6. Plots and Drawings

2016-04-018

```
$Version
```

```
10.0 for Mac OS X x86 (64-bit) (December 4, 2014)
```

6.1 Plotting Two - Dimensional Objects

6.1.1 Plotting Curves in Two - Dimensional Space. Plot[]

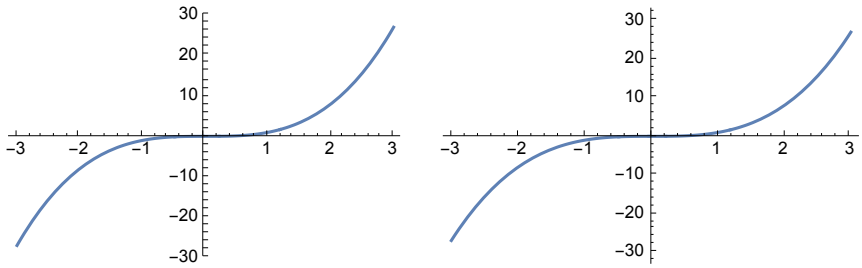
The general form of the Plot statement is :

Plot[f, {x, xmin, xmax}, Options] plot **f** as a function of **x** in the given range

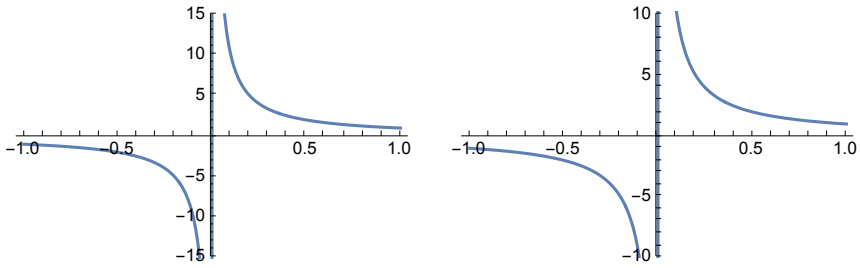
```
FullForm[p1]
```

```
p1
```

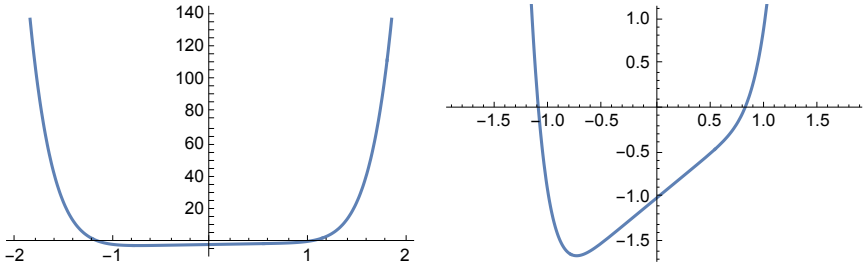
```
p1 = Plot[x^3, {x, -3, 3}];
p2 = Show[p1, PlotRange -> {-30, 30}];
GraphicsGrid[{{p1, p2}}, ImageSize → 450]
```




```
p1 = Plot[1 / x, {x, -1, 1}, PlotRange → {-15, 15} ];
p2 = Show[p1, PlotRange -> {-10, 10}];
Show[GraphicsRow[{p1, p2}], ImageSize → 450]
```



```
Clear[f, x]; f[x_] = x^8 + x - 1;
p1 = Plot[ f[y], {y, -2, 2}];
p2 = Show[p1, PlotRange -> {-1.6, 1} ];
Show[GraphicsRow[{p1, p2}], ImageSize → 450]
```



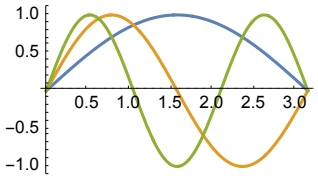
6.1.2 Plotting Several Curves in one Figure

The graphs of several functions f1[x], f2[x], may be represented in one figure. The general form of the statement is :

