // 1. Serie
            25 16 40 19)   // 2. Serie
  RadarChart(oval+symbol)
  FillStyle(;;transparent)
  BorderStyle(;;none)
  SymbolStyle(1;bullet;6;1;red)
  SymbolStyle(2;square;6;1;green)
  MajorGridLineColors(all;all;black)
CloseDrawing()
```



Die Texte der Achsenbeschriftungen werden mit der Funktion `AxisLabelText()` definiert, Details dazu finden sich im Abschnitt *Achsen*. Der Achsenindex ist bei Radardiagrammen stets 1, Achsenindizes größer 1 werden ignoriert. Standardmäßig beginnt die Achsenbeschriftung oben (12 Uhr) und verläuft im Uhrzeigersinn über einen Bereich von 360 Grad. Mit den Argumenten *startwinkel* und *öffnungswinkel* können sowohl die Position der ersten Achse als auch der Richtungssinn und der Öffnungswinkel der Achsen festgelegt werden. Durch Vorgabe eines Startwinkels größer Null verschieben sich alle Achsen im Uhrzeigersinn, durch einen

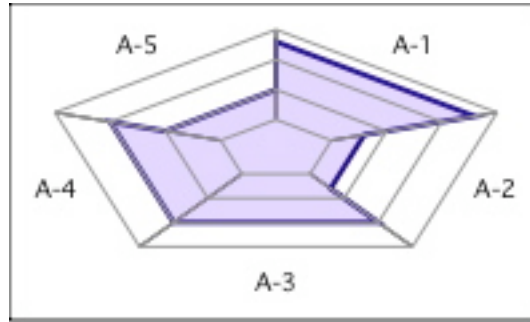
negativen Startwinkel gegen den Uhrzeigersinn. Alle Winkel sind im Bereich von  $\pm 360$  Grad anzugeben, dabei ist 0 Grad oben (12 Uhr), 90 Grad ist rechts (3 Uhr) und -90 Grad links (9 Uhr). Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(43 38 32 30 38)
  RadarChart(oval+shadow;90;-360)
  FillStyle(1;lightYellow)
  BorderStyle(1;poly;2;darkYellow)
  ShadowStyle(1;1;gray)
  GridLocation(all;front)
  AxisMajorTicks(;0)
  AxisLabelText(all;"A-|u|")
CloseDrawing()
```



Durch Aktivieren des 4. Arguments *intervalleVerschieben=on* werden die darzustellenden Werte nicht punktförmig entlang der radial verlaufenden Achsen aufgetragen, sondern keilförmig zwischen den Achsen. Zweckmäßigerweise wird anstelle der gekrümmten Rasterlinien ein polygonales Raster verwendet. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(18 8 15 15 10)
  RadarChart(oval;;;on)
  RadarChartOptions(0;poly) // nach RadarChart()!
  FillStyle(1;50 0 255 50)
  BorderStyle(1;poly;2;darkBlue)
  GridLocation(all;front)
  Scaling(x;linear;5;20;3)
  AxisLabelText(all;"A-|u|")
CloseDrawing()
```

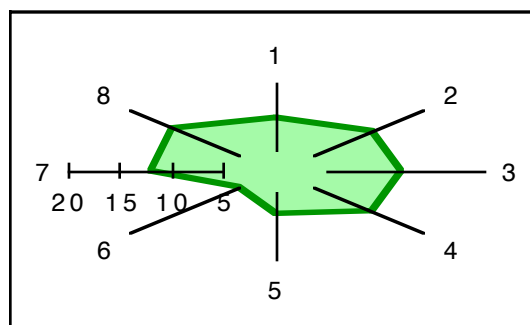


**RadarChartOptions(skalenachse;rasterform;  
pfeileHinzufügen;polygonNichtSchließen)**

Radardiagramme können durch die Funktion RadarChartOptions() auf vielfältige Weise variiert werden. Zu beachten ist, dass die Funktion RadarChartOptions() nach der Funktion RadarChart() anzuführen ist.

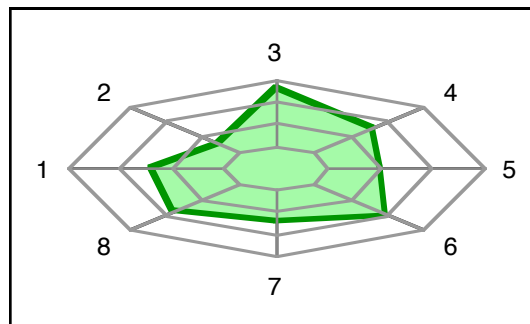
Mit dem ersten Argument *skalenachse* kann festgelegt werden, entlang welcher Achse die Skalierung aufgetragen wird. Standardmäßig wird die Skalierung der ersten Achse zugeordnet (*skalenachse=1*). Bei *skalenachse=0* wird keine Skala ausgegeben. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(12 13 12 13 10 5 12 14)
  RadarChart(oval)
  RadarChartOptions(7) // nach RadarChart(!)
  AxisOptions(all;front)
  FillStyle(1;lightGreen)
  BorderStyle(1;poly;2;green)
  GridLocation(all;none)
  Scaling(x;linear;5;20;3)
  AxisLabelText(all;"|u|")
CloseDrawing()
```

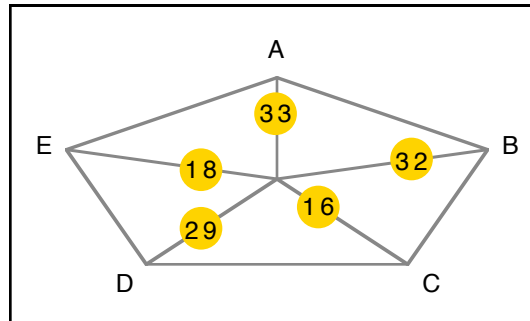


Mit dem zweiten Argument *rasterform* kann zwischen dem standardmäßigen ovalen Raster (*rasterform=oval*), einem polygonalen Raster (*rasterform=poly*) und einem ausgeblendeten Raster (*rasterform=none*) gewählt werden. Das Raster kann auch durch die Funktion *GridLocation( ;none)* ausgeblendet werden. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(12 8 18 13 10 15 12 14)
  RadarChart(oval;-90)
  RadarChartOptions(0;poly) // nach RadarChart(!
  FillStyle(1;lightGreen)
  BorderStyle(1;poly;2;green)
  GridLocation(all;front)
  Scaling(x;linear;5;20;3)
  AxisLabelText(all;"|u|")
CloseDrawing()
```

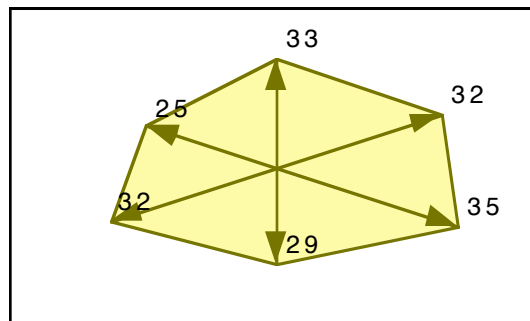


```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(33 32 16 29 18)
  RadarChart(oval+label+symbol)
  RadarChartOptions(0;poly) // nach RadarChart(!
  FillStyle(1;;transparent)
  BorderStyle(1;none)
  SymbolStyle(1;bullet;15;;darkYellow)
  LabelOptions(1;centerCenter)
  AxisLine(1;0) // keine Achsenlinien
  AxisLabelText(1;"A";"B";"C";"D";"E")
CloseDrawing()
```

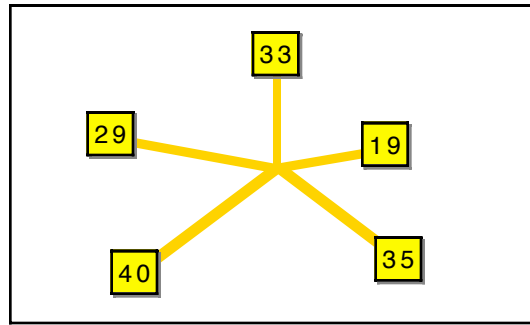


Optional können Radardiagrammen durch vom Zentrum ausgehende Pfeile ergänzt werden (*pfeileHinzufügen=on*). Die Gestaltung der Pfeile erfolgt durch die Funktion `ArrowStyle()` und wird im Abschnitt *Stile* erläutert. Beispiele:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(33 32 35 29 32 25)
RadarChart(oval+label)
RadarChartOptions(0;none;on) // nach RadarChart(!
FillStyle(1;lightYellow)
BorderStyle(1;;1;100 100 0)
ArrowStyle(1;1;100 100 0;;;12;8)
AxisOptions(all;none) // keine Achsen
CloseDrawing()
```



```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(33 19 35 40 29)
RadarChart(oval+label)
RadarChartOptions(0;none;on) // nach RadarChart(!
FillStyle(1;;transparent)
BorderStyle(1;none)
LabelBackground(1;yellow;;;1;black;;1)
LabelOptions(1;centerCenter)
ArrowStyle(1;3;darkYellow;;;0;0)
AxisOptions(all;none) // keine Achsen
CloseDrawing()
```

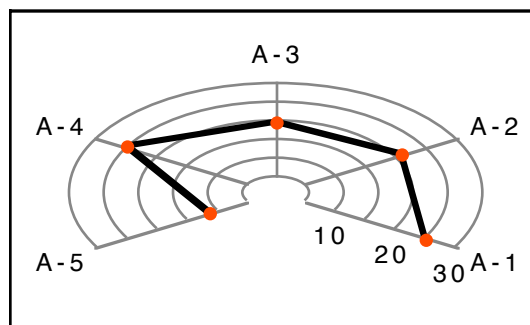


Durch die Funktion `FillStyle(;;transparent)` kann die Füllung ausgeblendet werden, die so verbleibende Berandung verbindet die Punkte in Form eines geschlossenen Linienzuges. Durch Aktivieren des 4. Arguments `polygonNichtSchließen=on` kann die Schlusslinie zwischen dem letzten und ersten Punkt ausgeblendet werden. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(25 21 19 25 11)
  RadarChart(symbol+oval;120;-240)
  RadarChartOptions(1;oval;off;on) // nach RadarChart()

  FillStyle(all;;transparent)
  BorderStyle(1;poly;2;black)
  SymbolStyle(1;bullet;4;1;red)

  Scaling(1;linear;5;30;5)
  AxisLabelText(1;"A-|i0|")
  AxisLine(1;0) // keine Achsenlinien
  AxisMajorTicks(1;0) // keine Skalen-Markierungen
  AxisMajorTickLabelOptions(1;in;;;2;2)
CloseDrawing()
```



```

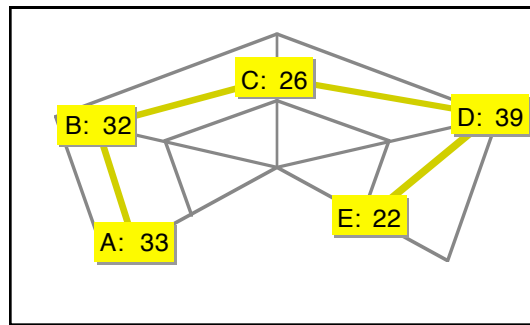
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(33 32 26 39 22)
  RadarChart(oval+label;-135;270)
  RadarChartOptions(0;poly;off;on) // nach RadarChart()

  LabelTexts(1;"A: |u|";"B: |u|";"C: |u|";
             "D: |u|";"E: |u|")
  LabelBackground(1;yellow;;0;;1)
  LabelOptions(1;centerCenter)

  FillStyle(1;;transparent)
  BorderStyle(1;;2;200 200 0)

  AxisOptions(all;none) // keine Achsen
CloseDrawing()

```



### HochTief-Diagramme:

```

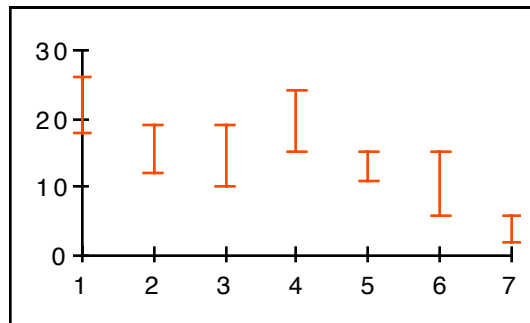
HighLowChart(darstellung;intervalleVersch.;diagrammTyp;
             hochMarkerLänge;hochMarkerAusrichtung;
             tiefMarkerLänge;tiefMarkerAusrichtung;
             schlussMarkerLänge;schlussMarkerAusrichtung;
             öffnenMarkerLänge;öffnenMarkerAusrichtung)

```

Die Funktion `HighLowChart()` ermöglicht die vielfältige Darstellung von Wertebereichen. Zur Festlegung der Bereichsgrenzen sind mindestens zwei Werteserien notwendig. Die Höchstwerte werden in der Funktion `ChartData()` als 1. Werteserie, die Tiefstwerte als 2. Werteserie angegeben.

Beispiel:

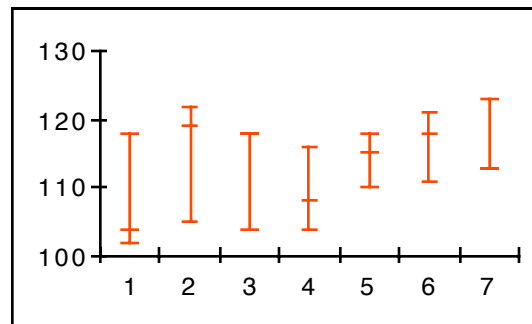
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(26 19 19 24 15 15 6; // Höchstwerte
            18 12 10 15 11 6 2) // Tiefstwerte
  HighLowChart()
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



Optional können unter Verwendung des 3. Arguments *diagrammTyp*, neben den standardmäßig verwendeten zwei Werten (*diagrammTyp=highLow*), auch drei Werte (*diagrammTyp=highLowClose*) oder auch vier Werte (*diagrammTyp=highLowCloseOpen*) angegeben werden. Da diese Art der Darstellung häufig für Börsenkurse verwendet wird, werden die vier Werte als Höchst-, Tiefst-, Schluss- und Eröffnungskurs bezeichnet. Die Schlusskurse werden in `ChartData()` als 3. Werteserie, die Eröffnungskurse als 4. Werteserie angeführt.

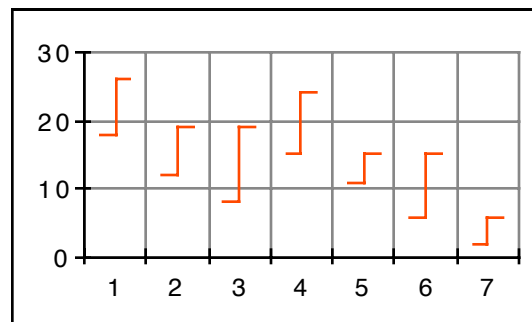
Weiters besteht die Möglichkeit, die Bereichslinien um die halbe Intervallbreite zu verschieben, so dass diese nicht an den Intervallgrenzen liegen, sondern in der Mitte der Intervalle. Dies wird durch Aktivieren des 2. Arguments *intervalleVerschieben=on* ermöglicht. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(118 122 118 116 118 121 123; //Höchstwerte
            102 105 104 104 110 111 113; //Tiefstwerte
            104 119 118 108 115 118 113) //Schlusswerte
  HighLowChart(;on;highLowClose)
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



Die Ausrichtung der Markierungen — die Markierungen für die Tiefst- und Eröffnungskurse werden üblicherweise nach links ausgerichtet, die Markierungen für die Höchst- und Schlusskurse nach rechts — wird mit den Argumenten *hochMarkerAusrichtung*, *tiefMarkerAusrichtung*, *schlussMarkerAusrichtung* und *öffnenMarkerAusrichtung* festgelegt. Alle Markierungen können wahlweise links, rechts oder mittig (default) angeordnet werden. Die Länge der Markierungen kann mit den Argumenten *hochMarkerLänge*, *tiefMarkerLänge*, *schlussMarkerLänge* und *öffnenMarkerLänge* kontrolliert werden und ist in Prozent der Intervallbreite einzugeben. Beispiele:

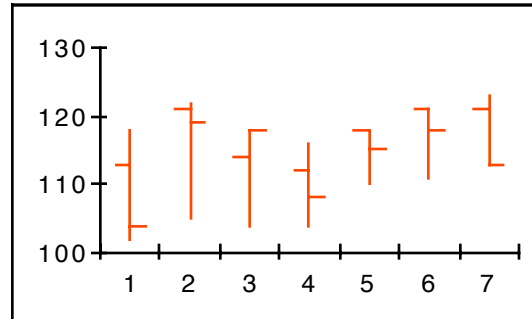
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(26 19 19 24 15 15 6; // Höchstwerte
            18 12 8 15 11 6 2) // Tiefstwerte
  HighLowChart(;on;highLow;;right;;left)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(118 122 118 116 118 121 123; //Höchstwerte
            102 105 104 104 110 111 113; //Tiefstwerte
            104 119 118 108 115 118 113; //Schlusswerte
            113 121 114 112 118 121 121) //Eröffnungsw.
  HighLowChart(;on;highLowCloseOpen;0;;0;;right;;left)
  GridLocation(all;none) // kein Raster
CloseDrawing()

```



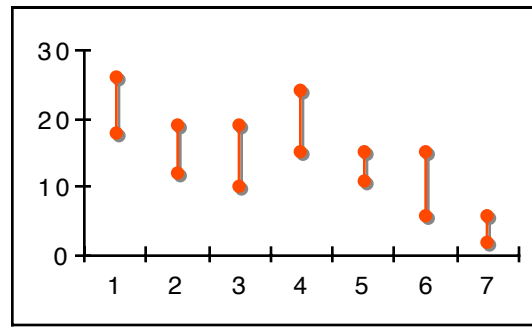
Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung mit Symbolen (*darstellung=symbol*), Schatten (*darstellung=shadow*) und Zahlenwerten (*darstellung=label*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" beliebig kombinieren.

Die Darstellung der Linien, Symbole und Schatten kann durch die Stilfunktionen `LineStyle()`, `SymbolStyle()` und `ShadowStyle()` variiert werden, die Darstellung der Zahlenwerte durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()` und `LabelOptions()`. Alle Stilfunktionen werden ausführlich im Abschnitt *Stile* behandelt. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(26 19 19 24 15 15 6; // Höchstwerte
            18 12 10 15 11 6 2) // Tiefstwerte
  HighLowChart(shadow+symbol;on;;0;;0)
  GridLocation(all;none) // kein Raster
  SymbolStyle(1;bullet;4)
  ShadowStyle(1;1;gray)
CloseDrawing()

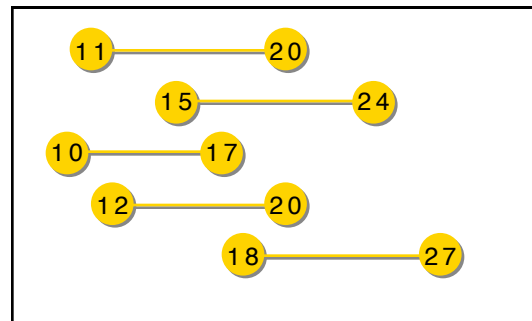
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(27 20 17 24 20; // Höchstwerte
            18 12 10 15 11) // Tiefstwerte
  HighLowChart(horizontal+shadow+label+symbol)
  LineStyle(1;poly;1;darkYellow)
  SymbolStyle(1;bullet;15;;darkYellow)
  ShadowStyle(1;1;gray)
  LabelOptions(1;centerCenter)
  GridLocation(all;none) // kein Raster
  AxisOptions(all;none) // keine Achsen
CloseDrawing()

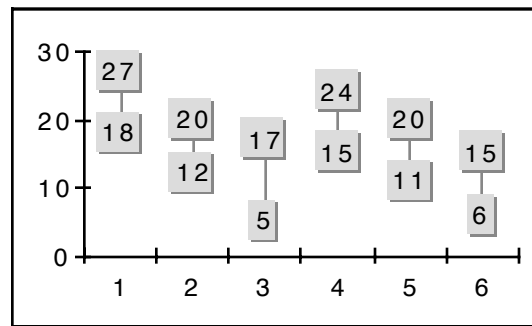
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(27 20 17 24 20 15; // Höchstwerte
            18 12 5 15 11 6) // Tiefstwerte
  HighLowChart(label;on)
  GridLocation(all;none) // kein Raster
  LineStyle(1;poly;1;gray)
  LabelBackground(all;lightGray;;0;;1)
  LabelOptions(all;centerCenter)
CloseDrawing()

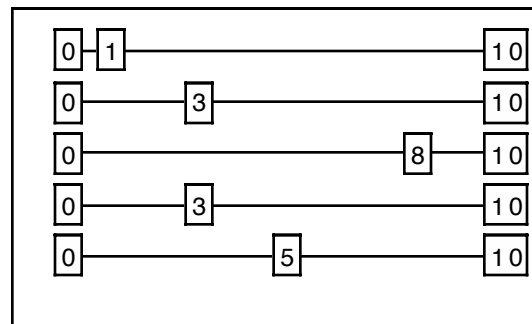
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(10 10 10 10 10; // Höchstwerte
          0 0 0 0 0; // Tiefstwerte
          5 3 8 3 1) // "Schlusswerte"
HighLowChart(label+horizontal;;highLowClose)
LineStyle(1;poly;1;black)
LabelBackground()
LabelOptions(all;centerCenter)
GridLocation(all;none) // kein Raster
AxisOptions(all;none) // keine Achsen
CloseDrawing()

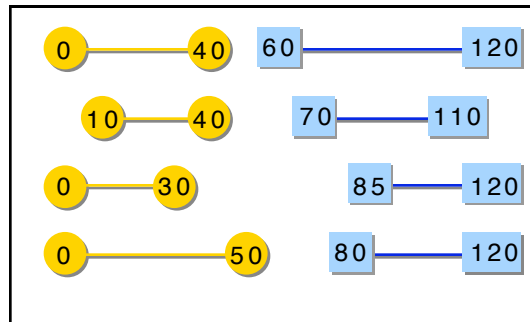
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(50 30 40 40; // Max. 1.Serie
            0 0 10 0; // Min. 1.Serie
            120 120 110 120; // Max. 2.Serie
            80 85 70 60) // Min. 2.Serie
  HighLowChart(label+horizontal+symbol+shadow;;;0;;0)
  LineStyle(1;poly;1;darkYellow)
  SymbolStyle(1;bullet;15;;darkYellow)
  ShadowStyle(all;1;gray)
  SymbolStyle(2;none)
  LabelBackground(2;lightBlue;;;0;;1)
  LabelOptions(all;centerCenter)
  GridLocation(all;none) // kein Raster
  AxisOptions(all;none) // keine Achsen
CloseDrawing()

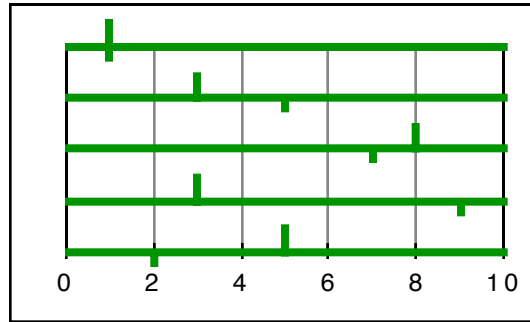
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(10 10 10 10 10; // Höchstwerte
            0 0 0 0 0; // Tiefstwerte
            5 3 8 3 1; // "Schlusswerte"
            2 9 7 5 1) // "Eröffnungswerte"
  HighLowChart(horizontal;off;highLowCloseOpen;
               0;; // Max.
               0;; // Min.
               50;left; // Schluss
               20;right) // Öffnen
  AxisOptions(y;none) // keine y-Achse
  LineStyle(1;poly;3;green)
  GridFrame()
CloseDrawing()

```

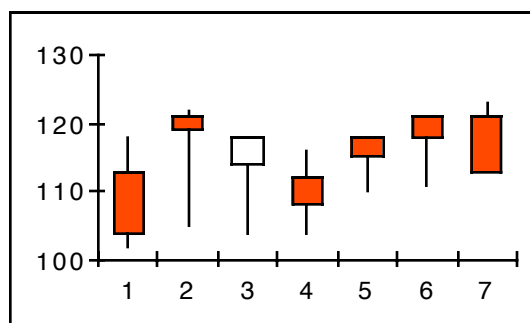


### Candlestick-Diagramme:

**CandlestickChart(darstellung;abstand;hochMarkerLänge;  
hochMarkerAusrichtung;tiefMarkerLänge;  
tiefMarkerAusrichtung)**

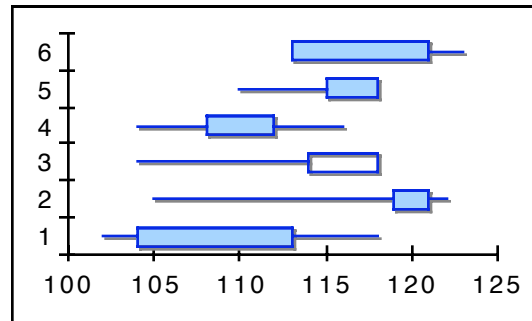
Die Funktion `CandlestickChart()` wird zur Darstellung von Börsenkursen verwendet. Die dazu in der Funktion `ChartData()` benötigten vier Werteserien entsprechen den Höchst-, Tiefst-, Schluss- und Eröffnungswerten. Dabei wird der höchste und der tiefste Kurswert durch eine Linie verbunden und der Bereich zwischen Eröffnungs- und Schlusskurs durch ein Rechteck dargestellt. Ist der Schlusskurs niedriger als der Eröffnungskurs, so wird die Rechteckfläche gefüllt dargestellt, umgekehrt, ist der Schlusskurs höher als der Eröffnungskurs, so wird die Rechteckfläche nicht gefüllt. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(118 122 118 116 118 121 123; //Höchstwerte
          102 105 104 104 110 111 113; //Tiefstwerte
          104 119 118 108 115 118 113; //Schlusswerte
          113 121 114 112 118 121 121) //Eröffnungsw.
CandlestickChart()
GridLocation(all;none) // kein Raster
CloseDrawing()
```



Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*) und andererseits die Ergänzung mit einem Schatten (*darstellung=shadow*). Die Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Die Darstellung der Rechteckfüllungen, der Berandungen und Schatten kann durch die Stilfunktionen `FillStyle()`, `PictureStyle()`, `BorderStyle()` und `ShadowStyle()` variiert werden. Alle Stilfunktionen werden im Abschnitt *Stile* behandelt. Beispiele:

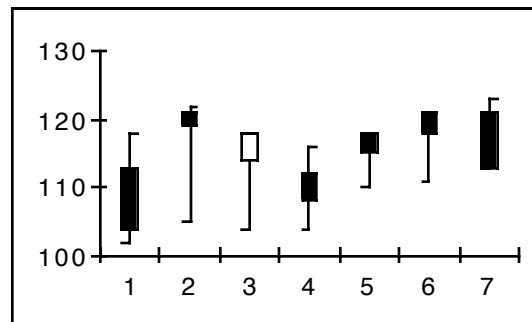
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(118 122 118 116 118 123; // Höchstwerte
            102 105 104 104 110 113; // Tiefstwerte
            104 119 118 108 115 113; // Schlusswerte
            113 121 114 112 118 121) // Eröffnungswerte
  CandlestickChart(shadow+horizontal)
  FillStyle(1;lightBlue)
  BorderStyle(1;;1;blue)
  ShadowStyle(1;1;gray)
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



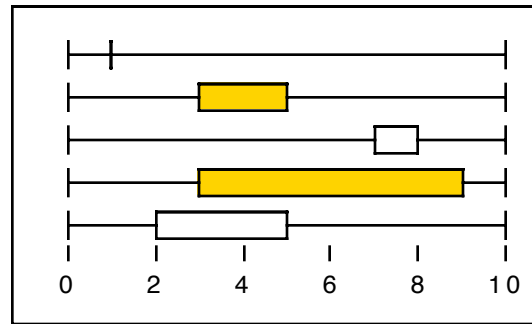
Mit dem 2. Argument *abstand* kann die Distanz zwischen den einzelnen "Candlesticks" bzw. die Breite der Rechtecke kontrolliert werden. Der Abstand wird dabei in Prozent der Rechteckbreite angegeben. Standardmäßig ist der Abstand gleich der Rechteckbreite (*abstand=100*). Bei Werten größer 100 (z.B. *abstand=400*) werden die Rechtecke schmaler und die Abstände dazwischen größer, bei einem Wert kleiner 100 (z.B. *abstand=50*) werden die Rechtecke breiter und die Abstände kleiner. Optional können die "Candlesticks" durch links, rechts oder mittig ausgerichtete Markierungen ergänzt werden. Mit dem 4. und 5. Argument *hochMarkerLänge* und *hochMarkerAusrichtung* werden die Länge und Ausrichtung der Markierungen für die Höchstwerte, mit dem 6. und 7. Argument *tiefMarkerLänge* und *tiefMarkerAusrichtung* die Länge und Aus-

richtung der Markierungen für die Tiefstwerte festgelegt. Die Länge ist dabei in Prozent bezogen auf die Rechteckbreite einzugeben. Beispiele:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(118 122 118 116 118 121 123; //Höchstwerte
          102 105 104 104 110 111 113; //Tiefstwerte
          104 119 118 108 115 118 113; //Schlusswerte
          113 121 114 112 118 121 121) //Eröffnungsw.
CandlestickChart(;300;50;right;50;left)
FillStyle(1;black)
GridLocation(all;none) // kein Raster
CloseDrawing()
```



```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(10 10 10 10 10; // Höchstwerte
          0 0 0 0 0; // Tiefstwerte
          5 3 8 3 1; // "Schlusswerte"
          2 9 7 5 1) // "Eröffnungswerte"
CandlestickChart(horizontal;60;100;center;100;center)
FillStyle(1;darkYellow)
GridLocation(all;none) // kein Raster
AxisLine(x;0) // keine x-Achsenlinie
AxisMajorTicks(x;;;;out) // Skala verschieben
AxisOptions(y;none) // keine y-Achse
CloseDrawing()
```



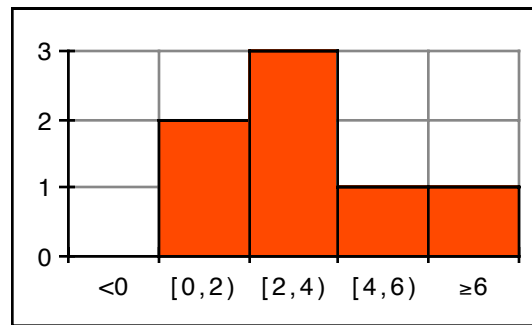
### Histogramme:

**Histogram(darstellung;kategorienabstand;serienabstand)**

**HistogramRange(minWert;maxWert;intervallanzahl)**

Die Funktion `Histogram()` dient zur Auswertung und Darstellung von Häufigkeitsverteilungen. Standardmäßig wird der Wertebereich durch den größten und kleinsten Wert in der Funktion `ChartData()` bestimmt und in zehn Intervalle unterteilt. Mit den Argumenten *minWert*, *maxWert* und *intervallanzahl* in der Funktion `HistogramRange()` können sowohl der auszuwertende Bereich als auch die Intervallanzahl vom Benutzer vorgegeben werden. Dabei ist zu beachten, dass standardmäßig noch zwei Randintervalle hinzugefügt werden, d.h. zum Beispiel bei *intervallanzahl*=3 werden insgesamt fünf Intervalle dargestellt. Die Beschriftung der Intervalle erfolgt durch die in Klammern angeführten Intervallgrenzen. Zum Beispiel, "[2,4)" bedeutet, das Intervall umfaßt alle Werte größer gleich zwei und kleiner vier. Zu beachten ist, dass die Funktion `HistogramRange()` nach der Funktion `Histogram()` anzuführen ist. Beispiele:

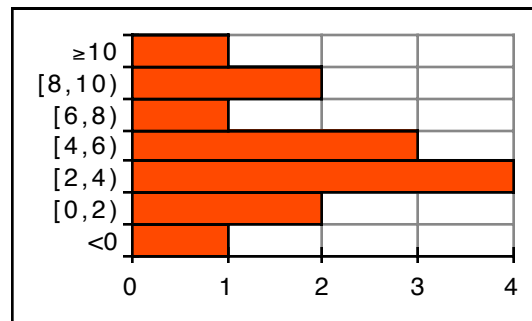
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(6.5 0 2.9 1.5 3 2.5 5.9)
  Histogram()
  HistogramRange(0;6;3) // nach Histogramm()!
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 2 0 3 8 4 5 -2 6 9 5 2 12 2)
  Histogram(horizontal)
  HistogramRange(0;10;5) // nach Histogramm()!
  ScalingOptions(x;;on) // nur ganzzahlige x-Skala
CloseDrawing()

```

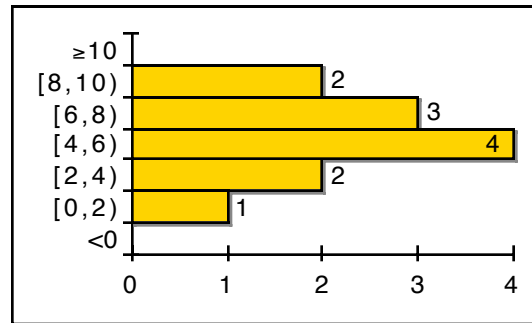


Das Aussehen der Balken wird durch die Argumente *darstellung*, *kategorienabstand* und *serienabstand* in der Funktion `Histogram()` kontrolliert. Diese Argumente werden ausführlich bei den *Balkendiagrammen* behandelt. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(2.1 4.8 0 7 9.9 5 7.2 6 4 8 2.5 5.1)
  Histogram(horizontal+shadow+label)
  HistogramRange(0;10;5) // nach Histogramm()!
  FillStyle(1;darkYellow)
  ShadowStyle(1;1;gray)
  ScalingOptions(x;;on) // nur ganzzahlige x-Skala
  GridLocation(xy;none) // kein Raster
CloseDrawing()

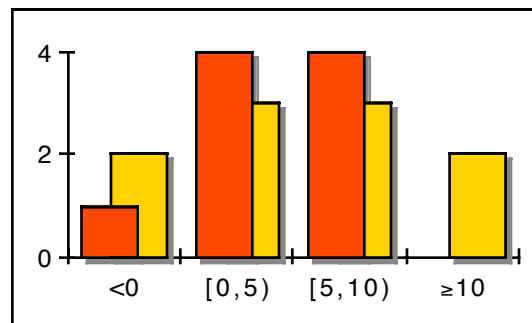
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 0 9.9 5 7.2 3 0.9 6 -.5; // 1.Serie
            6 9 2 7 2 12 -2 0.5 11 -1) // 2.Serie
  Histogram(shadow;50;-50)
  HistogramRange(0;10;2) // nach Histogram()!
  FillStyle(2;darkYellow)
  ShadowStyle(all;2;gray)
  ScalingOptions(y;;on) // nur ganzzahlige y-Skala
  GridLocation(xy;none) // kein Raster
CloseDrawing()

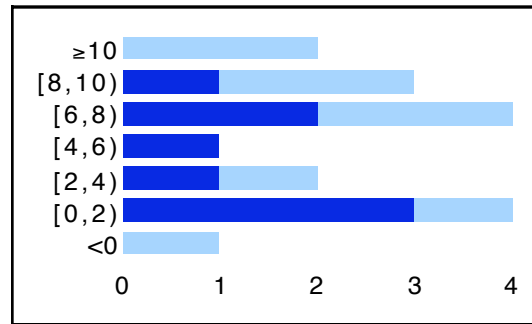
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 0 9.9 5 7.2 3 0.9 6; // 1.Serie
            6 9 8 7 2 12 -2 0.5 11) // 2.Serie
  Histogram(horizontal+stacked;40)
  HistogramRange(0;10;5) // nach Histogram()!
  FillStyle(1;blue)
  FillStyle(2;lightBlue)
  BorderStyle(all;none)
  AxisLine(all;0) // keine Achsenlinien
  AxisMajorTicks(all;0) // keine Skalenmarkierungen
  ScalingOptions(x;;on) // nur ganzzahlige x-Skala
  GridLocation(xy;none) // kein Raster
CloseDrawing()

```

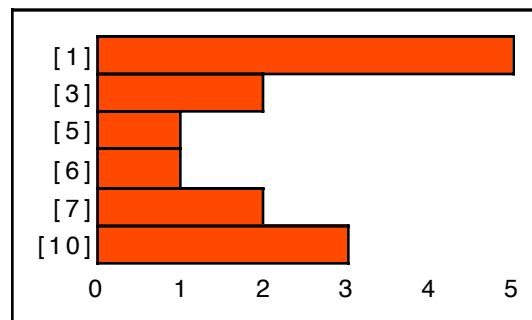


**HistogramOptions(werteZählen;inHöheresIntervall;  
endenHinzufügen;frequenzlinien)**

Optional können die Werte auch ohne Intervallbereiche, das heißt allein nach ihrer Häufigkeit, ausgewertet werden. Dies wird durch Aktivieren des ersten Arguments *werteZählen=on* ermöglicht. Die Funktion *HistogramRange()* wird in diesem Fall ignoriert, genauso wie die Argumente *inHöheresIntervall* und *endenHinzufügen*.

Die Funktion *HistogramOptions()* ist genauso wie die Funktion *HistogramRange()* nach der Funktion *Histogram()* anzuführen. Beispiele:

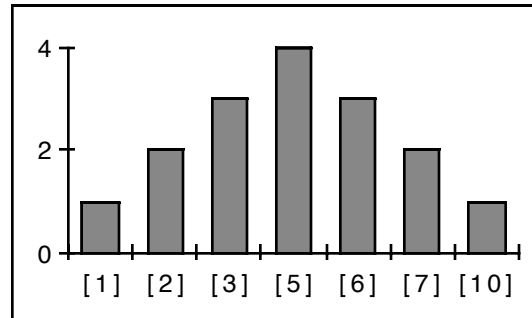
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 10 7 10 1 5 3 7 3 1 1 6 10 1)
  Histogram(horizontal)
  HistogramOptions(on)
  AxisLine(all;0)           // keine Achsenlinien
  AxisMajorTicks(all;0)    // keine Skalenmarkierungen
  ScalingOptions(y;on)     // y-Skala von oben nach unten
  ScalingOptions(x;on)     // nur ganzzahlige x-Skala
  GridLocation(xy;none)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(6 2 5 7 3 6 5 3 7 3 2 5 1 6 10 5)
  Histogram(;80)
  HistogramOptions(on) // nach Histogram()!
  FillStyle(1;gray)
  GridLocation(xy;none)
CloseDrawing()

```

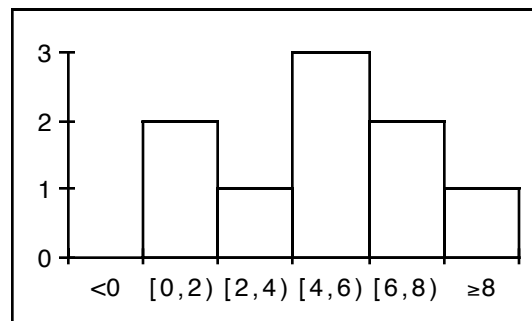


Mit dem zweiten Argument *inHöheresIntervall* kann die Zuordnung jener Werte, welche genau an einer Intervallgrenze liegen, kontrolliert werden. Standardmäßig werden diese "Grenzwerte" dem höheren Intervall zugeordnet (*inHöheresIntervall=on*), bei *inHöheresIntervall=off* dem niedrigeren Intervall. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 0 9 5 7 4 2 6 5)
  Histogram()
  HistogramRange(0;8;4) // nach Histogram()!
  HistogramOptions(;on) // nach Histogram()!
  FillStyle(1;;transparent)
  ScalingOptions(y;;on) // nur ganzzahlige y-Skala
  GridLocation(xy;none) // kein Raster
CloseDrawing()

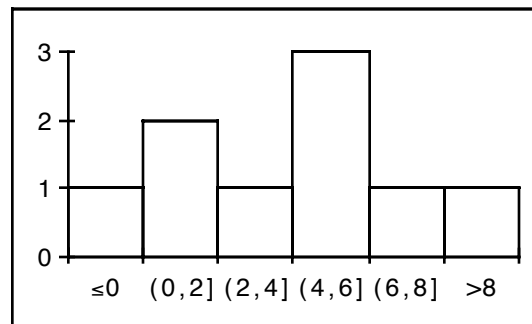
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 0 9 5 7 4 2 6 5)
  Histogram()
  HistogramRange(0;8;4) // nach Histogram()!
  HistogramOptions(;off) // nach Histogram()!
  FillStyle(1;;transparent)
  ScalingOptions(y;;on) // nur ganzzahlige y-Skala
  GridLocation(xy;none) // kein Raster
CloseDrawing()

```

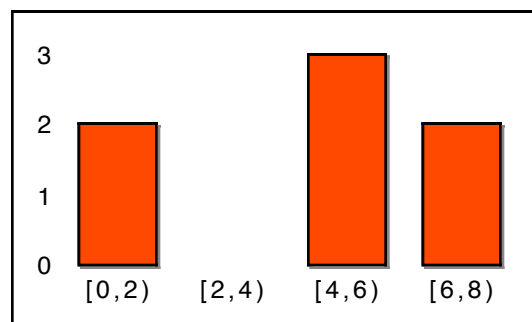


Die standardmäßig hinzugefügten Randintervalle können mit dem 3. Argument *endenHinzufügen=off* unterdrückt werden. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 0 9 5 7 4 8 6 5)
  Histogram(shadow;50)
  HistogramRange(0;8;4) // nach Histogram()!
  HistogramOptions(;;off) // nach Histogram()!
  ShadowStyle(1;1;gray)
  AxisLine(all;0) // keine Achsenlinien
  AxisMajorTicks(all;0) // keine Skalenmarkierungen
  ScalingOptions(x;;on) // nur ganzzahlige y-Skala
  GridLocation(xy;none) // kein Raster
CloseDrawing()

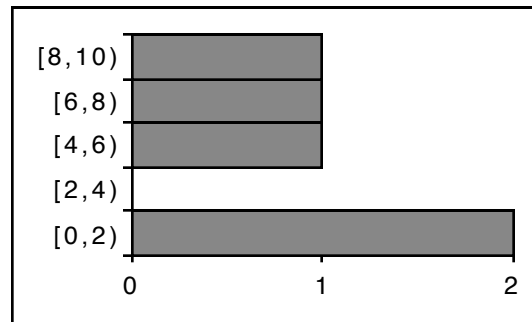
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 0 9.9 5 7.2)
  Histogram(horizontal)
  HistogramOptions(;;off) // nach Histogram()!
  HistogramRange(0;10;5) // nach Histogram()!
  FillStyle(all;gray)
  GridLocation(xy;none)
  ScalingOptions(x;;on) // nur ganzzahlige x-Skala
CloseDrawing()

```



Optional können Histogramme durch Frequenzlinien ergänzt werden. Dazu stehen als 4. Argument *frequenzlinien* die folgenden Konstanten zur Verfügung:

Konstante	Wert	Anmerkung
none	0	keine Frequenzlinien
frequency	1	Frequenzlinien
ogive	2	laufend aufsummierte Frequenzen
reverseOgive	3	rückwärts laufende Frequenzsummen

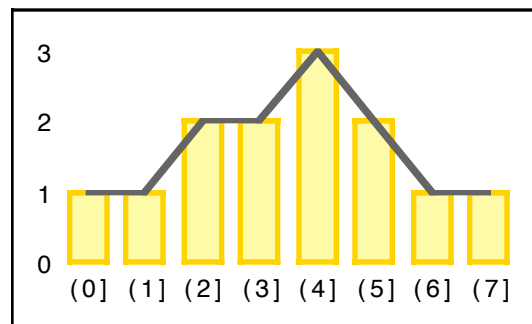
Die Frequenzlinien können durch die Funktion `LineStyle()` gestaltet werden. Die Funktion `LineStyle()` wird im Abschnitt *Stile* erläutert.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 4 9.9 5 4.2 3 4.9 6
            6 9 8 7 2 12 3 3.5 11)
  Histogram(;50)
  HistogramRange(0;8;8) // nach Histogram(!)
  HistogramOptions(;off;off;frequency)
  FillStyle(1;lightYellow)
  BorderStyle(1;;2;darkYellow)
  LineStyle(1;poly;2;darkGray)
  GridLocation(xy;none) // kein Raster
  AxisLine(all;0)       // keine Achsenlinien
  AxisMajorTicks(all;0) // keine Skalenmarkierungen
  AxisMajorTickLabelTexts(x;"(|u|]")
CloseDrawing()

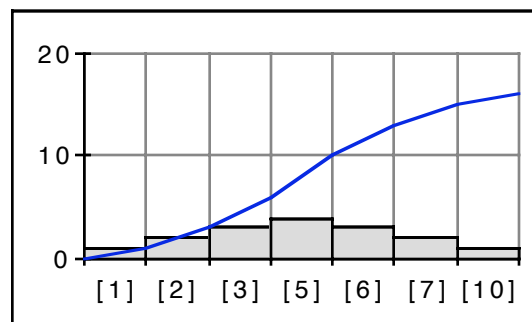
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(6 2 5 7 3 6 5 3 7 3 2 5 1 6 10 5)
  Histogram()
  HistogramOptions(on;;;ogive) // nach Histogram()
  FillStyle(1;lightGray)
  LineStyle(1;poly;1;blue)
CloseDrawing()

```



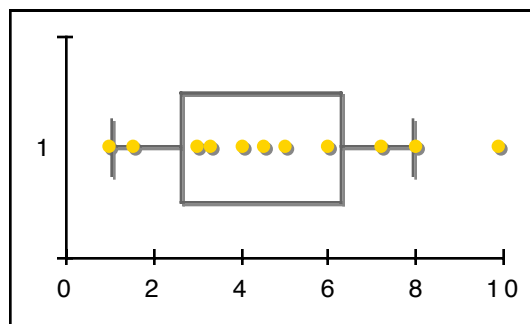
**Box Plots:**

Box Plots, auch als Box- und Whisker Plots bezeichnet, werden zur Darstellung statistischer Verteilungen verwendet.

**BoxPlots(darstellung;obereBoxPerzentile;  
untereBoxPerzentile;obereWhiskerPerzentile;  
untereWhiskerPerzentile)**

Das erste Argument *darstellung* ermöglicht einerseits die Drehung des Diagramms um 90 Grad (*darstellung=horizontal*), andererseits die Ergänzung mit Symbolen (*darstellung=symbol*) und Schatten (*darstellung=shadow*). Alle Darstellungsoptionen lassen sich durch ein Pluszeichen "+" kombinieren. Die Darstellung der Berandung, Symbole und Schatten kann durch die Stilfunktionen *BorderStyle()*, *SymbolStyle()* und *ShadowStyle()* variiert werden. Alle Stilfunktionen werden im Abschnitt *Stile* behandelt. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 1.5 8 9.9 4.5 7.2 3.3 4 5 3 6)
  BoxPlot(symbol+shadow+horizontal)
  SymbolStyle(all;bullet;4;1;darkYellow)
  BorderStyle(all;;1;darkGray)
  ShadowStyle(all;1;gray)
  GridLocation(xy;none)
CloseDrawing()
```



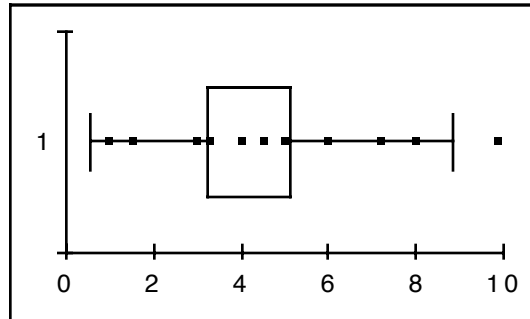
Die Box-Grenzwerte werden durch die Argumente *obereBoxPerzentile* (default: 75 Prozent) und *untereBoxPerzentile* (default: 25 Prozent) festgelegt, die Whisker-Grenzwerte werden durch die Argumente *obereWhiskerPerzentile* (default: 90 Prozent) und *untereWhiskerPerzentile* (default: 10 Prozent).

Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 1.5 8 9.9 4.5 7.2 3.3 4 5 3 6)
  BoxPlot(symbol+horizontal;65;35;95;5)
  SymbolStyle(all;square;2;1;black)
  GridLocation(xy;none)
CloseDrawing()

```



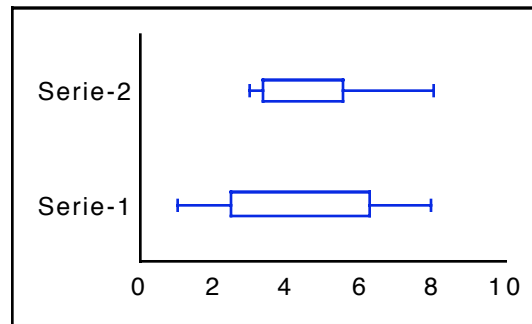
**BoxPlotOptions(boxabstand;istPerzentileGraph;boxFüllen;  
mittelwertZeigen;medianZeigen;nurAusreißerZeigen;  
nurMarkerZeigen;markerlänge;konfidenzintervall)**

Die Funktion `BoxPlotOptions()` ist nach der Funktion `BoxPlot()` anzuführen und eröffnet eine Vielzahl von Gestaltungsmöglichkeiten. Mit dem 1. Argument *boxabstand* kann der Abstand zwischen den einzelnen Serien kontrolliert werden. Der Abstand wird dabei in Prozent der Rechteckbreite angegeben. Standardmäßig ist der Abstand gleich der Rechteckbreite (*boxabstand=100*). Bei Werten größer 100 (z.B. *boxabstand=400*) werden die Rechtecke schmaler und die Abstände dazwischen größer, bei einem Wert kleiner 100 (z.B. *boxabstand=50*) werden die Rechtecke breiter und die Abstände kleiner. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6;      // 1.Serie
            3 5 3.2 4.1 8 9 5 3.5 4.9 6.1) // 2.Serie
  BoxPlot(horizontal)
  BoxPlotOptions(400) // nach BoxPlot()!
  BorderStyle(all;;1;blue)
  AxisMajorTicks(all;0)
  AxisMajorTickLabelTexts(y;"Serie-1";"Serie-2")
  GridLocation(xy;none)
CloseDrawing()

```

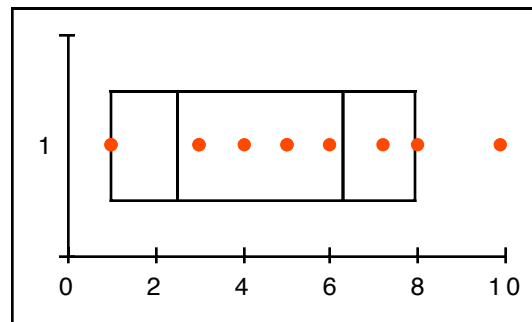


Durch Aktivierung des 2. Arguments *istPerzentileGraph=on* wird die Rechteckbox bis zur Whisker-Perzentile durchgezogen. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6)
  BoxPlot(horizontal+symbol)
  BoxPlotOptions(;on) // nach BoxPlot(!)
  SymbolStyle(1;bullet;4)
  GridLocation(xy;none)
CloseDrawing()

```

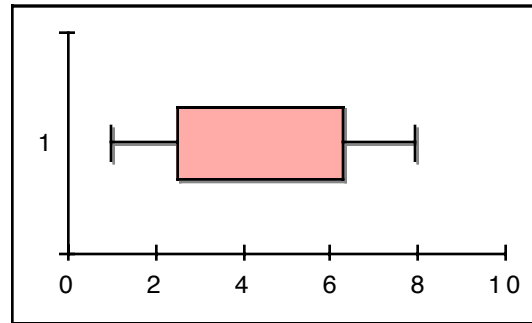


Mit dem 3. Argument *boxFüllen=on* und der Funktion `FillStyle()` bzw. `PictureStyle()` kann die Füllung der Rechteckbox kontrolliert werden. Die Funktionen `FillStyle()` und `PictureStyle()` werden im Abschnitt *Stile* erläutert. Beispiel:

```

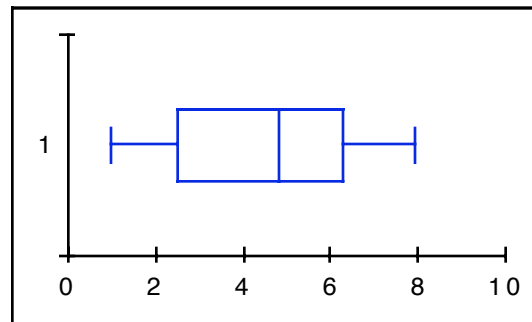
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6)
  BoxPlot(horizontal+shadow)
  BoxPlotOptions(200;off;on) // nach BoxPlot(!)
  FillStyle(1;lightRed)
  BorderStyle(1;;1)
  ShadowStyle(1;1;gray)
  GridLocation(xy;none)
CloseDrawing()

```

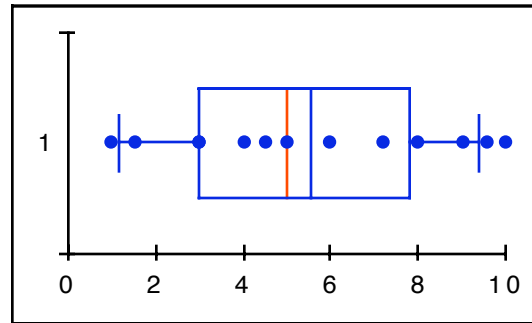


Weiters können Markierungslinien für den arithmetischen Mittelwert bzw. Median durch Aktivieren des Arguments *mittelwertZeigen=on* bzw. *medianZeigen=on* hinzugefügt werden. Die Darstellung der Mittelwertlinie ist ident der Berandung und wird durch die Funktion `BorderStyle()` kontrolliert. Das Aussehen der Medianlinie kann durch die Funktion `LineStyle()` kontrolliert werden. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6)
  BoxPlot(horizontal)
  BoxPlotOptions(200;off;off;on) // nach BoxPlot()
  BorderStyle(1;;1;blue)
  GridLocation(xy;none)
CloseDrawing()
```

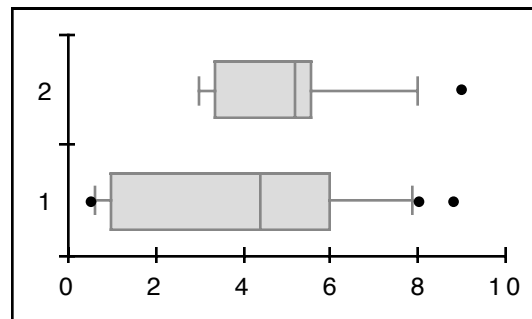


```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 1 8 9.6 5 7.2 10 9 3 4 4.5 3 6)
  BoxPlot(horizontal+symbol)
  BoxPlotOptions(100;off;off;on;on) // nach BoxPlot()
  BorderStyle(1;;1;blue)
  LineStyle(1;;1;red)
  SymbolStyle(1;bullet;4;1;blue)
  GridLocation(xy;none)
CloseDrawing()
```



Durch Aktivieren des Arguments *nurAusreißerZeigen=on* kann die Darstellung der Symbole auf die "Ausreißer" beschränkt werden, d.h. auf jene Werte, welche außerhalb der Whisker-Perzentile liegen. Die Darstellung der Symbole wird durch *darstellung=symbol* in der Funktion `BoxPlot()` aktiviert. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(0.5 1 1 8 8.8 5 7.2 3 4 5 3 6; // 1.Serie
          3 5 3.2 4.1 8 9 5 3.5 4.9 6.1) // 2.Serie
BoxPlot(horizontal+symbol)
BoxPlotOptions(100;off;on;on;off;on) //nach BoxPlot()
FillStyle(all;lightGray)
BorderStyle(all;;1;gray)
SymbolStyle(all;bullet;3;;black)
GridLocation(xy;none)
CloseDrawing()
```



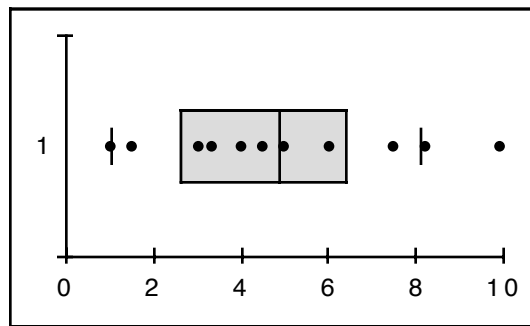
Mit dem Argument *nurMarkerZeigen=on* kann die Verbindungslinie zwischen Rechteckbox und Whisker-Markierung ausgeblendet werden.

Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 1.5 8.2 9.9 5 7.5 3 4 4.5 3.3 6)
  BoxPlot(horizontal+symbol)
  BoxPlotOptions(200;off;on;on;on;off;off;on)
  FillStyle(1;lightGray)
  BorderStyle(1;;1;)
  SymbolStyle(1;bullet;3;;black)
  GridLocation(xy;none)
CloseDrawing()

```

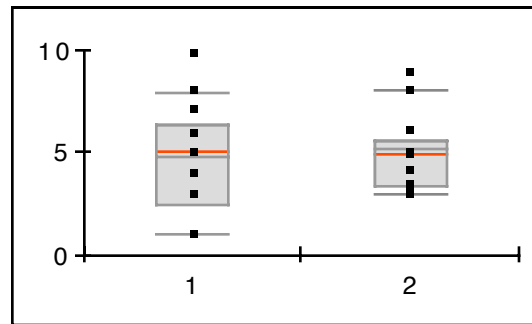


Die Markerlänge der Whisker-Perzentile kann mit dem Argument *marker-Länge* kontrolliert werden. Die Markerlänge ist in Prozent der Rechteckbreite anzugeben und ist standardmäßig 50, d.h. die halbe Rechteckbreite. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6;      // 1.Serie
            3 5 3.2 4.1 8 9 5 3.5 4.9 6.1) // 2.Serie
  BoxPlot(symbol)
  BoxPlotOptions(200;off;on;on;on;on;off;on;100)
  FillStyle(all;lightGray)
  BorderStyle(all;;1;gray)
  LineStyle(all;;1;red) // Median
  SymbolStyle(all;square;2;;black)
  GridLocation(xy;none)
CloseDrawing()

```

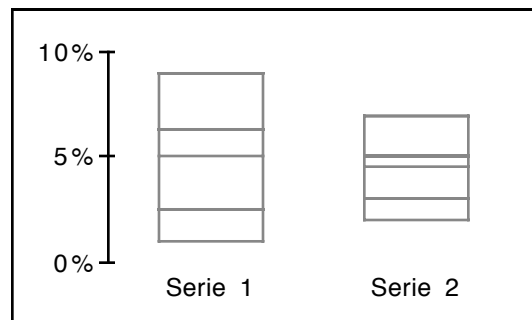


Box Plots können auch zur Darstellung von Quartilen verwendet werden. Dazu wird die obere Box-Perzentile gleich 75% (das entspricht der 3. Quartile) und die untere Box-Perzentile gleich 25% gesetzt (das entspricht der 1. Quartile). Die 2. Quartile (=50% Perzentile) entspricht dem Median. Die Quartilen werden begrenzt durch die obere Whisker-Perzentile gleich 100% und durch die untere Whisker-Perzentile gleich 0%. Beispiel:

```

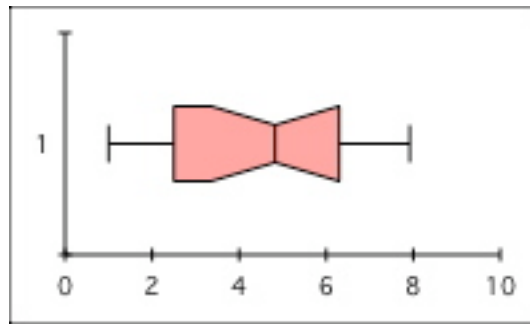
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 1 8 9 5 7.2 3 4 5 3 6;          // 1. Serie
            3 5 3 4.1 7 3 5 2 5 3.5 4.9 6) // 2. Serie
  BoxPlot(;75;25;100;0)
  BoxPlotOptions(;on; // istPerzentileGraph
                 off; // keine Füllung
                 off; // kein Mittelwert
                 on) // Median (= 2.Quartile)
  BorderStyle(all;;1;gray)
  LineStyle(all;;1;gray) // Median
  AxisLine(x;0)
  AxisMajorTicks(x;0)
  AxisMajorTickLabelTexts(y;"|u|%" )
  GridLocation(xy;none)
  AxisMajorTickLabelTexts(x;"Serie |u|")
CloseDrawing()

```



Optional kann mit dem Argument *konfidenzintervall* ein vorgegebener Konvidenzbereich durch Einschnüren der Rechteckbox sichtbar gemacht werden. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(1 1 8 9.9 5 7.2 3 4 5 3 6)
BoxPlot(horizontal)
BoxPlotOptions(200;off;on;
               on; // Mittelwert zeigen
               off;
               off;
               off;
               50; // Markerlänge
               90) // 90% Konvidenz
FillStyle(1;lightRed)
GridLocation(xy;none)
CloseDrawing()
```



## Stile

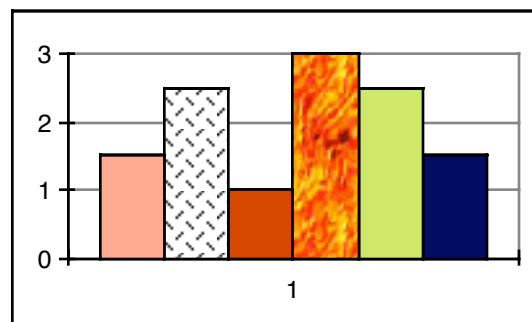
Stile erlauben die detaillierte Kontrolle über das Aussehen einzelner Diagrammserien. Folgende Komponenten können für jede Datenserie individuell gestaltet werden:

- Füllflächen
- Berandungen
- Linienzüge
- Symbole
- Schatten
- Beschriftungen
- Pfeile

### **FillStyle(*serienindex*; *farbe*; *muster*)**

Mittels der Funktion `FillStyle()` kann jeder Datenserie sowohl eine individuelle Füllfarbe als auch ein Füllmuster zugeordnet werden. Serien, denen nicht explizit ein Stil zugeordnet wird, werden in einer der 16 Standardfüllungen dargestellt. Ein Überblick über die vordefinierten Standardfüllungen sind in *xmReferenz* zu finden. Beispiele:

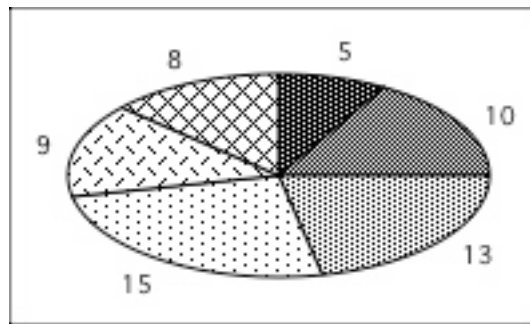
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(1.5; 2.5; 1; 3; 2.5; 1.5)
BarChart()
FillStyle(1;red;gray)
FillStyle(2;black;48)
FillStyle(3;darkRed)
FillStyle(4;;127)
FillStyle(5;200 231 89)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 10 13 15 9 8)
  PieChart(oval+label)
  BorderStyle(all;)
  FillStyle(1;black;darkGray)
  FillStyle(2;black;gray)
  FillStyle(3;black;lightGray)
  FillStyle(4;black;13)
  FillStyle(5;black;48)
  FillStyle(6;black;53)
CloseDrawing()

```

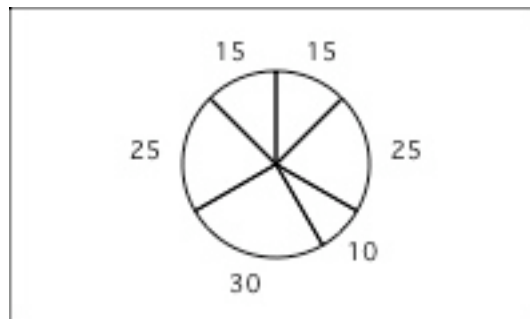


Wird kein Serienindex vorgegeben oder *serienindex=all* gesetzt, so wird allen Serien die gleiche Füllung zugewiesen. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(15 25 10 30 25 15)
  PieChart(label)
  FillStyle(;;transparent)
CloseDrawing()

```



**PictureStyle(*serienindex*;*quelle*;*name*;  
stapelnUndSkalieren)**

Anstelle einer Füllfarbe kann wahlweise auch ein Bild als Füllung verwendet werden. Dabei werden drei verschiedene Bildquellen unterstützt:

- *Zwischenablage*: (*quelle=clipboard*)

Die Zwischenablage wird verwendet, wenn als Quelle die Konstante *clipboard* eingegeben wird. Das Argument *name* wird ignoriert.

Beispiel: `PictureStyle(1;clipboard)`

- *Datei*: (*quelle=file*)

Es kann entweder der vollständige, absolute Dateipfad oder nur ein relativer Pfad vorgegeben werden. Der relative Pfad bezieht sich auf den Ordner in welchem sich die aktuelle FileMaker Pro Datei befindet. Beispiele:

`PictureStyle(1;file;"Bilder/Bild_01.png")`

`PictureStyle(2;file;"C:/Bilder/Bild_01.png")`

`PictureStyle(3;file;"Macintosh HD/Bilder/Bild_01.pdf")`

Unter Mac OS X können folgende Bildformate importiert werden: PDF, PICT, GIF, JPEG, PNG, BMP, TIFF.

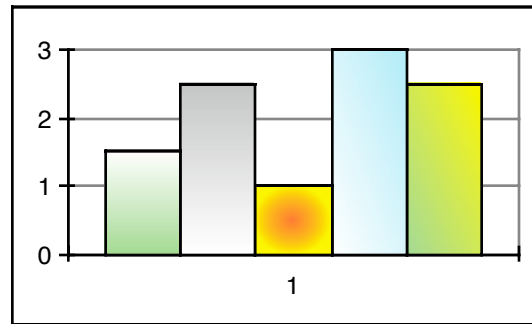
Unter Windows können die folgende Bildformate importiert werden: WMF, EMF, GIF, JPEG, PNG, BMP, TIFF.

- *Resource*: (*quelle=resource*)

Zur Zeit stehen insgesamt 42 in Ressourcen abgespeicherte Bilder zur Verfügung. Der Zugriff erfolgt mittels einer Indexnummer zwischen 1 und 42. Dabei ist zu beachten, dass die Indexnummer unter Hochkomma zu setzen ist. Ein Überblick über die vordefinierten Bildressourcen findet sich in *xmReferenz*. Weiters ist unter Windows bei der Verwendung von Ressourcen zu beachten, dass Diagramme zu einer Größe von mehreren MB anwachsen können. Dieses Problem tritt nur in Verbindung mit dem Standard-Ausgabeformat EMF auf, nicht jedoch bei der Ausgabe als Bitmap. EMF- und Bitmapformat werden im Zuge der Funktion `OpenDrawing()` im Abschnitt *Layout* behandelt.

Beispiele:

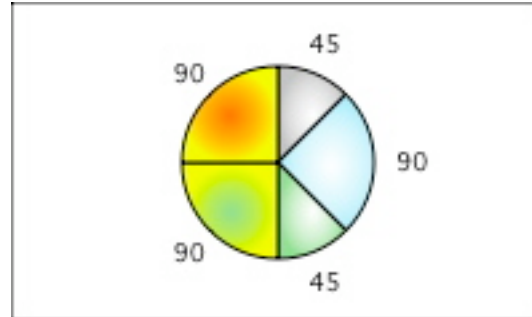
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5; 2.5; 1; 3; 2.5)
  BarChart()
  PictureStyle(1;resource;"5")
  PictureStyle(2;resource;"2")
  PictureStyle(3;resource;"42")
  PictureStyle(4;resource;"24")
  PictureStyle(5;resource;"30")
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(45 90 45 90 90)
  PieChart(label)
  PictureStyle(1;resource;"34")
  PictureStyle(2;resource;"36")
  PictureStyle(3;resource;"38")
  PictureStyle(4;resource;"40")
  PictureStyle(5;resource;"42")
CloseDrawing()

```

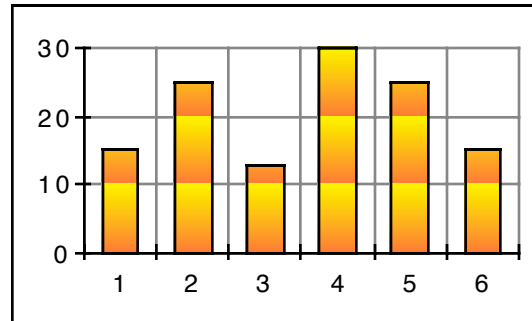


Bei Balkendiagrammen besteht zusätzlich noch die Möglichkeit, Bilder durch Vorgabe eines Skalierungswertes zu skalieren und zu stapeln. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(15 25 13 30 25 15)
  BarChart()
  PictureStyle(all;resource;"8";10)
CloseDrawing()

```



**BorderStyle(*serienindex*;*linienführung*;*strichstärke*;  
*farbe*;*muster*)**

Standardmäßig werden alle Ränder schwarz und ein Pixel breit dargestellt. Mit der Funktion `BorderStyle()` kann jeder Datenserie eine individuelle Berandung zugeordnet werden. Zusätzlich kann bei einigen Diagrammen über das Argument *linienführung* der Berandungsverlauf variiert werden:

Konstante	Wert	Diagrammart	Darstellung
none	0	alle Diagramme	Berandung ausblenden
jump	1	(wird nicht unterstützt)	nur horiz. od. vert. Linien
step	2	Areachart	stufenförm. Berandung
poly	3	alle Diagramme	geradlinige Berandung
smooth	4	Area-, Radar-, Polarchart	geglättete Berandung

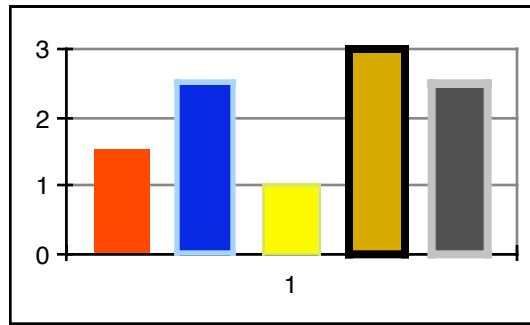
Der Grad der Glättung (*linienführung*=4) kann durch einen zusätzlichen Parameter kontrolliert werden. Des weiteren kann der Strichstärke optional durch Anfügen einer Liste von Strichlängen und -abständen ein Strichmuster zugeordnet werden. Beispiele dazu finden sich unter `LineStyle()`.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5; 2.5; 1; 3; 2.5)
  BarChart(;;50)
  BorderStyle(1;poly;0)
  BorderStyle(2;poly;2;lightBlue)
  BorderStyle(3;poly;1;200 231 89)
  BorderStyle(4;;3)
  BorderStyle(5;;3;gray;gray)
CloseDrawing()

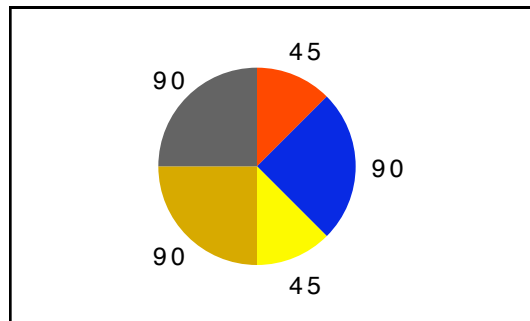
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(45 90 45 90 90)
  PieChart(label)
  BorderStyle(all;none)
CloseDrawing()

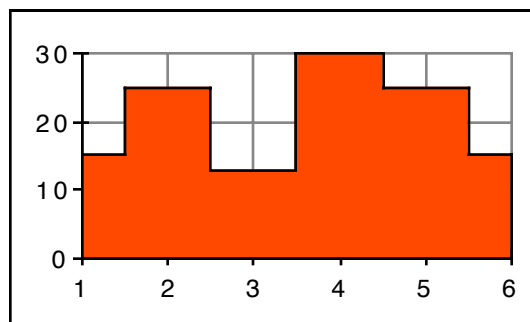
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(15 25 13 30 25 15)
  AreaChart()
  BorderStyle(1;step;1;black)
CloseDrawing()

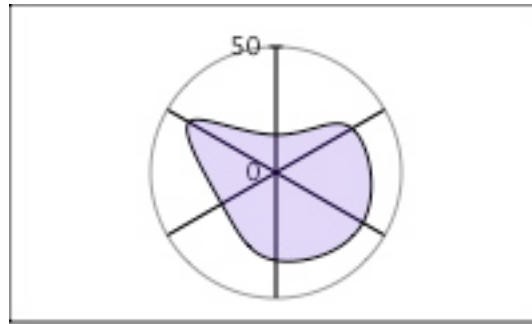
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(15 35 40 35 25 40)
  RadarChart()
  MajorGridLinePatterns(all;all;black)
  FillStyle(1;50 0 200 50)
  BorderStyle(1;smooth)
CloseDrawing()

```



**LineStyle(*serienindex*;*linienführung*;*strichstärke*;*farbe*;  
*muster*)**

Mittels der Funktion `LineStyle()` kann jeder Datenserie ein individueller Liniensstil zugeordnet werden. Serien, denen nicht explizit ein Stil zugeordnet wird, werden in einer der 16 Standardfarben dargestellt. Ein Überblick über die vordefinierten Farben sind in *xmReferenz* zu finden. Zusätzlich kann mit dem Argument *linienführung* der Kurvenverlauf variiert werden:

<i>Konstante</i>	<i>Wert</i>	<i>Darstellung</i>
none	0	Linienzug ausblenden
jump	1	nur horizontale oder vertikale Linien
step	2	stufenförmiger Linienzug
poly	3	geradliniger Linienzug ( <i>default</i> )
smooth	4	geglätteter Linienzug

Der Grad der Glättung (*linienführung*=4) kann durch einen zusätzlichen Parameter kontrolliert werden. Der Wertebereich liegt zwischen 0 (nicht geglättet, polygonal) und 2 (stark geglättet), der Defaultwert ist 1.

Beispiele:

```

LineStyle(all;4;1;darkBlue)      // glättung: 1.0 (default)
LineStyle(all;4 0;1;darkBlue)    // glättung: 0.0 (polygonal)
LineStyle(all;4 0.5;1;darkBlue)  // glättung: 0.5
LineStyle(all;smooth 0.5)        // nicht erlaubt!
LineStyle(all;smooth,0.5)        // nicht erlaubt!

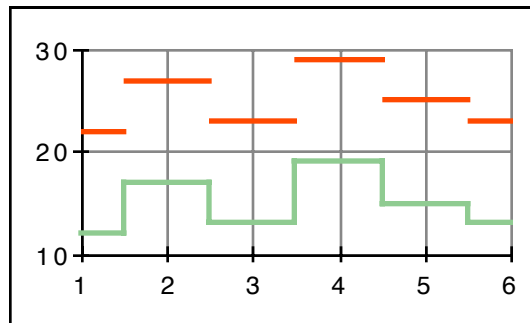
```

Zusätzlich zur Strichstärke kann optional durch Anfügen einer Liste von Strichlängen und -abständen ein Strichmuster festgelegt werden.

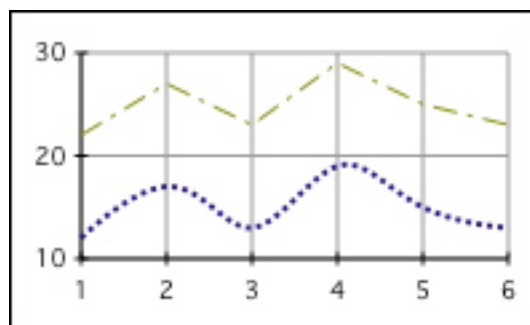
Beispiele:

```
LineStyle(all;poly;1)           // durchgehende Linie (default)
LineStyle(all;poly;1 2 2)       // punktierte Linie
LineStyle(all;poly;1 5 5)       // strichliert, 5 pixel lang
LineStyle(all;poly;2 9 4 2 4)   // strich-punktierte Linie
```

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(22 27 23 29 25 23;
          12 17 13 19 15 13)
LineChart()
LineStyle(1;jump;2)
LineStyle(2;step;2;green;gray)
CloseDrawing()
```



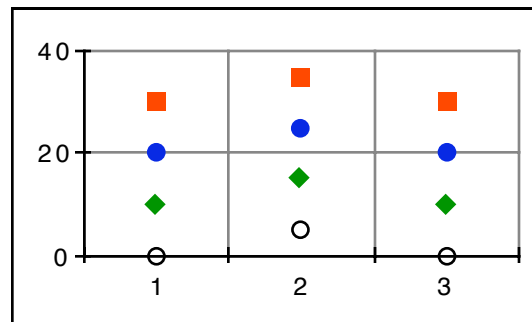
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(22 27 23 29 25 23;
          12 17 13 19 15 13)
LineChart()
LineStyle(1;poly;1 9 4 2 4;olive)
LineStyle(2;smooth;2 2 2;darkBlue)
CloseDrawing()
```



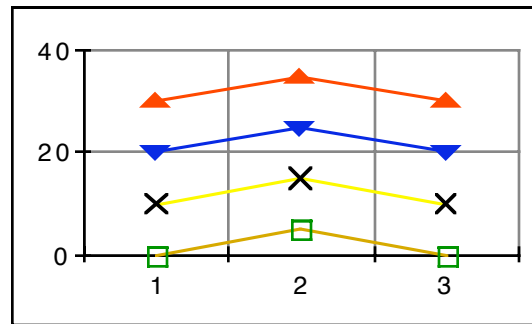
**SymbolStyle(*serienindex*;*typ*;*größe*;*strichstärke*;*farbe*;  
*muster*)**

Durch die Funktion SymbolStyle() kann jeder Datenserie ein individuelles Symbol zugeordnet werden. Serien, denen nicht explizit ein Symbol zugeordnet wird, werden in einem der 12 Standardsymbole dargestellt. Insgesamt stehen zur Zeit 18 Symbole zur Verfügung. Ein Überblick über die vordefinierten Symbole ist in *xmReferenz* zu finden. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(30 35 30; 20 25 20; 10 15 10; 0 5 0)
  ScatterChart(;on)
  SymbolStyle(1;square;6)
  SymbolStyle(2;bullet;6)
  SymbolStyle(3;diamond;6;;green)
  SymbolStyle(4;circle;6;;black)
CloseDrawing()
```



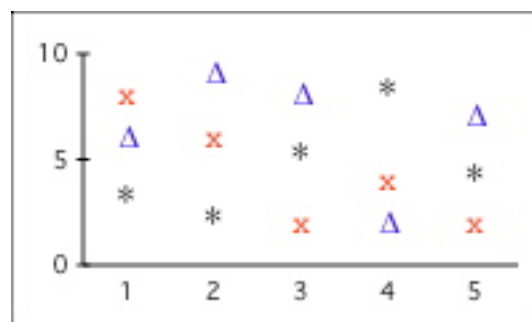
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(30 35 30;
            20 25 20;
            10 15 10;
            0 5 0)
  LineChart(symbol;on)
  SymbolStyle(1;upTriangle;10)
  SymbolStyle(2;downTriangle;10)
  SymbolStyle(3;cross;8;1;black)
  SymbolStyle(4;hollowSquare;6;1;green)
CloseDrawing()
```



Optional können als Symbole auch Zeichen aus einem Zeichensatz verwendet werden; dadurch ergibt sich eine Vielzahl neuer "Symbole", welche bei Verwendung von Postscript- oder TrueType-Fonts mit hoher Qualität ausgegeben werden können. Die Zeichen werden durch die Funktionen `LabelTexts()` und `LabelStyle()` festgelegt. Die Funktion `LabelOptions()` ermöglicht die exakte Platzierung der Zeichen.

Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 2 5 8 4; 8 6 2 4 2; 6 9 8 2 7)
  ScatterChart(label;on)
  SymbolStyle(all;none)
  LabelOptions(all;centerCenter)
  LabelTexts(1;"*")
  LabelStyle(1;"Times";16)
  LabelTexts(2;"x")
  LabelStyle(2;"Courier";12;;red)
  LabelTexts(3;"\u2206") // \xUUUU Unicode (hexadezimal)
  LabelStyle(3;"Times";12;;blue)
  AxisMajorTicks(x;0)
  GridLocation(;none)
CloseDrawing()
```



Symbole können auch unter Verwendung eines Blasendiagramms dargestellt werden. Dabei ergibt sich die Möglichkeit, mit der Funktion `PictureStyle()` Bilder aus einer Datei einzulesen und als Symbole zu verwenden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 1 6 8; 1 1 1 1;
            8 4 2 4; 1 1 1 1;
            6 9 8 2; 1 1 1 1)
  BubbleChart(;on)
  BubbleChartOptions(9)
  BorderStyle(all;0)
  AxisMajorTicks(x;0)
  GridLocation(;none)

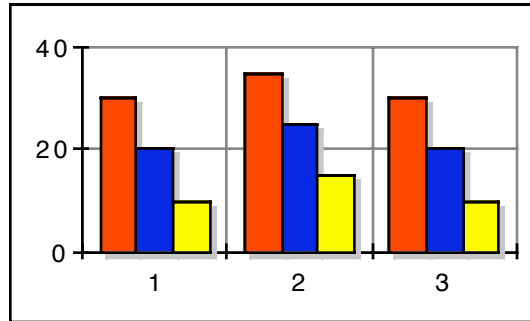
  // Unterstützte Dateiformate unter Mac OS X:
  // PDF, PICT, GIF, JPEG, PNG, BMP, TIFF
  PictureStyle(1;file;"Symbole/Symbol01.pdf")
  PictureStyle(2;file;"Macintosh HD/Symbole/Symb2.pdf")

  // Unterstützte Dateiformate unter Windows OS:
  // WMF, EMF, GIF, JPEG, PNG, BMP, TIFF
  PictureStyle(3;file;"C:/Symbole/Symbol03.png")
CloseDrawing()
```

### **ShadowStyle(serienindex;abstand;farbe;muster)**

Diagrammschatten werden standardmäßig in grau und in einem Abstand von drei Pixel dargestellt. Durch die Funktion `ShadowStyle()` kann die Darstellung des Schattens variiert werden. Beispiele:

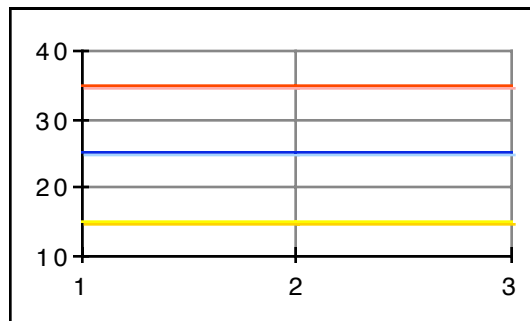
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(30 35 30;
            20 25 20;
            10 15 10)
  BarChart(shadow)
  ShadowStyle(all;2;200 200 200)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(35 35 35; 25 25 25;15 15 15)
  LineChart(shadow)
  ShadowStyle(1;1;lightRed)
  ShadowStyle(2;1;lightBlue)
  ShadowStyle(3;1;darkYellow)
CloseDrawing()

```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 2 3)
  PieChart(shadow+label;15)
  BorderStyle(all;none)
  ShadowStyle(all;3;gray)
CloseDrawing()

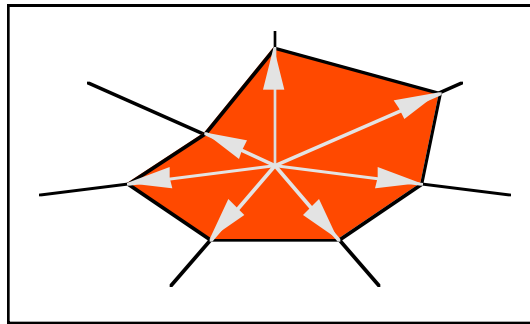
```



**ArrowStyle(*serienindex*;*strichstärke*;*farbe*;*muster*;  
*pfeilposition*;*pfeillänge*;*pfeilbreite*;*pfeilkerbe*;  
*istPfeilHohl*)**

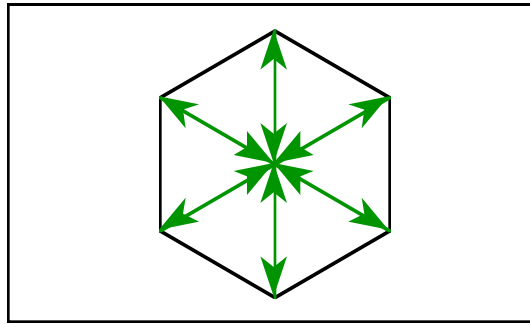
Radar- und Polardiagramme können optional durch Pfeile ergänzt werden. Diese verlaufen vom Zentrum zu den einzelnen Diagrammpunkten. Standardmäßig werden alle Pfeile in schwarz und 1 Pixel breit dargestellt. Durch die Funktion ArrowStyle() kann das Aussehen der Pfeile inkl. Pfeilspitzen für jede Datenserie getrennt kontrolliert werden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(35 35 25 25 25 25 15)
  RadarChart(oval)
  // keine Skalenbeschriftung, kein Raster
  RadarChartOptions(0;none;on)
  ArrowStyle(1;1;lightGray)
CloseDrawing()
```



Weiters kann durch das Argument *pfeilposition* die Pfeilspitze am Ende (default), am Anfang oder auch beidseitig angeordnet werden. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(40 40 40 40 40 40)
  RadarChart()
  // keine Skalenbeschriftung, kein Raster
  RadarChartOptions(0;none;on)
  AxisLine(all;0) // Achsen ausblenden
  FillStyle(1;;transparent) // Füllung ausblenden
  ArrowStyle(1;1;green;;begin+end;15;10;4)
CloseDrawing()
```

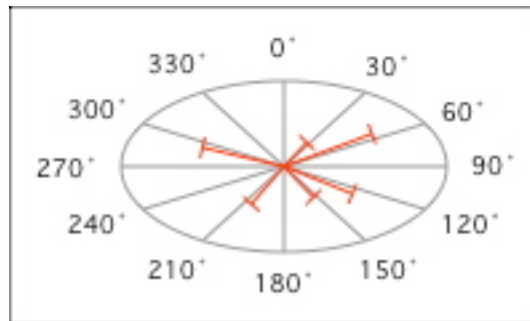


```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(-10 21 7  -25 9 27; -22 -16 14 11 -18 19)
  PolarChart(oval)
  PolarChartOptions(0;;on) // keine Skalenbeschriftung
  ArrowStyle(1;1;red;;end;0)
  AxisLabelText(all;"|u|°")

  AxisLine(all;0)           // Achsen ausblenden
  BorderStyle(1;none)       // Rand ausblenden
  FillStyle(1;;transparent) // Füllung ausblenden
CloseDrawing()

```

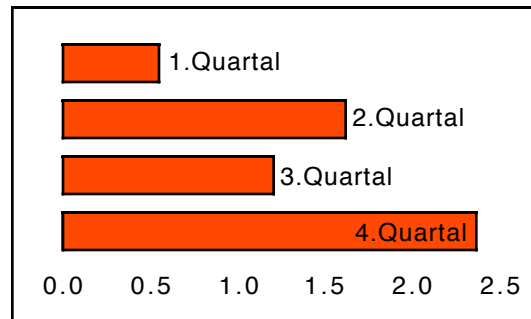


**LabelTexts(*serienindex;text1;text2...*)**

Die Funktion `LabelTexts()` ermöglicht die individuelle Beschriftung von Diagrammwerten. Die Texte können dabei sehr variable gestaltet werden:

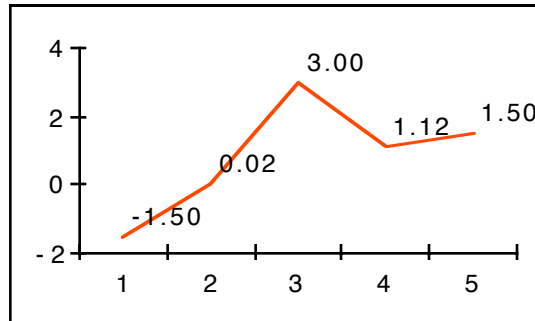
- Reiner Text:  
Zum Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(0.5555 1.61 1.2 2.365)
  BarChart(label+horizontal;50)
  LabelTexts(1;"1.Quartal";
             "2.Quartal";
             "3.Quartal";
             "4.Quartal")
  GridLocation(all;none)
  ScalingOptions(y;on)
  AxisOptions(y;none)
  AxisLine(x;0)
  AxisMajorTicks(all;0)
CloseDrawing()
```



- Formatanweisung:  
Eine Formatanweisung ist stets zwischen zwei vertikale Balken "|" zu setzen. Eine detaillierte Erklärung aller Formatanweisungen inklusive zahlreicher Beispiele ist in *xmReferenz* zu finden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(-1.5 0.024 3 1.121 1.5)
  LineChart(label;on)
  LabelTexts(1;"|f2|")
  GridLocation(all;none)
CloseDrawing()
```

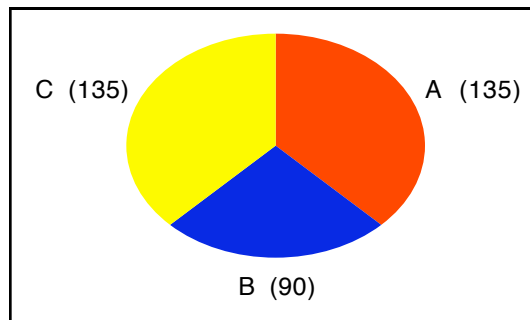


- Kombination aus Text und Formatanweisung:  
Vor und hinter einer Formatanweisung können optional noch Texte angefügt werden. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(135 90 135)
  PieChart(label+oval)
  BorderStyle(all;none)
  LabelTexts(1;"A (|u|)";"B (|u|)";"C (|u|)")
CloseDrawing()

```

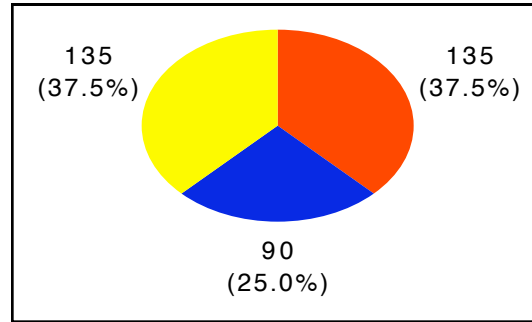


Bei Tortendiagrammen besteht zudem die Möglichkeit sowohl die absoluten Werte (default) als auch die prozentuellen Werte darzustellen. Dies geschieht durch Anführen von zwei Formatanweisungen. Die erste Formatanweisung beschreibt die absoluten Werte, die zweite Formatanweisung dient zur Formatierung der prozentuellen Werte. Beispiel:

```

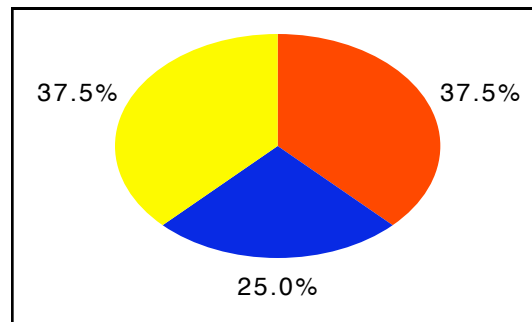
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(135 90 135)
  PieChart(label+oval)
  BorderStyle(all;none)
  LabelTexts(1;"|u|\n(|f1|%)")
  LabelStyle(all;;;center)
CloseDrawing()

```



Sollen nur die prozentuellen Werte dargestellt werden, muß die Ausgabe der absoluten Werte durch eine leere Formatanweisung "|" unterdrückt werden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(135 90 135)
  PieChart(label+oval)
  BorderStyle(all;none)
  LabelTexts(1;"| | |f1|%" )
CloseDrawing()
```

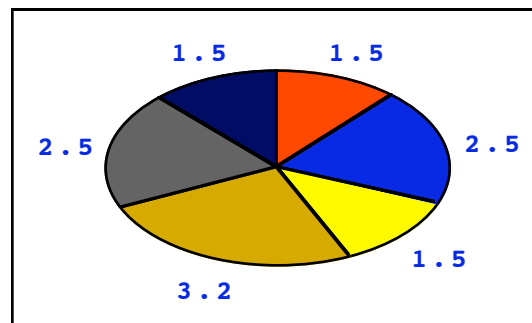


Die Texte bzw. Formatanweisungen werden periodisch wiederholt, falls die Anzahl der Diagrammwerte größer ist als die Anzahl der Textargumente. Wird kein Text übergeben, so wird die Default-Formatanweisung "|u|" verwendet. Weiters sind auch mehrzeilige Bezeichnungen durch Einfügen eines Zeilenumbruchs "\n" möglich.

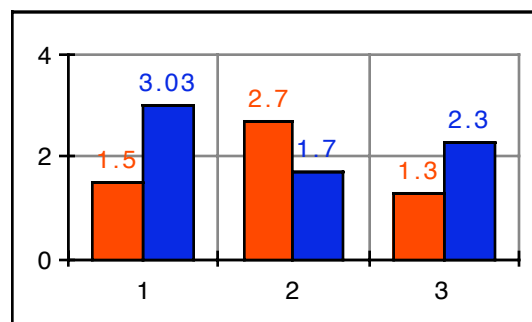
**LabelStyle**(serienindex;schrift;größe;stil;farbe;  
ausrichtung;neigungswinkel;maxBreite;  
maxHöhe;auslassungsPosition)

Mit der Funktion LabelStyle() kann den Diagrammwerten ein Schriftstil zugeordnet werden. Dabei kann für jede Datenserie ein individueller Schriftstil festgelegt werden. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1.5 3.2 2.5 1.5)
  PieChart(oval+label;)
  LabelStyle(1;"Courier";10:bold;blue)
CloseDrawing()
```

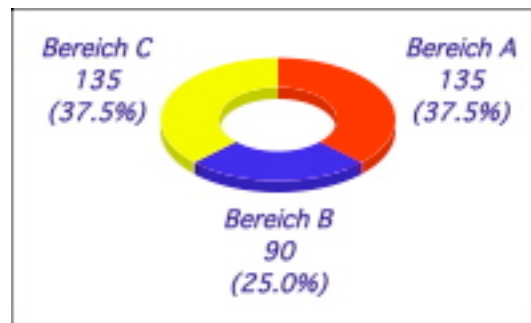


```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.7 1.3; 3.03 1.7 2.3)
  BarChart(label)
  LabelStyle(1;;9;;red)
  LabelStyle(2;;9;;blue)
CloseDrawing()
```



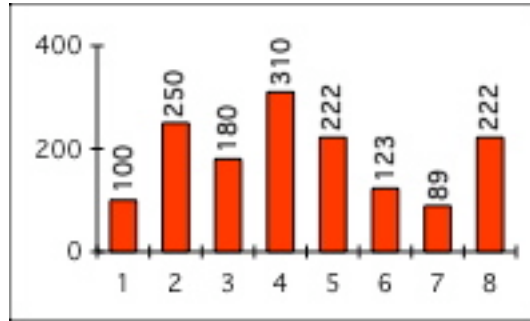
Mit dem Argument *ausrichtung* können mehrzeilige Texte entweder linksbündig (*ausrichtung=left*), mittig (*ausrichtung=center*) oder rechtsbündig (*ausrichtung=right*) ausgegeben werden. Standardmäßig werden mehrzeilige Beschriftungstexte linksbündig formatiert. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(135 90 135)
  PieChart(label+oval;10;50)
  BorderStyle(all;none)
  LabelTexts(1;"Bereich A\n|u|\n(|f1|%)";
             "Bereich B\n|u|\n(|f1|%)";
             "Bereich C\n|u|\n(|f1|%)")
  LabelStyle(all;;;bold+italic;darkBlue;center)
CloseDrawing()
```



Das 7. Argument *neigungswinkel* ermöglicht Texte in eine beliebige Richtung zu drehen. Bei Vorgabe eines Neigungswinkels größer Null werden die Texte im Uhrzeigersinn, bei einem negativen Neigungswinkel gegen den Uhrzeigersinn gedreht. Alle Winkel sind im Bereich von  $\pm 360$  Grad anzugeben, dabei ist 0 Grad horizontal (3 Uhr). Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(100 250 180 310 222 123 89 222)
  BarChart(label)
  LabelTexts(1;"|u|")
  LabelStyle;;;bold;;;-90)
  LabelOptions(1;out)
  GridLocation(xy;none) // Raster ausblenden
CloseDrawing()
```



Bei Vorgabe einer maximalen Breite (*maxBreite*>0) können längere, mehrzeilige Texte als Fließtext ausgegeben werden. Die Höhe eines Textblocks kann über das Argument *maxHöhe* kontrolliert werden. Texte, welche nicht vollständig in den durch *maxBreite* und *maxHöhe* vorgegebenen Bereich passen, werden entsprechend der Vorgabe durch das Argument *auslassungsPosition* gekürzt. Fünf Möglichkeiten stehen zur Verfügung:

*auslassungsPosition*=0: Text wird am Ende geklippt, ohne "..."

*auslassungsPosition*=1: Text wird am Anfang geklippt (...text)

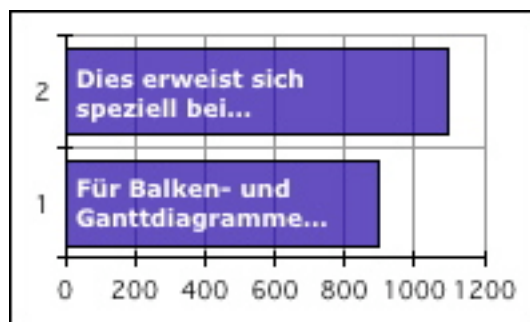
*auslassungsPosition*=2: Text wird in der Mitte geklippt (text...text)

*auslassungsPosition*=3: Text wird am Ende geklippt (text...) – default

*auslassungsPosition*=4: Text wird am Anfang und Ende geklippt (...text...)

Beispiele:

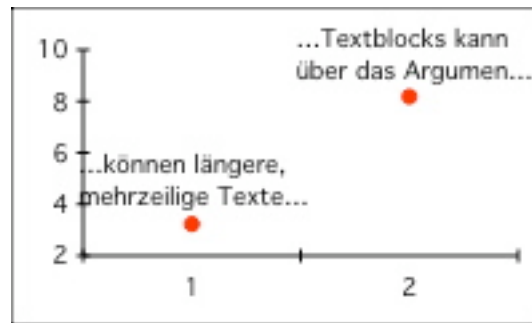
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(900 1100)
BarChart(label+horizontal;30)
FillStyle(1;0 0 150 200)
LabelStyle(1;"Verdana";;bold;white;left;;110;30;3)
LabelOptions(;begin)
LabelTexts(1;"Für Balken- und Ganttprogramme steht
noch eine weitere Option zur Verfügung." ; "Dies erweist sich
speziell bei Ganttprogrammen zur Projektverwaltung oder für
Terminplaner als vorteilhaft.")
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3.23 8.18)
  ScatterChart(label;on)
  SymbolStyle(1;bullet;5)
  LabelStyle(1;;;;;left;;85;30;4)
  LabelOptions(1;topCenter)
  LabelTexts(1;"Bei Vorgabe einer maximalen Breite
(maxBreite>0) können längere, mehrzeilige Texte als Fließtext
ausgegeben werden." ; "Die Höhe eines Textblocks kann über
das Argument maxHöhe kontrolliert werden.")
  GridLocation(xy;none)
CloseDrawing()

```



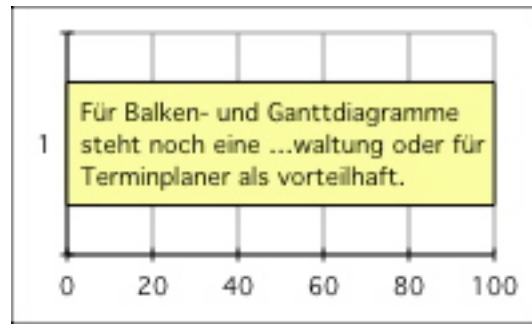
Standardmäßig werden *maxBreite=-1* und *maxHöhe=-1* gesetzt, d.h. keine Beschränkung auf eine bestimmte Breite oder Höhe und somit keine Fließtextunterstützung.

Für Balken- und Gantt-diagramme steht noch eine weitere Option zur Verfügung. Bei *maxBreite=-2* und/oder *maxHöhe=-2* wird die vorhandene Breite bzw. Höhe der Balken als Begrenzung für die Textblöcke verwendet. Dies erweist sich speziell bei Gantt-diagrammen zur Projektverwaltung oder für Terminplaner als vorteilhaft. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(100 )
  BarChart(label+horizontal;80)
  FillStyle(1;lightYellow)
  LabelStyle(1;;;;;-2;-2;2)
  LabelTexts(1;"Für Balken- und Gantt-diagramme steht
noch eine weitere Option zur Verfügung. Dies erweist sich
speziell bei Gantt-diagrammen zur Projektverwaltung oder für
Terminplaner als vorteilhaft.")
CloseDrawing()

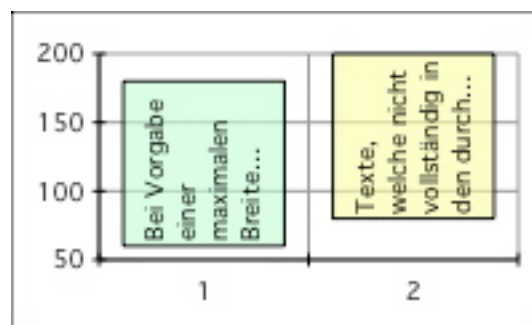
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(60 180;80 200)
  GanttChart(label+horizontal;30)
  FillStyle(1;200 255 200 150)
  FillStyle(2;255 255 150 150)
  LabelStyle(all;;;;;-90;-2;-2;3)
  LabelTexts(1;"Bei Vorgabe einer maximalen Breite
können längere, mehrzeilige Texte als Fließtext ausgegeben
werden. Die Höhe eines Textblocks kann über das Argument
maxHöhe kontrolliert werden.")
  LabelTexts(2;"Texte, welche nicht vollständig in den
durch maxBreite und maxHöhe vorgegebenen Bereich passen,
werden entsprechend der Vorgabe durch das Argument
auslassungsPosition gekürzt.")
CloseDrawing()

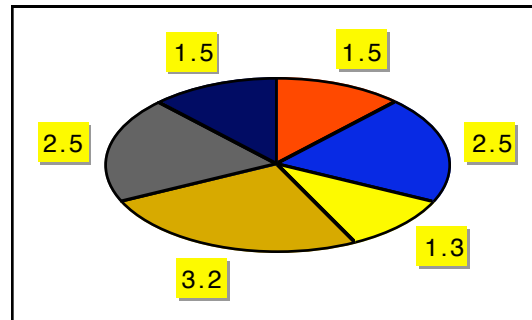
```



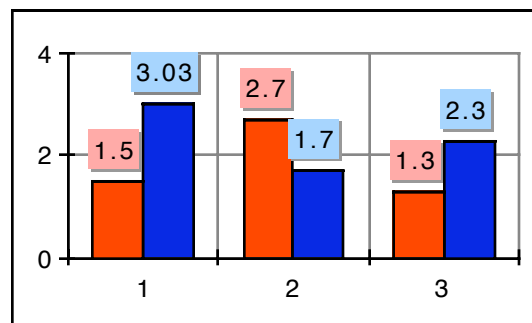
**LabelBackground(serienindex;füllfarbe;füllmuster;  
rahmenbreite;rahmenfarbe;rahmenmuster;  
schattenabstand;schattenfarbe;schattenmuster)**

Durch die Funktion LabelBackground( ) kann der Beschriftungshintergrund gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1.3 3.2 2.5 1.5)
  PieChart(oval+label)
  LabelBackground(1;yellow;;;0;;;1)
CloseDrawing()
```



```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.7 1.3; 3.03 1.7 2.3)
  BarChart(label)
  LabelBackground(1;lightRed;;;0;;;1)
  LabelBackground(2;lightBlue;;;0;;;1)
CloseDrawing()
```



**LabelOptions (serienindex;platzierung;hVersatz;vVersatz;  
unteresLimit;oberesLimit)**

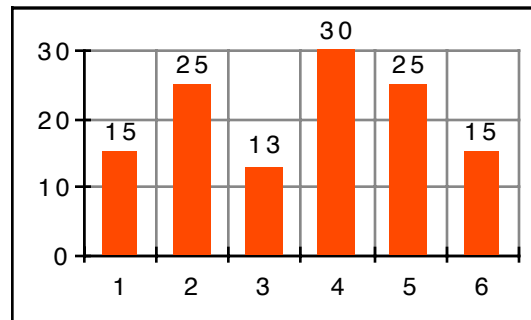
Die Funktion `LabelOptions ( )` stellt zahlreiche Möglichkeiten zur Platzierung und Feinabstimmung der Beschriftung zur Verfügung:

- *Platzierung der Beschriftung bei nicht gestapelten Balkendiagrammen:*  
Ist genügend Platz vorhanden, so wird standardmäßig die Beschriftung außerhalb der Balken platziert, ansonsten innerhalb. Mit dem Argument *platzierung* können bei nicht gestapelten Balkendiagrammen, nicht gestapelten Histogrammen und Gantt-Diagrammen neun Positionen definiert werden:

Konstante	Wert	Platzierung
<code>smartBegin</code>	1	Balkenanfang, falls genügend Platz
<code>smartCenter</code>	2	Balkenmitte, falls genügend Platz
<code>smartEnd</code>	3	Balkenende, falls genügend Platz
<code>begin</code>	4	Balkenanfang (zwingend)
<code>center</code>	5	Balkenmitte (zwingend)
<code>end</code>	6	Balkenende (zwingend)
<code>edge</code>	7	Rand am Balkenende (zwingend)
<code>smartOut</code>	8	außerhalb, falls genügend Platz (default)
<code>out</code>	9	außerhalb (zwingend)

Beispiele:

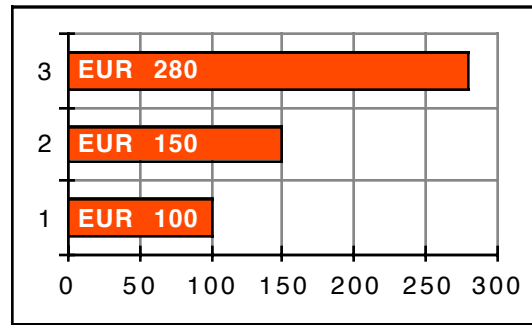
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(15 25 13 30 25 15)
  BarChart(label)
  BorderStyle(all;0)
  LabelOptions(1;out)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(100 150 280)
  BarChart(label+horizontal)
  LabelTexts(1;"EUR |u|")
  LabelStyle(;;;bold;white)
  LabelOptions(1;begin)
CloseDrawing()

```



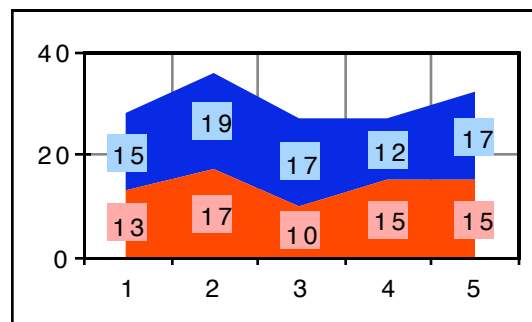
Weitere Beispiele finden sich im Abschnitt *Diagramme*.

- *Platzierung der Beschriftung bei gestapelten Diagrammen:*  
Bei gestapelten oder proportionalen Balken- oder Flächendiagrammen wird die Beschriftung standardmäßig in der Mitte der einzelnen Balken- bzw. Flächenabschnitte platziert; das Argument *platzierung* wird ignoriert. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(13 17 10 15 15; 15 19 17 12 17)
  AreaChart(stacked+label;on)
  BorderStyle(all;none)
  LabelBackground(1;lightRed;;0)
  LabelBackground(2;lightBlue;;0)
  GridFrame()
CloseDrawing()

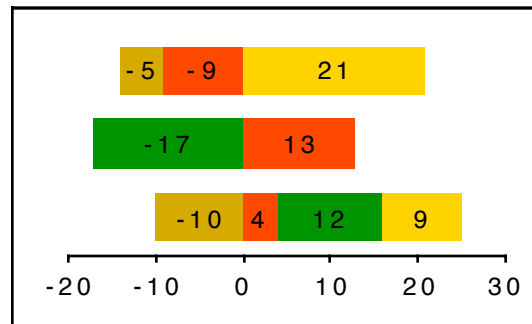
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4  13 -9;
            12 -17  0;
            9   0 21;
            -10  0 -5)
  BarChart(stacked+horizontal+label;50)
  BorderStyle(all;none)
  FillStyle(2;green)
  FillStyle(3;darkYellow)
  GridLocation(all;none) // kein Raster
  AxisOptions(y;none)    // keine y-Achse
  AxisMajorTicks(x;2;;;out)
CloseDrawing()

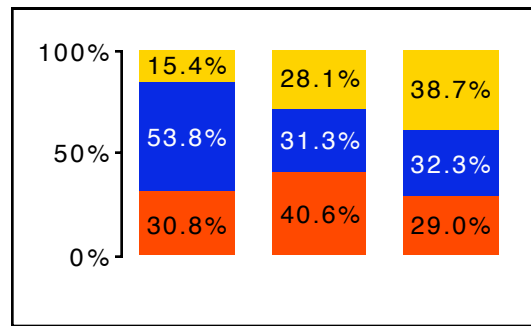
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 13 9;
            7 10 10;
            2 9 12)
  BarChart(proportional+label;40)
  Scaling(y;percent)
  BorderStyle(all;none) // keine Balkenberandung
  FillStyle(3;darkYellow)
  LabelTexts(all;"|2f1|")
  LabelStyle(2;;;white)
  GridLocation(all;none) // kein Raster
  AxisOptions(x;none)    // keine x-Achse
  AxisMajorTicks(y;2;;;out)
CloseDrawing()

```



Zusätzlich besteht die Möglichkeit, entweder die Gesamtsummen oder die laufenden Summen darzustellen. Die Darstellung und Platzierung der (laufenden) Summen werden durch die vier Stilfunktionen `LabelTexts()`, `LabelStyle()`, `LabelBackground()`, `LabelOptions()` in Verbindung mit `serienindex=-1` bzw. `serienindex=stacked` kontrolliert. Dieser spezielle Serienindex steht ausschließlich in den vier oben angeführten Stilfunktionen in Verbindung mit gestapelten oder proportionalen Diagrammen zur Verfügung. Dabei können die (laufenden) Summen entweder außerhalb (default) oder am Rand der einzelnen Balken- bzw. Flächenabschnitte platziert werden. Dazu stehen für das Argument *platzierung* die folgenden vier Konstanten zur Verfügung:

Konstante	Wert	Platzierung
<code>totalsOut</code>	1	die Gesamtsummen außerhalb
<code>totalsEdge</code>	2	die Gesamtsummen am Rand
<code>runningTotalsOut</code>	3	die laufenden Summen außerhalb
<code>runningTotalsEdge</code>	4	die laufenden Summen am Rand

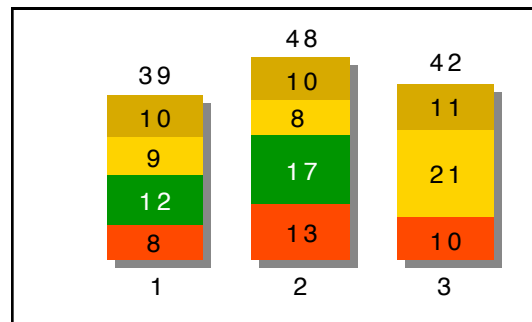
Die sonst üblichen Platzierungskonstanten werden bei `serienindex=-1` bzw. `serienindex=stacked` ignoriert.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 8  13 10;
             12 17  0;
             9   8 21;
             10 10 11)
  BarChart(stacked+label+shadow;50)
  LabelOptions(stacked;totalsOut)
  BorderStyle(all;none)
  LabelStyle(2;;;white)
  FillStyle(2;green)
  FillStyle(3;darkYellow)
  GridLocation(all;none) // kein Raster
  AxisOptions(y;none)    // keine y-Achse
  AxisMajorTicks(x;0)    // keine x-Achsenmarkierungen
  AxisLine(x;0)          // keine x-Achsenlinie
CloseDrawing()

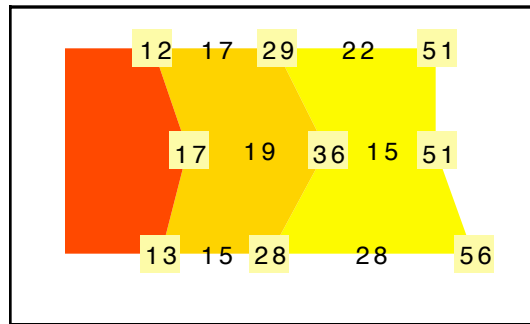
```



```

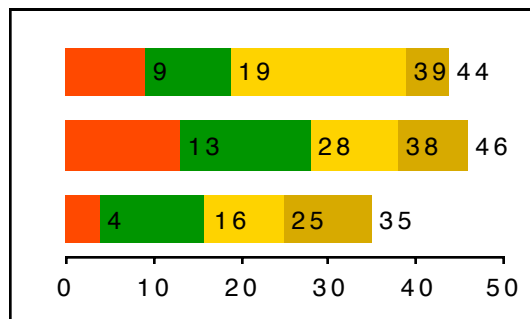
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(13 17 12;
             15 19 17;
             28 15 22)
  AreaChart(stacked+label+horizontal)
  LabelTexts(1;" ")
  BorderStyle(all;none)
  FillStyle(2;darkYellow)
  LabelOptions(-1;runningTotalsEdge)
  LabelBackground(stacked;lightYellow;;0)
  GridLocation(all;none) // Raster ausblenden
  AxisOptions(all;none)  // Achsen ausblenden
CloseDrawing()

```



Falls nur die Endsummen bzw. die laufenden Summen dargestellt werden sollen, kann man die "standardmäßige" Beschriftung durch Vorgabe einer "leeren Beschriftung", z.B. `LabelTexts(all;" ")`, ausblenden. Beispiele:

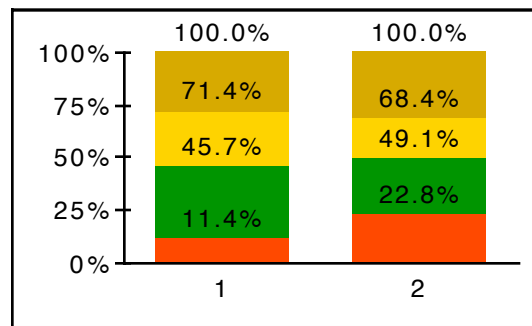
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13 9;
             12 15 10;
             9 10 20;
             10 8 5)
  BarChart(stacked+horizontal+label;50)
  LabelTexts(all;" ")
  LabelOptions(stacked;runningTotalsOut)
  BorderStyle(all;none)
  FillStyle(2;green)
  FillStyle(3;darkYellow)
  GridLocation(all;none) // kein Raster
  AxisOptions(y;none)    // y-Achse ausblenden
  AxisMajorTicks(x;2;;;out)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 13;
            12 15;
            9 11;
            10 18)
  BarChart(proportional+label;50)
  Scaling(y;percent;0;100;4)
  LabelTexts(all;" ")
  LabelTexts(stacked;"|2f1|%" )
  LabelOptions(stacked;runningTotalsOut)
  BorderStyle(all;none)
  FillStyle(2;green)
  FillStyle(3;darkYellow)
  GridLocation(all;none) // kein Raster
  AxisMajorTicks(x;0)
CloseDrawing()

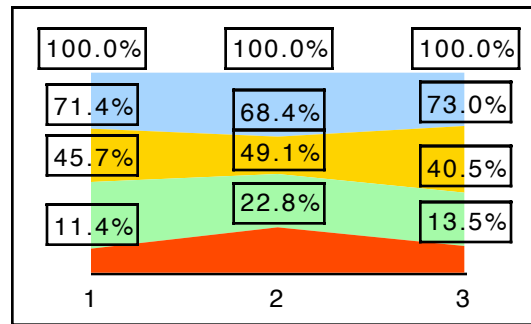
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenChart(30;25;140;75;on)
  ChartData(4 13 5; 12 15 10; 9 11 12; 10 18 10)
  AreaChart(proportional+label)
  LabelTexts(all;" ")
  LabelTexts(stacked;"|2f1|%" )
  LabelBackground(stacked;;transparent)
  LabelOptions(stacked;runningTotalsOut)
  BorderStyle(all;none)
  FillStyle(2;lightGreen)
  FillStyle(3;darkYellow)
  FillStyle(4;lightBlue)
  GridLocation(all;none)
  AxisOptions(y;none) // y-Achse ausblenden
  AxisMajorTicks(x;0)
CloseDrawing()

```



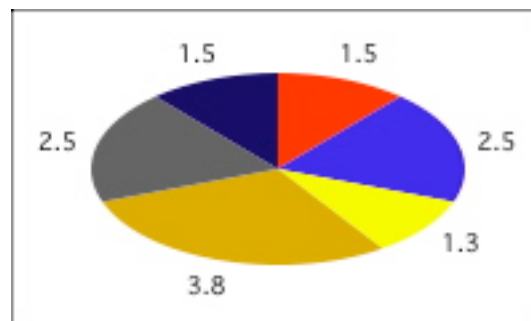
- *Platzierung der Beschriftung bei Tortendiagrammen:*

Bei Tortendiagrammen werden die Werte stets außerhalb des Diagramms platziert — das Argument *platzierung* wird ignoriert. Es besteht aber die Möglichkeit mit den Funktionen `PieChartInnerLabelTexts()` und `PieChartCenterLabelText()` zusätzlich Texte auch innerhalb und im Zentrum von Tortendiagrammen zu platzieren. Detaillierte Informationen dazu inkl. Beispiele finden sich im Abschnitt *Diagramme*. Beispiel:

```

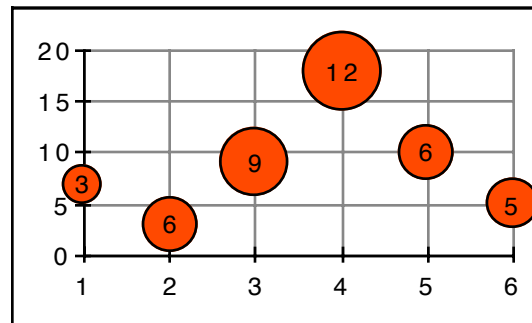
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(1.5 2.5 1.3 3.8 2.5 1.5)
PieChart(oval+label)
BorderStyle(all;none)
CloseDrawing()

```



- *Platzierung der Beschriftung bei Blasendiagrammen (bubble charts):*  
Standardmäßig wird die Beschriftung im Zentrum platziert (*platzierung=centerCenter*). Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(7 3 9 18 10 5; // Positionen
            3 6 9 12 6 5) // Durchmesser
  BubbleChart(label)
CloseDrawing()
```

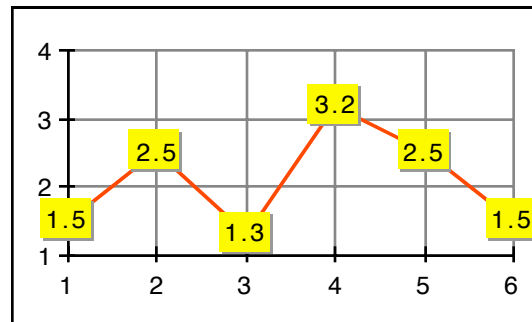


- *Platzierung der Beschriftung bei allen übrigen Diagrammen:*  
Standardmäßig wird die Beschriftung rechts oberhalb des Diagrammwerts positioniert. Mit dem Argument *platzierung* können neun Positionen definiert werden:

Konstante	Wert	Platzierung
topLeft	1	oben links
topCenter	2	oben mitte
topRight	3	oben rechts (default)
centerLeft	4	mitte links
centerCenter	5	mitte mitte
centerRight	6	mitte rechts
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

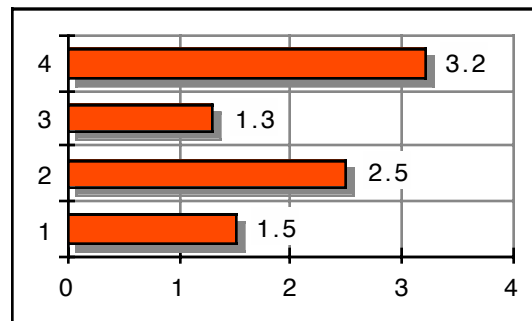
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1.3 3.2 2.5 1.5)
  LineChart(label)
  LabelBackground(1;yellow;;0;;1)
  LabelOptions(all;centerCenter)
CloseDrawing()
```



Optional kann noch mit den Argumenten *hVersatz* und *vVersatz* die Lage der Beschriftung feinabgestimmt werden. Positive Versatzwerte schieben die Beschriftung nach rechts unten, negative Werte nach links oben. Beispiel:

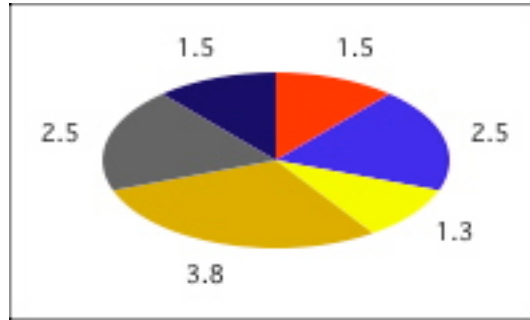
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1.3 3.2)
  BarChart(label+horizontal+shadow)
  LabelBackground(1;white;;0)
  LabelOptions(1;;3) // 3 Pixel nach rechts
CloseDrawing()
```



Bei Tortendiagrammen werden die Argumente *hVersatz* und *vVersatz* ignoriert; stattdessen ist es möglich, mit der Funktion `PieChartLabelOptions(;textversatzAußen)` einen radialen Versatzwert zu definieren. Ein positiven Versatzwert vergrößert den Abstand zwischen Berandung und Beschriftung, ein negativer Wert verkleinert den Abstand. Das Argument *textversatzAußen* ist in Prozent der Segmentlänge einzugeben.

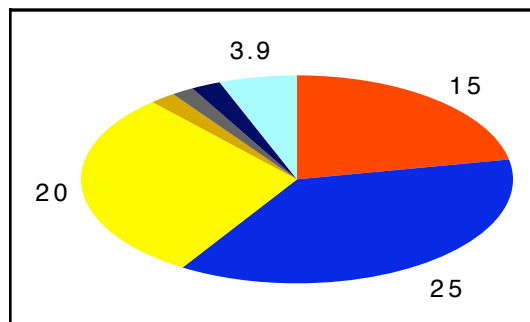
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1.3 3.8 2.5 1.5)
  PieChart(oval+label)
  PieChartLabelOptions(;10)
  BorderStyle(all;none)
CloseDrawing()
```



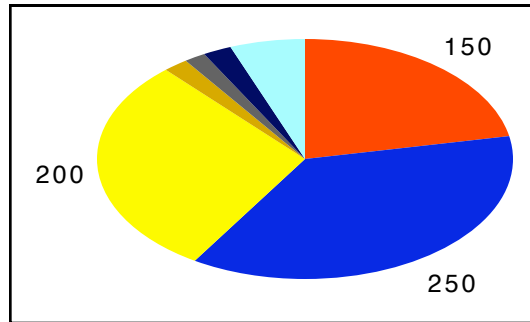
Mithilfe der Argumente *unteresLimit* und *oberesLimit* kann die Beschriftung von Diagrammwerten unterdrückt werden, d.h. nur Diagrammwerte größer als das untere Limit und kleiner als das obere Limit werden beschriftet. So kann zum Beispiel bei Tortendiagrammen durch Angabe eines unteren Limits die Beschriftung schmaler Segmente verhindert werden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(15 25 20 1.3 1.2 1.5 3.9)
  PieChart(label+oval)
  BorderStyle(all;none)
  LabelOptions(1;;0;0;3.0)
CloseDrawing()
```



Zusätzlich besteht bei Tortendiagrammen die Möglichkeit, mittels der Funktion `PieChartLabelOptions(relativeLimitsVerwenden)` zwischen absoluten und relativen Grenzwerten zu wählen. Relative Grenzwerte sind in Prozent bezogen auf die Gesamtsumme anzugeben. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(150 250 200 13 12 15 39)
PieChart(label+oval)
BorderStyle(all;none)
// alle Werte, die kleiner sind als
// 10% der Gesamtsumme ausblenden
LabelOptions(1;;0;0;10.0)
PieChartLabelOptions(on)
CloseDrawing()
```



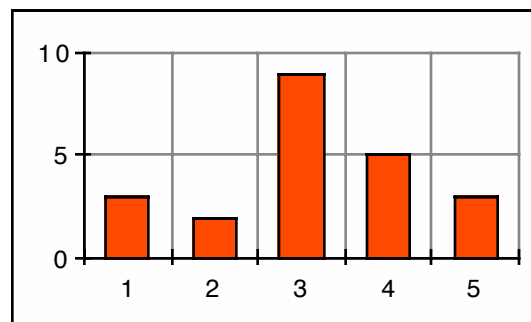
## Hintergrund

Zur Gestaltung des Hintergrundes stehen insgesamt vier Funktionen zur Verfügung, je zwei zur Gestaltung des "Gesamthintergrundes", d.h. der Hintergrund erstreckt sich über die gesamte Zeichnung oder des umschließenden Views, und zwei weitere Funktionen zur Gestaltung des eigentlichen Diagramm-Hintergrundes, d.h. der Hintergrund ist auf den Bereich des Diagrammrasters beschränkt.

**Background(füllfarbe;füllmuster;rahmenbreite;  
rahmenfarbe;rahmenmuster;schattenabstand;  
schattenfarbe;schattenmuster)**

Die Funktion Background() erlaubt die Festlegung eines einfarbigen Hintergrundes inklusive Berandung und Schatten. Wird die Funktion Background() ohne Argumente angeführt, so wird ein weißer Hintergrund mit einem 1 Pixel breiten schwarzen Rand ohne Schatten gezeichnet. Beispiel:

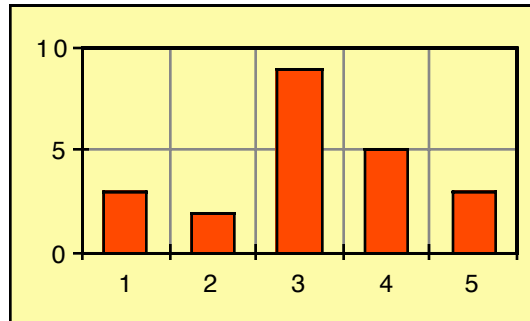
```
OpenDrawing(200;120)
  ChartData(3 2 9 5 3)
  BarChart()
  Background()
CloseDrawing()
```



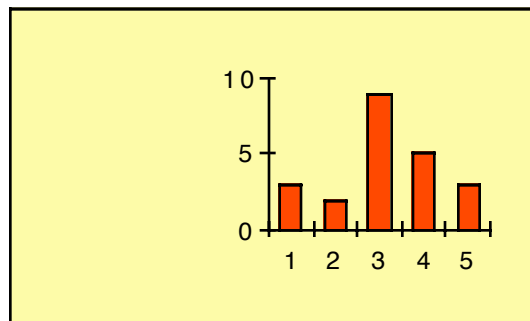
Werden keine Views verwendet oder wird die Hintergrundfunktion außerhalb der Views angeführt, bezieht sich die Funktion auf die gesamte Zeichnung. Views werden im Abschnitt *Layout* behandelt.

Beispiele:

```
OpenDrawing(200;120)
  ChartData(3 2 9 5 3)
  BarChart()
  GridFrame()
  Background(lightYellow)
CloseDrawing()
```



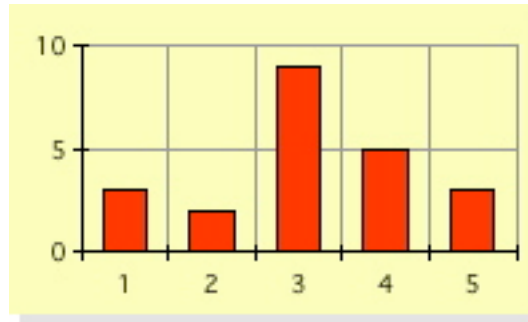
```
OpenDrawing(200;120)
  OpenView(70;10;120;100)
  ChartData(3 2 9 5 3)
  BarChart()
  GridLocation(all;none) // kein Raster
  CloseView()
  Background(lightYellow)
CloseDrawing()
```



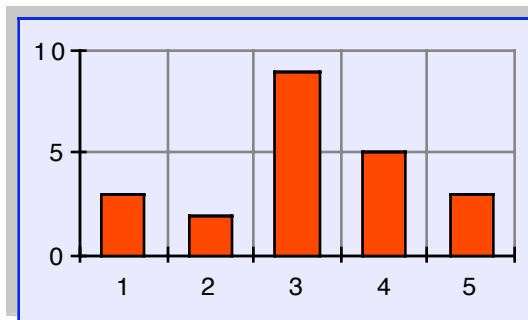
Ist der Schattenabstand positiv, so wird der Schatten rechts unten dargestellt, bei einem negativen Abstand, wird der Schatten links oben gezeichnet.

Beispiele:

```
OpenDrawing(200;120)
  ChartData(3 2 9 5 3)
  BarChart()
  Background(255 255 180;;;0;;;4;lightGray)
CloseDrawing()
```

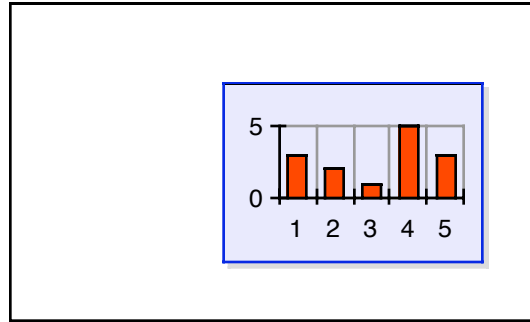


```
OpenDrawing(200;120)
  ChartData(3 2 9 5 3)
  BarChart()
  Background(230 230 255;;1;blue;;-4;200 200 200)
CloseDrawing()
```



Wird die Funktion Background( ) innerhalb eines Views angeführt, so bezieht sich die Funktion auf den Hintergrund des Views. Beispiele:

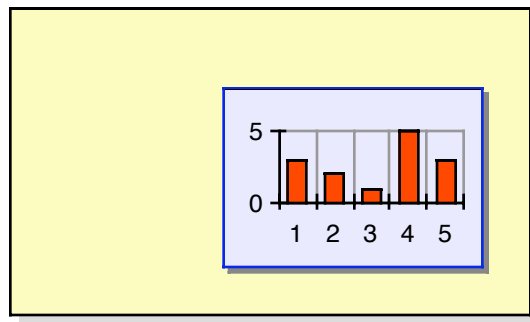
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenView(80;30;100;70)
    AddFrame(0;0;100;70)
    ChartData(3 2 1 5 3)
    BarChart()
    Background(230 230 255;;1;blue;;2;lightGray)
  CloseView()
CloseDrawing()
```



```

OpenDrawing(200;120)
  OpenView(80;30;100;70)
    AddFrame(0;0;100;70)
    ChartData(3 2 1 5 3)
    BarChart()
    Background(230 230 255;;1;blue;;2)
  CloseView()
  Background(255 255 180;;1;;4;lightGray)
CloseDrawing()

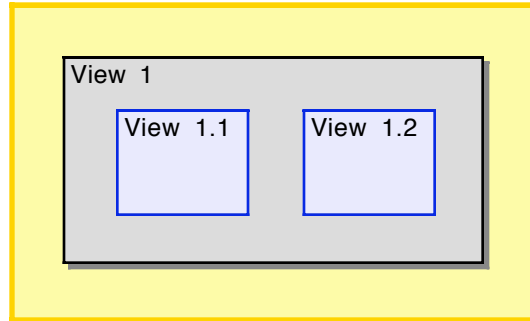
```



```

OpenDrawing(200;120)
  OpenView(20;20;160;80)
    Background(lightGray;;1;;2)
    AddText(3;10;"View 1")
    OpenView(20;20;50;40)
      Background(230 230 255;;1;blue)
      AddText(3;10;"View 1.1")
    CloseView()
    OpenView(90;20;50;40)
      Background(230 230 255;;1;blue)
      AddText(3;10;"View 1.2")
    CloseView()
  CloseView()
  Background(lightYellow;;2;darkYellow)
CloseDrawing()

```



**BackgroundPict(quelle;name;platzierung;angleichung;  
istProportional)**

Die Funktion BackgroundPict ( ) ermöglicht die Verwendung eines Bildes als Hintergrund. Dabei werden drei verschiedene Bildquellen unterstützt:

- *Zwischenablage: (quelle=clipboard)*

Die Zwischenablage wird verwendet, wenn als Quelle die Konstante *clipboard* eingegeben wird oder das argument *quelle* leer bleibt. Das 2. Argument *name* hat keine Bedeutung und wird ignoriert. Beispiele: BackgroundPict(clipboard)

BackgroundPict( )

- *Datei: (quelle=file)*

Es kann entweder der vollständige, absolute Dateipfad oder nur ein relativer Pfad vorgegeben werden. Der relative Pfad bezieht sich auf den Ordner in welchem sich die aktuelle FileMaker Pro Datei befindet. Beispiele:

BackgroundPict(file;"Bilder/Bild\_01.png")

BackgroundPict(file;"C:/Bilder/Bild\_01.png")

BackgroundPict(file;"Macintosh HD/Bilder/Bild\_01.pdf")

Unter Mac OS X können folgende Bildformate importiert werden: PDF, PICT, GIF, JPEG, PNG, BMP, TIFF.

Unter Windows können die folgende Bildformate importiert werden: WMF, EMF, GIF, JPEG, PNG, BMP, TIFF.

- *Resource: (quelle=resource)*

Zur Zeit stehen insgesamt 42 in Ressourcen abgespeicherte Farbverlaufbilder zur Verfügung. Der Zugriff erfolgt mittels einer Indexnummer zwischen 1 und 42. Dabei ist zu beachten, dass die Indexnummer unter Hochkomma zu setzen ist. Ein Überblick über die vordefinierten Bildressourcen findet sich in *xmReferenz*.

Weiters ist unter Windows bei der Verwendung von Ressourcen zu beachten, dass Diagramme zu einer Größe von mehreren MB anwachsen können. Dieses Problem tritt nur in Verbindung mit dem Standard-Ausgabeformat EMF auf, nicht jedoch bei der Ausgabe als Bitmap. EMF- und Bitmapformat werden im Zuge der Funktion OpenDrawing ( ) im Abschnitt *Layout* behandelt.

Das dritte Argument *platzierung* ermöglicht neun verschiedene Positionen zur Platzierung eines Hintergrundbildes.

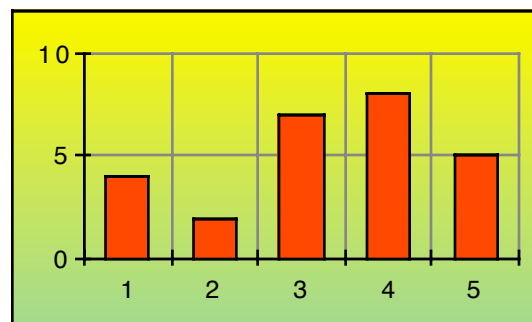
<i>Konstante</i>	<i>Wert</i>	<i>Anmerkung</i>
topLeft	1	oben links
topCenter	2	oben mitte
topRight	3	oben rechts
centerLeft	4	mitte links
centerCenter	5	mitte mitte (default)
centerRight	6	mitte rechts
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

Das 4. Argument *angleichung* stellt fünf Optionen zum Angleichen des Bildes an die Hintergrundfläche zur Verfügung.

<i>Konstante</i>	<i>Wert</i>	<i>Anmerkung</i>
crop	1	abschneiden
reduce	2	verkleinern
enlarge	3	vergrößern
reduceOrEnlarge	4	verkleinern od. vergrößern (default)
tile	5	aneinander reihen

Zusätzlich kann mit dem 5. Argument *istProportional* festgelegt werden, ob das Verhältnis Bildbreite zu Bildhöhe beim Anpassen an die Hintergrundfläche beibehalten werden soll. Bei *angleichung=crop* und *angleichung=tile* wird das Argument *istProportional* ignoriert. Beispiele:

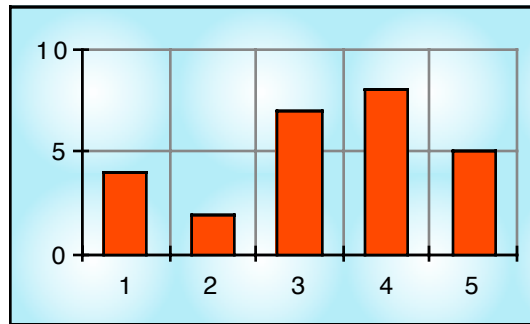
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(4 2 7 8 5)
BarChart()
BackgroundPict(resource;"7")
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 2 7 8 5)
  BarChart()
  BackgroundPict(resource;"36";;tile)
CloseDrawing()

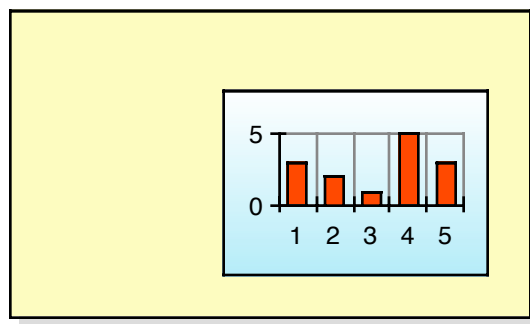
```



```

OpenDrawing(200;120)
  OpenView(80;30;100;70)
    AddFrame(0;0;100;70)
    ChartData(3 2 1 5 3)
    BarChart()
    BackgroundPict(resource;"3")
  CloseView()
  Background(255 255 180;;1;;;4;lightGray)
CloseDrawing()

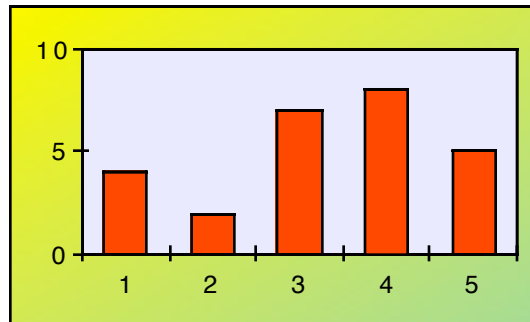
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 2 7 8 5)
  BarChart()
  GridLocation(all;none) // kein Raster
  ChartBackground(all;230 230 255)
  BackgroundPict(resource;"29")
CloseDrawing()

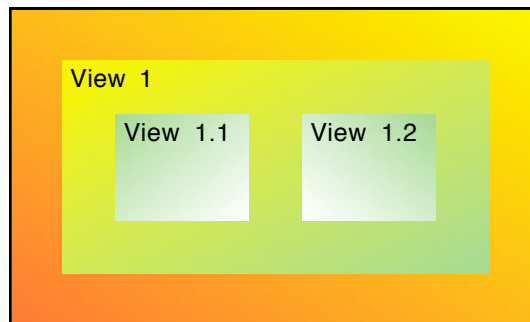
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenView(20;20;160;80)
    BackgroundPict(resource;"29")
    AddText(3;10;"View 1")
    OpenView(20;20;50;40)
      BackgroundPict(resource;"26")
      AddText(3;10;"View 1.1")
    CloseView()
    OpenView(90;20;50;40)
      BackgroundPict(resource;"28")
      AddText(3;10;"View 1.2")
    CloseView()
  CloseView()
  BackgroundPict(resource;"32")
CloseDrawing()

```



Weiters besteht die Möglichkeit, Hintergrundfunktionen mit Funktionen wie `AddFrame()` zu kombinieren. Beispiel:

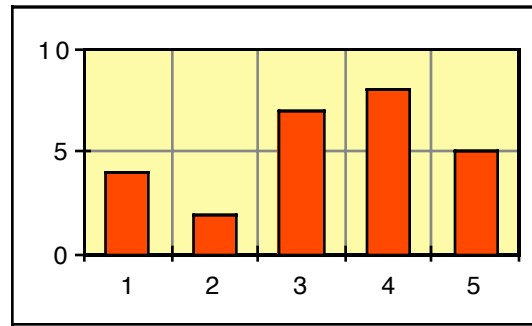
```
OpenDrawing(200;120)
  ChartData(4 2 7 8 5)
  BarChart()
  BackgroundPict(resource;"27")
  AddFrame(1;1;198;118;2;green)
CloseDrawing()
```



**ChartBackground(ebenenindex;füllfarbe;füllmuster;  
rahmenbreite;rahmenfarbe;rahmenmuster;  
schattenbreite;schattenfarbe;schattenmuster)**

Die Funktion `ChartBackground()` ist bis auf das 1. Argument *ebenenindex* ident aufgebaut wie die Funktion `Background()`. Im Gegensatz zur Funktion `Background()` ist die Hintergrundfläche bei der Funktion `ChartBackground()` auf den Bereich des Diagrammrasters begrenzt. Als erstes Argument *ebenenindex* ist bei 2-dimensionalen Diagrammen die Konstante *xy* oder *all* einzugeben. Bei 3-dimensionalen Balken- und Gantt-Diagrammen stehen noch zusätzlich die Konstanten *xz* und *yz* zur Verfügung. Die z-Achse zeigt dabei aus der Tiefe in die Richtung des Betrachters. Ansonst gelten die gleichen Überlegungen wie bei der Funktion `Background()` beschrieben. Beispiele:

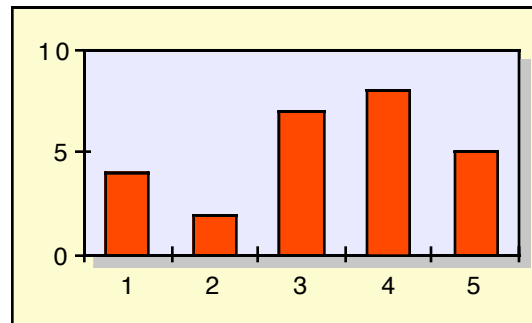
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 2 7 8 5)
  BarChart()
  GridFrame()
  ChartBackground(xy;lightYellow)
CloseDrawing()
```



```

OpenDrawing(200;120)
  ChartData(4 2 7 8 5)
  BarChart()
  GridLocation(all;none) // kein Raster
  ChartBackground(,230 230 255;;1;black;;4;200 200 200)
  Background(255 255 200)
CloseDrawing()

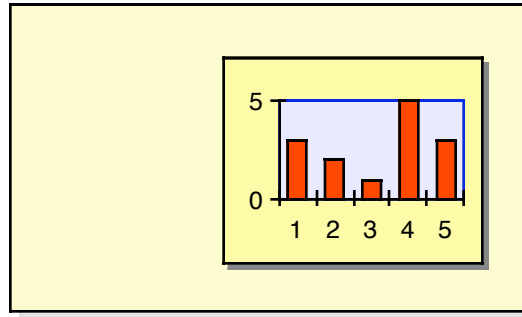
```



```

OpenDrawing(200;120)
  OpenView(80;20;100;80)
  ChartData(3 2 1 5 3)
  BarChart()
  GridLocation(all;none) // kein Raster
  ChartBackground(xy;230 230 255;;1;blue)
  Background(lightYellow;;;2)
  CloseView()
  Background(255 255 200;;1;;;4;lightGray)
CloseDrawing()

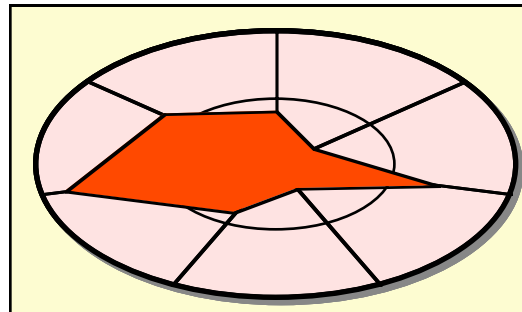
```



```

OpenDrawing(200;120)
  ChartData(4 2 7 2 4 9 6)
  RadarChart(oval)
  RadarChartOptions(0)
  MajorGridLineColors(all;all;black)
  ChartBackground(xy;255 220 220;;2;black;;2)
  Background(255 255 200;;1;;3;lightGray)
CloseDrawing()

```



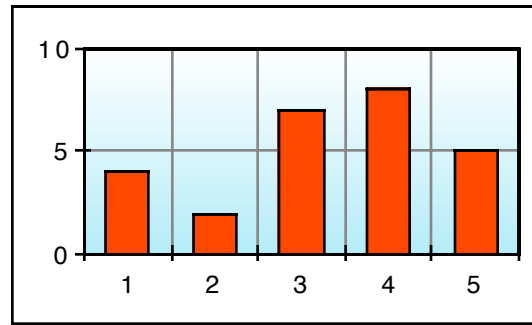
**ChartBackgroundPict(*ebenenindex*;*quelle*;*name*;  
                          *platzierung*;*angleichung*;*istProportional*)**

Mit der Funktion `ChartBackgroundPict()` kann analog zur Funktion `ChartBackground()` ein auf das Diagrammraster begrenzter Bildhintergrund definiert werden. Das 1. Argument *ebenenindex* ist ident wie in Funktion `ChartBackground()`, die restlichen Argumente *quelle*, *name*, *platzierung*, *angleichung* und *istProportional* sind gleich wie unter `BackgroundPict()` beschrieben. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 2 7 8 5)
  BarChart()
  GridFrame()
  ChartBackgroundPict(all;resource;"3")
CloseDrawing()

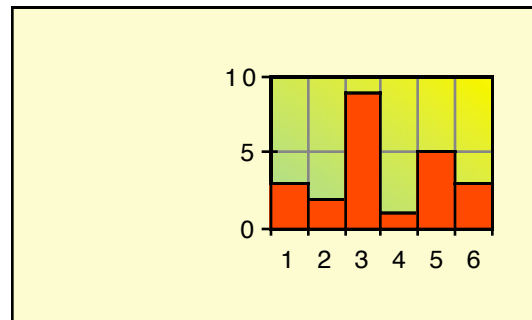
```



```

OpenDrawing(200;120)
  OpenView(70;10;120;100)
    ChartData(3 2 9 1 5 3)
    BarChart(;0)
    GridFrame()
    ChartBackgroundPict(xy;resource;"30")
  CloseView()
  Background(255 255 200)
CloseDrawing()

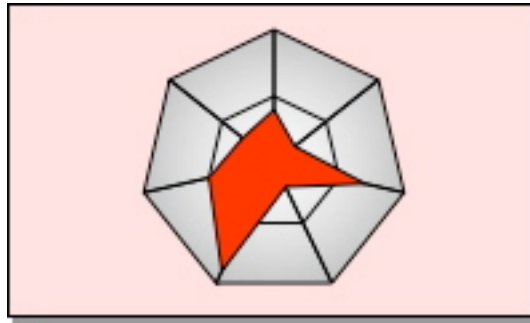
```



```

OpenDrawing(200;120)
  ChartData(4 2 7 2 9 5 3)
  RadarChart()
  RadarChartOptions(0;poly)
  MajorGridLineColors(all;all;black)
  ChartBackgroundPict(xy;resource;"34")
  Background(255 220 220;;1;black;;2;gray)
CloseDrawing()

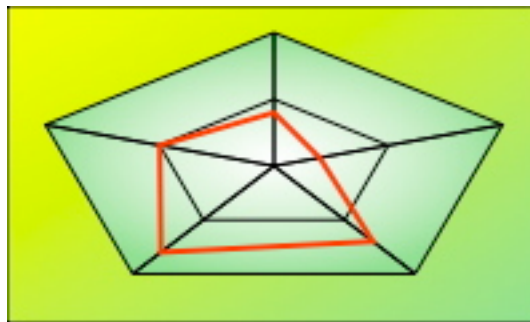
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 2 7 8 5)
  RadarChart(oval)
  RadarChartOptions(0;poly)
  FillStyle(all;;transparent)
  BorderStyle(1;poly;2;red)
  MajorGridLineColors(all;all;black)
  ChartBackgroundPict(xy;resource;"38")
  BackgroundPict(resource;"29")
CloseDrawing()

```



## Raster

Zur Gestaltung des Rasters stehen insgesamt 12 Funktionen zur Verfügung. Diese ermöglichen die Gestaltung von:

- Rasterlinien und Rasterflächen
- Grob- und Feinraster
- Rasterrahmen

Standardmäßig werden 1 Pixel breite, graue Rasterlinien verwendet.

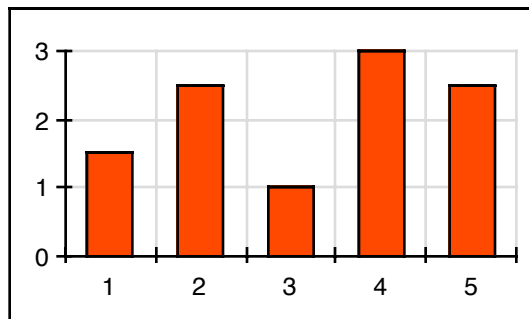
**MajorGridLineWidths(richtungsachse;verteilungsachse;  
strichstärke1;strichstärke2...)**

**MajorGridLineColors(richtungsachse;verteilungsachse;  
farbe1;farbe2...)**

**MajorGridLinePatterns(richtungsachse;verteilungsachse;  
muster1;muster2...)**

Rasterlinien können unterschiedlich in Farbe, Muster und Linienstärke definiert werden. Ist die Anzahl der Rasterlinien größer als die Anzahl der definierten Strichstärken, Farben oder Muster, so werden diese periodisch wiederholt. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1 3 2.5)
  BarChart()
  MajorGridLineColors(all;all;lightGray)
CloseDrawing()
```



Rasterlinien können durch Nullsetzen der Strichstärke unterdrückt werden. Falls weder die Richtungsachse noch die Verteilungsachse vorgegeben werden, beziehen sich die Funktionen auf alle Raster, d.h. bei zweidimensionalen kartesischen Diagrammen, sowohl auf das horizontale als auch auf das vertikale Raster. Der gezielte Zugriff auf eine spezielle Rasterrichtung erfolgt durch die beiden Argumente *richtungsachse* und *verteilungsachse*.