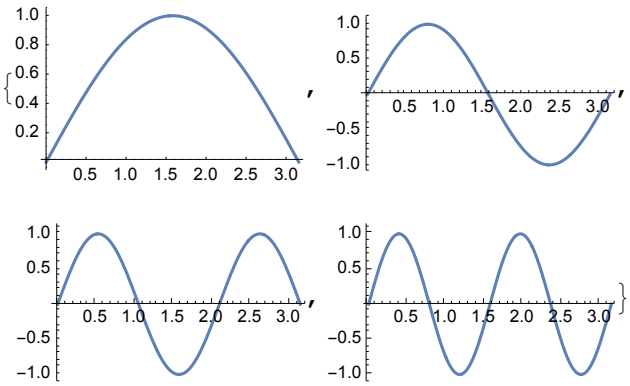
Plot[{f1[x], f2[x], ... }, {x, xmin, xmax}, Options]

The first argument of the statement must be a list of functions.

```
Plot[ {Sin[x], Sin[2 x], Sin[3 x]}, {x, 0, Pi}, ImageSize → 160 ]
```

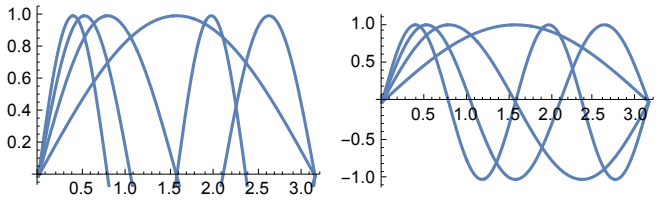



```
pp = Table[ Plot[Sin[n x], {x, 0, Pi}, ImageSize → 150], {n, 4}]
```



will produce figures for each of the harmonics.The following one contains all curves together {cf.end of § 5.2).

```
p1 = Show[pp, ImageSize → 160 ];
p2 = Show[Reverse[pp], ImageSize → 160];
Show[GraphicsRow[{p1, p2}]]
```



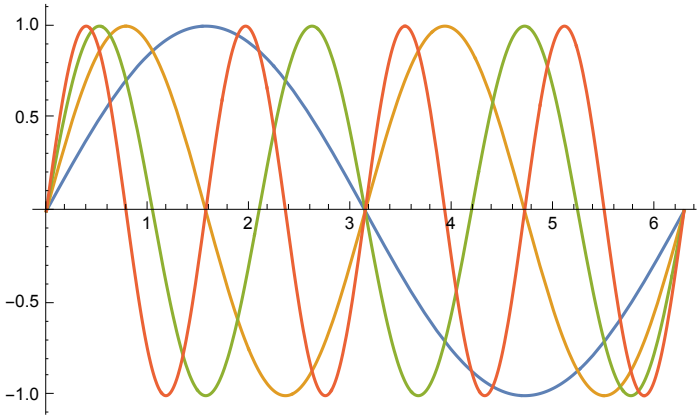
6.1.3 Plotting Lists of Functions

The list containing several functions to be plottend in one drawing may also be prepared by the command Table[].

```
pt = Table[ Sin[n x], {n, 4}]
```

```
{Sin[x], Sin[2 x], Sin[3 x], Sin[4 x]}
```

```
p1 = Plot[pt, {x, 0, 2 Pi}]
```



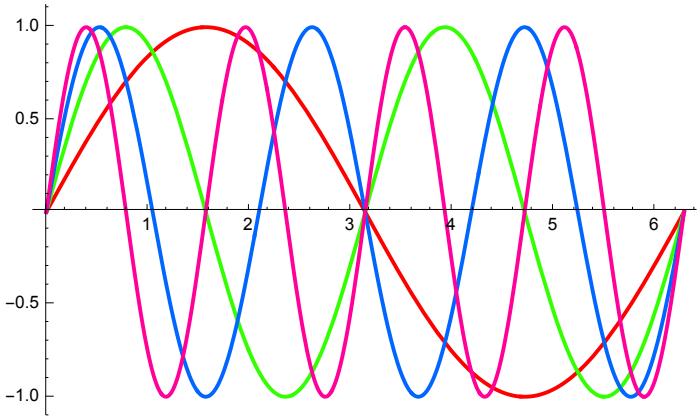

```
$Version
```

```
10.0 for Mac OS X x86 (64-bit) (December 4, 2014)
```

```
lc = Table[{Thick, Hue[k 0.1]}, {k, 0, 9, 3}]
```

```
{{Thickness[Large], ■}, {Thickness[Large], ■},
 {Thickness[Large], ■}, {Thickness[Large], ■}}
```

```
p1 = Plot[pt, {x, 0, 2 Pi}, PlotStyle → lc]
```