Bei 2-dimensionalen, kartesischen Diagrammen:

horizontales Raster: richtungsachse = x, verteilungsachse = y

vertikales Raster: richtungsachse = y, verteilungsachse = x

Bei 2-dimensionalen, polaren Diagrammen:

radiales Raster: richtungsachse = x, verteilungsachse = y

konzentr. Raster: richtungsachse = y, verteilungsachse = x

Bei 3-dimensionalen Balken- und Gantt-Diagrammen steht als Richtungs- bzw. Verteilungsachse noch zusätzlich die Konstante z zur Verfügung. Die z-Achse zeigt dabei aus der Tiefe in die Richtung des Betrachters.

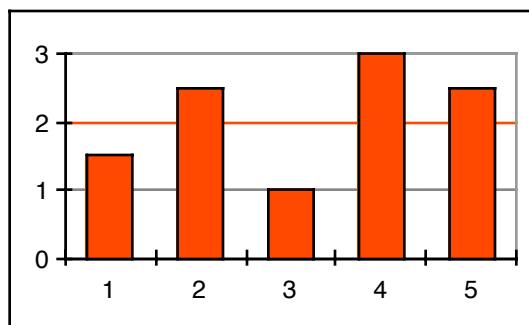
Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1 3 2.5)
  BarChart()

  // horizontale Rasterlinien
  MajorGridLineColors(x;y;gray;gray;red)

  // vertikale Rasterlinien ausblenden
  MajorGridLineWidths(y;x;0)

  // 1-Pixel breiter grauer Rahmen
  GridFrame(xy;1;gray)
CloseDrawing()
```



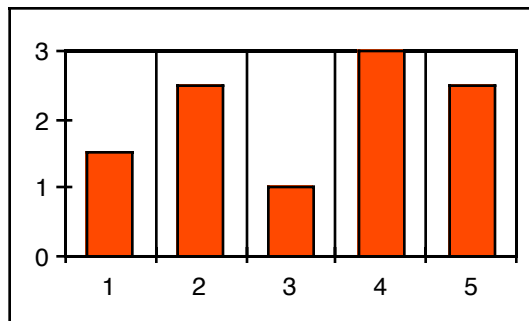
```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(1.5 2.5 1 3 2.5)
BarChart()
MajorGridLineColors(all;all;black)

// horizontale Rasterlinien ausblenden
MajorGridLineWidths(x;y;0)

// 1-Pixel breiter schwarzer Rahmen
GridFrame()
CloseDrawing()

```



```

MinorGridLineWidths(richtungsachse;verteilungsachse;  

                    strichstärke1;strichstärke2...)  

MinorGridLineColors(richtungsachse;verteilungsachse;  

                    farbe1;farbe2...)  

MinorGridLinePatterns(richtungsachse;verteilungsachse;  

                    muster1;muster2...)

```

Die Funktionsweise des Feinrasters ist gleich dem Grobraster. Zu beachten ist, dass das Feinraster erst durch Vorgabe einer Feinintervallanzahl größer 1 in der Funktion `Scaling()` — siehe Abschnitt *Achsen* — aktiviert wird.

Beispiele:

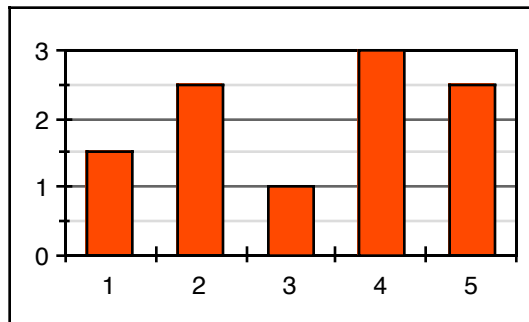
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1 3 2.5)
  BarChart()

  // 3 Grobintervalle, 2 Feinintervalle
  Scaling(y;linear;;;3;2)

  // Grobraster
  MajorGridLineColors(all;all;darkGray)
  // vertikale Rasterlinien ausblenden
  MajorGridLineWidths(y;x;0)

  // Feinraster
  MinorGridLineColors(all;all;lightGray)

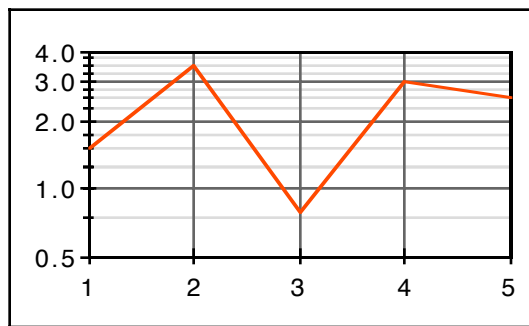
  // 1-Pixel breiter schwarzer Rahmen
  GridFrame()
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 3.5 0.8 3 2.5)
  LineChart()
  Scaling(y;log;;;3;4)
  // Grobraster
  MajorGridLineColors(all;all;darkGray)
  // Feinraster
  MinorGridLineColors(all;all;lightGray)
  // 1-Pixel breiter schwarzer Rahmen
  GridFrame()
CloseDrawing()

```



Zusätzlich zur Strichstärke kann optional durch Anfügen einer Liste von Strichlängen und -abständen ein Strichmuster festgelegt werden.  
Beispiel:

```

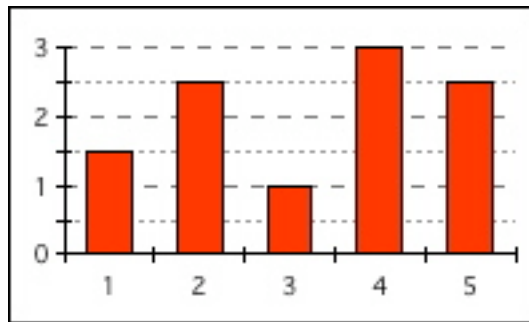
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1 3 2.5)
  BarChart()

  // 3 Grobintervalle, 2 Feinintervalle
  Scaling(y;linear;;;3;2)

  // Grobraster
  MajorGridLineColors(x;y;darkGray)
  MajorGridLineWidths(x;y;1 5 5)
  // vertikale Rasterlinien ausblenden
  MajorGridLineWidths(y;x;0)

  // Feinraster
  MinorGridLineColors(x;y;gray)
  MinorGridLineWidths(x;y;1 2 2)
CloseDrawing()

```



Weitere Beispiele betreffend Strichmuster finden sich im Abschnitt *Stile*, Funktion `LineStyle()`.

```
MajorGridStripeColors(richtungsachse;verteilungsachse;  
                        farbe1;farbe2...)  
MajorGridStripePatterns(richtungsachse;  
                         verteilungsachse;  
                         muster1;muster2...)  
MinorGridStripeColors(richtungsachse;verteilungsachse;  
                       farbe1;farbe2...)  
MinorGridStripePatterns(richtungsachse;  
                         verteilungsachse;  
                         muster1;muster2...)
```

Rasterstreifen können optional anstelle von Rasterlinien oder auch in Kombination mit Rasterlinien verwendet werden. Der Aufbau der Funktionen und deren Handhabung ist gleich den Rasterlinien-Funktionen. Rasterstreifen werden sowohl bei Diagrammen mit kartesischen Koordinaten wie Balken- oder Liniendiagrammen als auch bei Polar- und Radar-diagrammen unterstützt.

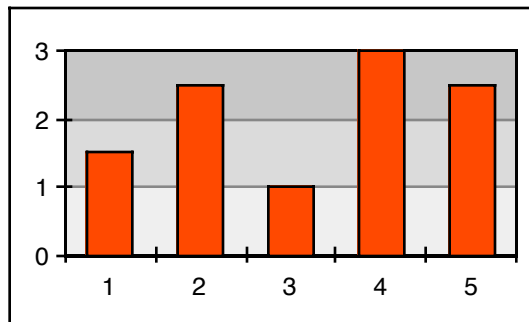
Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1 3 2.5)
  BarChart()
  // horizontale Rasterstreifen
  MajorGridStripeColors(x;y;240 240 240;
                        220 220 220;
                        200 200 200)

  // vertikale Rasterlinien ausblenden
  MajorGridLineWidths(y;x;0)
  // 1-Pixel breiter schwarzer Rahmen
  GridFrame()
CloseDrawing()

```



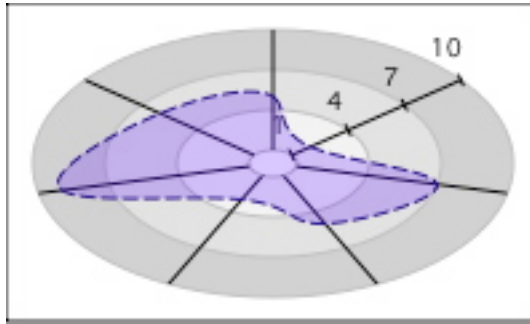
```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 2 7 5 3 9 6)
  RadarChart(oval)
  RadarChartOptions(2)
  Scaling(x;linear;1;10;3)
  FillStyle(1;50 0 255 70)
  BorderStyle(1;smooth;1 6 2;darkBlue)

  // Raster
  MajorGridLineWidths(x;y;0)
  MajorGridLineColors(y;x;180 180 180)
  MajorGridStripeColors(y;x;240 240 240;
                        220 220 220;
                        200 200 200)

CloseDrawing()

```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1 3 2.5)
  BarChart(horizontal)
  Scaling(x;linear;0;3;3;4)

  // Alle Marker ausblenden
  AxisMajorTicks(all;0)
  AxisMinorTicks(all;0)

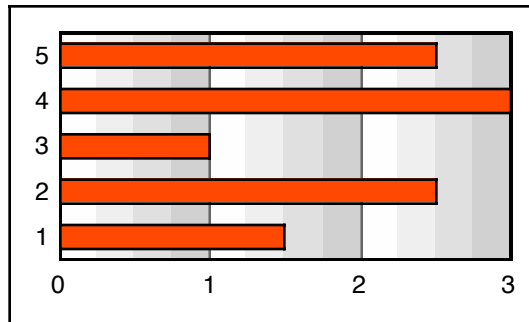
  // vertikale Rasterlinien
  MajorGridLineColors(y;x;darkGray)

  // vertikale Rasterstreifen
  MinorGridStripeColors(y;x;255 255 255;
                        240 240 240;
                        225 225 225;
                        210 210 210)

  // Rasterlinien ausblenden
  MajorGridLineWidths(x;y;0)
  MinorGridLineWidths(y;x;0)

  // 1-Pixel breiter schwarzer Rahmen
  GridFrame()
CloseDrawing()

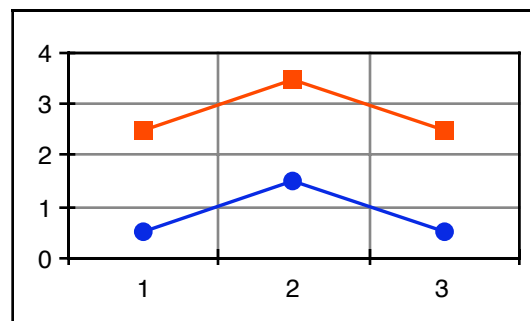
```



### **GridFrame(ebenenindex;strichstärke;farbe;muster)**

Durch die Funktion GridFrame( ) besteht die Möglichkeit, einen Rahmen um das Raster zu ziehen. Dabei kann die Breite, Farbe und das Muster des Rahmens kontrolliert werden. Als 1. Argument *ebenenindex* ist bei 2-dimensionalen Diagrammen die Konstante *xy* oder *all* einzugeben. Bei 3-dimensionalen Balken- und Gantt-Diagrammen stehen zusätzlich noch die Konstanten *xz* und *yz* zur Verfügung. Die z-Achse zeigt dabei aus der Tiefe in die Richtung des Betrachters. Beispiel:

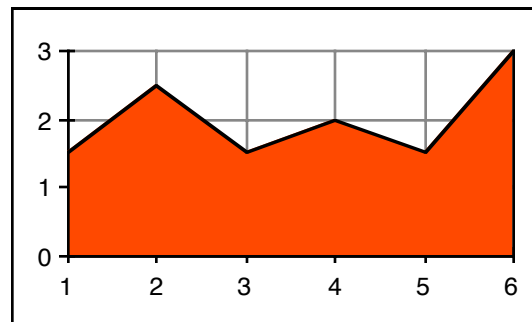
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(2.5 3.5 2.5; 0.5 1.5 0.5)
  LineChart(symbol;on)
  SymbolStyle(1;square;6)
  SymbolStyle(2;bulet;6)
  // 1-Pixel breiter schwarzer Rahmen
  GridFrame(xy;1;black)
CloseDrawing()
```



**GridLocation(ebenenindex;rasteranordnung)**

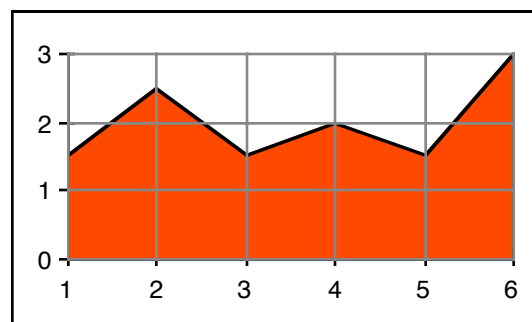
Standardmäßig wird das Raster stets im Hintergrund eines Diagramms dargestellt (*rasteranordnung=back*). Als 1. Argument *ebenenindex* ist bei 2-dimensionalen Diagrammen die Konstante *xy* oder *all* einzugeben. Bei 3-dimensionalen Balken- und Gantt-Diagrammen stehen zusätzlich noch die Konstanten *xz* und *yz* zur Verfügung. Die z-Achse zeigt dabei aus der Tiefe in die Richtung des Betrachters. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1.5 2 1.5 3)
  AreaChart()
  GridLocation(xy;back)
CloseDrawing()
```



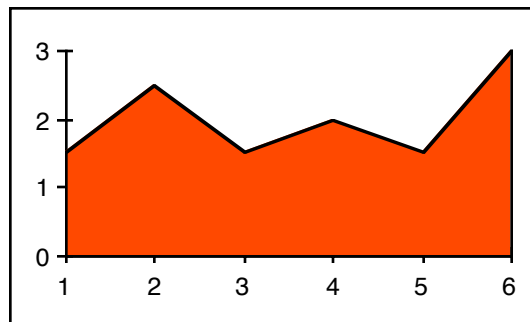
Durch *rasteranordnung=front* kann das Raster im Vordergrund platziert werden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1.5 2 1.5 3)
  AreaChart()
  GridLocation(xy;front)
CloseDrawing()
```



Optional können Raster durch *rasteranordnung=none* auf einfache Weise ausgeblendet werden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1.5 2.5 1.5 2 1.5 3)
  AreaChart()
  GridLocation(xy;none) // kein Raster
CloseDrawing()
```



## Achsen

Zur Gestaltung der Achsen stehen insgesamt 18 Funktionen zur Verfügung. Diese ermöglichen:

- die Festlegung unterschiedlicher Skalierungen
- die Gestaltung der Achsenlinien und Skalenmarkierungen
- ein flexibles Gestalten und Platzieren der Achsenbeschriftungen
- die Feinabstimmung mittels zahlreicher Optionen

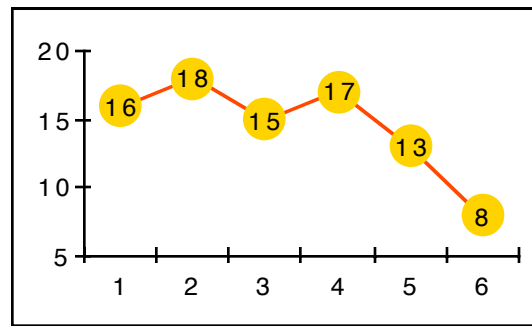
Bei allen 18 Funktionen wird als erstes Argument der Achsenindex festgelegt. Dazu stehen die vier Konstanten *x*, *y*, *z* und *all* zur Verfügung. Wird kein Achsenindex vorgegeben oder *achsenindex=all* gesetzt, bezieht sich die jeweilige Funktion auf alle Achsen, d.h. zum Beispiel bei zweidimensionalen Diagrammen auf die x- und y-Achse.

**Scaling(achsenindex;typ;minWert;maxWert;  
grobintervallAnzahl;feinintervallAnzahl;  
logBasis;äquidistanteLogSkala)**

Mit der Skalierungsfunktion `Scaling()` kann die Art der Skalierung, sowie die Anzahl der Unterteilungen und der Wertebereich festgelegt werden. Wird keine Skalierungsfunktion angeführt, so wird automatisch eine passende lineare Skalierung berechnet.

Das 2. Argument *typ* ermöglicht die Wahl zwischen linearer (*typ=linear*), prozentueller (*typ=percent*) und logarithmischer Skalierung (*typ=log*). Wird kein Typ vorgegeben, so wird eine linear Skalierung verwendet. Beispiele:

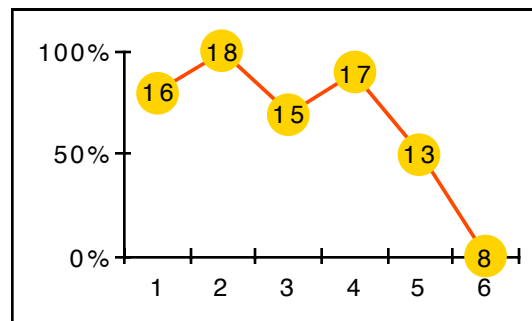
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 15 17 13 8)
  LineChart(symbol+label;on)
  Scaling(y;linear) // entspricht autom. Skalierung
  SymbolStyle(1;bullet;15;;darkYellow)
  LabelOptions(all;centerCenter)
  GridLocation(all;none)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 15 17 13 8)
  LineChart(symbol+label;on)
  Scaling(y;percent)
  SymbolStyle(1;bullet;15;;darkYellow)
  LabelOptions(all;centerCenter)
  GridLocation(all;none)
CloseDrawing()

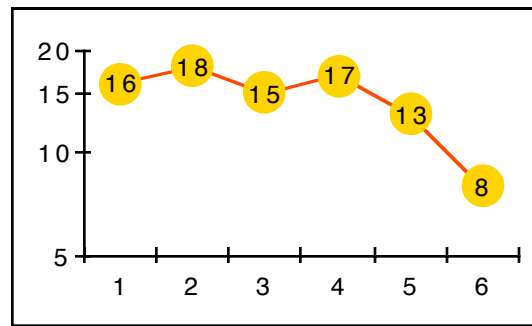
```



```

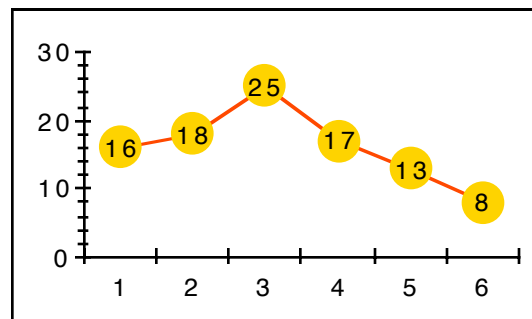
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 15 17 13 8)
  LineChart(symbol+label;on)
  Scaling(y;log)
  SymbolStyle(1;bullet;15;;darkYellow)
  LabelOptions(all;centerCenter)
  GridLocation(all;none)
CloseDrawing()

```



Durch das 3. und 4. Argument kann ein minimaler und maximaler Skalenwert vorgegeben werden. Wird der Minimalwert oder der Maximalwert nicht vorgegeben, so wird automatisch ein passender Wert gesucht. Mit dem 5. und 6. Argument kann die Anzahl der Grob- und Feinintervalle kontrolliert werden. Die Anzahl der Feinintervalle bezieht sich dabei auf die Anzahl der Intervalle innerhalb eines Grobintervalls, d.h. sollen zum Beispiel die Grobintervalle durch fünf Feinintervalle unterteilt werden, so ist *feinintervallAnzahl*=5 zu setzen; sollen keine Feinmarkierungen verwendet werden, so ist *feinintervallAnzahl*=1 (default). Falls die Anzahl der Grobintervalle nicht vorgegeben wird, wird die Skala standardmäßig in vier Grobintervalle unterteilt. Beispiele:

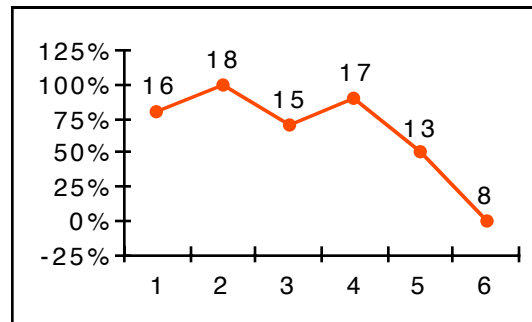
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 25 17 13 8)
  LineChart(symbol+label;on)
  Scaling(y;linear;0;30;3;5) // 3 Grob- u. 5 Feininter.
  SymbolStyle(1;bullet;15;;darkYellow)
  LabelOptions(all;centerCenter)
  GridLocation(all;none)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 15 17 13 8)
  LineChart(symbol+label;on)
  Scaling(y;percent;-25;125;6)
  SymbolStyle(1;bullet;4;;red)
  LabelOptions(all;topCenter)
  GridLocation(all;none)
CloseDrawing()

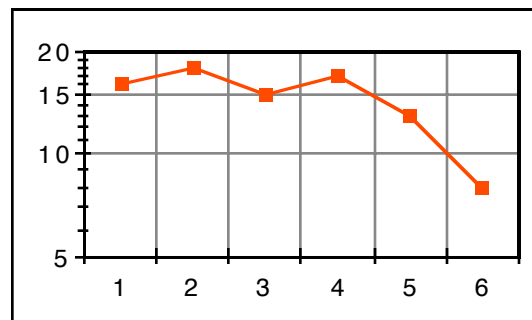
```



```

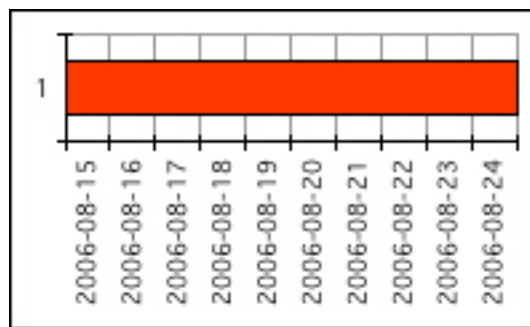
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 15 17 13 8)
  LineChart(symbol;on)
  Scaling(y;log;5;20;3;5) // 3 Grob- u. 5 Feininter.
  SymbolStyle(1;square;4;;red)
  MinorGridLineWidths(all;all;0) // kein Feinraster
  GridFrame()
CloseDrawing()

```

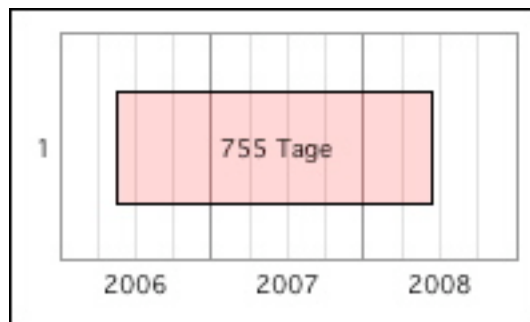


Als Minimal- und Maximalwert können auch Datumswerte und/oder Uhrzeiten verwendet werden (Zeitachse). Weitere Hinweise dazu finden sich im Abschnitt *Daten*, Kapitel *Eingabe von Datum und Uhrzeit*. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(2006-08-15 2006-08-24)
  GanttChart()
  Scaling(x;linear)
  AxisMajorTickLabelStyle(x;;;;;-90)
CloseDrawing()
```



```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(2006-05-22 2008-06-14)
  GanttChart(label)
  FillStyle(1;255 0 0 50)
  Labeltexts(1;"|u| Tage")
  Scaling(x;linear;2006/1/1;2008/12/31;;4)
  AxisLine(all;0)
  AxisMajorTicks(all;0)
  AxisMinorTicks(all;0)
  AxisMajorTickLabelTexts(x;"|YYYY|")
CloseDrawing()
```



Im Falle einer Zeitachse können die Grob- und Feinintervalle durch folgende Skalierungskonstanten definiert werden.

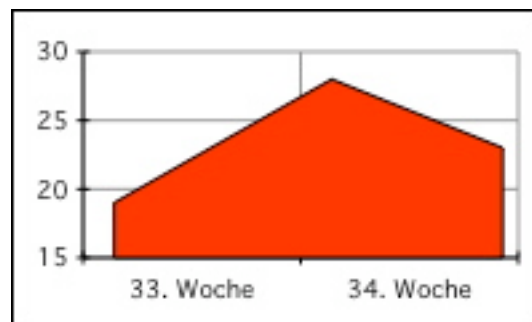
<i>Konstante</i>	<i>Wert</i>	<i>Anmerkung</i>
year	-1	Jahresintervall
quarter	-2	Quartalsintervall
month	-3	Monatsintervall
week	-4	Wochenintervall
day	-5	Tagesintervall
hour	-6	Stundenintervall
minute30	-7	30-Minutenintervall
minute20	-8	20-Minutenintervall
minute15	-9	15-Minutenintervall
minute10	-10	10-Minutenintervall
minute5	-11	5-Minutenintervall
minute	-12	Minutenintervall
second30	-13	30-Sekundenintervall
second20	-14	20-Sekundenintervall
second15	-15	15-Sekundenintervall
second10	-16	10-Sekundenintervall
second5	-17	5-Sekundenintervall
second	-18	Sekundenintervall

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  DateTimeOptions(ymd;2)
  ChartData(2006-08-15 2006-08-22 2006-08-27&12:00;
            19 28 23)
  AreaChart2D(;1)
  Scaling(x;linear;;;week)
  AxisMajorTickLabelTexts(x;"|WY|. Woche")
CloseDrawing()

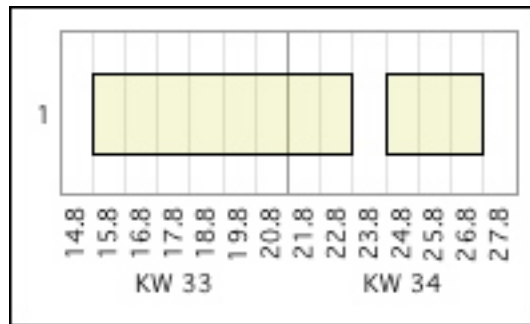
```



```

OpenDrawing(200;120;;2)
  AddFrame(0;0;200;120)
  DateTimeOptions(ymd;2)
  ChartData(2006-08-15 2006-08-22
            2006-08-24 2006-08-26)
  Ganttchart()
  FillStyle(1;200 200 0 50)
  Scaling(x;linear;;;week;day)
  AxisLine(all;0)
  AxisMajorTicks(all;0)
  AxisMinorTicks(all;0)
  AxisMajorTickLabelTexts(x;"KW |WY|")
  AxisMajorTickLabelOptions(x;out;;24)
  AxisMinorTickLabelTexts(x;"|D.M|")
  AxisMinorTickLabelStyle(x;;;;;-90)
CloseDrawing()

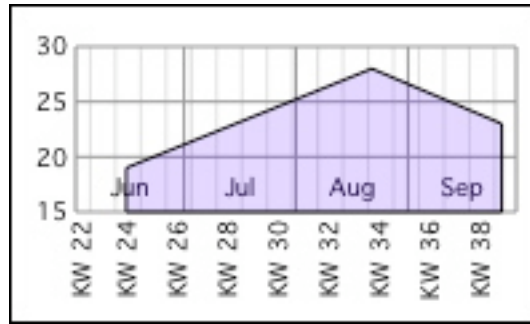
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  DateTimeOptions(ymd;2)
  ChartData(2006-06-15 2006-8-22 2006-9-27;
            19 28 23)
  AreaChart2D(;1)
  FillStyle(1;50 0 255 50)
  Scaling(x;linear;;;month;week)
  AxisLine(all;0)
  AxisMajorTicks(all;0)
  AxisMinorTicks(all;0)
  AxisMajorTickLabelTexts(x;"|Mon|")
  AxisMajorTickLabelOptions(x;in)
  AxisMinorTickLabelTexts(x;"KW |WY|")
  AxisMinorTickLabelStyle(x;;;;;-90)
  AxisMinorTickLabelOptions(x;;;;;2;off)
CloseDrawing()

```

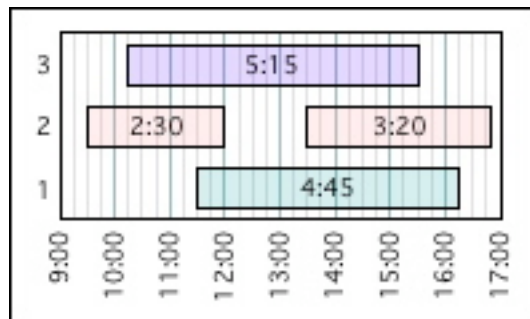


```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(11:30 16:15;
          9:30 12:00 13:30 16:50;
          10:15 15:30)
GanttChart(label;55)
FillStyle(1; 0 150 150 50)
FillStyle(2;255 150 150 50)
FillStyle(3; 50 0 255 50)

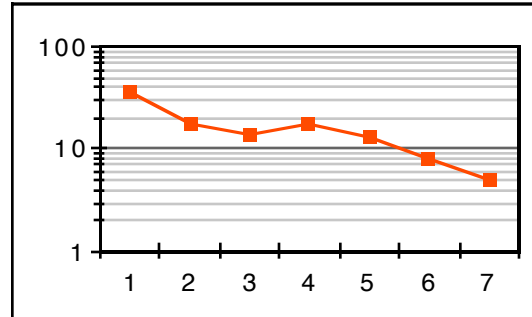
Scaling(x;linear;;;hour;minute15)
AxisLine(all;0)
AxisMajorTicks(all;0)
AxisMinorTicks(all;0)
AxisMajorTickLabelTexts(x;"|h:mm|")
AxisMajorTickLabelStyle(x;;;;;-90)
MajorGridLineWidths(x;y;0)
MajorGridLineColors(y;x;150 180 180)
GridFrame()
CloseDrawing()

```

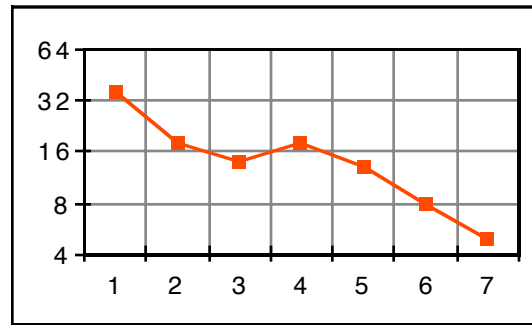


Die Argumente *logBasis* und *äquidistanteLogSkala* finden ausschließlich bei einer logarithmischen Skalierung Verwendung. Mit dem Argument *logBasis* wird die Basis des Logarithmus festgelegt. Falls keine Basis vorgegeben wird, wird der dekadische Logarithmus (*basis=10*) zur Skalierung verwendet. Durch Aktivieren des Arguments *äquidistanteLogSkala=on* finden nur Skalenwerte Verwendung, welche ein Vielfaches der Basis sind, zum Beispiel bei einer logarithmischen Skalierung zur Basis 10, werden nur Werte wie 0.01, 0.1, 1, 10, 100, 1000 usw. verwendet. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(36 18 14 18 13 8 5)
  LineChart(symbol;on)
  Scaling(y;log;;;9;10;on) // 9 Feinintervalle
  SymbolStyle(1;square;4;;red)
  MajorGridLineWidths(y;x;0) // kein vertikales Raster
  MajorGridLineColors;;;100 100 100)
  MinorGridLineColors;;;200 200 200)
  GridFrame()
CloseDrawing()
```



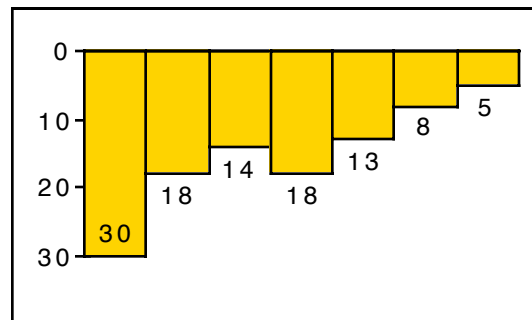
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(36 18 14 18 13 8 5)
  LineChart(symbol;on)
  Scaling(y;log;;;2;on) // Log. zur Basis 2
  SymbolStyle(1;square;4;;red)
  GridFrame()
CloseDrawing()
```



**ScalingOptions(achsenindex;umgekehrteSkala;  
nurGanzzahlen;nullNichtDarstellen)**

Durch Aktivieren des 2. Arguments *umgekehrteSkala=on* kann die Richtung der Skalierung umgedreht werden, d.h. die Skalierung der x-Achse verläuft nicht mehr aufsteigend von links nach rechts, sondern umgekehrt, aufsteigend von rechts nach links, die y-Achse entsprechend, aufsteigend von oben nach unten. Beispiele:

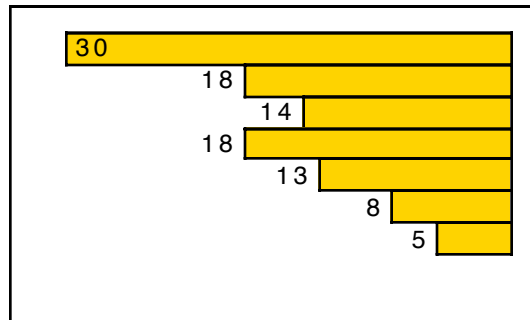
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(30 18 14 18 13 8 5)
  BarChart(label;0)
  FillStyle(1;darkYellow)
  ScalingOptions(y;on) // y-Skala umkehren
  AxisOptions(x;none) // keine x-Achse
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(30 18 14 18 13 8 5)
  BarChart(horizontal+label;0)
  FillStyle(1;darkYellow)
  ScalingOptions(all;on) // x- und y-Skala umkehren
  AxisOptions(all;none) // keine Achsen
  GridLocation(all;none) // kein Raster
CloseDrawing()

```

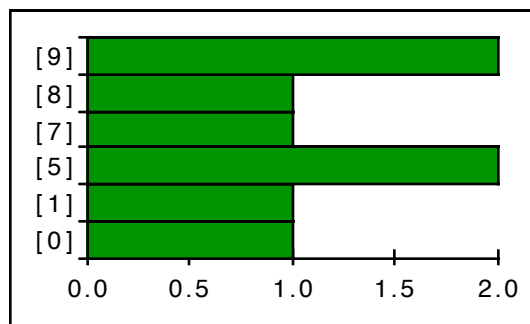


Durch Aktivieren des 3. Arguments *nurGanzzahlen=on* können nicht-ganzzahlige Skalierungswerte ausgeblendet werden. Dies erweist sich zweckmäßig zur Darstellung ganzzahliger Größen, wie zum Beispiel bei Häufigkeitsverteilungen. Beispiel:

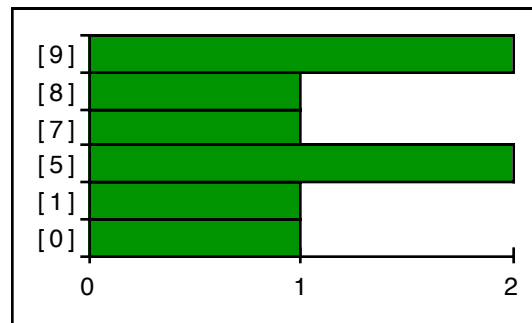
```

// ohne ScalingOptions()
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 9 0 9 5 5 7 8)
  Histogram(horizontal)
  HistogramOptions(on) // nach Histogramm(!)
  FillStyle(all;green)
  GridLocation(xy;none)
CloseDrawing()

```

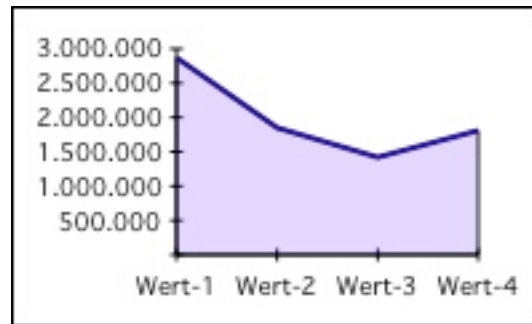


```
// mit ScalingOptions()
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 9 0 9 5 5 7 8)
  Histogram(horizontal)
  HistogramOptions(on) // nach Histogram(!)
  FillStyle(all;green)
  GridLocation(xy;none)
  ScalingOptions(x;;on) // nur ganzzahlige x-Skala
CloseDrawing()
```



Wird das 4. Argument *nullNichtDarstellen=on* gesetzt, wird die Skalenbeschriftung für den Wert 0 unterdrückt. Dies erweist sich bisweilen als vorteilhaft, wenn die Skalenbeschriftungen am Kreuzungspunkt von x- und y-Achse sehr knapp aneinander rücken. Beispiel:

```
OpenDrawing(200;120)
  SetThousandsSep(".")
  AddFrame(0;0;200;120)
  ChartData(2870000 1850000 1420000 1810000)
  AreaChart()
  AreaChartOptions(on)
  FillStyle(1;50 0 255 50)
  LineStyle(1;;2;darkBlue)
  Scaling(y;linear;0;3000000;6)
  ScalingOptions(y;;;on) // "0" ausblenden
  AxisMajorTickLabelTexts(x;"Wert-|u|")
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



**AxisLine(achsenindex;strichstärke;farbe;muster)**

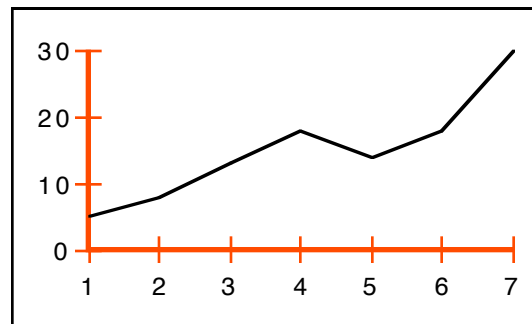
**AxisMajorTicks(achsenindex;länge;strichstärke;farbe;  
muster;platzierung)**

**AxisMinorTicks(achsenindex;länge;strichstärke;farbe;  
muster;platzierung)**

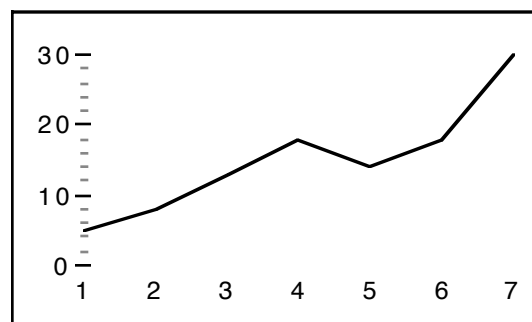
Durch die Funktion `AxisLine()` kann das Aussehen der Achsenlinien kontrolliert werden, durch die beiden Funktionen `AxisMajorTicks()` und `AxisMinorTicks()` das Aussehen der Grob- und Feinmarkierungen. Standardmäßig werden Achsenlinien und Skalenmarkierungen in schwarz und in einer Breite von 1 Pixel dargestellt. Die Strichstärke, die Farbe und das Muster können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Zusätzlich können bei den Grob- und Feinmarkierungen mit dem Argument *länge* die Markierungslänge in Pixel und mit dem Argument *platzierung* die Ausrichtung der Markierungen kontrolliert werden. Markierungen können entweder innerhalb des Diagramms platziert werden (*platzierung=in*), außerhalb des Diagramms (*platzierung=out*) oder zur Hälfte innerhalb und zur Hälfte außerhalb (*platzierung=center*). Die Grobmarkierungen sind standardmäßig 5 Pixel lang, die Feinmarkierungen 3 Pixel.

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 8 13 18 14 18 30)
  LineChart()
  LineStyle(1;poly;1;black)
  AxisLine(all;2;red)
  AxisMajorTicks(all;9;1;red)
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



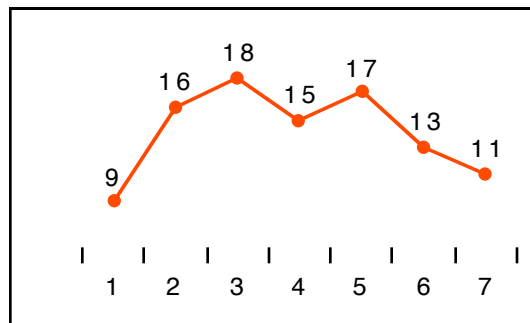
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 8 13 18 14 18 30)
  LineChart()
  Scaling(y;linear;0;30;3;5) // 3 Grob- u. 5 Feininter.
  LineStyle(1;poly;1;black)
  AxisLine(all;0) // keine Achsenlinien
  AxisMajorTicks(x;0) // keine x-Skalenmarkierungen
  AxisMajorTicks(y;5)
  AxisMinorTicks(y;2;1;gray)
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



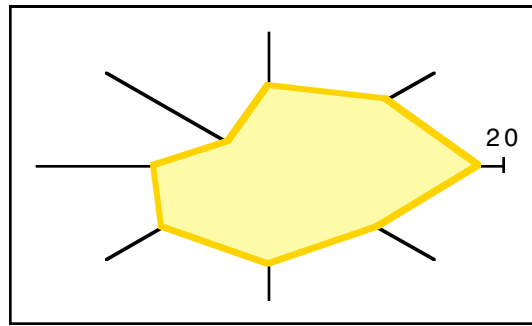
### **AxisOptions(achsenindex;achsenanordnung; achseVerschieben)**

Mit dem 2. Argument *achsenanordnung* können Achsen entweder hinter dem Diagramm angeordnet werden (*achsenanordnung=back*) oder davor (*achsenanordnung=front*). Letztere Anordnung erweist sich zum Beispiel zweckmäßig bei Polar- und Radardiagrammen. Optional können Achsen durch *achsenanordnung=none* komplett ausgeblendet werden. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(9 16 18 15 17 13 11)
  LineChart(symbol+label;on)
  SymbolStyle(1;bullet;4;;red)
  LabelOptions(all;topCenter)
  AxisOptions(y;none) // keine y-Achse
  AxisLine(x;0) // x-Achsenlinie ausblenden
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



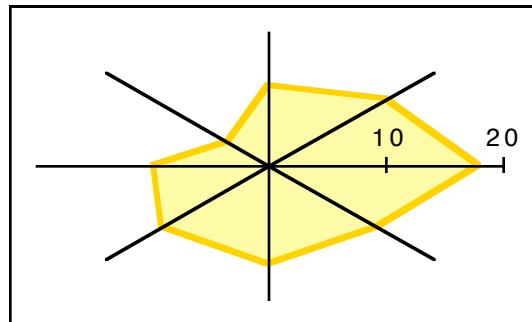
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(18 13 15 13 10 5 12 14)
  RadarChart(oval;90)
  FillStyle(1;lightYellow)
  BorderStyle(1;poly;2;darkYellow)
  AxisOptions(all;back) // Achsen im Hintergr.(default)
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(18 13 15 13 10 5 12 14)
  RadarChart(oval;90)
  FillStyle(1;lightYellow)
  BorderStyle(1;poly;2;darkYellow)
  AxisOptions(all;front) // Achsen in den Vordergrund
  GridLocation(all;none) // kein Raster
CloseDrawing()

```

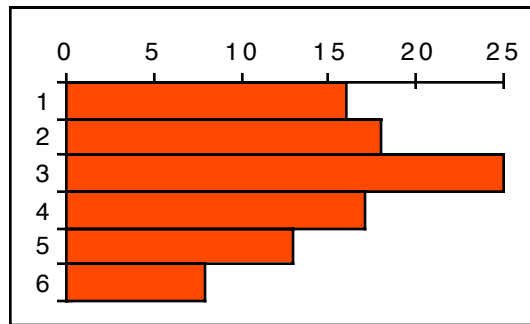


Durch Aktivieren des 3. Arguments *achseVerschieben=on* können Achsen auf die gegenüberliegende Seite verschoben werden, d.h. die x-Achse wird von unten nach oben verschoben, die y-Achse von links nach rechts. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 25 17 13 8)
  BarChart(horizontal;0)
  ScalingOptions(y;on) // y-Skalierung umkehren
  AxisOptions(x;;on) // x-Achse verschieben
  GridLocation(all;none) // kein Raster
CloseDrawing()

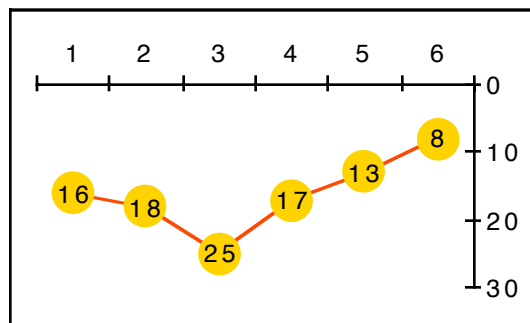
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 25 17 13 8)
  LineChart(symbol+label;on)
  ScalingOptions(y;on) // y-Skalierung umkehren
  SymbolStyle(1;bullet;15;;darkYellow)
  LabelOptions(all;centerCenter)
  AxisOptions(all;;on) // x- und y-Achse verschieben
  GridLocation(all;none) // kein Raster
CloseDrawing()

```



### **AxisLabelText(achsenindex;text1;text2...)**

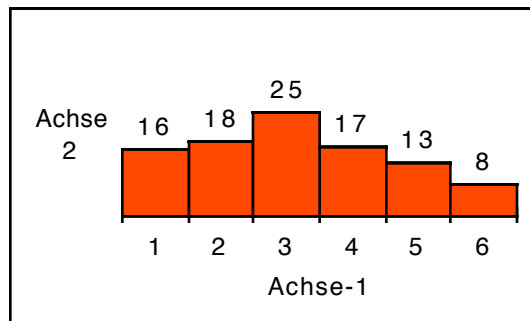
Durch die Funktion `AxisLabelText()` können Achsen mit einer Beschriftung versehen werden. Dabei sind auch mehrzeilige Bezeichnungen durch Einfügen eines Zeilenumbruchs "`\n`" möglich. Enthält der Text eine Formatanweisung, wie zum Beispiel "`|u|`", wird der Achsenindex ausgegeben, d.h. bei der x-Achsenbeschriftung "1", bei der y-Achsenbeschriftung "2". Die Handhabung von Texten in Verbindung mit Formatanweisungen wird im Abschnitt *Stile*, Funktion `LabelTexts()` erläutert.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 25 17 13 8)
  BarChart(label;0)
  AxisLine(y;0) // keine y-Achsenlinie
  AxisMajorTicks(y;0) // keine y-Markierungen
  AxisMajorTickLabelTexts(y;"") // keine y-Beschriftung
  AxisLabelText(x;"Achse-|u|")
  AxisLabelText(y;"Achse\n|u|")
  GridLocation(all;none) // kein Raster
CloseDrawing()

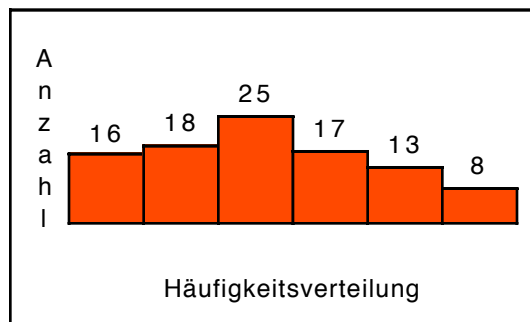
```



```

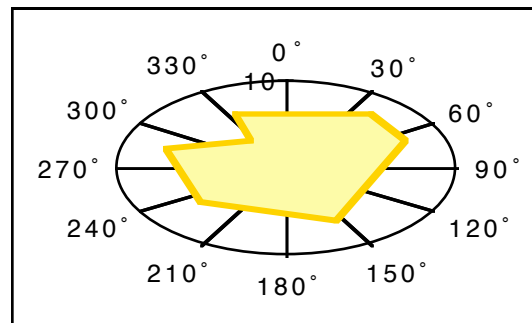
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 25 17 13 8)
  BarChart(label;0)
  AxisLine(all;0) // keine Achsenlinie
  AxisMajorTicks(all;0) // keine Markierungen
  AxisMajorTickLabelTexts(all;"") // keine Skalenbesch.
  AxisLabelText(x;"Häufigkeitsverteilung")
  AxisLabelText(y;"A\nn\nz\na\nh\nl")
  GridLocation(all;none) // kein Raster
CloseDrawing()

```

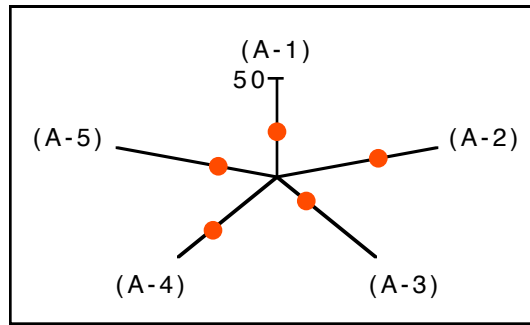


Bei Polar- und Radardiagrammen können auch mehrere Texte übergeben werden, die den einzelnen Achsen entsprechend zugeordnet werden. Ist die Anzahl der übergebenen Texte niedriger als die Anzahl der Achsen, so werden die Texte periodisch wiederholt. Enthält ein Text eine Formatanweisung, so wird bei Polardiagrammen der Winkel in Grad ausgegeben, bei Radardiagrammen der Achsenindex. Zu beachten ist, dass bei Polar- und Radardiagrammen der *achsenindex=1* oder *achsenindex=x* zu setzen ist, alle anderen Achsenindizes werden ignoriert. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(5 7 3 -5 -7 -2 -3; // x-Werte
            6 3 -6 -4 2 3 6) // y-Werte
  PolarChart(oval)
  FillStyle(1;lightYellow)
  BorderStyle(1;poly;2;darkYellow)
  MajorGridLineColors(all;all;black)
  AxisLabelText(all;"|u|°")
CloseDrawing()
```



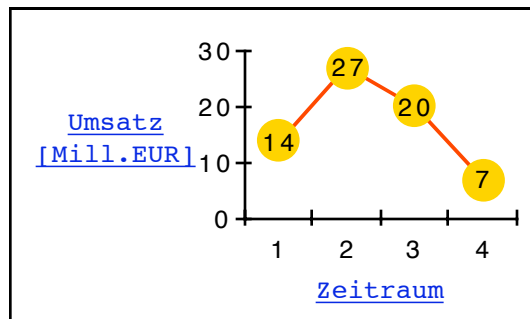
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 32 15 33 18)
  RadarChart(oval+symbol)
  FillStyle(;;transparent)
  BorderStyle(;;none)
  SymbolStyle(1;bullet;6;1;red)
  AxisLabelText(1;"(A-|u|)")
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



**AxisLabelStyle(achsenindex;schrift;größe;stil;farbe;  
ausrichtung;neigungswinkel;maxBreite;  
maxHöhe;auslassungsPosition)**

Mit der Funktion `AxisLabelStyle()` kann der Achsenbeschriftung ein Schriftstil zugeordnet werden. Die Attribute *neigungswinkel*, *maxBreite*, *maxHöhe* und *auslassungsPosition* werden im Abschnitt *Stile*, Funktion `LabelStyle()` erläutert. Beispiele:

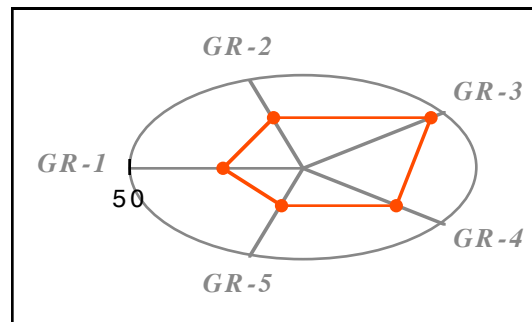
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(14 27 20 7)
  LineChart(symbol+label;on)
  SymbolStyle(1;bullet;15;;darkYellow)
  LabelOptions(all;centerCenter)
  AxisLabelText(x;"Zeitraum")
  AxisLabelText(y;"Umsatz\n[Mill.EUR]")
  AxisLabelStyle(all;"Courier";10;underline;blue)
  GridLocation(all;none)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 29 45 33 21)
  RadarChart(oval+symbol;-90)
  FillStyle(all;;transparent)
  BorderStyle(1;poly;1;red)
  SymbolStyle(1;bullet;4;1;red)
  AxisLine(1;0)
  AxisLabelText(1;"GR-|u|")
  AxisLabelStyle(1;"Times";10;bold+italic;gray)
CloseDrawing()

```



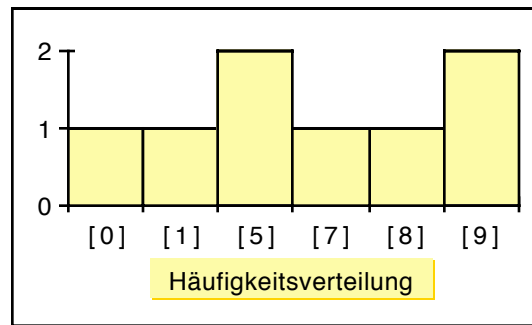
**AxisLabelBackground(achsenindex;füllfarbe;füllmuster;  
rahmenbreite;rahmenfarbe;rahmenmuster;  
schattenabstand;schattenfarbe;schattenmuster)**

Durch die Funktion AxisLabelBackground() kann der Hintergrund der Achsenbeschriftung gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 9 0 9 5 5 7 8)
  Histogram()
  HistogramOptions(on) // nach Histogramm(!)
  FillStyle(all;lightYellow)
  ScalingOptions(y;;on) // nur ganzzahlige y-Skala
  AxisLabelText(x;" Häufigkeitsverteilung ")
  AxisLabelBackground(x;lightYellow;;0;;1;darkYellow)
  GridLocation(xy;none)
CloseDrawing()

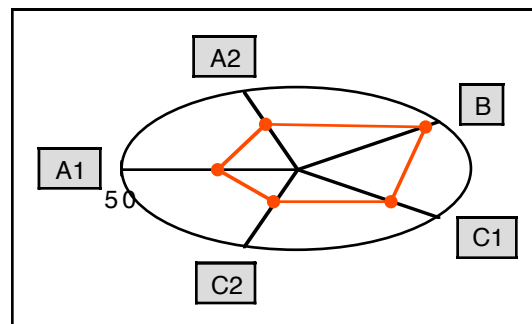
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(23 29 45 33 21)
RadarChart(oval+symbol;-90)
FillStyle(all;;transparent)
BorderStyle(1;poly;1;red)
SymbolStyle(1;bullet;4;1;red)
AxisLine(1;0)
AxisLabelText(1;" A1 ";" A2 ";" B ";" C1 ";" C2 ")
AxisLabelBackground(x;lightGray;;1)
MajorGridLineColors(all;all;black)
CloseDrawing()

```



**AxisLabelOptions(achsenindex;platzierung;hVersatz;  
vVersatz)**

Durch die Funktion AxisLabelOptions() kann die Platzierung der Achsenbeschriftung kontrolliert werden. Das Argument *platzierung* erlaubt die Auswahl unter neun vorgegebenen Positionen.

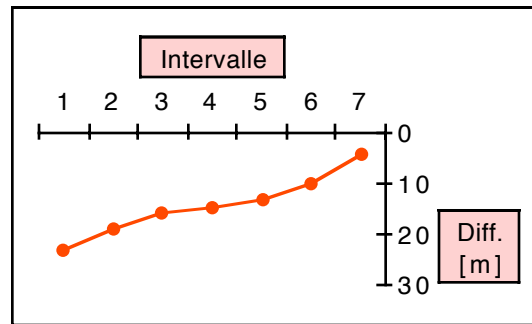
<i>Konstante</i>	<i>Wert</i>	<i>Anmerkung</i>
topLeft	1	oben links
topCenter	2	oben mitte
topRight	3	oben rechts
centerLeft	4	mitte links
centerCenter	5	mitte mitte
centerRight	6	mitte rechts
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

Standardmäßig wird die Beschriftung der x-Achse in der Mitte unterhalb der Achsenlinie platziert (*platzierung=bottomCenter*), die Beschriftung der y-Achse in der Mitte links von der Achsenlinie (*platzierung=centerLeft*). Bei Polar- und Radardiagrammen wird die Achsenbeschriftung immer außerhalb des Diagramms platziert, das Argument *platzierung* wird ignoriert. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 19 16 15 13 10 4)
  LineChart(symbol;on)
  SymbolStyle(1;bullet;4)
  ScalingOptions(y;on) // y-Skalierung umkehren
  AxisOptions(all;;on) // x- und y-Achse verschieben
  AxisLabelText(x;" Intervalle ")
  AxisLabelText(y;" Diff. \n[m]")
  AxisLabelBackground(all;255 200 200)
  AxisLabelOptions(x;topCenter)
  AxisLabelOptions(y;bottomRight)
  GridLocation(all;none)
CloseDrawing()

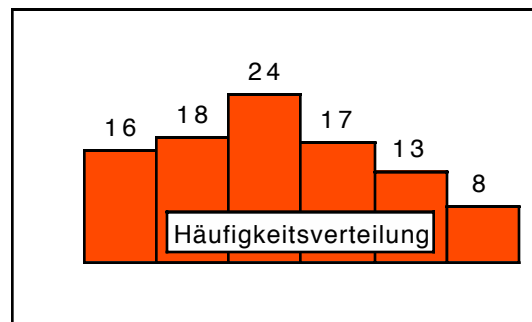
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 24 17 13 8)
  BarChart(label;0)
  AxisMajorTicks(x;0)           // keine x-Markierungen
  AxisMajorTickLabelTexts(x;"") // keine x-Skalenbesch.
  AxisOptions(y;none)
  AxisLabelText(x;"Häufigkeitsverteilung")
  AxisLabelBackground(x)
  AxisLabelOptions(x;topCenter)
  AxisOptions(x;front)
  GridLocation(all;none) // kein Raster
CloseDrawing()

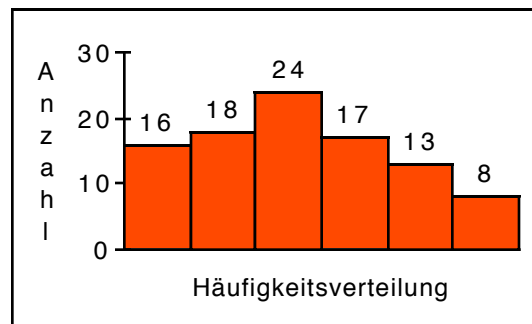
```



Mit den Argumenten *hVersatz* und *vVersatz* kann die Lage der Beschriftung feinabgestimmt werden. Positive Versatzwerte schieben die Beschriftung nach rechts unten, negative Werte nach links oben.

Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(16 18 24 17 13 8)
  BarChart(label;0)
  AxisLabelText(x;"Häufigkeitsverteilung")
  AxisLabelText(y;"A\nn\nz\na\nh\nl")
  AxisMajorTicks(x;0)           // keine x-Markierungen
  AxisMajorTickLabelTexts(x;"") // keine x-Skalenbesch.
  AxisLabelOptions(x;;;-10)     // 10 Pixel nach oben
  AxisLabelOptions(y;;;-6)      // 6 Pixel nach links
  GridLocation(all;none)        // kein Raster
CloseDrawing()
```

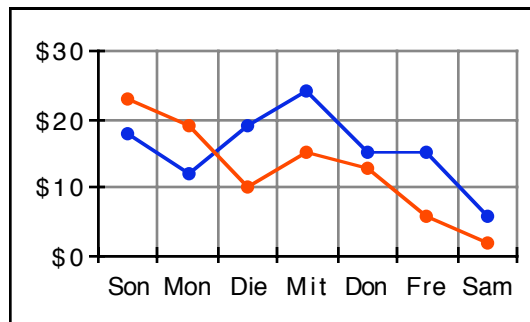


### **AxisMajorTickLabelTexts(achsenindex;text1;text2...)**

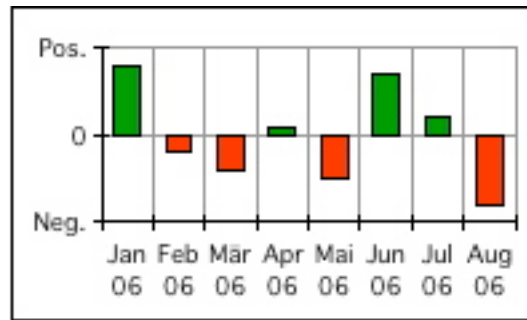
Die Beschriftung der Skalenmarkierungen besteht standardmäßig aus den jeweiligen Skalenwert. Durch die Funktion `AxisMajorTickLabelTexts()` kann der Beschriftungstext der Grobmarkierungen kontrolliert werden — so ist es möglich, jeder Skalenmarkierung einen individuellen Text zuzuordnen. Die Texte werden periodisch wiederholt, falls die Anzahl der Grobmarkierungen größer ist als die Anzahl der übergebenen Texte. Enthält der Text eine Formatanweisung, wie zum Beispiel "`|f1|`", so wird der Skalenwert entsprechend formatiert ausgegeben. Die Handhabung von Texten in Verbindung mit Formatanweisungen wird im Abschnitt *Stile*, Funktion `LabelTexts()` erläutert. Mehrzeilige Texte sind durch Einfügen eines Zeilenumbruchs "`\n`" möglich.

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 19 10 15 13 6 2;
            18 12 19 24 15 15 6)
  LineChart(symbol;on)
  SymbolStyle(1;bullet;4)
  SymbolStyle(2;bullet;4)
  AxisMajorTickLabelTexts(y;"$|u|")
  AxisMajorTickLabelTexts(x;"Son";"Mon";"Die";"Mit";
                           "Don";"Fre";"Sam")
CloseDrawing()
```



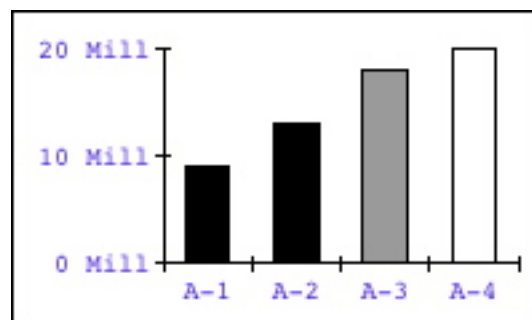
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(8 -2 -4 1 -5 7 2 -8)
  BarChart()
  BarChartOptions(;;on)
  Scaling(y;linear;-10;10;2)
  FillStyle(1;green)
  FillStyle(2;red)
  AxisMajorTickLabelTexts(y;"Neg."; "0"; "Pos.")
  AxisMajorTickLabelTexts(x;"Jan\n06"; "Feb\n06";
                           "Mär\n06"; "Apr\n06";
                           "Mai\n06"; "Jun\n06";
                           "Jul\n06"; "Aug\n06")
CloseDrawing()
```



**AxisMajorTickLabelStyle(achsenindex;schrift;größe;stil;  
farbe;ausrichtung;neigungswinkel;  
maxBreite;maxHöhe;auslassungsPosition)**

Mit der Funktion `AxisMajorTickLabelStyle()` kann der Schriftstil der Skalenbeschriftung variiert werden. Die Attribute *neigungswinkel*, *maxBreite*, *maxHöhe* und *auslassungsPosition* werden im Abschnitt *Stile*, Funktion `LabelStyle()` erläutert. Beispiel:

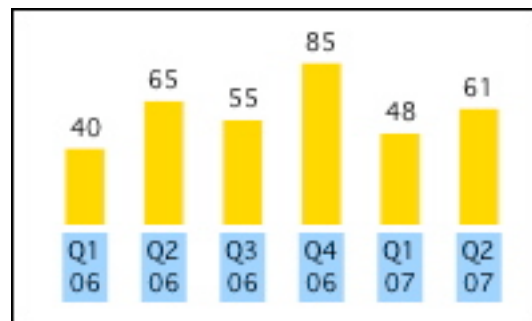
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 9000000 0 0 0;
           0 13000000 0 0;
           0 0 18000000 0;
           0 0 0 20000000)
BarChart(stacked)
FillStyle(all;black)
FillStyle(3;gray)
FillStyle(4;;transparent)
AxisMajorTickLabelTexts(x;"A-1";"A-2";
                        "A-3";"A-4")
AxisMajorTickLabelTexts(y;"|-6u| Mill")
AxisMajorTickLabelStyle(all;"Courier";10;plain;blue)
GridLocation(all;none) // kein Raster
CloseDrawing()
```



**AxisMajorTickLabelBackground(achsenindex;füllfarbe;  
füllmuster;rahmenbreite;rahmenfarbe;  
rahmenmuster;schattenabstand;  
schattenfarbe;schattenmuster)**

Durch die Funktion AxisMajorTickLabelBackground() kann der Hintergrund der Skalenbeschriftung gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren. Beispiele:

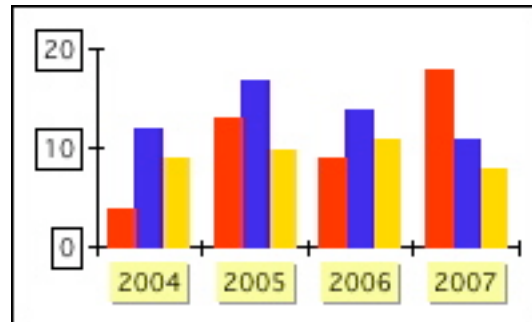
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(40 65 55 85 48 61)
  BarChart(label)
  FillStyle(1;darkYellow)
  BorderStyle(1;none)
  LabelOptions(1;out) // alle Beschriftungen außer-
                      // halb der Balken platzieren.
  AxisLine(all;0) // keine Achsenlinien
  AxisMajorTicks(all;0) // kein Skalenmarkierungen
  AxisMajorTickLabelTexts(y;"") // keine y-Beschriftung
  AxisMajorTickLabelTexts(x;"Q1\n06";"Q2\n06";
                          "Q3\n06";"Q4\n06";
                          "Q1\n07";"Q2\n07")
  AxisMajorTickLabelBackground(x;lightBlue;;0)
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 4 13 9 18;
           12 17 14 11;
           9 10 11 8)
BarChart()
FillStyle(3;darkYellow)
BorderStyle(all;none)
AxisMajorTickLabelTexts(x;"2004";"2005";
                        "2006";"2007")
AxisMajorTickLabelBackground(x;lightYellow;;0;;;1)
AxisMajorTickLabelBackground(y)
GridLocation(all;none)
CloseDrawing()

```



**AxisMajorTickLabelOptions(achsenindex;platzierung;  
hVersatz;vVersatz;beschriftungJeMarker;  
beschriftungAbMarker)**

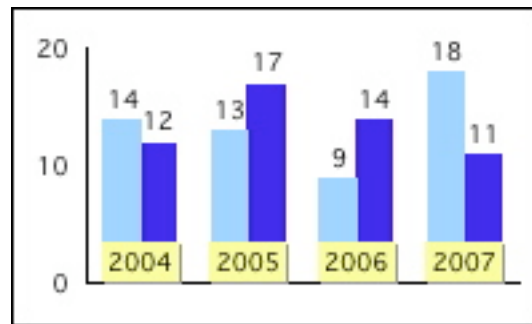
Die Funktion `AxisMajorTickLabelOptions()` dient zur Platzierung und Feinabstimmung der Skalenbeschriftung. Das Argument *platzierung* erlaubt die Positionierung der Beschriftung entweder innerhalb des Diagramms (*platzierung=in*) oder außerhalb des Diagramms (*platzierung=out*) oder auch direkt auf der Achsenlinie (*platzierung=center*). Mit den Argumenten *hVersatz* und *vVersatz* kann die Lage der Beschriftung feinabgestimmt werden. Positive Versatzwerte schieben die Beschriftung nach rechts unten, negative Werte nach links oben.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(14 13 9 18; 12 17 14 11)
  BarChart(label)
  FillStyle(1;lightBlue)
  BorderStyle(all;none)
  LabelOptions(all;out) // alle Beschriftungen außer-
                        // halb der Balken platzieren.
  AxisMajorTicks(all;0) // keine Skalenmarkierungen
  AxisMajorTickLabelTexts(x;"2004";"2005";
                        "2006";"2007")
  AxisMajorTickLabelBackground(x;lightYellow;;0;;1)
  AxisMajorTickLabelOptions(x;in;;2) //2 Pxl nach unten
  AxisMajorTickLabelOptions(y;;-3) //3 Pxl nach links
  AxisOptions(x;front) // Achsenbeschriftung vor Balken
  GridLocation(all;none)
CloseDrawing()

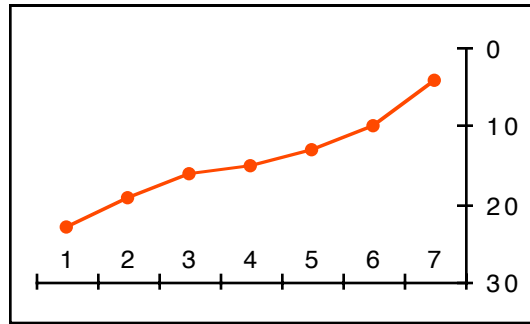
```



```

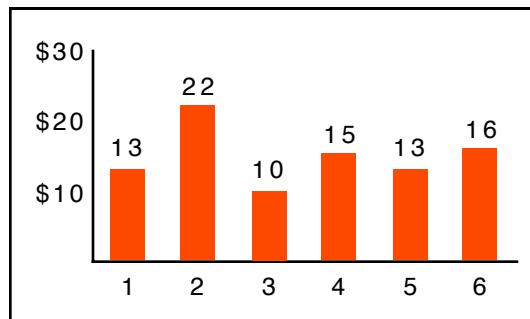
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(23 19 16 15 13 10 4)
  LineChart(symbol;on)
  SymbolStyle(1;bullet;4)
  ScalingOptions(y;on) // y-Skalierung umkehren
  AxisOptions(y;on) // y-Achse verschieben
  AxisMajorTickLabelOptions(x;in;;3) //3 Pxl nach unten
  AxisMajorTickLabelOptions(y;;3) //3 Pxl nach rechts
  GridLocation(all;none)
CloseDrawing()

```



Durch die Argumente *beschriftungJeMarker* und *beschriftungAbMarker* können Skalenbeschriftungen ausgeblendet werden. So bewirken zum Beispiel die Argumente *beschriftungJeMarker=2* und *beschriftungAbMarker=3*, dass nur die 3. Markierung, 5. Markierung, 7. Markierung usw. dargestellt werden. Beispiele:

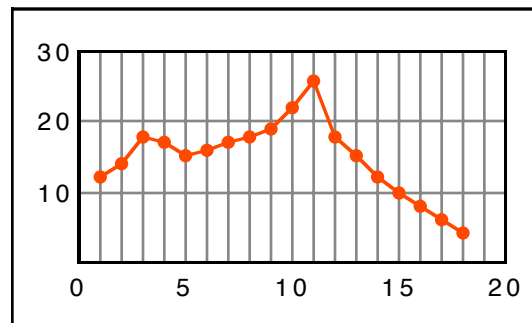
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(13 22 10 15 13 16)
  BarChart(label)
  BorderStyle(all;none)
  AxisMajorTicks(all;0) // keine Skalenmarkierungen
  AxisMajorTickLabelTexts(y;"$|u|")
  AxisMajorTickLabelOptions(y;;;1;2) //"$0" ausblenden
  GridLocation(all;none)
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(12 14 18 17 15 16 17 18 19
          22 26 18 15 12 10 8 6 4)
LineChart(symbol)
SymbolStyle(all;bullet;4)
Scaling(x;linear;0;20;20)
AxisMajorTicks(all;0)
AxisMajorTickLabelOptions(x;;;5) // jeder 5. Wert.
AxisMajorTickLabelOptions(y;;;1;2) // "0" ausblenden
GridFrame()
CloseDrawing()

```



```

AxisMinorTickLabelTexts(achsenindex;text1;text2...)
AxisMinorTickLabelStyle(achsenindex;schrift;größe;stil;
                        farbe;ausrichtung;neigungswinkel;
                        maxBreite;maxHöhe;auslassungsPosition)
AxisMinorTickLabelBackground(achsenindex;füllfarbe;
                             füllmuster;rahmenbreite;rahmenfarbe;
                             rahmenmuster;schattenabstand;
                             schattenfarbe;schattenmuster)
AxisMinorTickLabelOptions(achsenindex;platzierung;
                          hVersatz;vVersatz;beschriftungJeMarker;
                          beschriftungAbMarker;
                          beschriftungsmusterWiederholen)

```

Die Funktionen zur Beschriftung der Feinmarkierungen sind ident aufgebaut wie die Funktionen zur Beschriftung der Grobmarkierungen.

Beispiele:

```

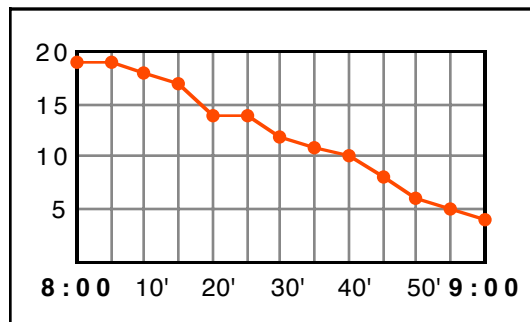
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 1  2  3  4  5  6  7  8  9 10 11 12 13;
            19 19 18 17 14 14 12 11 10  8  6  5  4)

  LineChart2D(symbol)
  Scaling(x;linear;1;13;1;12)
  SymbolStyle(1;bullet;4)

  // Achsen
  AxisMajorTicks(all;0) // keine Grobskalenmarkierungen
  AxisMinorTicks(all;0) // keine Feinskalenmarkierungen
  AxisMajorTickLabelTexts(x;"8:00";"9:00")
  AxisMajorTickLabelStyle(x;;;bold)
  AxisMinorTickLabelTexts(x;" 10'";" 20'";" 30'";
                          " 40'";" 50'")
  AxisMinorTickLabelOptions(x;;;2;2) // jeder 2. Wert.
  AxisMajorTickLabelOptions(y;;;1;2) // "0" ausblenden

  // Raster
  MajorGridLinePatterns(all;all;black)
  MajorGridLineColors(all;all;gray)
  MinorGridLinePatterns(all;all;black)
  MinorGridLineColors(all;all;gray)
  GridFrame()
CloseDrawing()

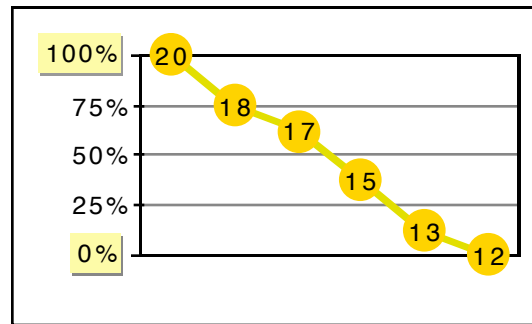
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(20 18 17 15 13 12)
  LineChart(symbol+label;on)
  Scaling(y;percent;0;100;1;4)
  LineStyle(1;poly;2;220 220 0)
  SymbolStyle(1;bullet;15;;darkYellow)
  LabelOptions(all;centerCenter)
  AxisOptions(x;none) // keine x-Achse
  AxisMajorTickLabelBackground(y;lightYellow;;0;;1)
  AxisMinorTickLabelTexts(y;"|u|%" )
  MajorGridLineWidths(y;x;0) // kein vertikales Raster
  GridFrame()
CloseDrawing()

```



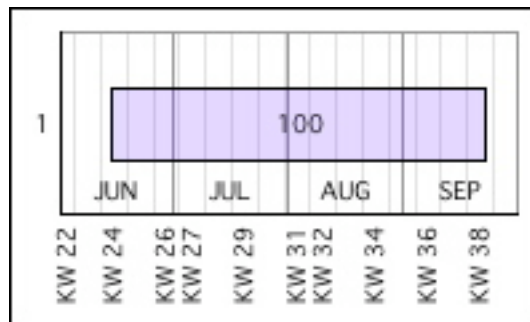
Die Funktion `AxisMinorTickLabelOptions()` besitzt im Gegensatz zur Funktion `AxisMajorTickLabelOptions()` noch als weiteres Argument *beschriftungsmusterWiederholen*. Standardmäßig wird das durch die Argumente *beschriftungJeMarker* und *beschriftungAbMarker* festgelegte Beschriftungsmuster der Feinmarkierungstexte bei jeder Grobmarkierung zurückgesetzt und beginnt wieder von vorne (*beschriftungsmusterWiederholen=on*). Wird *beschriftungsmusterWiederholen=off* gesetzt, dann wird das Beschriftungsmuster nicht wiederholt, sondern läuft ohne Rücksicht auf die Grobmarkierung über die gesamte Skalierung durch. Dies erweist sich z.B. bei einer Wochenskalieureung als vorteilhaft, da die Feinskalierung (Woche) nicht exakt mit der Grobskalierung (Jahr, Quartal oder Monat) zusammenpaßt.

Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  DateTimeOptions(ymd;2)
  ChartData(2006-6-15 2006-9-22)
  GanttChart(label;150)
  FillStyle(1;50 0 255 50)
  Scaling(x;linear;;;month;week)
  AxisMajorTicks(all;0)
  AxisMinorTicks(all;0)
  AxisMajorTickLabelTexts(x;"|MON|")
  AxisMajorTickLabelOptions(x;in)
  AxisMinorTickLabelTexts(x;"KW |WY|")
  AxisMinorTickLabelStyle(x;;;;;-90)
  AxisMinorTickLabelOptions(x;;;;;2;on) // aktiviert
CloseDrawing()

```

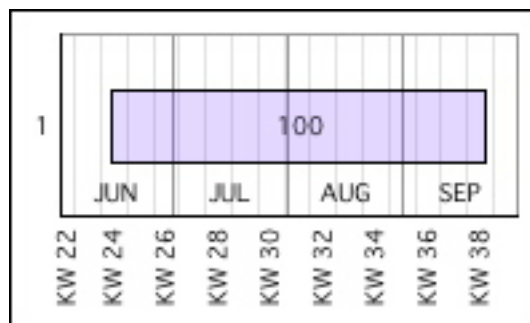


Das gleiche Beispiel mit *beschriftungsmusterWiederholen=off*:

```

...
AxisMinorTickLabelOptions(x;;;;;2;off) // deaktiviert
...

```



## Legende

Zur Gestaltung der Legende stehen vier Funktionen zur Verfügung. Diese ermöglichen:

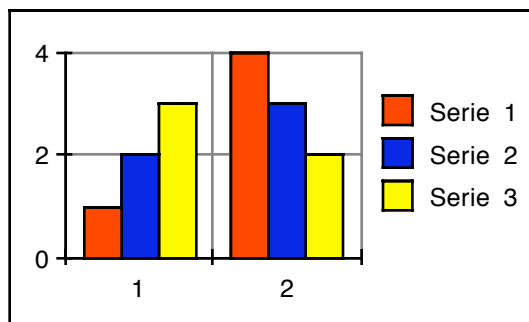
- das Definieren der Legendentexte
- die Festlegung des Schriftstils
- die Gestaltung des Hintergrundes
- das flexible Platzieren der Legende
- die Feinabstimmung mittels zahlreicher Optionen

Eine Legende wird nicht automatisch per default erstellt, sondern erst durch Aufrufen einer der vier nachstehend angeführten Funktionen.

### **LegendTexts(text1;text2...)**

Durch die Funktion `LegendTexts()` kann jeder Datenserie ein Namen zugeordnet werden. Dabei sind auch mehrzeilige Bezeichnungen durch Einfügen eines Zeilenumbruchs "`\n`" möglich. Falls keine oder zu wenige Legendentexte zur Verfügung gestellt werden, wird für die fehlenden Namen die Standardbezeichnung "Serie" plus einer laufenden Nummer verwendet. Beispiele:

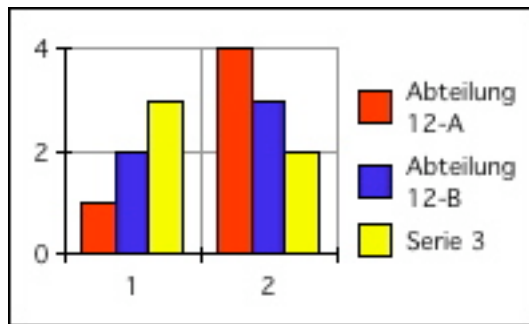
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 4; 2 3; 3 2)
  BarChart()
  LegendTexts()
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 4; 2 3; 3 2)
  BarChart()
  LegendTexts("Abteilung\n12-A";
              "Abteilung\n12-B")
CloseDrawing()

```



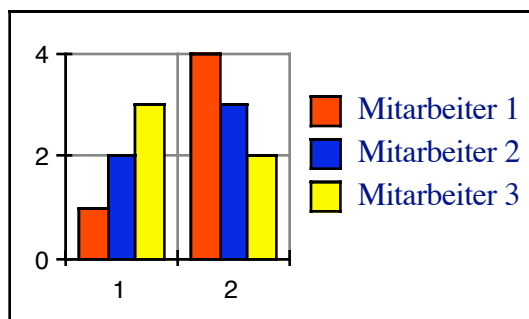
**LegendStyle(schrift;größe;stil;farbe;ausrichtung;  
neigungswinkel;maxBreite;maxHöhe;  
auslassungsPosition)**

Mit der Funktion `LegendStyle()` wird der Schriftstil der Legende festgelegt. Die Attribute *neigungswinkel*, *maxBreite*, *maxHöhe* und *auslassungsPosition* werden im Abschnitt *Stile*, Funktion `LabelStyle()` erläutert. Zum Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 4; 2 3; 3 2)
  BarChart()
  LegendTexts("Mitarbeiter 1";
              "Mitarbeiter 2";
              "Mitarbeiter 3")
  LegendStyle("Times";12;plain;darkBlue)
CloseDrawing()

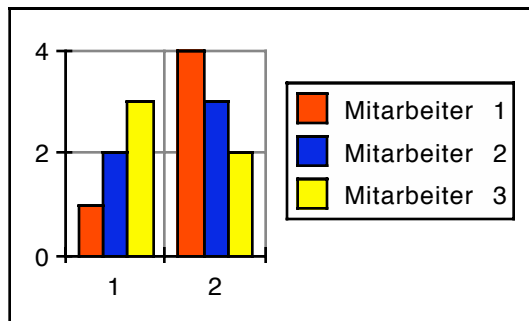
```



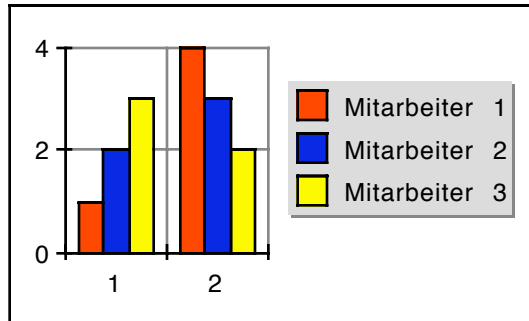
**LegendBackground(füllfarbe;füllmuster;rahmenbreite;  
rahmenfarbe;rahmenmuster;schattenabstand;  
schattenfarbe;schattenmuster)**

Durch die Funktion LegendBackground( ) kann der Legendenhintergrund gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 4; 2 3; 3 2)
  BarChart()
  LegendTexts("Mitarbeiter 1";
              "Mitarbeiter 2";
              "Mitarbeiter 3")
  LegendBackground()
CloseDrawing()
```



```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 4; 2 3; 3 2)
  BarChart()
  LegendTexts("Mitarbeiter 1";
              "Mitarbeiter 2";
              "Mitarbeiter 3")
  LegendBackground(lightGray;;0;;;1)
CloseDrawing()
```



**LegendOptions(platzierung;innerhalbDiagramm;hVersatz;  
vVersatz;anordnung;symboltyp;symbolbreite;  
symbolhöhe;symbolabstand;zeilenabstand;  
spaltenabstand;textPlatzierung)**

Die Funktion LegendOptions() stellt zahlreiche Möglichkeiten zur Platzierung und Feinabstimmung der Legende zur Verfügung. Standardmäßig wird die Legende an der rechten Seite des Diagramms platziert. Das erste Argument *platzierung* erlaubt die Auswahl unter neun vorgegebenen Positionen:

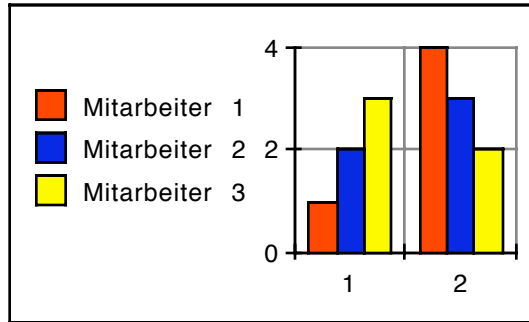
Konstante	Wert	Anmerkung
topLeft	1	oben links
topCenter	2	oben mitte
topRight	3	oben rechts
centerLeft	4	mitte links
centerCenter	5	mitte mitte
centerRight	6	mitte rechts (default)
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

Beispiel:

```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(1 4; 2 3; 3 2)
BarChart()
LegendTexts("Mitarbeiter 1";
            "Mitarbeiter 2";
            "Mitarbeiter 3")
LegendOptions(centerLeft)
CloseDrawing()

```



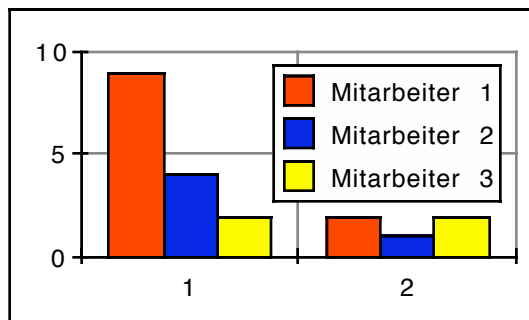
Zusätzlich besteht die Möglichkeit mittels *platzierung=0* die Legende relativ zum Koordinatenursprung zu platzieren. Der Koordinatenursprung befindet sich in der linken oberen Ecke der Zeichnung bzw. bei Vorhandensein eines Views, in der linken oberen Ecke des Views.

Das 2. Argument *innerhalbDiagramm* erlaubt die Positionierung der Legende innerhalb der Diagrammfläche. Optional kann noch durch *hVersatz* und *vVersatz* die Lage der Legende feinabgestimmt werden. Positive Versatzwerte schieben die Legende nach rechts unten, negative Werte nach links oben. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(9 2; 4 1; 2 2)
  BarChart()
  LegendTexts("Mitarbeiter 1";
              "Mitarbeiter 2";
              "Mitarbeiter 3")
  LegendBackground()
  LegendOptions(topRight;on;3;-3)
CloseDrawing()

```



Die Anordnung der Legendentexte kann mit dem Argument *anordnung* sehr vielseitig gestaltet werden. Das Argument *anordnung* kann aus bis zu 6 Elementen bestehen, die durch Leerzeichen, Tabs oder Zeilenumbrüche zu trennen sind.

Element	Funktion	Bereich	Defaultwert
[1]	Zeilenanzahl	-1..1000	10
[2]	Spaltenanzahl	-1..1000	-1
[3]	Anordnung Spalte für Spalte	0..1	0
[4]	Umgekehrt anordnen	0..1	0
[5]	Idente Spaltenbreiten	0..1	0
[6]	Idente Zeilenhöhen	0..1	0

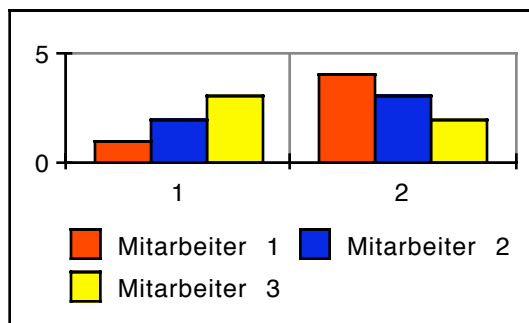
(-1...variabel Zeilen- bzw. Spaltenanzahl)

Beispiele:

```
LegendOptions(;;;2)           // 2 Zeilen (variable Spaltenanz.)
LegendOptions(;;;2 -1)        // 2 Zeilen (wie oben)
LegendOptions(;;;-1 2)        // 2 Spalten (variable Zeilenanz.)
LegendOptions(;;;3 -1 1)      // 3 Zeilen (Spalte für Spalte)
LegendOptions(;;;-1 1 0 1)    // 1 Spalte, von unten nach oben
```

Beispiele:

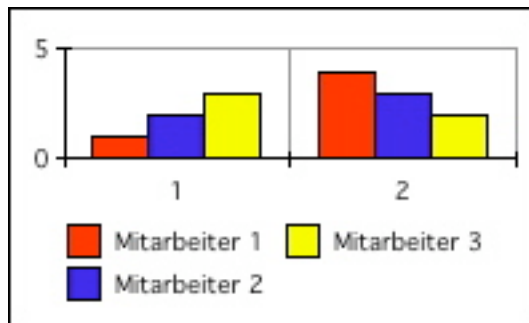
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(1 4; 2 3; 3 2)
  BarChart()
  LegendTexts("Mitarbeiter 1";
              "Mitarbeiter 2";
              "Mitarbeiter 3")
  // zeilenweise
  LegendOptions(bottomCenter;;;2)
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(1 4; 2 3; 3 2)
BarChart()
LegendTexts("Mitarbeiter 1";
            "Mitarbeiter 2";
            "Mitarbeiter 3")
// spaltenweise
LegendOptions(bottomCenter;;;2 -1 1)
CloseDrawing()

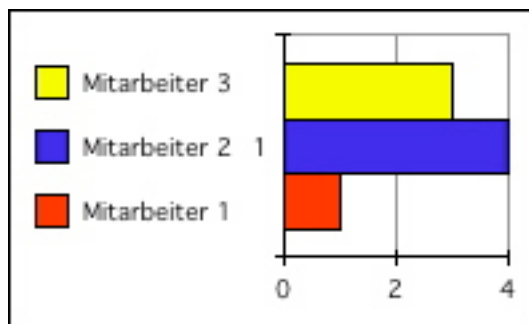
```



```

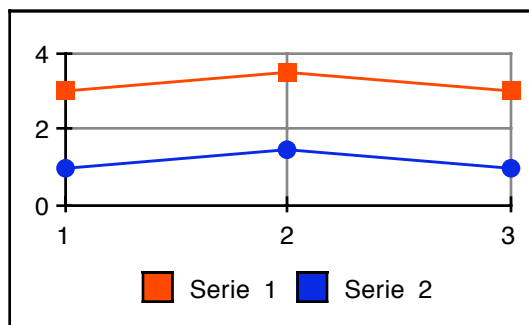
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(1; 4; 3)
BarChart(horizontal)
LegendTexts("Mitarbeiter 1";
            "Mitarbeiter 2";
            "Mitarbeiter 3")
// spaltenweise von unten nach oben
LegendOptions(centerLeft;;;-8;
            -1 1 1 1;
            ;;;;12)
CloseDrawing()

```

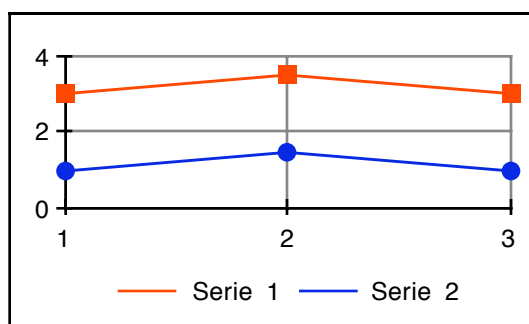


Das Argument *symboltyp* dient zum Festlegen des Legendensymbols. Als Legendensymbol kann ein Rechteck, eine Linie oder ein Diagrammsymbol verwendet werden — auch Kombinationen sind möglich. Wird kein Symboltyp vorgegeben, wird automatisch ein passender Typ gewählt, zum Beispiel bei Balken- und Kuchendiagrammen ein Quadrat, bei Liniendiagrammen mit Symbolen, eine Linie mit einem Symbol überlagert. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 3.5 3; 1 1.5 1)
  LineChart(symbol)
  SymbolStyle(1;square;6)
  SymbolStyle(2;bullet;6)
  LegendOptions(bottomCenter;;;1;rect)
CloseDrawing()
```



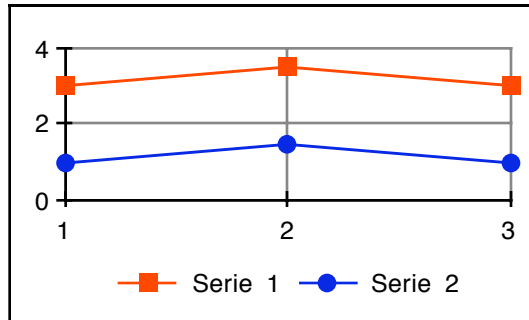
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 3.5 3; 1 1.5 1)
  LineChart(symbol)
  SymbolStyle(1;square;6)
  SymbolStyle(2;bullet;6)
  LegendOptions(bottomCenter;;;1;line)
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(3 3.5 3; 1 1.5 1)
LineChart(symbol)
SymbolStyle(1;square;6)
SymbolStyle(2;bullet;6)
LegendOptions(bottomCenter;;;1;line+symbol)
CloseDrawing()

```

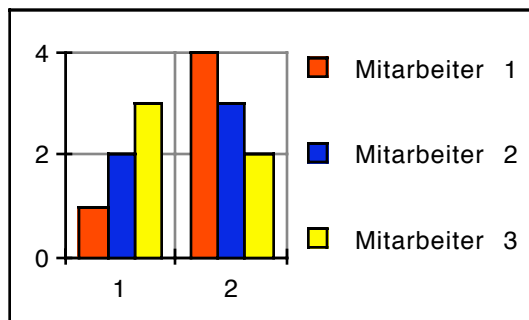


Die Argumente *symbolhöhe*, *symbolbreite*, *symbolabstand*, *zeilenabstand* und *spaltenabstand* kontrollieren die Größe des Legendensymbols, den Abstand zwischen Legendensymbol und Legendentext, sowie den Abstand zwischen den Legendentexten. Beispiel:

```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(1 4; 2 3; 3 2)
BarChart()
LegendTexts("Mitarbeiter 1";
            "Mitarbeiter 2";
            "Mitarbeiter 3")
LegendOptions(;;;;;8;8;10;20)
CloseDrawing()

```



Mit dem Argument *textPlatzierung* kann die Anordnung des Legendentextes relativ zum Legendensymbol kontrolliert werden. Neun Platzierungen stehen dazu zur Verfügung. Standardmäßig werden die Legendentexte rechts vom Legendensymbol angeführt (*textPlatzierung=centerRight*).