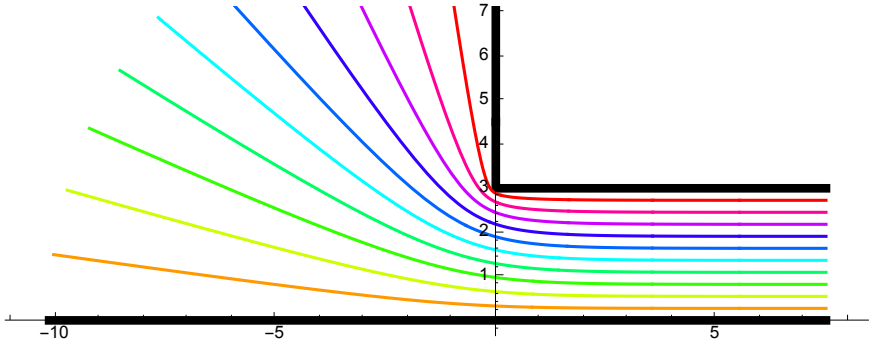


Now we show a mapping by a conformal map. The complex mapping function is:

```
af[z_] = 3/Pi (I (Pi + 2 Sqrt[Exp[z] - 1]) -
  Log[(1 + I Sqrt[Exp[z] - 1])/(1 - I Sqrt[Exp[z] - 1])]) ;

neq = 11 (* Number of equipotential lines - 1 *) ;
b = 3.37465 ; e = -8.46758 ;
```

Note a different syle for each curve:

```
styli = {{Thickness[.01], GrayLevel[0]},
          Hue[1.],Hue[.9],Hue[.8],Hue[.7],Hue[.6],
          Hue[.5],Hue[.4],Hue[.3],Hue[.2],Hue[.1],
         {Thickness[.01], GrayLevel[0]}};

fuli = Table[{Re[af[ k/neq Pi I + x]],
              Im[af[ k/neq Pi I + x]]}, {k,0,neq}];

ParametricPlot[fuli,{x,b,e}, PlotStyle -> styli,ImageSize -> 450]
```



The figure above shows the equipotential lines for a configuration where a voltage is applied to the metallic electrodes shown in black.

6.1.4 Plotting Real Branches of Functions Having Real and Complex Branches

Plot[] may run into difficulties in plotting functions having real and complex branches. These may be avoided in using the package: **RealOnly.m** . Miscellaneous`RealOnly` was available as an add-on package in previous versions of *Mathematica* and is now available on the web at library.wolfram.com/infocenter/MathSource/6771.

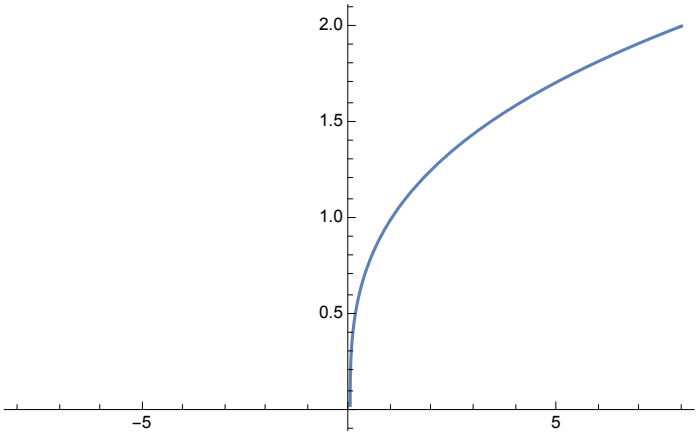
```
t = (-8)^(1/3)
```

$2\ (-1)^{1/3}$

```
N[%]
```

```
1. + 1.73205 i
```

```
p2 = Plot[x^(1/3), {x, -8, 8}]
```