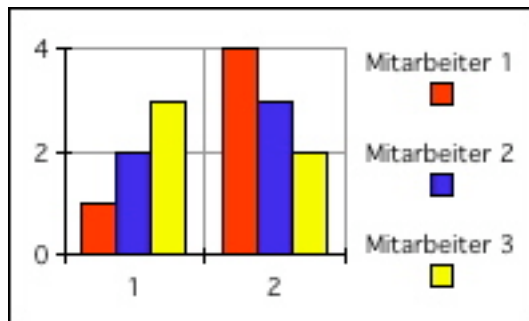
Konstante	Wert	Anmerkung
topLeft	1	oben links
topCenter	2	oben mitte
topRight	3	oben rechts
centerLeft	4	mitte links
centerCenter	5	mitte mitte
centerRight	6	mitte rechts (default)
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

Beispiele:

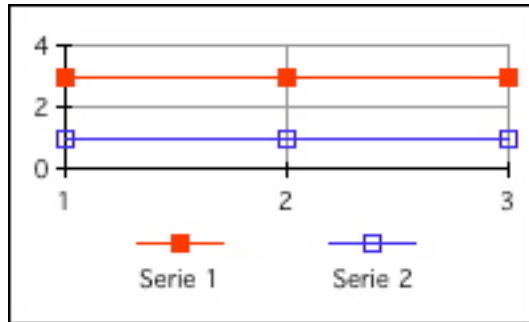
```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(1 4; 2 3; 3 2)
BarChart()
LegendTexts("Mitarbeiter 1";
            "Mitarbeiter 2";
            "Mitarbeiter 3")
LegendOptions(;;;;;
              8;8; // Symbolgröße
              2;  // Abstand zw. Symbol und Text
              12;; // Zeilenabstand
              topCenter)
CloseDrawing()

```



```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 3.0 3; 1 1.0 1)
  LineChart(symbol)
  SymbolStyle(1;square;6)
  SymbolStyle(2;hollowSquare;6)
  LegendOptions(bottomCenter;;;1;line+symbol)
  LegendOptions(bottomCenter;;;
    1;      // Zeilenzahl
    line+symbol;
    18;12; // Symbolgröße
    4;      // Abstand zw. Symbol und Text
    0;40;   // Spaltenabstand
    bottomCenter)
CloseDrawing()
```



# Titel

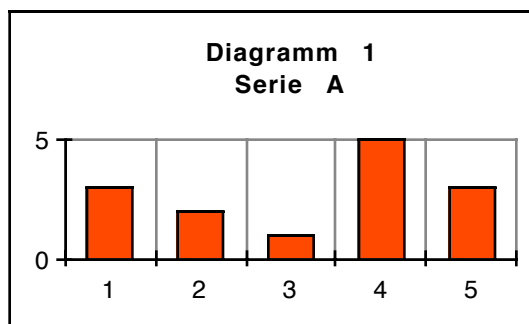
Zur Titelgestaltung stehen insgesamt fünf Funktionen zur Verfügung. Diese ermöglichen:

- das Definieren eines Titeltexs inkl. Untertitel
- getrennte Schriftstile für Titel und Untertitel
- die Gestaltung des Hintergrundes
- ein flexibles Platzieren des Titels

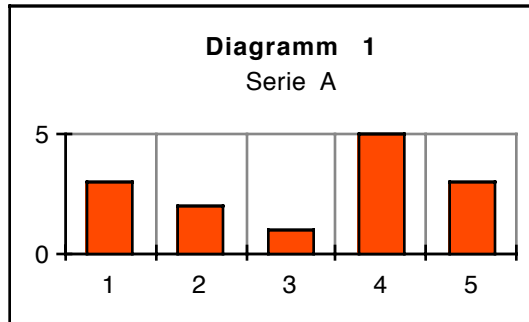
## **TitleText(titel;untertitel)**

Durch die Funktion `TitleText()` wird der Text des Titels festgelegt; optional besteht die Möglichkeit als 2. Argument einen Untertitel zu definieren. Durch Einfügen eines Zeilenumbruchs "`\n`" können auch mehrzeilige Texte ausgegeben werden. Beispiele:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(3 2 1 5 3)
BarChart()
TitleText("Diagramm 1\nSerie A")
CloseDrawing()
```



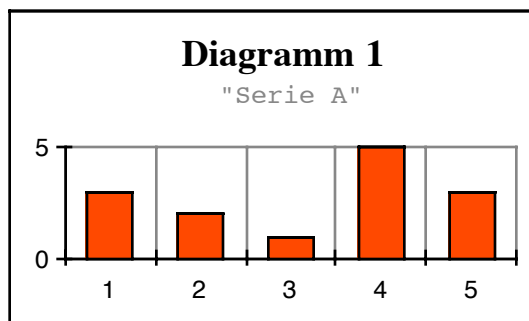
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(3 2 1 5 3)
BarChart()
TitleText("Diagramm 1";"Serie A")
CloseDrawing()
```



```
TitleStyle(schrift;größe;stil;farbe;ausrichtung;  
          neigungswinkel;maxBreite;maxHöhe;  
          auslassungsPosition)  
TitleSubStyle(schrift;größe;stil;farbe;ausrichtung;  
              neigungswinkel;maxBreite;maxHöhe;  
              auslassungsPosition)
```

Durch die Funktion `TitleStyle()` bzw. `TitleSubStyle()` wird der Schriftstil des Titels bzw. Untertitels festgelegt. Die Attribute *neigungswinkel*, *maxBreite*, *maxHöhe* und *auslassungsPosition* werden im Abschnitt *Stile*, Funktion `LabelStyle()` erläutert. Beispiel:

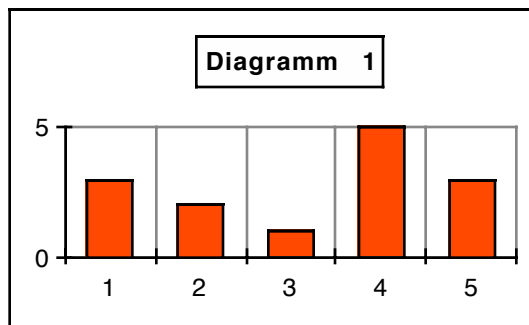
```
OpenDrawing(200;120)  
  AddFrame(0;0;200;120)  
  ChartData(3 2 1 5 3)  
  BarChart()  
  TitleText("Diagramm 1";"\ "Serie A\ "  
  TitleStyle("Times";14;bold)  
  TitleSubStyle("Courier";10;plain;gray)  
CloseDrawing()
```



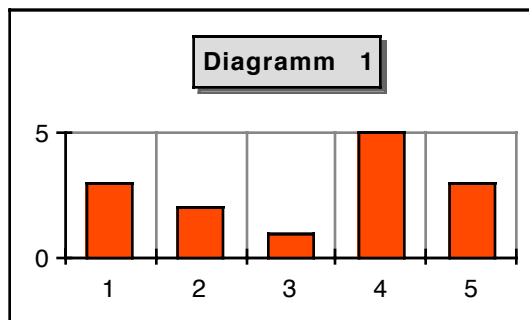
**TitleBackground(füllfarbe;füllmuster;rahmenbreite;  
rahmenfarbe;rahmenmuster;schattenabstand;  
schattenfarbe;schattenmuster)**

Durch die Funktion TitleBackground() kann der Titelhintergrund gestaltet werden. Dabei ist es möglich, Farbe, Muster, Berandung und Schatten zu variieren. Beispiele:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(3 2 1 5 3)
BarChart()
TitleText("Diagramm 1")
// weißer Hintergrund mit einem
// 1 Pixel breiten, schwarzen Rand.
TitleBackground()
CloseDrawing()
```



```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(3 2 1 5 3)
BarChart()
TitleText("Diagramm 1")
// hellgrauer Hintergrund mit einem 1 Pixel
// breiten, schwarzen Rand und einem 2 Pixel
// breiten, dunkelgrauen Schatten
TitleBackground(lightGray;;1;;2;darkGray)
CloseDrawing()
```



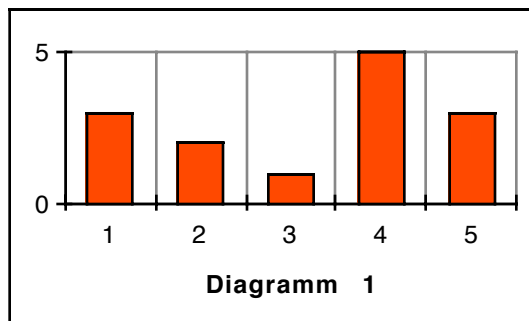
**TitleOptions(platzierung;innerhalbDiagramm;hVersatz;  
vVersatz;vUntertitelversatz;titelausrichtung)**

Die Funktion TitleOptions() dient zur Platzierung und Ausrichtung des Titels. Standardmäßig wird der Titel in der Mitte oberhalb des Diagramms platziert. Das 1. Argument *platzierung* erlaubt die Auswahl unter neun vorgegebenen Positionen:

Konstante	Wert	Anmerkung
topLeft	1	oben links
topCenter	2	oben mitte (default)
topRight	3	oben rechts
centerLeft	4	mitte links
centerCenter	5	mitte mitte
centerRight	6	mitte rechts
bottomLeft	7	unten links
bottomCenter	8	unten mitte
bottomRight	9	unten rechts

Beispiel:

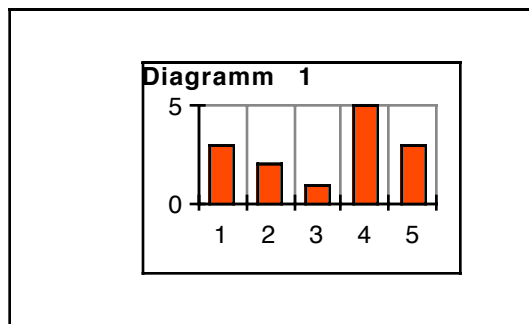
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(3 2 1 5 3)
BarChart()
TitleText("Diagramm 1")
TitleOptions(bottomCenter)
CloseDrawing()
```



Zusätzlich besteht die Möglichkeit mittels *platzierung=0* den Titel relativ zum Koordinatenursprung zu platzieren. Der Koordinatenursprung befindet sich in der linken oberen Ecke der Zeichnung oder, falls das Diagramm innerhalb eines Views platziert wird, in der linken oberen Ecke des Views.

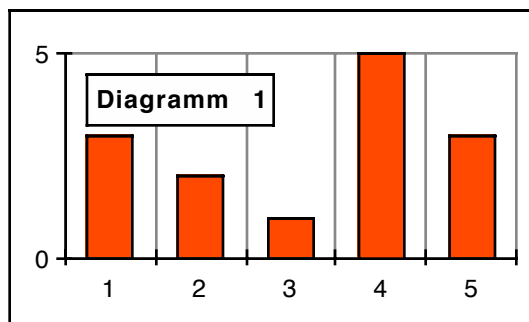
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  OpenView(50;20;120;80)
    AddFrame(0;0;120;80)
    ChartData(3 2 1 5 3)
    BarChart()
    TitleText("Diagramm 1")
    TitleOptions(0)
  CloseView()
CloseDrawing()
```



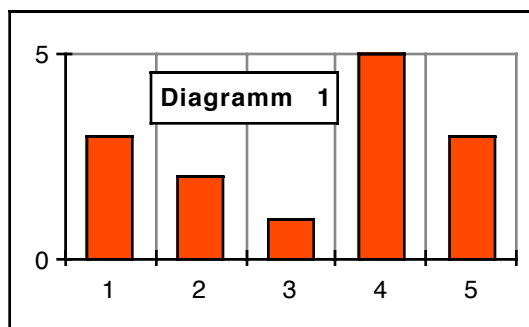
Das 2. Argument *innerhalbDiagramm* erlaubt die Positionierung des Titels innerhalb der Diagrammfläche. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 2 1 5 3)
  BarChart()
  TitleText("Diagramm 1")
  TitleBackground()
  TitleOptions(topLeft;on)
CloseDrawing()
```



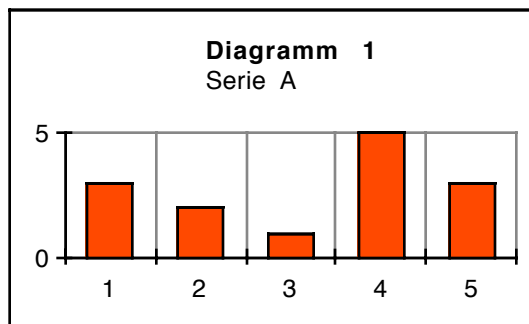
Optional kann durch *hVersatz* und *vVersatz* die Lage des Titels feinabgestimmt werden. Positive Versatzwerte schieben den Titel nach rechts unten, negative Werte nach links oben. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 2 1 5 3)
  BarChart()
  TitleText("Diagramm 1")
  TitleBackground()
  TitleOptions(topLeft;on;24;-1)
CloseDrawing()
```



Der Abstand und die Ausrichtung zwischen Titel und Untertitel kann durch *vUntertitelversatz* und *titelausrichtung* kontrolliert werden. Ein positiver Versatzwert vergrößert den Abstand zwischen Titel und Untertitel, ein negativer Wert verkleinert den Abstand. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 2 1 5 3)
  BarChart()
  TitleText("Diagramm 1";"Serie A")
  TitleOptions(topCenter;off;0;0;-2;left)
CloseDrawing()
```



## Hilfslinien

Punkt-, Linien- und Blasendiagramme können optional durch Hilfslinien ergänzt werden. Dazu stehen insgesamt vier Funktionen zur Verfügung. Diese dienen:

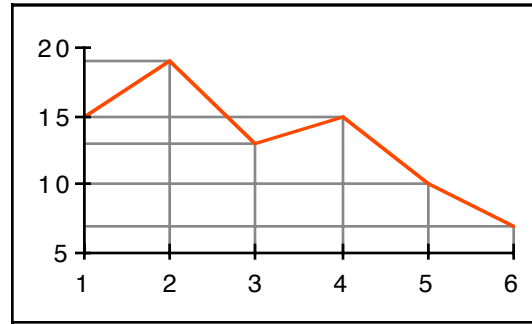
- zur Gestaltung der Hilfslinien
- zum Festlegen eines Referenzpunktes
- zum Festlegen einer Referenzlinie
- zum Festlegen von Referenzserien

Bei allen vier Funktionen wird als erstes Argument der Serienindex festgelegt. Wird kein Serienindex vorgegeben oder *serienindex=all* gesetzt, bezieht sich die jeweilige Funktion auf alle Datenserien.

**DropLineStyle(*serienindex*;*achsenindex*;*strichstärke*;  
*farbe*;*muster*)**

Das 2. Argument *achsenindex* legt die Richtung der Hilfslinien fest. Bei *achsenindex=x* werden die Hilfslinien auf die x-Achse bezogen, bei *achsenindex=y* auf die y-Achse. Wird kein Achsenindex vorgegeben bzw. ist *achsenindex=all*, werden die Hilfslinien sowohl in Richtung der x-Achse als auch in Richtung der y-Achse gezeichnet. Die Strichstärke, die Farbe und das Muster der Hilfslinien können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Wenn nicht anders vorgegeben, werden Hilfslinien als graue, 1 Pixel breite Linien dargestellt. Beispiele:

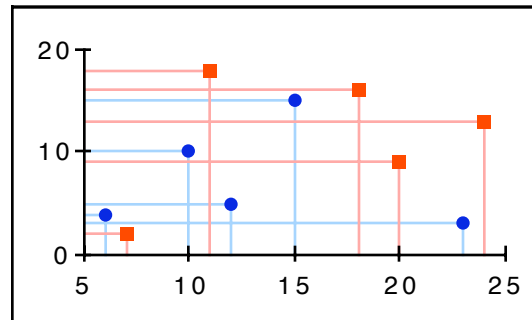
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(15 19 13 15 10 7)
  LineChart()
  DropLineStyle()
  AxisOptions(all;front) // Achsen vor Hilfslinien
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 7 11 18 20 24; // x-Werte 1.Serie
           2 18 16 9 13; // y-Werte 1.Serie
           23 10 15 12 6; // x-Werte 2.Serie
           3 10 15 5 4) // y-Werte 2.Serie
ScatterChart2D()
SymbolStyle(1;square;4;;red)
SymbolStyle(2;bulet;4;;blue)
DropLineStyle(1;all;1;lightRed)
DropLineStyle(2;all;1;lightBlue)
AxisOptions(all;front) // Achsen vor Hilfslinien
GridLocation(all;none) // kein Raster
CloseDrawing()

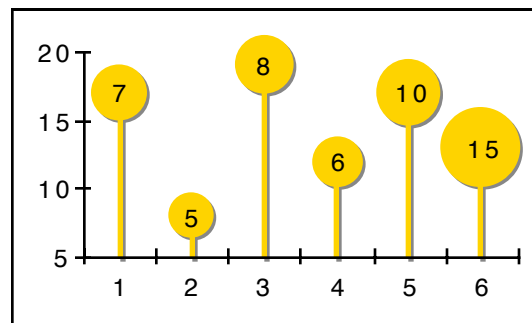
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(17 8 19 12 17 13; // y-Positionen
          7 5 8 6 10 15) // Durchmesser
BubbleChart(label+shadow;on)
FillStyle(1;darkYellow)
BorderStyle(1;none)
ShadowStyle(1;1;gray)
DropLineStyle(1;x;2;darkYellow)
AxisOptions(all;front) // Achsen vor Hilfslinien
GridLocation(all;none) // kein Raster
CloseDrawing()

```



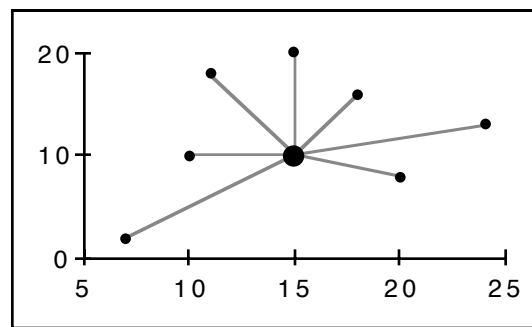
**DropLineReferencePoint(*serienindex*; *xPunkt*; *yPunkt*;  
*symboltyp*; *symbolgröße*; *strichstärke*;  
*symbolfarbe*; *symbolmuster*)**

Die Funktion `DropLineReferencePoint()` dient zur Festlegung eines Referenzpunktes, von welchem alle Hilfslinien strahlenförmig ausgehen. In diesem Fall werden als Argument *achsenindex* in der Funktion `DropLineStyle()` alle Indizes größer 1 ignoriert, d.h. es wird nur *achsenindex=x* oder *achsenindex=1* akzeptiert. Die Argumente *xPunkt* und *yPunkt* legen die Position des Referenzpunktes fest. Die Gestaltung des Referenzpunktes erfolgt durch die Argumente *symboltyp*, *symbolgröße*, *strichstärke*, *symbolfarbe* und *symbolmuster*. Insgesamt stehen zur Zeit 18 Symbole zur Verfügung. Ein Überblick über die vordefinierten Symbole ist in *xmReferenz* zu finden. Standardmäßig wird der Referenzpunkt durch einen schwarzen, kreisförmigen Punkt (*symboltyp=bullet*) mit 9 Pixel Durchmesser dargestellt.

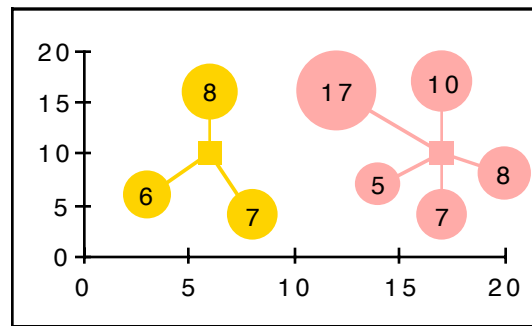
Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 7 11 18 20 24 10 15; // x-Werte
            2 18 16 8 13 10 20) // y-Werte
  ScatterChart2D()
  SymbolStyle(1;bullet;3;;black)

  DropLineReferencePoint(1;15;10)
  GridLocation(xy;none) // kein Raster
CloseDrawing()
```



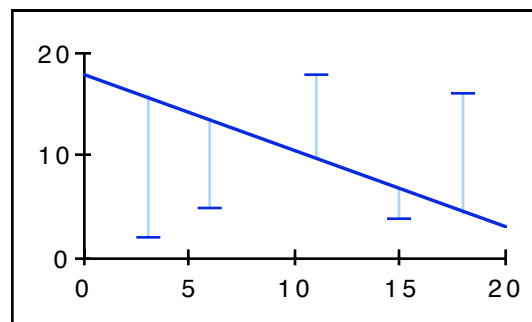
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(17 14 20 12 17; // x-Positionen 1.Serie
            4 7 8 16 17; // y-Positionen 1.Serie
            7 5 8 17 10; // Durchmesser 1.Serie
            8 3 6; // x-Positionen 2.Serie
            4 6 16; // y-Positionen 2.Serie
            7 6 8) // Durchmesser 2.Serie
  BubbleChart2D(label)
  BorderStyle(all;none)
  FillStyle(1;lightRed)
  FillStyle(2;darkYellow)
  DropLineStyle(1;;1;lightRed)
  DropLineStyle(2;;1;darkYellow)
  DropLineReferencePoint(1;17;10;square;8;;lightRed)
  DropLineReferencePoint(2;6;10;square;8;;darkYellow)
  GridLocation(xy;none) // kein Raster
CloseDrawing()
```



**DropLineReferenceLine(*serienindex*; *xStart*; *yStart*;  
*xEnde*; *yEnde*; *strichstärke*; *farbe*;  
*muster*)**

Die Funktion `DropLineReferenceLine()` dient zur Festlegung einer Referenzlinie. Die Argumente *xStart*, *yStart* und *xEnde*, *yEnde* definieren die Referenzlinie. Die Gestaltung der Referenzlinie erfolgt durch die Argumente *strichstärke*, *farbe* und *muster*. Beispiele:

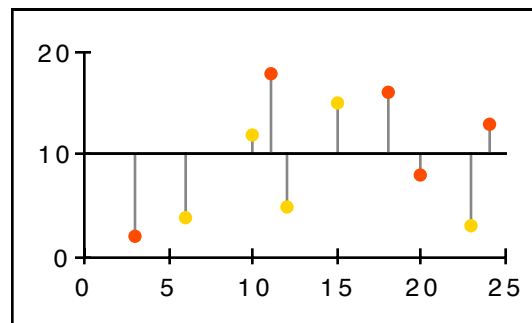
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(3 11 18 15 6; // x-Werte
          2 18 16 4 5) // y-Werte
ScatterChart2D()
SymbolStyle(1;hBar;8;;blue)
DropLineStyle(1;x;1;lightBlue)
DropLineReferenceLine(1;0;18;20;3;1;blue)
GridLocation(all;none) // kein Raster
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 3 11 18 20 24; // x-Werte 1.Serie
           2 18 16  8 13; // y-Werte 1.Serie
           23 10 15 12  6; // x-Werte 2.Serie
           3 12 15  5  4) // y-Werte 2.Serie
ScatterChart2D()
SymbolStyle(1;bulet;4;;red)
SymbolStyle(2;bulet;4;;darkYellow)
DropLineReferenceLine(all;0;10;25;10)
GridLocation(all;none) // kein Raster
CloseDrawing()

```



**DropLineReferenceSeries(*serienindex*;  
*referenzSerienindex1*;  
*referenzSerienindex2...*)**

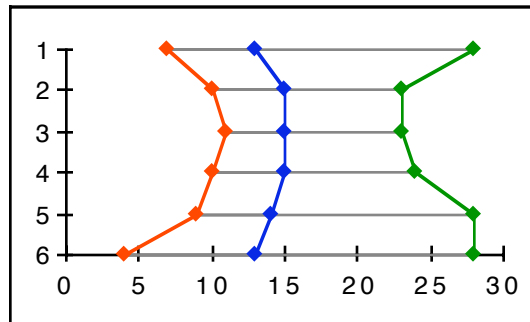
Datenpunkte unterschiedlicher Serien können unter Verwendung der Funktion `DropLineReferenceSeries()` durch Hilfslinien verbunden werden. Die "Bezugsreihe" wird durch das 1. Argument *serienindex* festgelegt.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 7 10 11 10 9 4; // 1.Serie
            13 15 15 15 14 13; // 2.Serie
            28 23 23 24 28 28) // 3.Serie
  LineChart(horizontal+symbol)
  LineStyle(all;poly;1)
  LineStyle(3;poly;1;green)
  SymbolStyle(all;diamond;4;1)
  SymbolStyle(3;diamond;4;1;green)
  DropLineReferenceSeries(all;1)
  ScalingOptions(y;on) // y-Skala von oben nach unten
  GridLocation(all;none) // kein Raster
CloseDrawing()

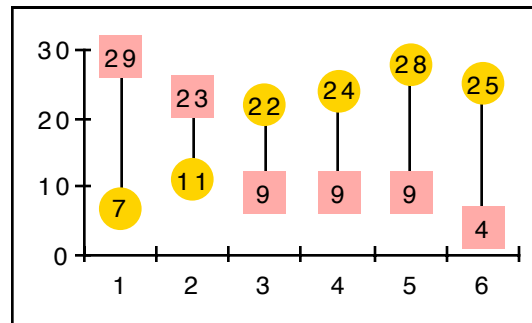
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 7 11 22 24 28 25; // 1.Serie
            29 23 9 9 9 4) // 2.Serie
  ScatterChart(label;on)
  SymbolStyle(1;bullet;15;;darkYellow)
  SymbolStyle(2;square;15;;lightRed)
  LabelOptions(all;centerCenter)
  DropLineStyle(all;x;1;black)
  DropLineReferenceSeries(1;2)
  GridLocation(all;none) // kein Raster
CloseDrawing()

```



## Gleitende Durchschnitte

Punkt- und Liniendiagramme können optional durch gleitende Durchschnitte ergänzt werden. Dazu stehen insgesamt drei Funktionen zur Verfügung. Diese ermöglichen:

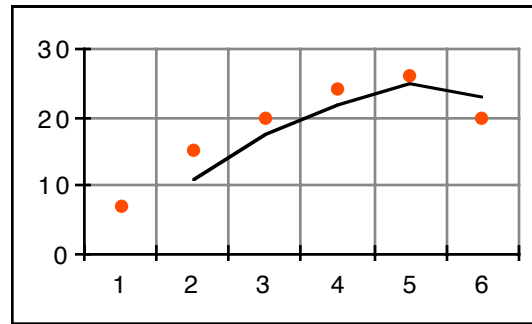
- die Festlegung unterschiedlicher Berechnungsmethoden
- die Gestaltung der Durchschnittslinien
- die Feinabstimmung mittels zahlreicher Optionen

Bei allen drei Funktionen werden als erstes Argument der Serienindex, als zweites Argument die Intervallanzahl und als drittes Argument die Berechnungsmethode festgelegt. Die Intervallanzahl bestimmt die Anzahl der zur Berechnung des Durchschnitts herangezogenen Datenpunkte. Soll zum Beispiel ein 50-Tage-Durchschnitt berechnet werden, so ist *intervallanzahl=50* zu setzen. Wird keine Intervallanzahl vorgegeben, so werden standardmäßig die Durchschnittswerte aus zwei Datenpunkten errechnet (*intervallanzahl=2*). Wird kein Serienindex vorgegeben oder *serienindex=all* gesetzt, bezieht sich die jeweilige Funktion auf alle vorhandenen Datenserien. Zur Zeit stehen drei Methoden zur Berechnung des gleitenden Durchschnitts zur Verfügung. Details dazu finden sich bei Funktion `MovingAverageOptions()`.

**`MovingAverage(serienindex;intervallanzahl;  
berechnungsmethode;wichtungsfaktoren)`**

Das erste Argument *serienindex* in der Funktion `MovingAverage()` legt fest, für welche Datenserie der gleitende Durchschnitt zu berechnen ist. Das zweite Argument *intervallanzahl*, definiert die Anzahl der Datenpunkte zur Berechnung der Durchschnittswerte. Beispiele:

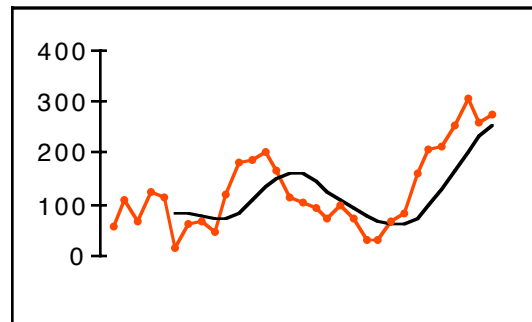
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(7 15 20 24 26 20)
  ScatterChart(;on)
  SymbolStyle(all;bullet;4;1;red)
  MovingAverage()
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 57 108 70 125 116 13 63 67 45 121 181
          189 201 166 115 105 91 75 99 72 33 29 69
          85 162 210 211 256 304 258 274)
LineChart(symbol)
SymbolStyle(1;bullet;2)
MovingAverage(1;6)      // 6 Pkte zur Mittelwertbildg.
AxisOptions(x;none)     // keine x-Achse
GridLocation(all;none)  // kein Raster
CloseDrawing()

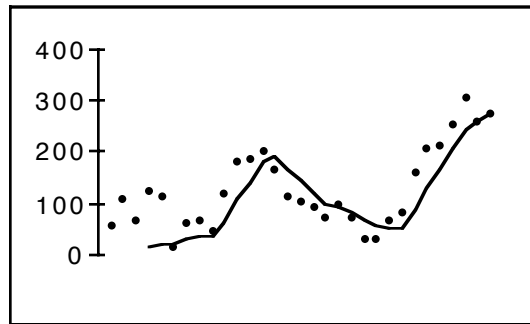
```



Optional kann als viertes Argument eine Liste von Wichtungsfaktoren angeführt werden. Dabei wird der 1. Datenpunkt mit dem 1. Wichtungsfaktor multipliziert, der 2. Datenpunkt mit dem 2. Wichtungsfaktor, usw. Ein Wichtungsfaktor kleiner 1 verringert den Einfluss eines Datenpunktes zur Mittelwertbildung, ein Wichtungsfaktor größer 1 vergrößert den Einfluss. Ist die Anzahl der Wichtungsfaktoren kleiner als die Anzahl der Datenpunkte, so werden die fehlenden Faktoren durch den neutralen Wert 1 ergänzt. Die Wichtungsfaktoren sind analog wie in der Funktion `ChartData()` getrennt durch Leerzeichen, Tabs oder Zeilenvorschübe einzugeben.

Beispiel:

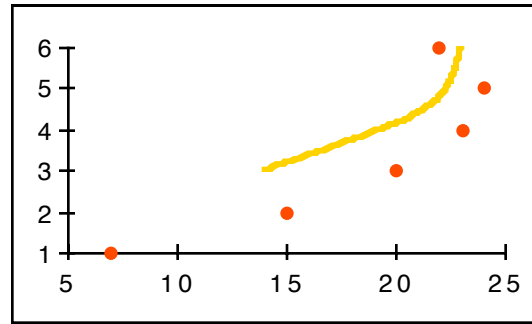
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 57 108 70 125 116 13 63 67 45 121 181
          189 201 166 115 105 91 75 99 72 33 29 69
          85 162 210 211 256 304 258 274)
ScatterChart()
SymbolStyle(1;bullet;2;1;black)
MovingAverage(1;4;;.1 .1 .1 .3 .3 .5 .7 .8 .9 1 1.2)
AxisOptions(x;none) // keine x-Achse
GridLocation(all;none) // kein Raster
CloseDrawing()
```



**MovingAverageLineStyle(*serienindex*;*intervallanzahl*;  
*berechnungsmethode*;*linienführung*;  
*strichstärke*;*farbe*;*muster*)**

Mit der Funktion `MovingAverageLineStyle()` kann das Aussehen der gleitenden Durchschnittslinien kontrolliert werden. Durch das 4. Argument *linienführung* kann der Kurvenverlauf — geglättet, polygonartig oder stufenförmig — gesteuert werden; Details dazu finden sich unter der Funktion `LineStyle()` im Abschnitt *Stile*. Die Strichstärke, die Farbe und das Muster der Durchschnittslinien können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Wenn nicht anders vorgegeben, werden Durchschnittslinien als schwarze, 1 Pixel breite Linien dargestellt. Beispiel:

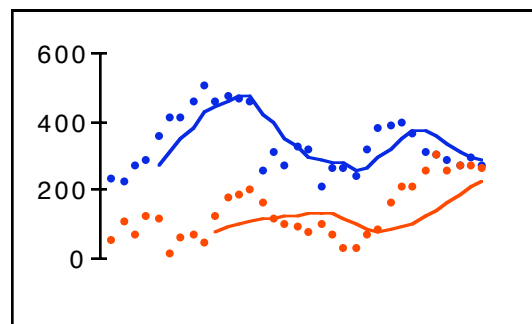
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(7 15 20 23 24 22)
ScatterChart(horizontal)
SymbolStyle(all;bullet;4;1;red)
MovingAverage(1;3;average)
MovingAverageLineStyle(1;3;;smooth;2;darkYellow)
GridLocation(all;none) // kein Raster
CloseDrawing()
```



Zur Zeit stehen drei Methoden zur Berechnung des gleitenden Durchschnitts zur Verfügung. Details dazu finden sich bei Funktion `MovingAverageOptions()`. Durch mehrfaches Aufrufen der Funktion `MovingAverage()` können gleichzeitig mehrere gleitende Durchschnitte sowohl für unterschiedliche Datenserien als auch für eine unterschiedliche Anzahl an Intervallen dargestellt werden. Beispiele:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 57 108 70 125 116 13 63 67 45 121 181
           189 201 166 115 105 91 75 99 72 33 29 69
           85 162 210 211 256 304 258 274 270 263;
           233 229 269 285 362 410 411 456 504 458
           474 470 463 257 308 270 325 316 213 263
           267 245 321 381 389 401 366 315 305 291
           275 299 272)

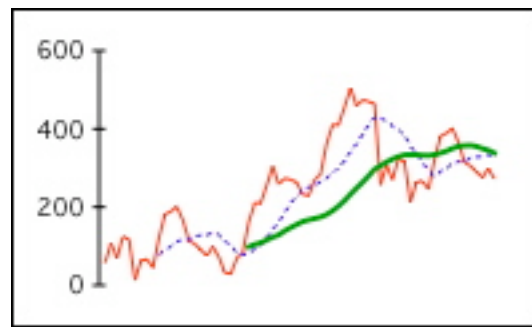
ScatterChart()
SymbolStyle(all;bullet;2)
MovingAverage(1;10;average)
MovingAverage(2;5;average)
MovingAverageLineStyle(1;10;average;poly;1;red)
MovingAverageLineStyle(2;5;average;poly;1;blue)
AxisOptions(x;none) // keine x-Achse
GridLocation(all;none) // kein Raster
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 57 108 70 125 116 13 63 67 45 121 181
          189 201 166 115 105 91 75 99 72 33 29 69
          85 162 210 211 256 304 258 274 270 263
          233 229 269 285 362 410 411 456 504 458
          474 470 463 257 308 270 325 316 213 263
          267 245 321 381 389 401 366 315 305 291
          275 299 272)
LineChart()
MovingAverage(1;10;average)
MovingAverage(1;25;average)
MovingAverageLineStyle(1;10;average;poly;1 2 2;blue)
MovingAverageLineStyle(1;25;average;poly;2;green)
AxisOptions(x;none) // keine x-Achse
GridLocation(all;none) // kein Raster
CloseDrawing()

```



**MovingAverageOptions(*serienindex*;*intervallanzahl*;  
*berechnungsmethode*;*ausrichtung*;*extrapolieren*;  
*hVersatz*;*vVersatz*;*istRelHVersatz*;  
*istRelVVersatz*)**

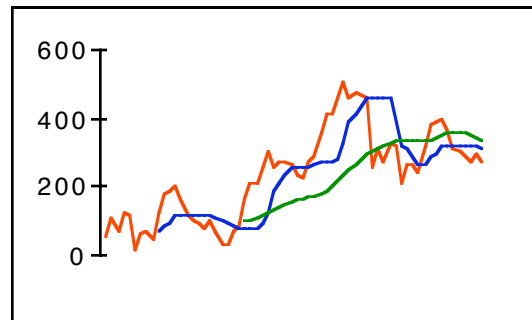
Durch die Funktion `MovingAverageOptions()` kann die Berechnung und Darstellung gleitender Durchschnitte vielfältig variiert werden. So kann mit dem 3. Argument *berechnungsmethode* unter drei verschiedenen Methoden zur Berechnung der Durchschnittswerte gewählt werden. Standardmäßig wird die Durchschnittslinie aus den arithmetischen Mittelwerten errechnet (*berechnungsmethode*=*average*). Anstelle der Mittelwerte ist es auch möglich, die Mediane zu verwenden (*berechnungsmethode*=*median*).

Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 57 108 70 125 116 13 63 67 45 121 181
            189 201 166 115 105 91 75 99 72 33 29 69
            85 162 210 211 256 304 258 274 270 263
            233 229 269 285 362 410 411 456 504 458
            474 470 463 257 308 270 325 316 213 263
            267 245 321 381 389 401 366 315 305 291
            275 299 272)
  LineChart()
  MovingAverage(1;10;median)
  MovingAverage(1;25;average)
  MovingAverageLineStyle(1;10;median;poly;1;blue)
  MovingAverageLineStyle(1;25;average;poly;1;green)
  MovingAverageOptions(1;10;median)
  MovingAverageOptions(1;25;average)
  AxisOptions(x;none) // keine x-Achse
  GridLocation(all;none) // kein Raster
CloseDrawing()

```



Die dritte Möglichkeit besteht in der Berechnung eines exponentiell gewichteten Durchschnitts (*berechnungsmethode=exponential*) nach der Formel:

```

w[i+1] = w[i] + a*(y[i+1]-w[i])
mit: a.....Glättungskonstante zwischen 0 und 1
     y[ ]...Originalwert
     w[ ]...geglätteter Wert

```

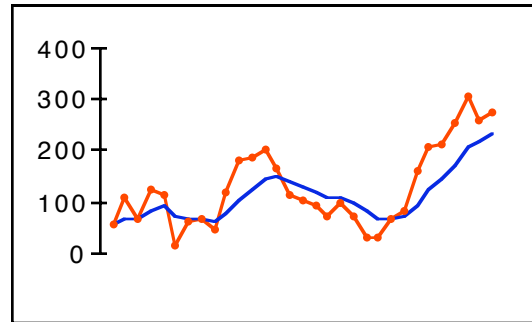
Die dazu benötigte Glättungskonstante wird als viertes Argument in der Funktion `MovingAverage()` anstelle der Wichtungsfaktoren übergeben. Zusätzlich kann zur Glättungskonstante noch optional ein Startwert hinzugefügt werden. Im Falle einer exponentiellen Glättung werden die Argumente *ausrichtung* und *extrapolieren* ignoriert.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 57 108 70 125 116 13 63 67 45 121 181
            189 201 166 115 105 91 75 99 72 33 29
            69 85 162 210 211 256 304 258 274)
  LineChart(symbol)
  SymbolStyle(all;bullet;2)
  // 0,25...Glättungskonstante
  MovingAverage(1;;exponential;0,25)
  MovingAverageLineStyle(1;2;exponential;poly;1;blue)
  MovingAverageOptions(1;;exponential)
  AxisOptions(x;none) // keine x-Achse
  GridLocation(all;none) // kein Raster
CloseDrawing()

```

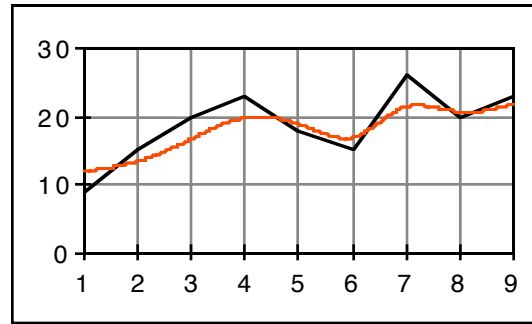


```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(9 15 20 23 18 15 26 20 23)
  LineChart()
  LineStyle(1;poly;1;black)

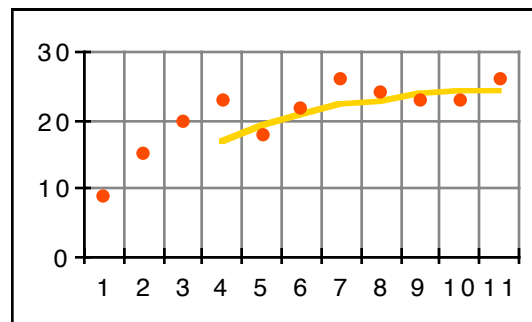
  // 0,5...Glättungskonstante
  // 15 ...Startwert
  MovingAverage(1;;exponential;0,5 15)
  MovingAverageLineStyle(1;;exponential;smooth;1;red)
  MovingAverageOptions(1;;exponential)
  GridFrame()
CloseDrawing()

```



Durch das 4. Argument *ausrichtung* kann die Position der Durchschnittsline kontrolliert werden. Durchschnittslinien — ausgenommen bei exponentieller Glättung — sind immer "etwas kürzer" als die Originalkurve, da die Punkteanzahl zur Darstellung der Durchschnittslinien stets um den Wert *intervallanzahl-1* geringer ist als die Punkteanzahl der Originalkurve. D.h., zum Beispiel eine 50-Tage-Durchschnittsline besitzt um 49 Punkte weniger als die Ausgangskurve. Zur Ausrichtung der kürzeren Durchschnittslinien stehen vier Konstanten zur Verfügung. Standardmäßig wird die Durchschnittsline nach dem letzten Datenpunkt ausgerichtet (*ausrichtung=backward*). Beispiel:

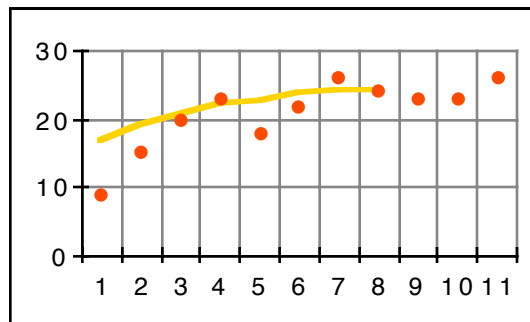
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(9 15 20 23 18 22 26 24 23 23 26)
  ScatterChart(;on)
  SymbolStyle(all;bullet;4;1;red)
  MovingAverage(1;4;average)
  MovingAverageLineStyle(1;4;average;poly;2;darkYellow)
  MovingAverageOptions(1;4;average;backward)
CloseDrawing()
```



Die Durchschnittsline wird bündig mit dem ersten Datenpunkt ausgerichtet, falls *ausrichtung=forward* gesetzt wird.

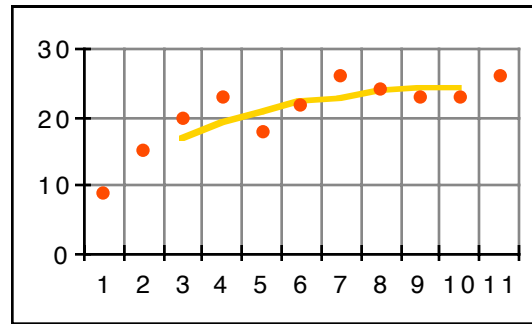
Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(9 15 20 23 18 22 26 24 23 23 26)
  ScatterChart(;on)
  SymbolStyle(all;bullet;4;1;red)
  MovingAverage(1;4;average)
  MovingAverageLineStyle(1;4;average;poly;2;darkYellow)
  MovingAverageOptions(1;4;average;forward)
CloseDrawing()
```



Bei *ausrichtung=centeredBackward* bzw. *ausrichtung=centeredForward* wird die Durchschnittsline "mittig" ausgerichtet. Das heißt, zum Beispiel bei einer Intervallanzahl von 50 wird bei *ausrichtung=centeredBackward* die Durchschnittsline um 25 Punkte eingerückt, bei *ausrichtung=centeredForward* um 24 Punkte. Falls die Anzahl der fehlenden Punkte geradzahlig ist, liefern die beiden Konstanten *centeredForward* und *centeredBackward* das gleiche Ergebnis. Beispiele:

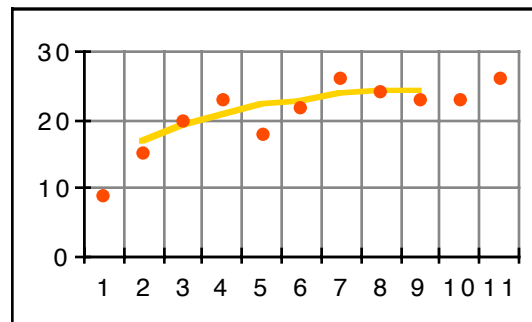
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(9 15 20 23 18 22 26 24 23 23 26)
  ScatterChart(;on)
  SymbolStyle(all;bullet;4;1;red)
  MovingAverage(1;4;average)
  MovingAverageLineStyle(1;4;average;poly;2;darkYellow)
  MovingAverageOptions(1;4;average;centeredBackward)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(9 15 20 23 18 22 26 24 23 23 26)
  ScatterChart(;on)
  SymbolStyle(all;bullet;4;1;red)
  MovingAverage(1;4;average)
  MovingAverageLineStyle(1;4;average;poly;2;darkYellow)
  MovingAverageOptions(1;4;average;centeredForward)
CloseDrawing()

```

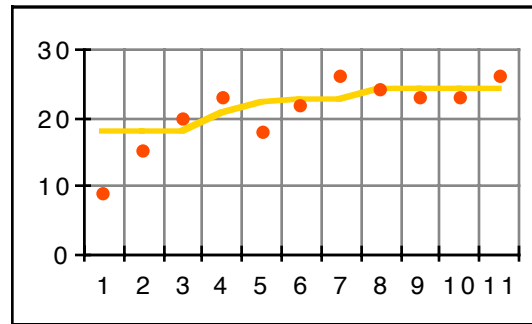


Durch Aktivieren des Arguments *extrapolieren=on* ist es möglich, die Durchschnittsline über den gesamten Datenbereich zu verlängern.  
Beispiel:

```

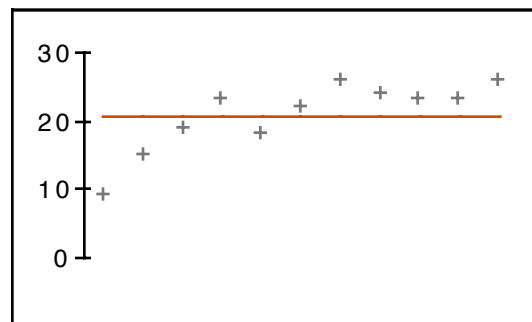
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(9 15 20 23 18 22 26 24 23 23 26)
  ScatterChart(;on)
  SymbolStyle(all;bullet;4;1;red)
  MovingAverage(1;6;average)
  MovingAverageLineStyle(1;6;average;poly;2;darkYellow)
  MovingAverageOptions(1;6;average;centeredForward;on)
CloseDrawing()

```



Als vorteilhaft erweist sich *extrapolieren=on* bei der Darstellung des Gesamtdurchschnitts. Der Gesamtdurchschnitt ergibt sich allgemein für eine aus  $n$  Werten bestehende Datenserie, indem *intervallanzahl* gleich  $n$  gesetzt wird. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(9 15 19 23 18 22 26 24 23 23 26)
  ScatterChart(;on)
  SymbolStyle(all;plus;5;1;darkGray)
  MovingAverage(1;11;average)
  MovingAverageLineStyle(1;11;average;poly;1;darkRed)
  MovingAverageOptions(1;11;average;;on)
  AxisOptions(x;none) // keine x-Achse
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



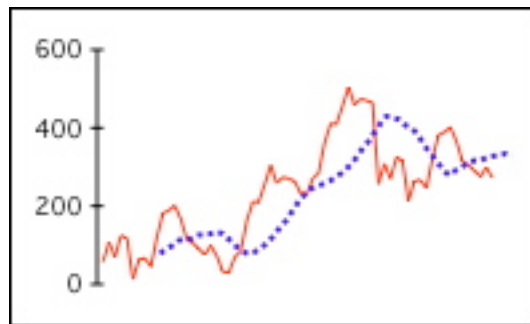
Gleitende Durchschnitte können optional durch Vorgabe eines horizontalen Versatzwertes (*hVersatz*) und/oder eines vertikalen Versatzwertes (*vVersatz*) verschoben dargestellt werden. Dabei können Versatzwerte entweder als absolute Größen (*istRelHVersatz=off* bzw. *istRelVVersatz=off*) vorgegeben werden (default) oder prozentuell bezogen auf die Durchschnittswerte (*istRelHVersatz=on* bzw. *istRelVVersatz=on*).

Beispiele:

```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 57 108 70 125 116 13 63 67 45 121 181
           189 201 166 115 105 91 75 99 72 33 29 69
           85 162 210 211 256 304 258 274 270 263
           233 229 269 285 362 410 411 456 504 458
           474 470 463 257 308 270 325 316 213 263
           267 245 321 381 389 401 366 315 305 291
           275 299 272)
LineChart()
MovingAverage(1;10;average)
MovingAverageLineStyle(1;10;average;poly;2 2 2;blue)
// um 5% nach rechts verschieben
MovingAverageOptions(1;10;average;;;5;on)
AxisOptions(x;none) // keine x-Achse
GridLocation(all;none) // kein Raster
CloseDrawing()

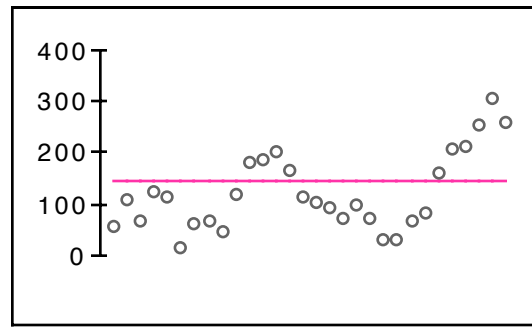
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 57 108 70 125 116 13 63 67
           45 121 181 189 201 166 115
           105 91 75 99 72 33 29 69 85
           162 210 211 256 304 258)
ScatterChart()
SymbolStyle(all;circle;4;1;darkGray)
MovingAverage(1;30;average)
MovingAverageLineStyle(1;30;average;poly;1;purple)
// um 20 Einheiten nach oben verschieben
MovingAverageOptions(1;30;average;;on;20)
AxisOptions(x;none) // keine x-Achse
GridLocation(all;none) // kein Raster
CloseDrawing()

```



## Kurvenanpassung

Zur Kurvenanpassung bei Punkt- und Liniendiagrammen stehen drei Funktionen zur Verfügung. Diese ermöglichen:

- die Festlegung unterschiedlicher Kurvenfunktionen
- die Gestaltung der Kurven
- die Feinabstimmung mittels zahlreicher Optionen

Bei allen drei Funktionen werden als erstes Argument der Serienindex und als zweites Argument der Kurvenfunktionstyp festgelegt. Wird kein Serienindex vorgegeben oder *serienindex=all* gesetzt, bezieht sich die jeweilige Funktion auf alle vorhandenen Datenserien. Folgende Kurvenfunktionen stehen zur Verfügung:

- Polynomfunktionen bis max. 10. Ordnung. (*typ=1...10*)  

$$f(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2 + c_3 \cdot x^3 + \dots + c_N \cdot x^N$$
Für den häufig benötigten Fall eines linearen Polynoms (Gerade)  $f(x) = c_0 + c_1 \cdot x$  steht zusätzlich die Konstante *linear* zur Verfügung (*typ=linear*).
- Potenzfunktionen (*typ=pow*):  

$$f(x) = c_0 \cdot x^{c_1}$$
- Exponentialfunktion (*typ=exp*):  

$$f(x) = c_0 \cdot \exp(c_1 \cdot x)$$
- Logarithmusfunktion (*typ=log*):  

$$f(x) = c_0 + c_1 \cdot \ln(x)$$

Die Berechnung der Kurvenkoeffizienten  $c_0 \dots c_{10}$  erfolgt nach der Methode der kleinsten Fehlerquadrate.

Mit der externen Funktion `xmCH_GetCFValue()` können der Korrelationskoeffizient  $r^2$  und die Kurvenkoeffizienten  $c_0$  bis max.  $c_{10}$  in eine FileMaker Pro Datei importiert werden. Die Funktion `xmCH_GetCFValue()` besitzt fünf Argumente:

```
xmCH_GetCFValue(Diagrammindex;
                 Serienindex;
                 Kurvenfunktionsindex;
                 Werteindex;
                 Formatanweisung)
```

- Diagrammindex:  
Da innerhalb einer Zeichnung mehrere Diagramme gleichzeitig dargestellt werden können, ist es notwendig, Diagramme durch einen Index zu spezifizieren. Der Index entspricht der Reihenfolge der Diagrammdefinitionen innerhalb von `OpenDrawing()`

und `CloseDrawing()`. Besteht die Zeichnung nur aus einem einzigen Diagramm, so ist der Diagrammindex gleich 1.

- **Serienindex:**  
Mit dem Serienindex wird festgelegt, auf welche Serie sich die Abfrage bezieht. Dies ist notwendig, da Kurvenanpassungen für mehrere Serien gleichzeitig durchgeführt werden können.
- **Kurvenfunktionsindex:**  
Mit dem Kurvenfunktionsindex wird festgelegt, auf welche Kurvenfunktion zugegriffen werden soll. Dies ist erforderlich, da für eine Datenserie mehrere Kurvenanpassungen gleichzeitig berechnet werden können. Der Kurvenfunktionsindex ist ident dem Argument *typ* und besitzt einen Wert zwischen -4 im Falle einer logarithmische Kurvenanpassung und 10 für ein Polynom 10. Ordnung.
- **Werteindex:**  
Der Werteindex erlaubt den Zugriff auf die einzelnen Koeffizienten. Werteindex gleich 1 liefert den Korrelationskoeffizienten  $r^2$ , Werteindex gleich 2 den Kurvenkoeffizienten  $c_0$ , der Werteindex 3 den Kurvenkoeffizienten  $c_1$ , usw.
- **Formatanweisung:**  
Optional kann der von der Funktion `xmCH_GetCFValue()` retournierte Koeffizient durch Vorgabe einer Formatanweisung entsprechend formatiert werden. Wird keine Formatanweisung vorgegeben, wird das Defaultformat "|u|" verwendet. Sämtliche Formatanweisungen inkl. zahlreicher Beispiele finden sich in *xmReferenz*.

Die Arbeitsweise der externen Funktion `xmCH_GetCFValue()` kann anhand der Datei *xmCFDemo.fp7* studiert werden.

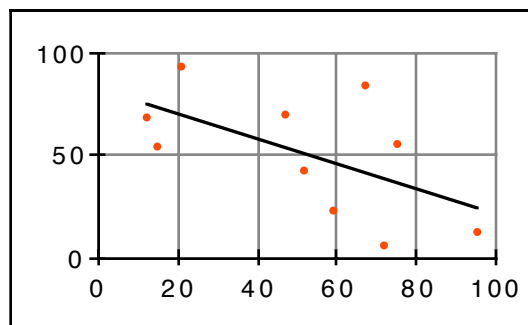
### **CurveFitting(*serienindex*;*typ*)**

Das erste Argument *serienindex* in der Funktion `CurveFitting()` legt fest, für welche Datenserie die Kurvenanpassung zu berechnen ist. Das zweite Argument *typ* definiert die Kurvenfunktion. Dazu stehen die folgenden Konstanten zur Verfügung:

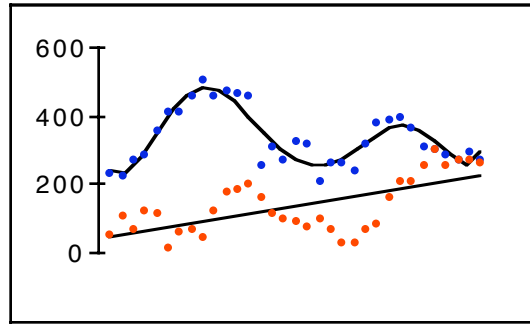
<i>Konstante</i>	<i>Wert</i>	<i>Funktion</i>
log	-4	$f(x) = c_0 + c_1 \cdot \ln(x)$
exp	-3	$f(x) = c_0 \cdot \exp(c_1 \cdot x)$
pow	-2	$f(x) = c_0 \cdot x^{c_1}$
linear	1	$f(x) = c_0 + c_1 \cdot x$ (default)

Wird *typ* nicht vorgegeben, so wird ein Polynom 1. Ordnung (Ausgleichsgerade) berechnet. Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(15 59 72 12 47 21 52 75 67 95; // x-Werte
            54 23 7 69 70 94 43 56 85 13) // y-Werte
  ScatterChart2D()
  SymbolStyle(all;bullet;2)
  CurveFitting()
CloseDrawing()
```



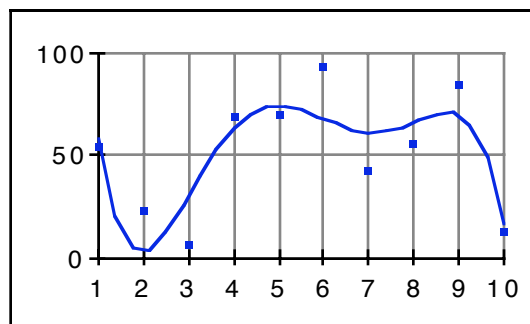
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 57 108 70 125 116 13 63 67 45 121 181
            189 201 166 115 105 91 75 99 72 33 29 69
            85 162 210 211 256 304 258 274 270 263;
            233 229 269 285 362 410 411 456 504 458
            474 470 463 257 308 270 325 316 213 263
            267 245 321 381 389 401 366 315 305 291
            275 299 272)
  ScatterChart()
  SymbolStyle(all;bullet;2)
  CurveFitting(1;1)
  CurveFitting(2;7)
  AxisOptions(x;none) // keine x-Achse
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



**CurveFittingLineStyle(*serienindex*;*typ*;*strichstärke*;  
*farbe*;*muster*)**

Mit der Funktion `CurveFittingLineStyle()` kann das Aussehen der Kurven kontrolliert werden. Die Strichstärke, die Farbe und das Muster können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Wenn nicht anders vorgegeben, werden die Kurven schwarz und 1 Pixel breit dargestellt. Beispiele:

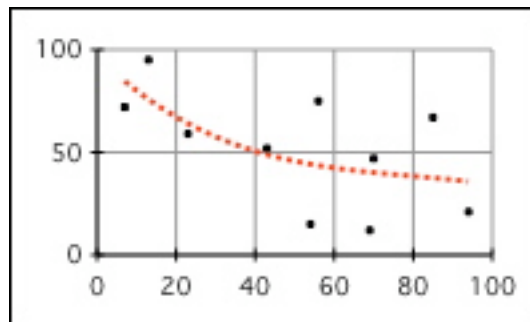
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(54 23 7 69 70 94 43 56 85 13)
  ScatterChart()
  SymbolStyle(all;square;2;1;blue)
  CurveFitting(1;5)
  CurveFittingLineStyle(1;5;1;blue)
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(54 23 7 69 70 94 43 56 85 13; // x-Werte
          15 59 72 12 47 21 52 75 67 95) // y-Werte
ScatterChart2D()
SymbolStyle(all;square;2;1;black)
CurveFitting(1;3)
CurveFittingLineStyle(1;3;2 2 2;red)
CloseDrawing()

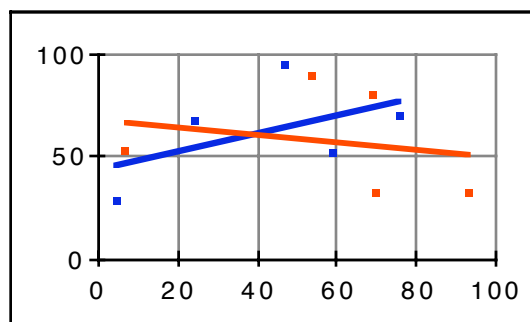
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(54 93 7 69 70; // x-Werte 1.Serie
          90 33 53 80 33; // y-Werte 1.Serie
          5 59 76 24 47; // x-Werte 2.Serie
          29 52 70 67 95) // y-Werte 2.Serie
ScatterChart2D()
SymbolStyle(all;square;2)
CurveFitting(all;1)
CurveFittingLineStyle(1;1;2;red)
CurveFittingLineStyle(2;1;2;blue)
CloseDrawing()

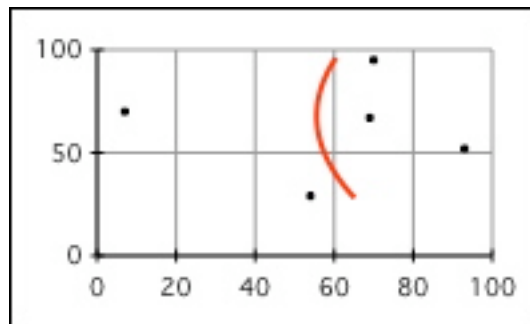
```



**CurveFittingOptions(*serienindex*;*typ*;*achsenVertauschen*;  
*extrapolieren*;*nulldurchgang*)**

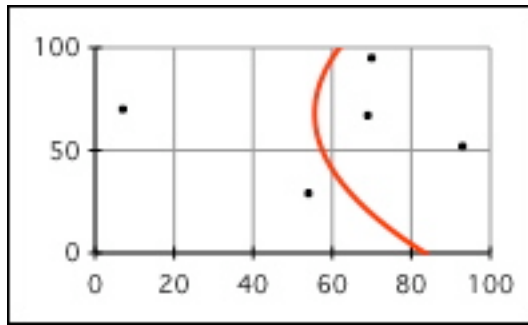
Die Funktion `CurveFittingOptions()` ermöglicht Variationen in der Berechnung und Darstellung der Kurven. Durch Aktivieren des Arguments *achsenVertauschen=on* werden die Koordinaten der zur Kurvenberechnung verwendeten Punkte vertauscht. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(54 93 7 69 70; // x-Werte
            29 52 70 67 95) // y-Werte
  ScatterChart2D()
  SymbolStyle(all;square;2;1;black)
  CurveFitting(all;2)
  CurveFittingLineStyle(1;2;2;red)
  CurveFittingOptions(all;2;on)
CloseDrawing()
```



Durch Aktivieren des Arguments *extrapolieren=on* können Kurven bis zum Rand des Diagramms verlängert werden. Beispiele:

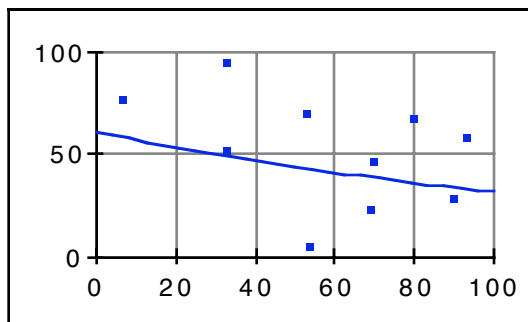
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(54 93 7 69 70; // x-Werte
            29 52 70 67 95) // y-Werte
  ScatterChart2D()
  SymbolStyle(all;square;2;1;black)
  CurveFitting(all;2)
  CurveFittingLineStyle(1;2;2;red)
  CurveFittingOptions(all;2;on;on)
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(54 93 7 69 70 90 33 53 80 33;
            5 59 76 24 47 29 52 70 67 95)
  ScatterChart2D()
  SymbolStyle(all;square;2;1;blue)
  CurveFitting(1;exp)
  CurveFittingLineStyle(1;exp;1;blue)
  CurveFittingOptions(1;exp;;on)
CloseDrawing()

```



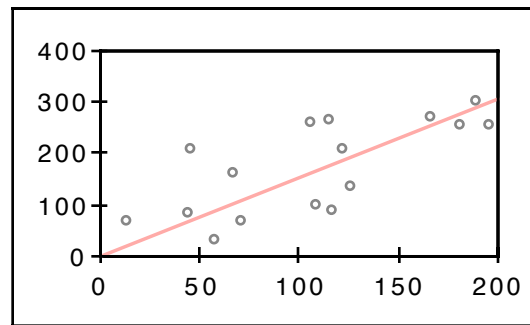
Polynomfunktionen verlaufen durch den Koordinatenursprung falls das Argument *nulldurchgang=on* gesetzt wird. Bei allen anderen Kurvenfunktionen wird das Argument *nulldurchgang* ignoriert.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(57 108 70 125 116 13 43 67 45 121
            181 189 195 166 115 105 91;
            35 99 72 139 90 69 85 162 210 211
            256 304 258 274 270 263)
  ScatterChart2D()
  SymbolStyle(all;circle;3;1;gray)
  CurveFitting(1;linear)
  CurveFittingLineStyle(1;linear;1;lightRed)
  CurveFittingOptions(1;linear;;on;on)
  MajorGridLineWidths(all;all;0) // hide grid
  GridLocation(xy;front)
  GridFrame()
CloseDrawing()

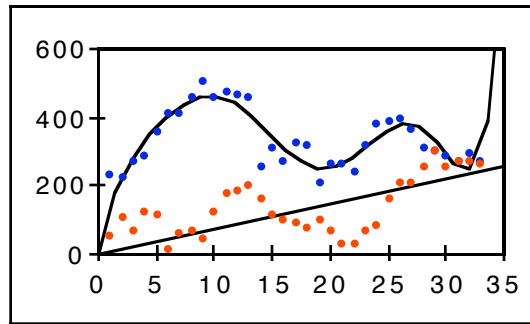
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 57 108 70 125 116 13 63 67 45 121 181
            189 201 166 115 105 91 75 99 72 33 29 69
            85 162 210 211 256 304 258 274 270 263;
            233 229 269 285 362 410 411 456 504 458
            474 470 463 257 308 270 325 316 213 263
            267 245 321 381 389 401 366 315 305 291
            275 299 272)
  ScatterChart()
  SymbolStyle(all;bullet;2)
  CurveFitting(1;linear)
  CurveFitting(2;7)
  CurveFittingOptions(1;1;;on;on)
  CurveFittingOptions(2;7;;on;on)
  MajorGridLineWidths(all;all;0) // Raster ausblenden
  GridFrame()
CloseDrawing()

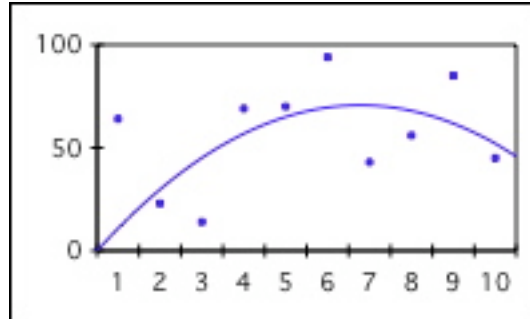
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(64 23 14 69 70 94 43 56 85 45)
  ScatterChart(;on)
  SymbolStyle(all;square;2;1;blue)
  CurveFitting(1;2)
  CurveFittingLineStyle(1;2;1;blue)
  CurveFittingOptions(1;2;off;on;on)
  MajorGridLineWidths(all;all;0) // Raster ausblenden
  GridFrame()
CloseDrawing()

```



## Fehlerbalken

Punkt-, Linien- und Balkendiagramme können durch Fehlerbalken ergänzt werden. Zur Berechnung und Darstellung der Fehlerbalken stehen vier Funktionen zur Verfügung.

**ErrorBars(*serienindex*;*achsenindex*;*fehlerrichtung*;*typ*;  
*wert1*;*wert2*;*wert3*;*wert4*)**

Durch die Funktion `ErrorBars()` können alle wichtigen Größen zur Berechnung der Fehlerbalken kontrolliert werden. Das erste Argument *serienindex* legt fest, auf welche Datenserie sich die Funktion bezieht. Mit dem zweiten Argument *achsenindex* kann bei zweidimensionalen Diagrammen die Richtung der Fehlerbalken vorgegeben werden. Wird kein Achsenindex vorgegeben oder *achsenindex=all* gesetzt, erfolgt die Berechnung der Fehlerbalken sowohl in x- als auch in y-Richtung. Mit dem 3. Argument *fehlerrichtung* kann festgelegt werden, ob nur die positiven Fehler (*fehlerrichtung=plus*) oder nur die negativen Fehler (*fehlerrichtung=minus*) oder beide Fehler (*fehlerrichtung=both*) dargestellt werden sollen. Standardmäßig werden beide Fehler dargestellt (*fehlerrichtung=both*). Die Art der Fehlerberechnung wird durch das Argument *typ* festgelegt. Die folgenden Fehlerberechnungen sind möglich:

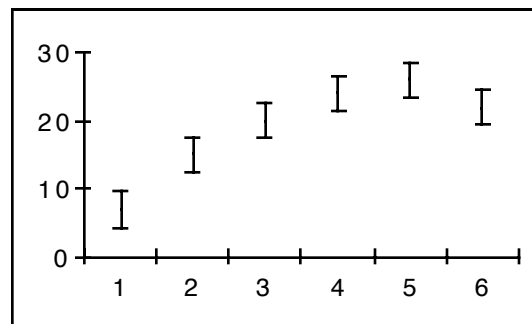
- Standardfehler: (*typ=stdError*)
- Standardabweichung: (*typ=stdDev*)
- Prozentuelle Fehler: (*typ=percent*)
- Konstante Fehler: (*typ=constant*)
- Benutzerdefinierte Fehlerlisten: (*typ=valueList*)

Wird *typ* nicht vorgegeben, so wird per default der Standardfehler berechnet. Benutzerdefinierte Fehlerlisten (*typ=valueList*) werden durch die Funktion `ErrorBarData()` übergeben. Die Bedeutung der Argumente *wert1* bis *wert4* ist abhängig von der Art der Fehlerberechnung:

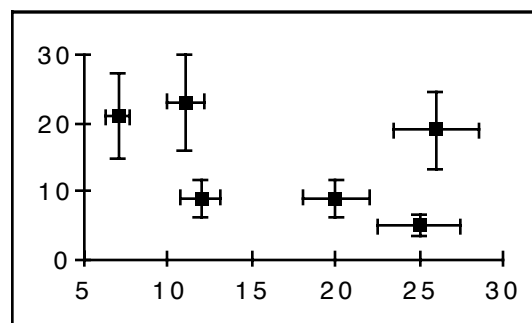
- Standardfehler:  
*wert1...wert4* haben keine Bedeutung und werden ignoriert.
- Standardabweichung:  
*wert1...wert4* sind die Standardabweichungen, default: 1
- Prozentuelle Fehler:  
*wert1...wert4* sind Prozentwerte, default: 5 [%]
- Konstante Fehler:  
*wert1...wert4* sind konstante Werte, default: 1
- Benutzerdefinierte Fehlerlisten:  
*wert1...wert4* haben keine Bedeutung und werden ignoriert.

Das Argument *wert1* bezieht sich auf den positiven Fehler in x-Richtung, das Argument *wert2* auf den negativen Fehler in x-Richtung, die Argumente *wert3* und *wert4* beziehen sich auf den positiven bzw. negativen Fehler in y-Richtung. Beispiele:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(7 15 20 24 26 22)
ScatterChart(;on)
SymbolStyle(all;none)
ErrorBars() // default: Standardfehler
GridLocation(all;none) // kein Raster
CloseDrawing()
```



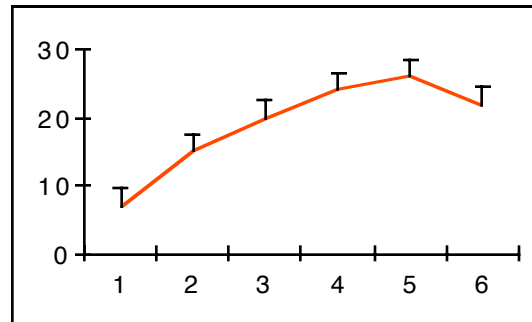
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData( 7 11 12 20 26 25; // x-Werte
          21 23 9 9 19 5) // y-Werte
ScatterChart2D()
SymbolStyle(all;square;4;1;black)
ErrorBars(1;all;both;percent;10;10;30;30)
GridLocation(all;none) // kein Raster
CloseDrawing()
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(7 15 20 24 26 22)
LineChart(;on)
ErrorBars(1;;plus)      // default: Standardfehler
GridLocation(all;none) // kein Raster
CloseDrawing()

```



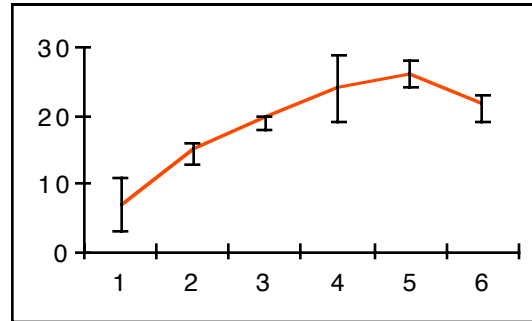
**ErrorBarData(*serienindex*;*werteliste1*;*werteliste2*;  
*werteliste3*;*werteliste4*)**

Mittels der Funktion `ErrorBarData()` können benutzerdefinierte Fehlerwerte übergeben werden. Korrespondierend dazu ist in der Funktion `ErrorBars()` das 4. Argument *typ=valueList* zu setzen, andernfalls wird die Funktion `ErrorBarData()` ignoriert. Das Argument *werteliste1* enthält die positiven Fehlerwerte in x-Richtung, das Argument *werteliste2* die negativen Fehlerwerte in x-Richtung. Analog repräsentieren die Argumente *werteliste3* und *werteliste4* die positiven bzw. negativen Fehlerwerte in y-Richtung. Die Werte einer Liste sind analog wie in der Funktion `ChartData()` getrennt durch Leerzeichen, Tabs oder Zeilenumbrüche einzugeben. Ist die Anzahl der übergebenen Fehlerwerte kleiner als die Anzahl der Datenpunkte, so werden die fehlenden Werte durch Nullen ergänzt. Beispiele:

```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
ChartData(7 15 20 24 26 22)
LineChart(;on)
ErrorBars(1;y:both;valueList)
ErrorBarData(1;;;4 1 0 5 2 1;4 2 2 5 2 3)
GridLocation(all;none) // kein Raster
CloseDrawing()

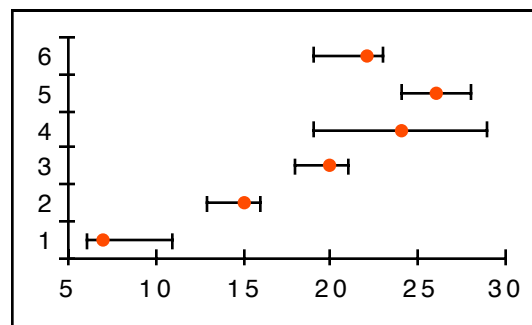
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(7 15 20 24 26 22)
  ScatterChart(horizontal;on)
  SymbolStyle(all;bulet;4;1;red)
  ErrorBars(1;;;valueList)
  ErrorBarData(1;4 1 1 5 2 1; 1 2 2 5 2 3)
  GridLocation(all;none) // kein Raster
CloseDrawing()

```

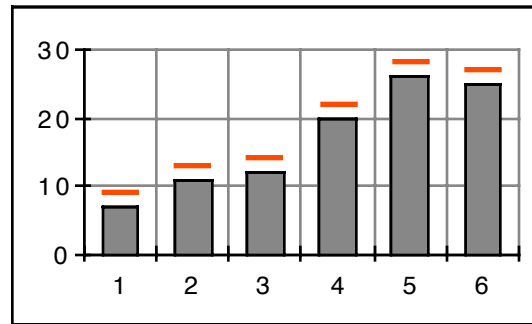


**ErrorBarStyle(*serienindex*; *achsenindex*; *nurMarkerZeigen*; *markerlänge*; *strichstärke*; *farbe*; *muster*)**

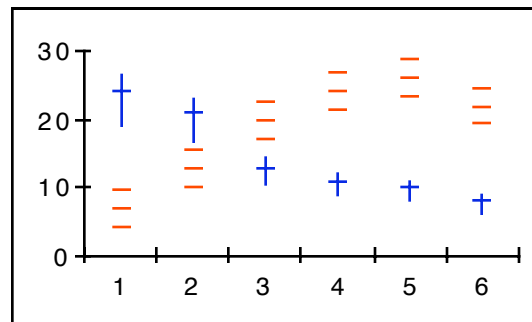
Mit der Funktion `ErrorBarStyle()` kann das Aussehen der Fehlerbalken kontrolliert werden. Das erste Argument *serienindex* legt fest, auf welche Datenserie sich die Funktion bezieht. Das zweite Argument *achsenindex* legt fest, auf welche Fehlerbalkenrichtung sich die Funktion bezieht. Durch Aktivieren des Arguments *nurMarkerZeigen=on* können die Verbindungslinien zwischen den Fehlermarkern unterdrückt werden. Die Länge der Fehlermarker (in Pixels) kann mit dem Argument *markerlänge* kontrolliert werden. Die Strichstärke, die Farbe und das Muster können durch die Argumente *strichstärke*, *farbe* und *muster* variiert werden. Wenn nicht anders vorgegeben, werden Fehlerbalken schwarz und 1 Pixel breit dargestellt.

Beispiele:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(7 11 12 20 26 25)
  BarChart()
  FillStyle(all;gray)
  ErrorBars(1;all;plus;constant;;;2)
  ErrorBarStyle(all;;;on;13;2;red)
CloseDrawing()
```



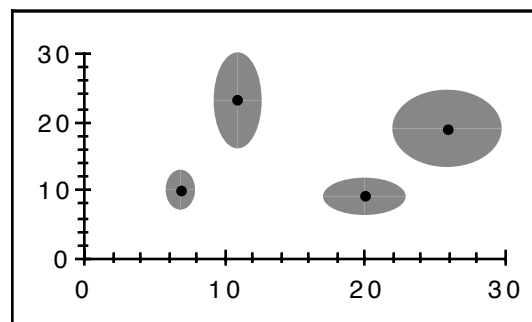
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 7 13 20 24 26 22; // 1.Serie
            24 21 13 11 10 8) // 2.Serie
  ScatterChart(;on)
  SymbolStyle(all;hBar;6)
  ErrorBars(1;y:both;stdError)
  ErrorBars(2;y:both;percent;;;10;20)
  ErrorBarStyle(1;y;on ;6;1;red)
  ErrorBarStyle(2;y;off;0;1;blue)
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



**ErrorBarStyle2D(*serienindex*; *form*; *füllfarbe*; *füllmuster*; *rahmenbreite*; *rahmenfarbe*; *rahmenmuster*)**

Durch die Funktion `ErrorBarStyle2D()` können bei zweidimensionalen Diagrammen die Fehlerwerte optional auch durch Ellipsen oder Rechtecke dargestellt werden. Das Argument *form* ermöglicht die Wahl zwischen elliptischen Fehlerflächen (*form=oval*) oder rechteckigen Fehlerflächen (*form=rect*). Wird *form* nicht vorgegeben, so werden standardmäßig Ellipsen zur Darstellung der Fehler verwendet. Die Fläche kann durch die Argumente *füllfarbe* und *füllmuster*, die Berandung durch die Argumente *rahmenbreite*, *rahmenfarbe* und *rahmenmuster* variiert werden. Standardmäßig werden Ellipsen durch einen 1 Pixel breiten, schwarzen Rand und ohne Füllung dargestellt. Beispiele:

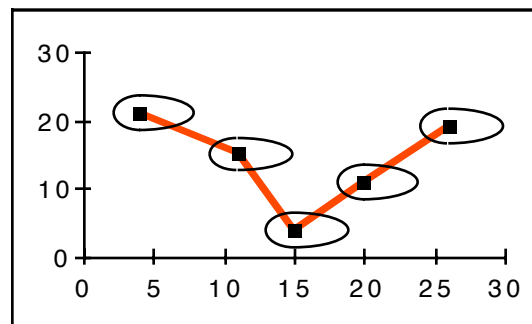
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 7 11 20 26; // x-Werte
            10 23 9 19) // y-Werte
  ScatterChart2D()
  Scaling(all;linear;0;30;3;5)
  SymbolStyle(all;bullet;3;1;black)
  ErrorBars(1;all;both;percent;15;15;30;30)
  ErrorBarStyle(1;all;on;0)
  ErrorBarStyle2D(1;oval;gray;black;0)
  GridLocation(all;none) // kein Raster
CloseDrawing()
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(4 11 15 20 26; 21 15 4 11 19)
  LineChart2D(symbol)
  LineStyle(1;poly;2)
  SymbolStyle(all;square;4;1;black)
  ErrorBars(1;x;both;constant;4;2)
  ErrorBars(1;y;both;stdError)
  ErrorBarStyle(1;all;on;0)
  ErrorBarStyle2D(1;oval)
  GridLocation(all;none) // kein Raster
CloseDrawing()

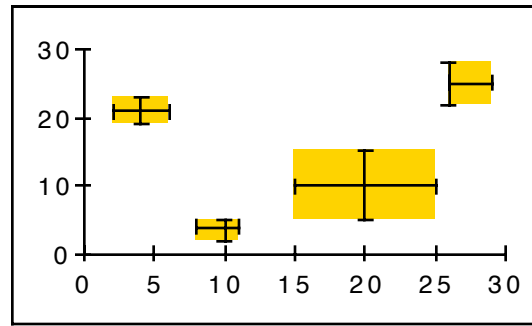
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData( 4 10 26 20; // x-Werte
            21  4 25 10) // y-Werte
  ScatterChart2D()
  SymbolStyle(all;none;3;1;black)
  ErrorBars(1;all;both;valueList)
  ErrorBarData(1;2 1 3 5; // positive x-Fehler
              2 2 0 5; // negative x-Fehler
              2 1 3 5; // positive y-Fehler
              2 2 3 5) // negative y-Fehler
  ErrorBarStyle2D(1;rect;darkYellow;black;0)
  GridLocation(all;none) // kein Raster
CloseDrawing()

```



## Grafische Basisfunktionen

Zusätzlich zu den Diagrammfunktionen steht ein Satz von über 20 grafischen Basisfunktionen zum Zeichnen von Linien, Rechtecken, Ellipsen, Polygonen, beliebigen durch Pfade definierten Objekten, Bitmaps, Texten etc. zur Verfügung. Diagramme können dadurch mit Texten, Logos usw. ergänzt werden. Es können aber auch allein aus den Basisfunktionen komplette Zeichnungen aufgebaut werden. Die Mächtigkeit und die Vielzahl der Basisfunktionen ermöglichen einen äußerst vielfältigen Einsatz.

Die Koordinaten beziehen sich bei allen Basisfunktionen auf die linke obere Ecke der Zeichnung. Falls Funktionen innerhalb eines Views angeführt werden, beziehen sich die Koordinaten auf die linke obere Ecke des Views. Views können auch dazu verwendet werden, mehrere Funktionen zu Gruppen zusammenzufassen. Die Gruppen können dann einfach durch Ändern der Viewpositionen auf der Zeichnung verschoben werden.

**AddText(hPosition;vPosition;text;schrift;größe;stil;  
farbe;hAusrichtung;vAusrichtung;neigungswinkel;  
maxBreite;maxHöhe;auslassungsPosition)**

Bei linksbündiger Ausrichtung des Textes (default) wird durch *hPosition* der linke Rand des Textes festgelegt, bei mittiger Ausrichtung (*hAusrichtung=center*) definiert *hPosition* die Mitte des Textes, bei rechtsbündiger Ausrichtung (*hAusrichtung=right*) legt *hPosition* den rechten Rand des Textes fest. Mit dem Argument *vPosition* wird die vertikale Position des Textes in Abhängigkeit von *vAusrichtung* festgelegt. Wird *vAusrichtung=baseline* gesetzt (default) dann bezieht sich *vPosition* auf die Grundlinie der ersten Textzeile. Bei *vAusrichtung=top* wird durch *vPosition* der obere Rand des Textes festgelegt, bei *vAusrichtung=center* definiert *vPosition* die vertikale Mitte des Textes, bei *vAusrichtung=bottom* legt *vPosition* den unteren Rand des Textes fest.

Mehrzeilige Texte werden durch Einfügen eines Zeilenumbruchs "\n" erstellt. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddText( 10;25;"Text1")
  AddText( 50;25;"Text2";"Courier";12)
  AddText(100;25;"Text3";"Times";12:bold+underline)
  AddText(150;25;"Text4";;12;plain;blue)
  AddText(10;60;"Mehrzeiliger\nText\nlinksbündig";
    ;;red)
  AddText(100;60;"Mehrzeiliger\nText\nmittig";
    ;;italic;green;center)
  AddText(190;60;"Mehrzeiliger\nText\nrechtsbündig";
    ;;underline;blue;right)
  AddText(60;115;"Copyright by X2max Software")
CloseDrawing()
```



Bei *neigungswinkel=0* (in Grad) wird der Text horizontal angeordnet (default). Bei einem *neigungswinkel>0* wird der Text im Uhrzeigersinn gedreht, bei *neigungswinkel<0* entgegengesetzt dem Uhrzeigersinn. Alle Winkel sind im Bereich von  $\pm 360$  Grad anzugeben, dabei ist 0 Grad horizontal (3 Uhr). Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
AddText(100;60;"Gedrehter Text";;14;;255 0 0 255;2;2;-60)
AddText(100;60;"Gedrehter Text";;14;;255 0 0 210;2;2;-30)
AddText(100;60;"Gedrehter Text";;14;;255 0 0 170;2;2; 0)
AddText(100;60;"Gedrehter Text";;14;;255 0 0 140;2;2; 30)
AddText(100;60;"Gedrehter Text";;14;;255 0 0 120;2;2; 60)
AddText(100;60;"Gedrehter Text";;14;;255 0 0  80;2;2; 90)
CloseDrawing()
```



Bei Vorgabe einer maximalen Breite (*maxBreite>0*) können längere, mehrzeilige Texte als Fließtext ausgegeben werden. Die Höhe eines Textblocks kann über das Argument *maxHöhe* kontrolliert werden. Texte, welche nicht vollständig in den durch *maxBreite* und *maxHöhe* vorgegebenen Bereich passen, werden entsprechend der Vorgabe durch das Argument *auslassungsPosition* gekürzt. Fünf Möglichkeiten stehen zur Verfügung:

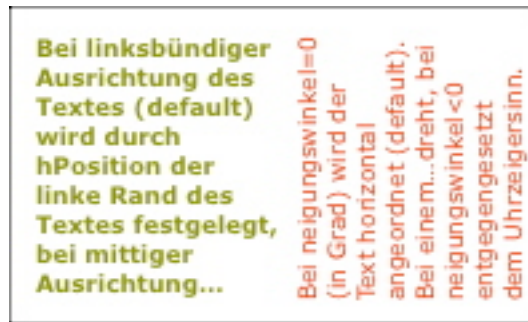
*auslassungsPosition=0*: Text wird am Ende geclippt, ohne "..."  
*auslassungsPosition=1*: Text wird am Anfang geclippt (...text)  
*auslassungsPosition=2*: Text wird in der Mitte geclippt (text...text)  
*auslassungsPosition=3*: Text wird am Ende geclippt (text...) – default  
*auslassungsPosition=4*: Text wird am Anfang und Ende geclippt (...text...)

Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddText(10;10;"Bei linksbündiger Ausrichtung des
Textes (default) wird durch hPosition der linke Rand des
Textes festgelegt, bei mittiger Ausrichtung (hAusrichtung=
center) definiert hPosition die Mitte des Textes, bei
rechtsbündiger Ausrichtung (hAusrichtung=right) legt
hPosition den rechten Rand des Textes fest.");
    "Verdana";9;bold;olive;left;top;0;90;100)
  AddText(100;25;"Bei neigungswinkel=0 (in Grad) wird
der Text horizontal angeordnet (default). Bei einem
neigungswinkel>0 wird der Text im Uhrzeigersinn gedreht, bei
neigungswinkel<0 entgegengesetzt dem Uhrzeigersinn.");
    "Verdana";9;plain;darkRed;left;;-90;100;90;2)
CloseDrawing()

```

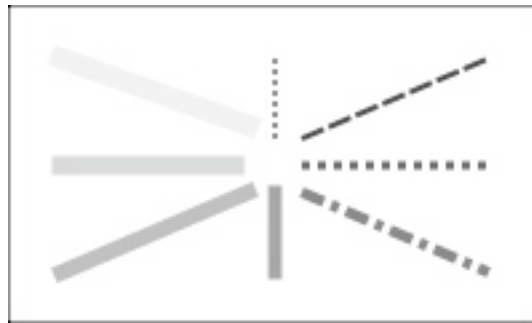


Standardmäßig werden *maxBreite=-1* und *maxHöhe=-1* gesetzt, d.h. keine Beschränkung auf eine bestimmte Breite oder Höhe und somit keine Fließtextunterstützung.

**AddLine(hStart;vStart;hEnde;vEnde;strichstärke;farbe;muster)**

Eventuell benötigte Linien können durch die Funktion AddLine() hinzugefügt werden; wobei die Strichstärke, die Farbe und das Muster variiert werden können. Der Koordinatenursprung befindet sich in der linken, oberen Ecke der Zeichnung bzw. des Views. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddLine(100;50;100; 20;1 2 2; 0 0 0)
  AddLine(110;50;180; 20;2 9 2; 60 60 60)
  AddLine(110;60;180; 60;3 3 3; 90 90 90)
  AddLine(110;70;180;100;4 9 3 3 3;120 120 120)
  AddLine(100;70;100;100;5;150 150 150)
  AddLine( 90;70; 20;100;6;180 180 180)
  AddLine( 85;60; 20; 60;7;210 210 210)
  AddLine( 90;45; 20; 20;8;240 240 240)
CloseDrawing()
```



**AddFrame(links;oben;breite;höhe;rahmenbreite;farbe;muster)**

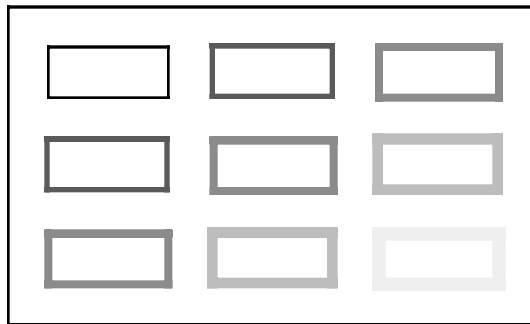
Die Funktion AddFrame ( ) ermöglicht die Erstellung rechteckiger Rahmen, wobei die Rahmenbreite, die Farbe und das Muster variiert werden können. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)

  // linke Spalte
  AddFrame(15;15;46;20;1; 0 0 0)
  AddFrame(15;50;46;20;2; 90 90 90)
  AddFrame(15;85;46;20;3;140 140 140)

  // mittlere Spalte
  AddFrame(77;15;46;20;2; 90 90 90)
  AddFrame(77;50;46;20;3;140 140 140)
  AddFrame(77;85;46;20;4;190 190 190)

  // rechte Spalte
  AddFrame(139;15;46;20;3;140 140 140)
  AddFrame(139;50;46;20;4;190 190 190)
  AddFrame(139;85;46;20;5;240 240 240)
CloseDrawing()
```



**AddRect(links;oben;breite;höhe;füllfarbe;füllmuster)**

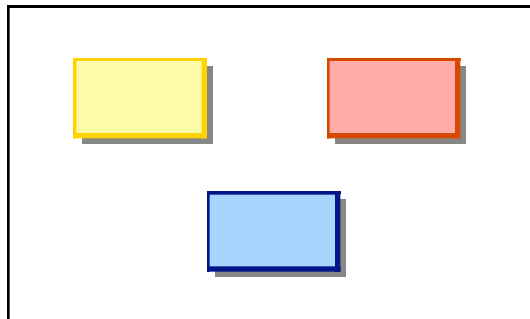
Die Funktion AddRect() dient zur Erstellung rechteckiger Flächen, wobei die Farbe und das Muster variiert werden können. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)

  OpenView(25;20;53;33) // links oben
    AddRect(3;3;50;30;gray)
    AddRect(0;0;50;30;lightYellow)
    AddFrame(0;0;50;30;2;darkYellow)
  CloseView()

  OpenView(120;20;53;33) // rechts oben
    AddRect(3;3;50;30;gray)
    AddRect(0;0;50;30;lightRed)
    AddFrame(0;0;50;30;2;darkRed)
  CloseView()

  OpenView(75;70;53;33) // unten Mitte
    AddRect(3;3;50;30;gray)
    AddRect(0;0;50;30;lightBlue)
    AddFrame(0;0;50;30;2;darkBlue)
  CloseView()
CloseDrawing()
```



**AddRoundFrame(links;oben;breite;höhe;hKrümmung;  
vKrümmung;rahmenbreite;farbe;muster)**

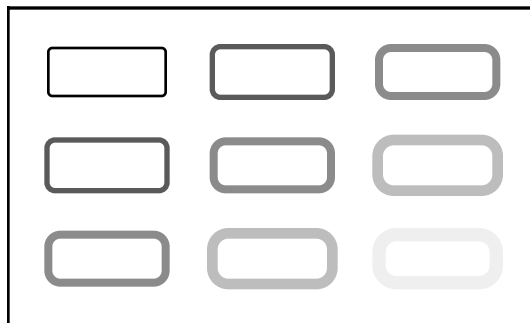
Die Funktion AddRoundFrame() dient zur Erstellung von Rahmen mit abgerundeten Ecken, wobei die Rahmenbreite, die Farbe und das Muster variiert werden können. Die Stärke der Rundung wird durch die Argumente *hKrümmung* und *vKrümmung* kontrolliert. Je größer *hKrümmung* bzw. *vKrümmung* desto stärker die Rundung. Bei *hKrümmung*=0 bzw. *vKrümmung*=0 verschwindet die Rundung und AddRoundFrame() geht in AddFrame() über. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)

  // linke Spalte
  AddRoundFrame(15;15;46;20; 4; 4;1; 0 0 0)
  AddRoundFrame(15;50;46;20; 8; 8;2; 90 90 90)
  AddRoundFrame(15;85;46;20;12;12;3;140 140 140)

  // mittlere Spalte
  AddRoundFrame(77;15;46;20; 8; 8;2; 90 90 90)
  AddRoundFrame(77;50;46;20;12;12;3;140 140 140)
  AddRoundFrame(77;85;46;20;16;16;4;190 190 190)

  // rechte Spalte
  AddRoundFrame(139;15;46;20;12;12;3;140 140 140)
  AddRoundFrame(139;50;46;20;16;16;4;190 190 190)
  AddRoundFrame(139;85;46;20;20;20;5;240 240 240)
CloseDrawing()
```



**AddRoundRect(links;oben;breite;höhe;hKrümmung;  
vKrümmung;füllfarbe;füllmuster)**

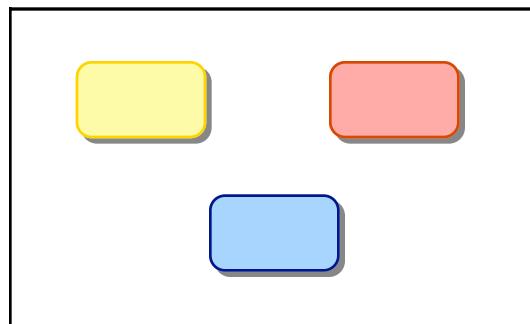
Die Funktion AddRoundRect ( ) ermöglicht die Erstellung von abgerundeten Rechteckflächen, wobei die Farbe und das Muster variiert werden können. Die Stärke der Rundung wird durch die Argumente *hKrümmung* und *vKrümmung* kontrolliert. Je größer *hKrümmung* bzw. *vKrümmung* desto stärker die Rundung. Bei *hKrümmung*=0 bzw. *vKrümmung*=0 verschwindet die Rundung und AddRoundRect ( ) geht in AddRect ( ) über. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)

  OpenView(25;20;52;32) // links oben
    AddRoundRect(2;2;50;30;;;gray)
    AddRoundRect(0;0;50;30;;;lightYellow)
    AddRoundFrame(0;0;50;30;;;1;darkYellow)
  CloseView()

  OpenView(120;20;52;32) // rechts oben
    AddRoundRect(2;2;50;30;;;gray)
    AddRoundRect(0;0;50;30;;;lightRed)
    AddRoundFrame(0;0;50;30;;;1;darkRed)
  CloseView()

  OpenView(75;70;52;32) // unten Mitte
    AddRoundRect(2;2;50;30;;;gray)
    AddRoundRect(0;0;50;30;;;lightBlue)
    AddRoundFrame(0;0;50;30;;;1;darkBlue)
  CloseView()
CloseDrawing()
```



**AddEllipse(links;oben;breite;höhe;strichstärke;farbe;  
muster)**

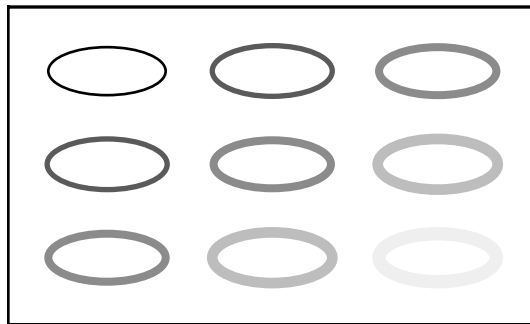
Die Funktion AddEllipse() ermöglicht die Erstellung ovaler Rahmen, wobei die Strichstärke, die Farbe und das Muster variiert werden können. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)

  // linke Spalte
  AddEllipse(15;15;46;20;1; 0 0 0)
  AddEllipse(15;50;46;20;2; 90 90 90)
  AddEllipse(15;85;46;20;3;140 140 140)

  // mittlere Spalte
  AddEllipse(77;15;46;20;2; 90 90 90)
  AddEllipse(77;50;46;20;3;140 140 140)
  AddEllipse(77;85;46;20;4;190 190 190)

  // rechte Spalte
  AddEllipse(139;15;46;20;3;140 140 140)
  AddEllipse(139;50;46;20;4;190 190 190)
  AddEllipse(139;85;46;20;5;240 240 240)
CloseDrawing()
```



**AddOval(links;oben;breite;höhe;füllfarbe;füllmuster)**

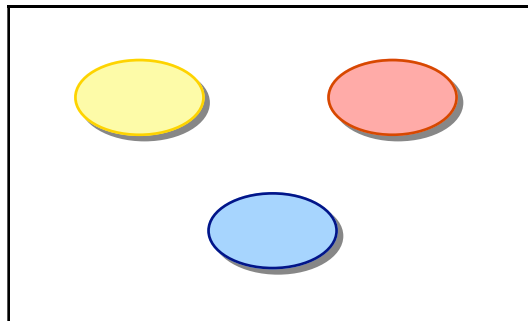
Die Funktion AddOval( ) ermöglicht die Erstellung ovaler Flächen, wobei die Farbe und das Muster variiert werden können. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)

  OpenView(25;20;52;32) // links oben
    AddOval(2;2;50;30;gray)
    AddOval(0;0;50;30;lightYellow)
    AddEllipse(0;0;50;30;1;darkYellow)
  CloseView()