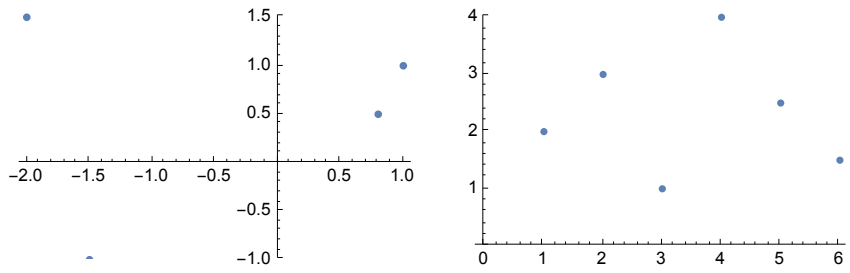
6.1.5 Plotting Points and Polygons: ListPlot[], ListLinePlot[]

General form of the Statement:

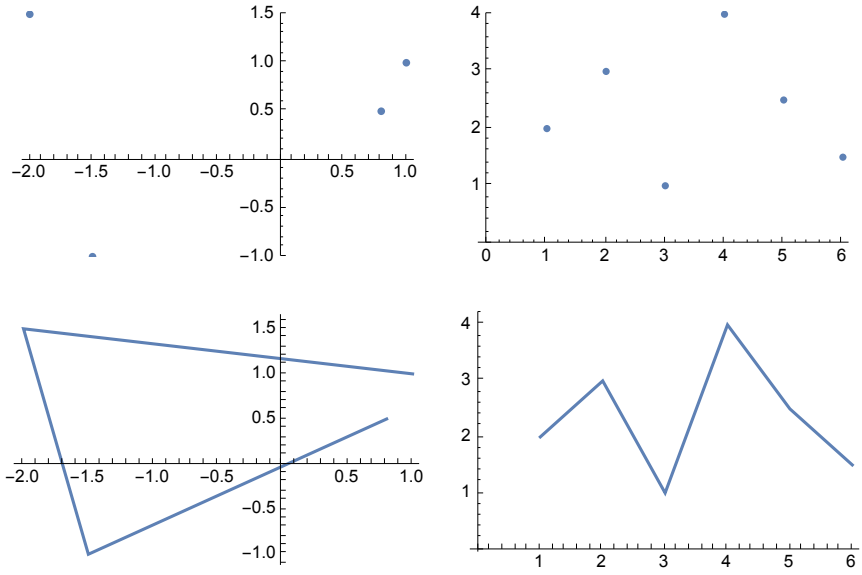
```
ListPlot[ list, Options  ]

ListLinePlot[List, Options]
```

list must be a list of points, i.e. a list of sublists giving points. So each sublist must contain a pair of numbers.
One may also submit a list of simple numbers; then these are used for the ordinates, while the corresponding position in the list serves a the abscissa. **ListLinePlot[]** connects adjoining points by straight lines. This can also be done by ListPlot[] with the option Joined → True.

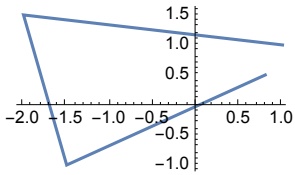
```
l1 = {{1, 1}, {-2, 1.5}, {-1.5, -1}, {.8, .5} };
p1 = ListPlot[ l1, PlotRange → {-1, 1.5} ];
l2 = {2, 3, 1, 4, 2.5, 1.5};
p2 = ListPlot[l2, PlotRange → {0, 4} ];
Show[GraphicsRow[{p1, p2}], ImageSize → 450]
```



```
ps1 = ListLinePlot[ l1];
ps2 = ListLinePlot[ l2];
Show[GraphicsRow[{p1,p2}], ImageSize -> 450]
Show[GraphicsRow[{ps1,ps2}], ImageSize -> 450]
```

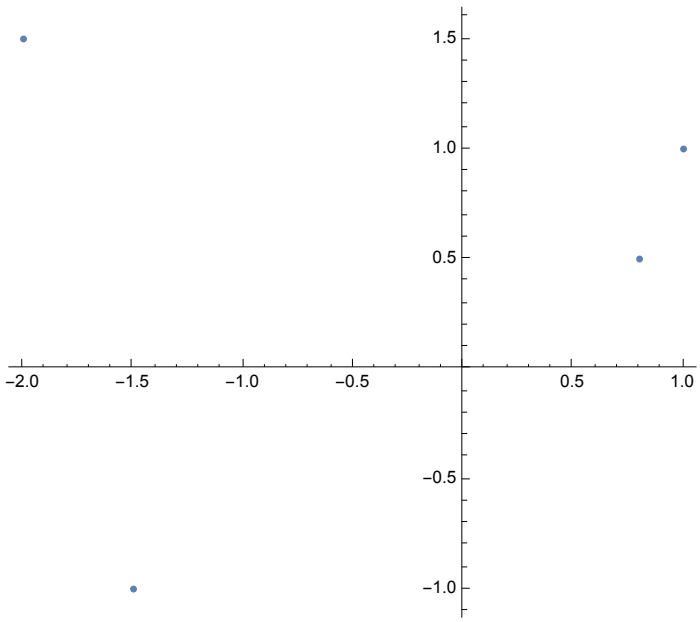



```
ListLinePlot[l1, ImageSize → 150]
```



The size of a point can be chosen by the option **PlotStyle -> PointSize[num].**
num gives the size as a fraction relative to the total width of the figure. For example, ***num* = 0.01** means the size of the points is 1 percent of the width of the figure. For other size commands see § 6.5.2.

```
ListPlot[l1, PlotStyle -> PointSize[0.01], AspectRatio -> Automatic ]
```



ListPlot[] may be used to show the disribution of roots of a function in the complex plane.

Since this command requires real coordinates, complex numbers must be transformed into pairs of real numbers as shown:

```
f = x^8 + x - 1;  s8 = Solve[f == 0.,x]
```

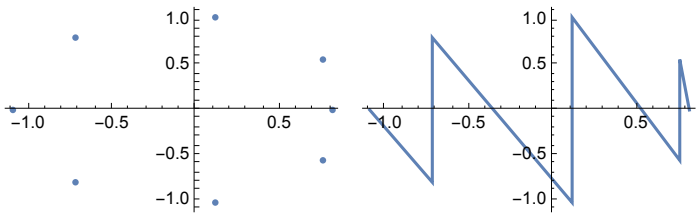
```
{{x → -1.09698}, {x → -0.723237 - 0.807112 i}, {x → -0.723237 + 0.807112 i},
 {x → 0.111621 - 1.03344 i}, {x → 0.111621 + 1.03344 i},
 {x → 0.75428 - 0.562241 i}, {x → 0.75428 + 0.562241 i}, {x → 0.811652}}
```

```
r8 = {Re[x],Im[x]} /. s8
```

```
{{-1.09698, 0}, {-0.723237, -0.807112},
 {-0.723237, 0.807112}, {0.111621, -1.03344}, {0.111621, 1.03344},
 {0.75428, -0.562241}, {0.75428, 0.562241}, {0.811652, 0}}
```

```
p1 = ListPlot[r8]; p2 = ListLinePlot[r8];
```

```
Show[GraphicsGrid[{{p1,p2}}]]
```