  OpenView(120;20;52;32) // rechts oben
    AddOval(2;2;50;30;gray)
    AddOval(0;0;50;30;lightRed)
    AddEllipse(0;0;50;30;1;darkRed)
  CloseView()

  OpenView(75;70;53;33) // unten Mitte
    AddOval(2;2;50;30;gray)
    AddOval(0;0;50;30;lightBlue)
    AddEllipse(0;0;50;30;1;darkBlue)
  CloseView()
CloseDrawing()
```



**AddArc(links;oben;breite;höhe;startwinkel;  
öffnungswinkel;strichstärke;farbe;muster)**

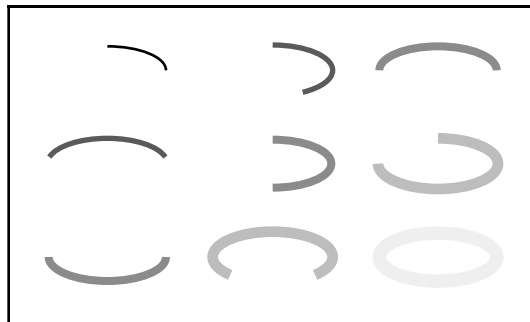
Die Funktion AddArc() ermöglicht die Erstellung ovaler Bögen, wobei die Strichstärke, die Farbe und das Muster variiert werden können. Start- und Öffnungswinkel sind im Bereich von  $\pm 360$  Grad anzugeben; dabei ist *startwinkel*=0 oben (12 Uhr), *startwinkel*=90 rechts (3 Uhr) und *startwinkel*=-90 links (9 Uhr). Ist der Öffnungswinkel positiv, wird der Bogen im Uhrzeigersinn aufgetragen, ist der Öffnungswinkel negativ, wird der Bogen gegen den Uhrzeigersinn aufgetragen. Beispiel:

```
OpenDrawing(200;120)
AddFrame(0;0;200;120)

// linke Spalte
AddArc(15;15;46;20; 0; 90;1; 0 0 0)
AddArc(15;50;46;20;-75;150;2; 90 90 90)
AddArc(15;85;46;20; 90;180;3;140 140 140)

// mittlere Spalte
AddArc(77;15;46;20; 0;150;2; 90 90 90)
AddArc(77;50;46;20; 0;180;3;140 140 140)
AddArc(77;85;46;20;-135;270;4;190 190 190)

// rechte Spalte
AddArc(139;15;46;20;-90;180;3;140 140 140)
AddArc(139;50;46;20; 0;270;4;190 190 190)
AddArc(139;85;46;20; 0;360;5;240 240 240)
CloseDrawing()
```



**AddSlice(links;oben;breite;höhe;startwinkel;  
öffnungswinkel;innenradius;füllfarbe;füllmuster)**

Die Funktion AddSlice() dient zur Erstellung oval begrenzter Sektoren, wobei die Farbe und das Muster variiert werden können. Die Erklärung der Argumente *startwinkel* und *öffnungswinkel* findet sich bei der Funktion AddArc(). Das Argument *innenradius* ermöglicht die Darstellung von ringförmigen Sektoren. Der Innenradius ist dabei in Prozent der Halbachse — zwischen 0 (default) und 100 — einzugeben. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)

  OpenView(25;20;52;32) // links oben
    AddSlice(2;2;50;30;-135;270;;gray)
    AddSlice(0;0;50;30;-135;270;;lightYellow)
    AddArc(0;0;50;30;-135;270;1;darkYellow)
  CloseView()

  OpenView(120;20;52;32) // rechts oben
    AddSlice(2;2;50;30;-90;180;50;gray)
    AddSlice(0;0;50;30;-90;180;50;lightRed)
    AddArc(0;0;50;30;-90;180;1;darkRed)
  CloseView()

  OpenView(75;70;53;33) // unten Mitte
    AddSlice(2;2;50;30;45;270;;gray)
    AddSlice(0;0;50;30;45;270;;lightBlue)
    AddArc(0;0;50;30;45;270;1;darkBlue)
  CloseView()
CloseDrawing()
```



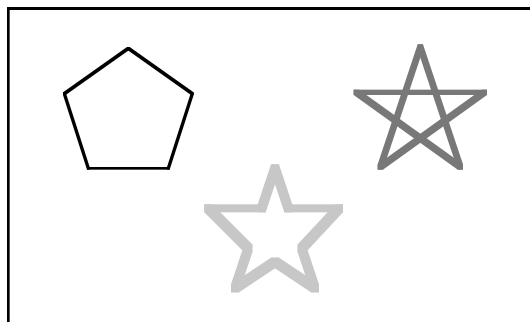
**AddPolyline(koordReihenfolge;koordListe;strichstärke;  
farbe;muster)**

Die Funktion AddPolyline( ) ermöglicht die Erstellung von Linienzügen, wobei die Strichstärke, die Farbe und das Muster variiert werden können. Das Argument *koordReihenfolge* ermöglicht die Eingabe der Koordinaten auf zwei Arten: Entweder werden zuerst alle x-Koordinaten und dann alle y-Koordinaten angeführt (*koordReihenfolge=xyxy*) oder es werden die x- und y-Koordinaten des 1. Polygonpunktes angeführt, dann die x- und y-Koordinaten des 2. Polygonpunktes, usw. (*koordReihenfolge=xxyy*). Alle Koordinatenwerte sind getrennt durch Leerzeichen, Tabs oder Zeilenumbrüche einzugeben. Die folgenden zwei Funktionen definieren den gleichen Linienzug:

```
(1) AddPolyline(xxyy;45 69 60 30 21 45
                  15 32 60 60 32 15)
(2) AddPolyline(xyxy;45 15
                  69 32
                  60 60
                  30 60
                  21 32
                  45 15)
```

Beispiel:

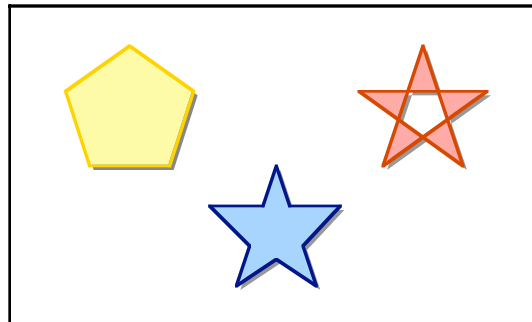
```
OpenDrawing(200;120)
AddFrame(0;0;200;120)
AddPolyline(xyxy;45 15 69 32 60 60 30 60
              21 32 45 15; 1)
AddPolyline(xyxy;155 15 140 60 179 32 131 32
              170 60 155 15; 2;120 120 120)
AddPolyline(xyxy;100 60 95 75 75 75 90 90
              85 105 100 95 115 105 110 90
              124 75 105 75 100 60;
              3;200 200 200)
CloseDrawing()
```



**AddPolygon(koordReihenfolge;koordListe;füllfarbe;  
füllmuster)**

Die Funktion AddPolygon() ermöglicht die Darstellung polygonal begrenzter Flächen, wobei die Farbe und das Muster variiert werden können. Die Argumente *koordReihenfolge* und *koordListe* werden bei der Funktion AddPolyline() erklärt. Beispiel:

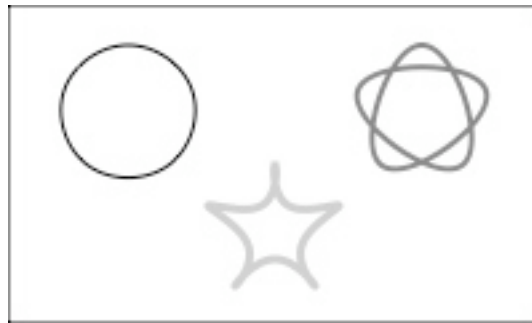
```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddPolygon(xyxy;47 17 71 34 62 62 32 62
             23 34 47 17; gray)
  AddPolygon(xyxy;45 15 69 32 60 60 30 60
             21 32 45 15; lightYellow)
  AddPolyline(xyxy;45 15 69 32 60 60 30 60
              21 32 45 15; 1; darkYellow)
  AddPolygon(xyxy;157 17 142 62 181 34 133 34
             172 62 157 17; gray)
  AddPolygon(xyxy;155 15 140 60 179 32 131 32
             170 60 155 15; lightRed)
  AddPolyline(xyxy;155 15 140 60 179 32 131 32
              170 60 155 15; 1; darkRed)
  AddPolygon(xyxy;102 62 97 77 77 77 92 92
             87 107 102 97 117 107 112 92
             126 77 107 77 102 62;
             gray)
  AddPolygon(xyxy;100 60 95 75 75 75 90 90
             85 105 100 95 115 105 110 90
             124 75 105 75 100 60;
             lightBlue)
  AddPolyline(xyxy;100 60 95 75 75 75 90 90
              85 105 100 95 115 105 110 90
              124 75 105 75 100 60;
              1; darkBlue)
CloseDrawing()
```



**AddSmoothPolyline(koordReihenfolge;koordListe;  
strichstärke;farbe;muster)**

Die Funktion AddSmoothPolyline() ermöglicht die Erstellung von geglätteten Linienzügen, wobei die Strichstärke, die Farbe und das Muster variiert werden können. Die Argumente *koordReihenfolge* und *koordListe* werden bei der Funktion AddPolyline() erklärt. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddSmoothPolyline(xyxy;45 15 69 32 60 60 30 60
                    21 32 45 15;1)
  AddSmoothPolyline(xyxy;155 15 140 60 179 32 131 32
                    170 60 155 15; 2;120 120 120)
  AddSmoothPolyline(xyxy;100 60 95 75 75 75 90 90
                    85 105 100 95 115 105 110 90
                    124 75 105 75 100 60;
                    3;200 200 200)
CloseDrawing()
```



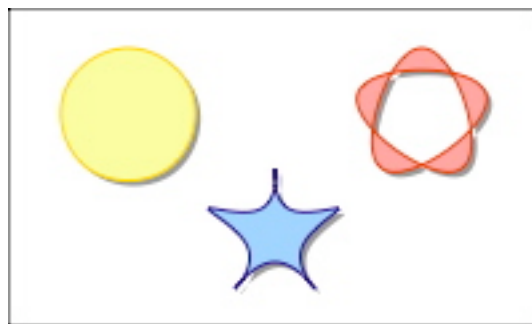
### **AddSmoothPolygon(koordReihenfolge;koordListe;füllfarbe; füllmuster)**

Die Funktion AddSmoothPolygon() ermöglicht die Darstellung von glatt berandeten Flächen, wobei die Farbe und das Muster variiert werden können. Die Argumente *koordReihenfolge* und *koordListe* werden bei der Funktion AddPolyline() erklärt. Beispiel:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddSmoothPolygon(xyxy;47 17 71 34 62 62 32 62
                    23 34 47 17; gray)
  AddSmoothPolygon(xyxy;45 15 69 32 60 60 30 60
                    21 32 45 15; lightYellow)
  AddSmoothPolyline(xyxy;45 15 69 32 60 60 30 60
                    21 32 45 15; 1;darkYellow)
  AddSmoothPolygon(xyxy;157 17 142 62 181 34 133 34
                    172 62 157 17; gray)
  AddSmoothPolygon(xyxy;155 15 140 60 179 32 131 32
                    170 60 155 15; lightRed)
  AddSmoothPolyline(xyxy;155 15 140 60 179 32 131 32
                    170 60 155 15; 1;darkRed)
  AddSmoothPolygon(xyxy;102 62 97 77 77 77 92 92
                    87 107 102 97 117 107 112 92
                    126 77 107 77 102 62; gray)
  AddSmoothPolygon(xyxy;100 60 95 75 75 75 90 90
                    85 105 100 95 115 105 110 90
                    124 75 105 75 100 60;
                    lightBlue)
  AddSmoothPolyline(xyxy;100 60 95 75 75 75 90 90
                    85 105 100 95 115 105 110 90
                    124 75 105 75 100 60;
                    1;darkBlue)
CloseDrawing()

```



**AddPath(pfadDaten;füllfarbe;füllmuster;rahmenbreite;  
rahmenfarbe;rahmenmuster;schattenabstand;  
schattenfarbe;schattenmuster)**

Die Funktion AddPath() ist eine sehr allgemeine und gleichzeitig äußerst mächtige Funktion zur Darstellung 2-dimensionaler Objekte. Durch das 1. Argument *pfadDaten* wird die Form des Objektes eindeutig festgelegt. Dazu stehen insgesamt 6 Pfadkonstanten (1...6) zur Verfügung. Unmittelbar anschließend an eine Pfadkonstante werden die dazu benötigten Werte angeführt. In der Regel sind das ein oder mehrere Koordinatenpaare.

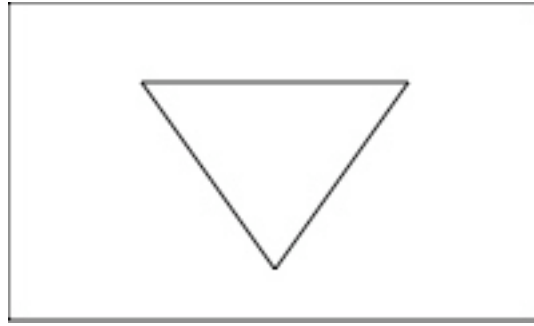
<i>Pfadkonstante</i>	<i>anschließende Werte</i>
1...Pfad schließen	(keine)
2...Versetzen nach	xPkt yPkt      z.B.: 2 100 100
3...Linie nach	xPkt yPkt      z.B.: 3 100 100
4...Quadrat. Bézier nach	xKontrollPkt yKontrollPkt xEndPkt yEndPkt z.B.: 4 50 20 100 100
5...Kub. Bézier nach	xKontrollPkt1 yKontrollPkt1 xKontrollPkt2 yKontrollPkt2 xEndPkt yEndPkt z.B.: 5 25 20 75 50 100 100
6...Ellipt. Bogen nach	xEndPkt yEndPkt xRadius yRadius x-Achsenneigung im Uhrzeigersinn [Grad] bogenFlag:      0...Winkel ≤ 180 Grad 1...Winkel > 180 Grad richtungsFlag: 0...gegen Uhrzeigersinn 1...im Uhrzeigersinn z.B.: 6 100 100 50 50 0 0 1

Pfadkonstanten und die dazugehörigen Werte sind durch Leerzeichen, Tabs oder Zeilenumbrüche zu trennen. Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddPath(2  50  30 // Versetzen nach 50 30
          3 150  30 // Linie nach 150 30
          3 100 100 // Linie nach 100 100
          1)       // Pfad schließen
CloseDrawing()

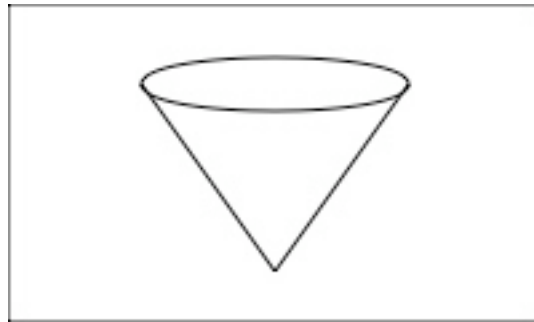
```



```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddPath(2  50  30                // Versetzen nach 50 30
          6 150  30 50 10 0 0 1 // Bogen nach 150 30
          6  50  30 50 10 0 0 1 // Bogen nach  50 30
          3 100 100                // Linie nach 100 100
          3 150  30)              // Linie nach 150 30
CloseDrawing()

```



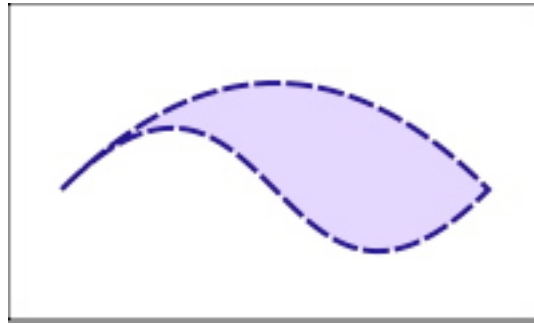
Die Argumente *füllfarbe*, *füllmuster*, *rahmenbreite*, *rahmenfarbe*, *rahmenmuster*, *schattenabstand*, *schattenfarbe* und *schattenmuster* ermöglichen Objekte optional mit Füllung, Berandung und Schatten darzustellen.

Beispiele:

```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddPath(2  20  70                // Versetzen nach 20 70
          4 100 -10 180  70 // Quad. Bézier nach 180 70
          5 100 150 100 -10 20 70; // Kub. Bézier 20 70
          50 0 255 50;;        // Füllung
          2 10 2;darkBlue)     // Berandung
CloseDrawing()

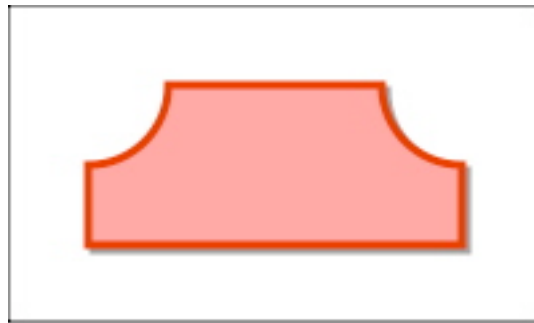
```



```

OpenDrawing(200;120)
AddFrame(0;0;200;120)
AddPath(2 30 90 // Versetzen nach 30 90
        3 30 60 // Linie nach 30 60
        6 60 30 30 30 0 0 0 // Bogen nach 60 30
        3 140 30 // Linie nach 140 30
        6 170 60 30 30 0 0 0 // Bogen nach 170 60
        3 170 90 // Linie nach 170 90
        1; // Pfad schließen
        255 155 155;; // Füllung
        3;darkRed;; // Berandung
        2;150 150 150) // Schatten
CloseDrawing()

```



Eine Pfaddefinition kann auch aus mehreren Teilpfaden aufgebaut sein, d.h. das Argument *pfadDaten* enthält mehrere geschlossene Pfade. Damit wird es möglich Objekte mit Löchern zu definieren.

Beispiel:

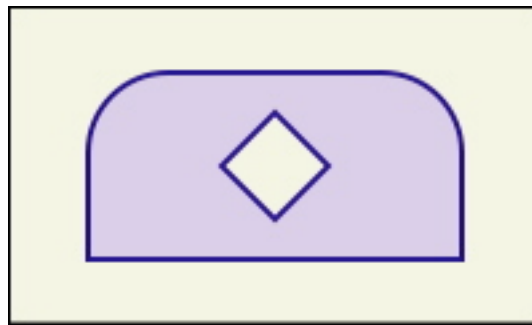
```

OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddPath(2 30 95          // Versetzen nach 30 95
          3 30 55          // Linie nach 30 55
          6 60 25 30 30 0 0 1 // Bogen nach 60 25
          3 140 25         // Linie nach 140 25
          6 170 55 30 30 0 0 1 // Bogen nach 170 55
          3 170 95         // Linie nach 170 95
          1                // Äußeren Pfad schließen

          // Raute ausstanzen
          2 100 40          // Versetzen nach 100 40
          3 120 60          // Linie nach 120 60
          3 100 80          // Linie nach 100 80
          3 80 60           // Linie nach 80 60
          1;                // Inneren Pfad schließen

          50 0 255 50;;     // Füllung
          2;darkBlue)       // Berandung
  Background(242 242 222)
CloseDrawing()

```



**AddSymbol(hPosition;vPosition;symboltyp;symbolgröße;  
strichstärke;farbe;muster)**

Die Funktion AddSymbol() ermöglicht die Darstellung von Symbolen, wobei die Symbolgröße, die Strichstärke, die Farbe und das Muster variiert werden können. Insgesamt stehen zur Zeit 18 Symbole zur Verfügung. Ein Überblick über die vordefinierten Symbole ist in *xmReferenz* zu finden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)

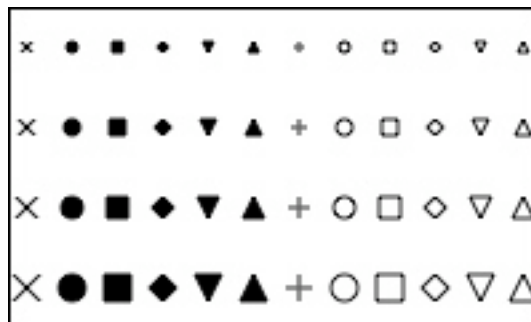
  // 1.Zeile
  AddSymbol( 7;15; 1;4)
  AddSymbol( 24;15; 2;4)
  AddSymbol( 41;15; 3;4)
  AddSymbol( 58;15; 4;4)
  AddSymbol( 75;15; 5;4)
  AddSymbol( 92;15; 6;4)
  AddSymbol(109;15; 7;4)
  AddSymbol(126;15; 8;4)
  AddSymbol(143;15; 9;4)
  AddSymbol(160;15;10;4)
  AddSymbol(177;15;11;4)
  AddSymbol(193;15;12;4)

  // 2.Zeile
  AddSymbol( 7;45; 1;6)
  AddSymbol( 24;45; 2;6)
  AddSymbol( 41;45; 3;6)
  AddSymbol( 58;45; 4;6)
  AddSymbol( 75;45; 5;6)
  AddSymbol( 92;45; 6;6)
  AddSymbol(109;45; 7;6)
  AddSymbol(126;45; 8;6)
  AddSymbol(143;45; 9;6)
  AddSymbol(160;45;10;6)
  AddSymbol(177;45;11;6)
  AddSymbol(193;45;12;6)
```

```
// 3.Zeile
AddSymbol( 7;75; 1;8)
AddSymbol( 24;75; 2;8)
AddSymbol( 41;75; 3;8)
AddSymbol( 58;75; 4;8)
AddSymbol( 75;75; 5;8)
AddSymbol( 92;75; 6;8)
AddSymbol(109;75; 7;8)
AddSymbol(126;75; 8;8)
AddSymbol(143;75; 9;8)
AddSymbol(160;75;10;8)
AddSymbol(177;75;11;8)
AddSymbol(193;75;12;8)

// 4.Zeile
AddSymbol( 7;105; 1;10)
AddSymbol( 24;105; 2;10)
AddSymbol( 41;105; 3;10)
AddSymbol( 58;105; 4;10)
AddSymbol( 75;105; 5;10)
AddSymbol( 92;105; 6;10)
AddSymbol(109;105; 7;10)
AddSymbol(126;105; 8;10)
AddSymbol(143;105; 9;10)
AddSymbol(160;105;10;10)
AddSymbol(177;105;11;10)
AddSymbol(193;105;12;10)

CloseDrawing()
```



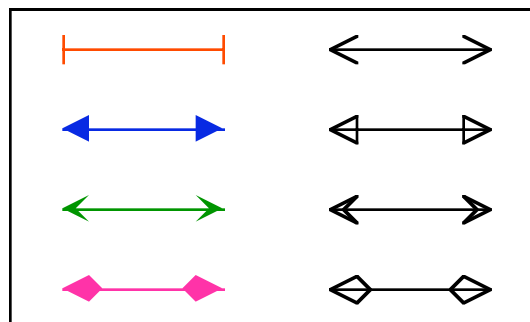
```
AddArrow(hStart;vStart;hEnde;vEnde;strichstärke;farbe;  
muster;pfeilposition;pfeillänge;pfeilbreite;  
pfeilkerbe;istPfeilHohl)
```

Die Funktion `AddArrow()` ermöglicht die Darstellung von Pfeilen. Dabei können sowohl die Strichstärke, die Farbe, das Muster und die Pfeilspitze variiert werden. Weiters kann durch das Argument *pfeilposition* die Pfeilspitze am Ende (default), am Anfang oder auch beidseitig angeordnet werden. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)

  AddArrow( 20; 15; 80; 15;1;red;;begin+end;0;10;0)
  AddArrow(120; 15;180; 15;1;;;begin+end;10;10;10)
  AddArrow( 20; 45; 80; 45;1;blue;;begin+end;10;10)
  AddArrow(120; 45;180; 45;1;;;begin+end;10;10;0;on)
  AddArrow( 20; 75; 80; 75;1;green;;begin+end;10;10;5)
  AddArrow(120; 75;180; 75;1;;;begin+end;10;10;5;on)
  AddArrow( 20;105;80;105;1;purple;;begin+end;10;10;-5)
  AddArrow(120;105;180;105;1;;;begin+end;10;10;-5;on)

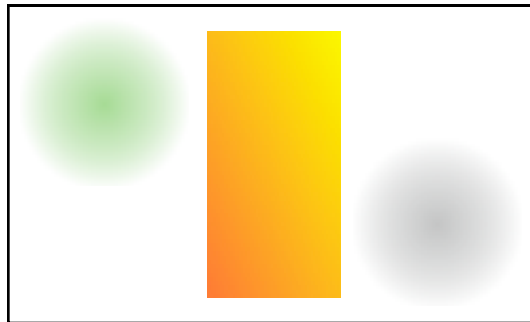
CloseDrawing()
```



**AddPicture(links;oben;breite;höhe;quelle;name;  
platzierung;angleichung;istProportional)**

Mittels der Funktion AddPicture() ist es möglich, eine Zeichnung durch ein Bild wie z.B. einem Firmenlogo zu ergänzen. Dabei werden drei verschiedene Bildquellen unterstützt: die Zwischenablage (*quelle=clipboard*), eine Datei (*quelle=file*) oder eine Resource (*quelle=resource*). Die Argumente *quelle* und *name* werden ausführlich im Zuge der Funktion PictureStyle() im Abschnitt *Stile* erläutert. Die linke obere Ecke des Bildes wird entsprechend der Argumente *links* und *oben* platziert. Der Koordinatenursprung befindet sich in der linken, oberen Ecke der Zeichnung bzw. des Views. Die Argumente *breite* und *höhe* dienen zum Skalieren des Bildes. Wird die Breite bzw. die Höhe nicht vorgegeben, so wird zur Darstellung die tatsächliche Breite bzw. Höhe des Bildes verwendet. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddPicture(75;10;50;100;resource;"32")
  AddPicture( 5; 5;;;resource;"37")
  AddPicture(130;50;;;resource;"33")
CloseDrawing()
```



Die Argumente *platzierung*, *angleichung* und *istProportional* werden im Abschnitt *Hintergrund*, Funktion BackgroundPict() erklärt.

Die Sichtbarkeit der Basisfunktionen kann durch Anführen sogenannter Clippingfunktionen auf eine bestimmte Fläche (Clippingfläche) eingeschränkt werden.

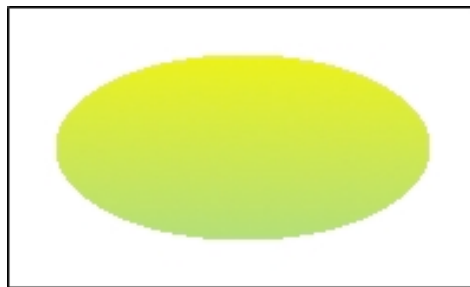
*Zur Zeit werden die Clippingfunktionen nur unter Windows OS unterstützt.*

Die folgenden Funktionen können zur Festlegung der Clippingfläche herangezogen werden:

```
AddClipRect(typ; links; oben; breite; höhe)  
AddClipRoundRect(typ; links; oben; breite; höhe; hKrümmung;  
                   vKrümmung)  
AddClipOval(typ; links; oben; breite; höhe)  
AddClipSlice(typ; links; oben; breite; höhe; startwinkel;  
              öffnungswinkel; innenradius)  
AddClipPolygon(typ; koordReihenfolge; koordListe)  
AddClipSmoothPolygon(typ; koordReihenfolge; koordListe)
```

Der Aufbau der Clippingfunktionen und die Bedeutung der Argumente sind identisch mit den Basisfunktionen. Das Argument *typ* ermöglicht die Kombination mehrerer Clippingfunktionen. Standardmäßig wird jede Clippingfunktion mit der aktuellen Clippingfläche geschnitten (*typ=sect*), d.h. die neue Clippingfläche kann nie größer werden als die zuvor vorhandene Clippingfläche. Beispiel:

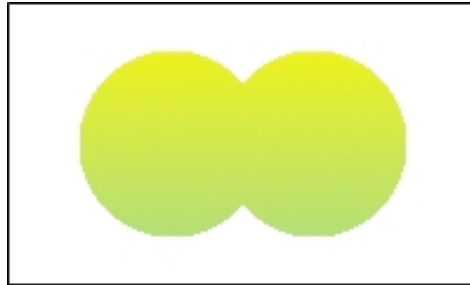
```
OpenDrawing(200;120)  
  AddFrame(0;0;200;120)  
  AddClipOval(sect;20;20;160;80)  
  AddPicture(0;0;200;120;resource;"7")  
CloseDrawing()
```



Clippingflächen werden addiert, falls *typ=union* gesetzt wird; das heißt die vorhandene Clippingfläche wird um die durch die Clippingfunktion definierte Fläche erweitert.

Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddClipOval(sect;30;20;80;80)
  AddClipOval(union;90;20;80;80)
  AddPicture(0;0;200;120;resource;"7")
CloseDrawing()
```



Falls *typ=diff* gesetzt wird, ergibt sich die neue Clippingfläche aus der Differenz zwischen der vorhandenen Clippingfläche und der durch die Clippingfunktion definierten Fläche. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  AddClipOval(sect;60;20;80;80)
  AddClipOval(diff;90;20;80;80)
  AddPicture(0;0;200;120;resource;"7")
CloseDrawing()
```



### **AddClipReset()**

Die Funktion `AddClipReset()` setzt die Clippingfläche auf die Breite und Höhe der Zeichnung zurück oder, falls die Funktion `AddClipReset()` innerhalb eines Views angeführt wird, auf die Breite und Höhe des Views.

## Ausgabe

Ausgabefunktionen ermöglichen das Sichern einer Zeichnung in eine Datei. Dabei stehen zur Zeit die folgenden Formate zur Verfügung:

- Mac OS X: PDF, PICT, JPEG, PNG, BMP, TIFF
- Windows: EMF, GIF, JPEG, PNG, BMP, TIFF

PDF und EMF-Format sind Vektorformate mit hoher Druckqualität, im Gegensatz zu GIF, JPG, PNG, BMP und TIFF-Format. Letztere stellen reine Bitmap-Formate dar, mit geringer Auflösung (z.B. 72dpi) und können plattformübergreifend verwendet werden.

Wird keine Ausgabefunktion angeführt, so wird standardmäßig die Zeichnung in ein FileMaker Pro Medienfeld kopiert. Beispiel:

```
OpenDrawing(200;120)
  AddFrame(0;0;200;120)
  ChartData(3 2 1 5 3)
  BarChart()
CloseDrawing()
```

Alle Ausgabefunktionen können an beliebiger Stelle, das heißt sowohl innerhalb als auch außerhalb der Funktionen `OpenDrawing()` und `CloseDrawing()` angeführt werden.

### **SendToClipboard()**

Die Funktion `SendToClipboard()` besitzt keine Argumente und wird dann angeführt, wenn eine Zeichnung in die Zwischenablage kopiert werden soll.

## Beispiel 1:

```
// Die Zeichnung wird sowohl in ein FileMaker Pro
// Medienfeld als auch in die Zwischenablage kopiert.
OpenDrawing(200;120)
    AddFrame(0;0;200;120)
    ChartData(3 2 1 5 3)
    BarChart()
CloseDrawing()
SendToClipboard()
```

## Beispiel 2:

```
// Die Zeichnung wird sowohl in ein FileMaker Pro
// Medienfeld als auch in eine PNG Datei gesichert
// und auch noch in die Zwischenablage kopiert.
OpenDrawing(200;120)
    AddFrame(0;0;200;120)
    ChartData(3 2 1 5 3)
    BarChart()
CloseDrawing()
SaveAsPNGFile("Zeichnung.png")
SendToClipboard()
```

**SaveAsPDFFile(dateiname;dateiflag;dateityp)**

Die Funktion `SaveAsPDFFile()` steht zur Zeit nur unter Mac OS X zur Verfügung und ermöglicht das Sichern einer Zeichnung in eine PDF Datei. Unter Windows wird die Funktion `SaveAsPDFFile()` ignoriert. Mit dem ersten Argument *dateiname* wird der Name der Datei inklusive optionalem Dateipfad definiert. Als Dateipfad-Trennzeichen ist ein Schrägstrich "/" zu verwenden, *ohne* zusätzliche Leerzeichen vor und nach dem Schrägstrich. Zum Beispiel:

```
SaveAsPDFFile("Macintosh HD/Bilder/Zeichnung.pdf")
```

Es kann entweder der vollständige, absolute Dateipfad oder nur ein relativer Pfad vorgegeben werden. Der relative Pfad bezieht sich auf den Ordner in welchem sich die aktuelle FileMaker Pro Datei befindet. Ist eine Datei mit dem gleichen Namen bereits vorhanden, so stehen mit dem zweiten Argument *dateiflag* folgende Optionen zur Verfügung:

- *dateiflag=addCounter*: (default)  
Standardmäßig wird ein Zähler an den Dateinamen angefügt, z.B. Diagramm-1.pdf, Diagramm-2.pdf, usw.
- *dateiflag=replace*:  
Die bereits vorhandene Datei wird überschrieben.
- *dateiflag=throwError*:  
Abbruch mit der Fehlermeldung *"Datei existiert bereits"*.

Mit dem dritten Argument *dateityp* kann optional unter Mac OS X noch der Dateityp (file creator) definiert werden. Damit wird festgelegt, mit welcher Applikation die Datei standardmäßig, d.h. beim Doppelklicken auf die Datei, geöffnet werden soll. Standardmäßig ist kein Dateityp (*dateityp*="????") vorgegeben. Unter Windows wird das Argument *dateityp* ignoriert. Beispiele:

```
SaveAsPDFFile("Diagramm.pdf")
SaveAsPDFFile("Diagramme/Chart.pdf";replace)
SaveAsPDFFile("Macintosh HD/Diagramme/Plot.pdf")
SaveAsPDFFile("Plot.pdf";;"8BIM") // Sichern als
                                   // "Photoshop"-Datei.
```

### **SaveAsEMFFile(dateiname;dateiflag)**

Die Funktion *SaveAsEMFFile()* steht nur unter Windows zur Verfügung und ermöglicht das Sichern einer Zeichnung in eine EMF-Datei (Enhanced Metafile Format). Unter Mac OS X wird die Funktion *SaveAsEMFFile()* ignoriert. Mit dem ersten Argument *dateiname* wird der Name der Datei inklusive optionalem Dateipfad definiert. Als Dateipfad-Trennzeichen ist ein Schrägstrich "/" zu verwenden, *ohne* zusätzliche Leerzeichen vor und nach dem Schrägstrich. Zum Beispiel:

```
SaveAsEMFFile("C:/Programme/Plots/Plot-1.emf")
```

Es kann entweder der vollständige, absolute Dateipfad oder nur ein relativer Pfad vorgegeben werden. Der relative Pfad bezieht sich auf den Ordner in welchem sich die aktuelle FileMaker Pro Datei befindet. Ist eine Datei mit dem gleichen Namen bereits vorhanden, so stehen mit dem zweiten Argument *dateiflag* folgende Optionen zur Verfügung:

- *dateiflag=addCounter:* (default)  
Standardmäßig wird ein Zähler an den Dateinamen angefügt, z.B. Diagramm-1.emf, Diagramm-2.emf, usw.
- *dateiflag=replace:*  
Die bereits vorhandene Datei wird überschrieben.
- *dateiflag=throwError:*  
Abbruch mit der Fehlermeldung "*Datei existiert bereits*".

Beispiele:

```
SaveAsEMFFile("Diagramm.emf")
SaveAsEMFFile("Diagramme/Chart.emf";replace)
SaveAsEMFFile("C:/Programme/FileMaker/Grafik.emf")
```

### **SaveAsPICTFile(dateiname;dateiflag;dateityp)**

Die Funktion *SaveAsPICTFile()* steht nur unter Mac OS X zur Verfügung und ermöglicht das Sichern einer Zeichnung in eine PICT Datei. Unter

Windows wird die Funktion `SaveAsPICTFile()` ignoriert. `SaveAsPICTFile()` ist ident aufgebaut wie die Funktion `SaveAsPDFFile()`.

Zum Beispiel:

```
SaveAsPICTFile("Diagramm.pct")
SaveAsPICTFile("Diagramme/Chart.pct";replace)
SaveAsPICTFile("Macintosh HD/Diagramme/Plot.pct")
SaveAsPICTFile("Plot.pct";;"8BIM") // Sichern als
                                   // "Photoshop"-Datei.
```

### **SaveAsGIFFile(dateiname;dateiflag)**

Die Funktion `SaveAsGIFFile()` steht zur Zeit nur unter Windows zur Verfügung und ist ident aufgebaut wie die Funktion `SaveAsEMFFile()`.

Zum Beispiel:

```
SaveAsGIFFile("Diagramm.gif")
SaveAsGIFFile("Diagramme/Chart.gif";replace)
SaveAsGIFFile("C:/Programme/FileMaker/Grafik.gif")
```

### **SaveAsJPGFile(dateiname;dateiflag;dateityp;qualität)**

Die Funktion `SaveAsJPGFile()` steht sowohl unter Mac OS X als auch unter Windows zur Verfügung und ist ident aufgebaut wie die Funktion `SaveAsPDFFile()`. Zusätzlich kann mit dem 4. Argument *qualität* die Bildqualität der JPEG-Datei kontrolliert werden. Dazu stehen die folgenden fünf Konstanten zur Verfügung:

<i>Konstante</i>	<i>Wert</i>	<i>Anmerkung</i>
min	1	
low	2	
normal	3	(default)
high	4	
max	5	

Hohe Bildqualität bedeutet größere Dateien zufolge geringerer Komprimierung und umgekehrt, niedrige Bildqualität führt zu kleineren Dateien bei starker Komprimierung der Bildinformation. Wenn nicht anders vorgegeben, wird die erstellte JPEG-Datei mit *qualität=normal* weggeschrieben. Beispiele:

```
SaveAsJPGFile("Chart1.jpg";replace)
SaveAsJPGFile("Diagramme/Diagramm.jpg";;;high)
SaveAsJPGFile("Macintosh HD/Diagramme/Chart.jpg")
SaveAsJPGFile("C:/Programme/Plots/Plot1.jpg";;;max)
```

**SaveAsPNGFile(dateiname;dateiflag;dateityp)**

Die Funktion SaveAsPNGFile() steht sowohl unter Mac OS X als auch unter Windows zur Verfügung und ist ident aufgebaut wie die Funktion SaveAsPDFFile(). Zum Beispiel:

```
SaveAsPNGFile("Diagramm.png")
SaveAsPNGFile("Diagramme/Chart.png";replace)
SaveAsPNGFile("Macintosh HD/Plots/Grafik.png")
SaveAsPNGFile("C:/Programme/Plots/Grafik.png")
```

**SaveAsBMPFile(dateiname;dateiflag;dateityp)**

Die Funktion SaveAsBMPFile() steht sowohl unter Windows als auch unter Mac OS X zur Verfügung und ist ident aufgebaut wie die Funktion SaveAsPDFFile().

Beispiele:

```
SaveAsBMPFile("Diagramm.bmp")
SaveAsBMPFile("Diagramme/Chart.bmp";replace)
SaveAsBMPFile("Macintosh HD/Plots/Grafik.bmp")
SaveAsBMPFile("C:/Programme/Plots/Grafik.bmp")
```

**SaveAsTIFFFile(dateiname;dateiflag;dateityp)**

Die Funktion SaveAsTIFFFile() steht sowohl unter Windows als auch unter Mac OS X zur Verfügung und ist ident aufgebaut wie die Funktion SaveAsPDFFile().

Beispiele:

```
SaveAsTIFFFile("Diagramm.tif")
SaveAsTIFFFile("Diagramme/Chart.tif";replace)
SaveAsTIFFFile("Macintosh HD/Plots/Grafik.tif")
SaveAsTIFFFile("C:/Programme/Plots/Grafik.tif")
```

## Diverses

### **DateTimeOptions(datumsschreibweise;wochenbeginn)**

Durch das erste Argument *datumsschreibweise* kann die Reihenfolge von Tag, Monat und Jahr bei der Eingabe von Datumswerten vorgegeben werden. Die Funktion ist vor `ChartData()` bzw. vor `ChartDataRead()` anzuführen. Da `DateTimeOptions()` eine "globale" Funktion ist, ist es zweckmäßig diese gleich am Skriptanfang, d.h. unmittelbar nach `OpenDrawing()` anzuführen. Beispiel:

```
OpenDrawing(...)
  DateTimeOptions(...)
  OpenChart(...)
    ... // Diagramm 1
  CloseChart()
  OpenChart(...)
    ... // Diagramm 2
  CloseChart()
CloseDrawing()
```

Weitere Informationen und Beispiele zur Datumsschreibweise finden sich im Abschnitt *Daten*, Kapitel *Eingabe von Datum und Uhrzeit*.

Mit dem zweiten Argument *wochenbeginn* kann der Beginn der Kalenderwoche durch einen Wert zwischen 1 und 7 festgelegt werden. Dabei ist 1=Sonntag (default), 2=Montag, 3=Dienstag usw.

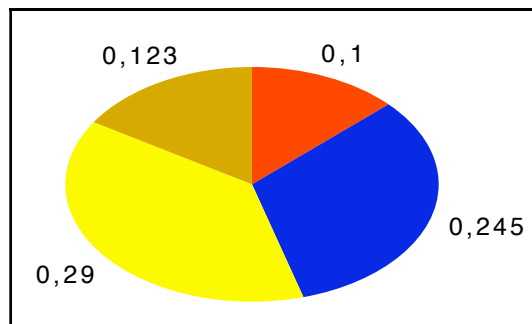
Beispiele:

```
DateTimeOptions(mdy;1) // US Datumsformat
DateTimeOptions(;2) // Wochenbeginn am Montag (ISO 8601)
```

**SetDecimalPoint(zeichen)**

Durch die Funktion `SetDecimalPoint()` kann das Dezimalpunkt-Zeichen kontrolliert werden. `SetDecimalPoint()` ist eine "globale" Funktion, d.h. alle in einer Zeichung dargestellten Dezimalzahlen sind davon betroffen, unabhängig ob es sich dabei um Achsenbeschriftungen, Beschriftungen von Diagrammwerten, etc. handelt, auch unabhängig von eventuell vorhandenen Formatanweisungen. Beispiel:

```
OpenDrawing(200;120)
  SetDecimalPoint(",")
  AddFrame(0;0;200;120)
  ChartData(0.1 0.245 0,29 0,123)
  PieChart(label+oval)
  BorderStyle(all;none)
CloseDrawing()
```



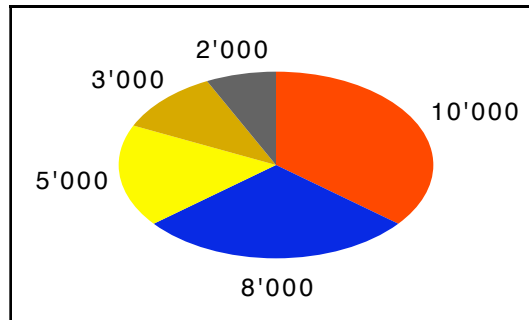
Die Eingabe von Dezimalzahlen wird durch die Funktion `SetDecimalPoint()` nicht beeinflusst, d.h. sowohl "." als auch "," werden automatisch bei der Eingabe von Dezimalzahlen via `ChartData()` als Dezimalpunkt erkannt.

**SetThousandsSep(zeichen)**

Die Funktion `SetThousandsSep()` eröffnet die Möglichkeit, Zahlen mit Tausender-Trennzeichen auszugeben. `SetThousandsSep()` ist eine "globale" Funktion, d.h. alle in einer Zeichung dargestellten Zahlen sind davon betroffen, unabhängig ob es sich dabei um Achsenbeschriftungen, Beschriftungen von Diagrammwerten, etc. handelt, auch unabhängig von eventuell vorhandenen Formatanweisungen.

Beispiel:

```
OpenDrawing(200;120)
  SetThousandsSep(" ")
  AddFrame(0;0;200;120)
  ChartData(10000 8000 5000 3000 2000)
  PieChart(label+oval)
  BorderStyle(all;none)
CloseDrawing()
```



Tausender-Trennzeichen sind bei der Eingabe von Zahlen nicht erlaubt — siehe Funktion `ChartData()` bzw. `ChartDataRead()` im Abschnitt *Daten*.

# Index

# Index

## A

Achsen	193
AddArc	294
AddArrow	306
AddClipOval	308
AddClipPolygon	308
AddClipRect	308
AddClipReset	309
AddClipRoundRect	308
AddClipSlice	308
AddClipSmoothPolygon	308
AddEllipse	292
AddFrame	288
AddLine	287
AddOval	293
AddPath	300
AddPicture	307
AddPolygon	297
AddPolyline	296
AddRect	289
AddRoundFrame	290
AddRoundRect	291
AddSlice	295
AddSmoothPolygon	299
AddSmoothPolyline	298
AddSymbol	304
AddText	284
AreaChart	47
AreaChart2D	54
AreaChartOptions	52
ArrowStyle	146
Ausgabe	310
AxisLabelBackground	213
AxisLabelOptions	215
AxisLabelStyle	212

AxisLabelText	209
AxisLine	205
AxisMajorTickLabelBackground	220
AxisMajorTickLabelOptions	221
AxisMajorTickLabelStyle	219
AxisMajorTickLabelTexts	217
AxisMajorTicks	205
AxisMinorTickLabelBackground	224
AxisMinorTickLabelOptions	224
AxisMinorTickLabelStyle	224
AxisMinorTickLabelTexts	224
AxisMinorTicks	205
AxisOptions	207
 <b>B</b>	
Background	169
BackgroundPict	173
BarChart	58
BarChartOptions	68
Betriebssystem	8
BorderStyle	138
BoxPlotOptions	127
BoxPlots	126
BubbleChart	75
BubbleChart2D	78
BubbleChartOptions	78
 <b>C</b>	
CandlestickChart	115
ChartBackground	177
ChartBackgroundPict	179
ChartData	27
ChartDataLowerLimits	30
ChartDataOptions	28
ChartDataRead	32
ChartDataUpperLimits	30
ChartDataWrite	33
CloseChart	26
CloseDrawing	21
CloseView	22
CurveFitting	267
CurveFittingLineStyle	269
CurveFittingOptions	271

**D**

Daten	27
DateTimeOptions	315
Datum und Uhrzeit	35
Diagramme	37
Diverses	315
DropLineReferenceLine	249
DropLineReferencePoint	247
DropLineReferenceSeries	250
DropLineStyle	245

**E**

Ein erstes Beispiel	13
ErrorBarData	277
ErrorBars	275
ErrorBarStyle	278
ErrorBarStyle2D	280

**F**

Fehlerbalken	275
FileMaker Pro Runtime	9
FileMaker Pro Version	8
FillStyle	134

**G**

GanttChart	70
Gleitende Durchschnitte	253
Grafische Basisfunktionen	283
GridFrame	190
GridLocation	191
Grundlagen	11

**H**

HighLowChart	108
Hilfslinien	245
Hintergrund	169
Histogram	118
HistogramOptions	121
HistogramRange	118

**K**

Kurvenanpassung	266
-----------------	-----

**L**

LabelBackground	156
-----------------	-----

LabelOptions	157
LabelStyle	151
LabelTexts	148
Layout	19
LegendBackground	230
Legende	228
LegendOptions	231
LegendStyle	229
LegendTexts	228
LineChart	42
LineChart2D	45
LineStyle	140
 <b>M</b>	
MajorGridLineColors	182
MajorGridLinePatterns	182
MajorGridLineWidths	182
MajorGridStripeColors	187
MajorGridStripePatterns	187
MinorGridLineColors	184
MinorGridLinePatterns	184
MinorGridLineWidths	184
MinorGridStripeColors	187
MinorGridStripePatterns	187
MovingAverage	253
MovingAverageLineStyle	255
MovingAverageOptions	257
 <b>O</b>	
OpenChart	22
OpenDrawing	19
OpenView	21
 <b>P</b>	
PictureStyle	136
PieChart	80
PieChartAuxLines	88
PieChartCenterLabelBackground	91
PieChartCenterLabelStyle	91
PieChartCenterLabelText	91
PieChartExplodeDepths	87
PieChartExplodes	85
PieChartInnerLabelBackground	90
PieChartInnerLabelStyle	90
PieChartInnerLabelTexts	90

PieChartLabelOptions	89
PolarChart	92
PolarChartOptions	96
<b>Q</b>	
Quartilen	132
<b>R</b>	
RadarChart	100
RadarChartOptions	104
Raster	182
<b>S</b>	
SaveAsBMPFile	314
SaveAsEMFFile	312
SaveAsGIFFile	313
SaveAsJPGFile	313
SaveAsPDFFile	311
SaveAsPICTFile	312
SaveAsPNGFile	314
SaveAsTIFFFile	314
Scaling	193
ScalingOptions	202
ScatterChart	37
ScatterChart2D	40
SendToClipboard	310
SetDecimalPoint	316
SetThousandsSep	316
ShadowStyle	144
Stile	134
Support	10
SymbolStyle	142
<b>T</b>	
Titel	239
TitleBackground	241
TitleOptions	242
TitleStyle	240
TitleSubStyle	240
TitleText	239
<b>U</b>	
Übersicht	18

**W**

Was ist xmCHART	11
Wie funktioniert xmCHART	11