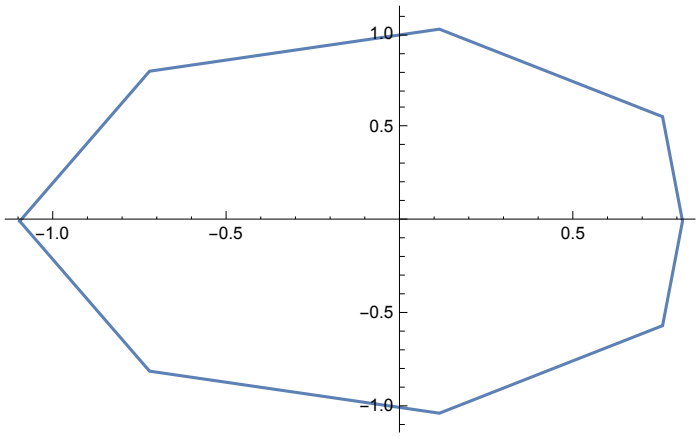


In order to get a closed polygon the order of the pairs in the list is rearranged by hand:

```
n8 = { r8[[1]],  r8[[2]], r8[[4]], r8[[6]], r8[[8]],
       r8[[7]],  r8[[5]], r8[[3]], r8[[1]] }
```

```
{{-1.09698, 0}, {-0.723237, -0.807112}, {0.111621, -1.03344},
 {0.75428, -0.562241}, {0.811652, 0}, {0.75428, 0.562241},
 {0.111621, 1.03344}, {-0.723237, 0.807112}, {-1.09698, 0}}
```

```
p8 = ListPlot[ n8,  Joined -> True]
```

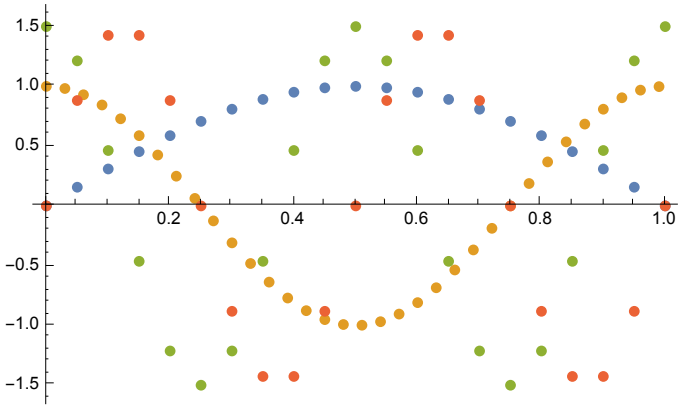


Automatic listing by criteria is treated in § 5.7.

6.1.5.1 Plotting points or polygons belonging to several curves

```
list1 = Table[{x, Sin[ Pi x]}, {x, 0, 1, 0.05}];
list2 = Table[{x, Cos[2 Pi x]}, {x, 0, 1, 0.03}];
list3 = Table[{x, 1.5 Cos[4 Pi x]}, {x, 0, 1, 0.05}];
list4 = Table[{x, 1.5 Sin[4 Pi x]}, {x, 0, 1, 0.05}];
ims = 350;
```

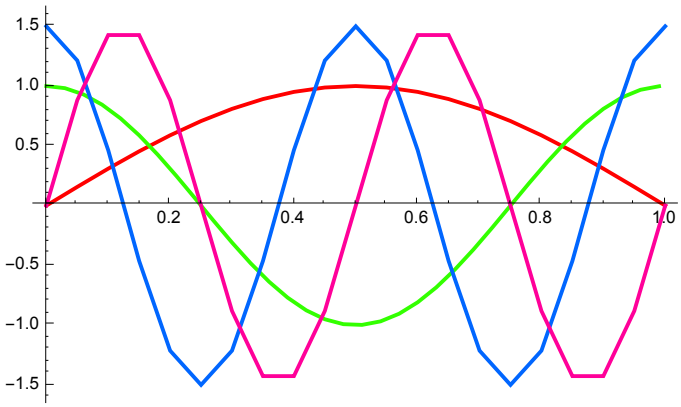
```
ListPlot[{list1, list2, list3, list4}, ImageSize → ims]
```




```
lc = Table[{Thick, Hue[k 0.1]}, {k, 0, 9, 3}]
```

```
{{Thickness[Large], ■}, {Thickness[Large], ■},
 {Thickness[Large], ■}, {Thickness[Large], ■}}
```

```
ListPlot[{list1, list2, list3, list4}, Joined → True,
PlotStyle → lc, ImageSize → ims]
```



6.1.6 Drawing Curves in Parametric Representation

General form of the statement:

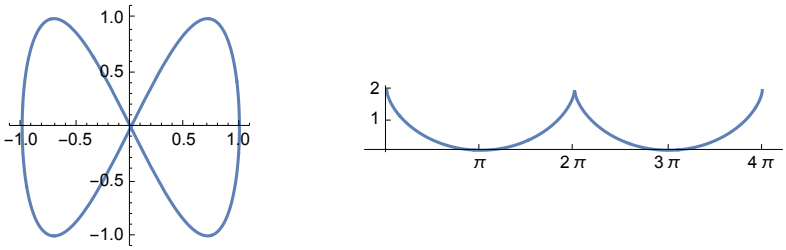
```
ParametricPlot[ {x[t], y[t]},  {t, tmin, tmax}, Options ]
```

The first argument of the command must be a list of two functions. This list may also be defined before `ParametricPlot[]` is called.

```
p1 = ParametricPlot[ {Sin[t], Sin[2t]}, {t, 0, 2 Pi} ];
```

```
Clear[x,y,t]; x = t - Sin[t]; y = 1 + Cos[t];
p2 = ParametricPlot[ {x,y}, {t,0, 4π},Ticks -> {π Range[0,4],Range[0,2]} ];
```

```
Show[GraphicsRow[{p1,p2}],ImageSize -> 450]
```



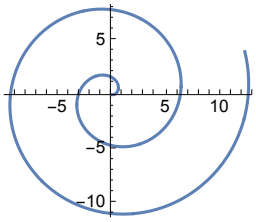
6.1.7 Plotting a Plane Curve in Polar Coordinates

Plane curves may be given in polar coordinates such that the radius **r** is a given function of the azimuth ϕ. Denoting the angle by **t** the general form of the command is:

```
PolarPlot[  r[t],  {t, tmin, tmax}, Options ]
```

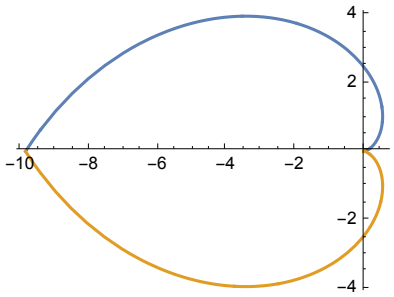
The first argument of the command may even be a list of functions.

```
r = t;
PolarPlot[r, {t, 0, 4.1 Pi}, ImageSize -> 130 ]
```



The first argument of the command may even be a list of functions:

```
PolarPlot[ {t^2, -(Pi - t)^2}, {t,0,Pi}, ImageSize -> 200]
```

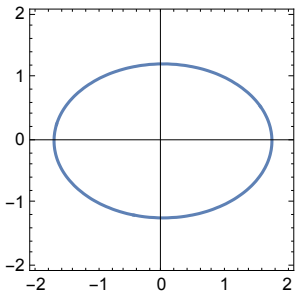


6.1.8 Plotting Implicit Functions

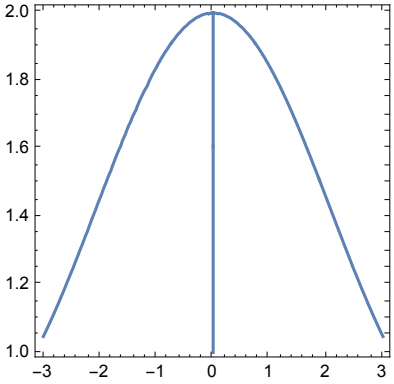
ContourPlot[*eqn*, {x, xmin, xmax},{y, ymin, ymax}, Options]
plots the solution to *eqn* using the **Solve[]** method, with **x** ranging from xmin to xmax.

(* **ImplicitPlot[{*eq1,eq2,...*}, ranges, options] *)** obsolete already in Mathematica 6 and 7

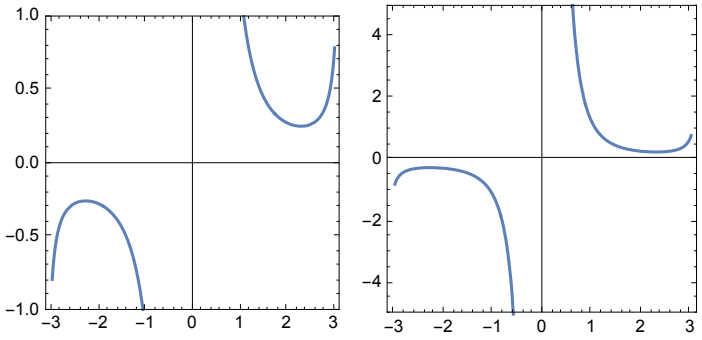
```
Clear[x,y]
p1 = ContourPlot[ x^2 + 2 y^2 == 3, {x,-2,2},{y,-2,2},ImageSize -> 150,
                  Axes -> True ]
```



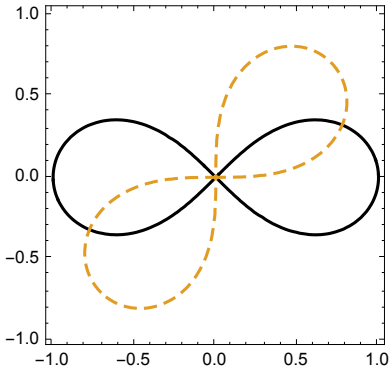

```
ContourPlot[y x == x  + Sin[x], {x,-3,3},{y, 1,2},ImageSize ->  200 ]
```




```
p1 = ContourPlot[y Sin[x] == 1/x^2, {x,-3,3},{y,-1,1},
                 Axes -> True, PlotRange -> {-1,1}];
p2 = ContourPlot[y Sin[x] == 1/x^2, {x,-3,3},{y,-5,5},
                 Axes -> True, PlotRange -> 5{-1,1}];
Show[GraphicsRow[{p1,p2}]]
```

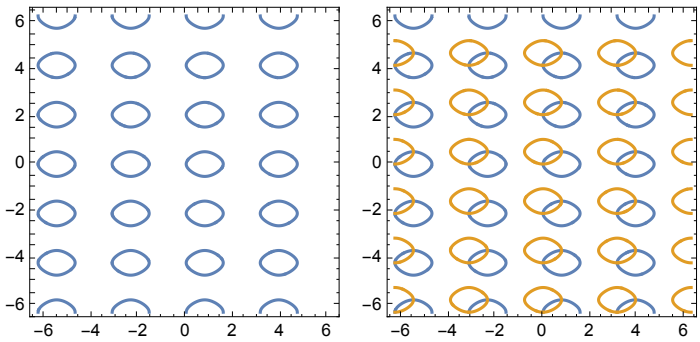


```
ContourPlot[{(x^2 + y^2)^2 == x^2 - y^2, (x^2 + y^2)^2 == 2 x y}, {x, -1, 1}, {y, -1, 1},
 ContourStyle → {GrayLevel[0], Dashing[{.03}]}, ImageSize → 200]
```




```
p1 = ContourPlot[ Sin[2 x] + Cos[3 y] == 1,
                  {x,-2Pi,2Pi},{y,-2Pi,2Pi},PlotPoints -> 50];

p2 = ContourPlot[ {Sin[2 x] + Cos[3 y] == 1,
                 Cos[2 x] + Sin[3 y] == 1},
                  {x,-2Pi,2Pi},{y,-2Pi,2Pi},PlotPoints -> 50];
```

```
Show[GraphicsRow[{p1,p2}]]
```



6.1.9 RegionPlot

```
?? RegionPlot
```

RegionPlot[*pred*, {*x*, x_{min}, x_{max}}, {*y*, y_{min}, y_{max}}] makes a plot showing the region in which *pred* is True. >>

```
Attributes[RegionPlot] = {HoldAll, Protected, ReadProtected}
```

```
Options[RegionPlot] = {AlignmentPoint → Center, AspectRatio → 1, Axes → False,
  AxesLabel → None, AxesOrigin → Automatic, AxesStyle → {}, Background → None,
  BaselinePosition → Automatic, BaseStyle → {}, BoundaryStyle → Automatic,
  ColorFunction → Automatic, ColorFunctionScaling → True, ColorOutput → Automatic,
  ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
  DisplayFunction :→ $DisplayFunction, Epilog → {}, Evaluated → Automatic,
  EvaluationMonitor → None, FormatType :→ TraditionalForm, Frame → True,
  FrameLabel → None, FrameStyle → {}, FrameTicks → Automatic, FrameTicksStyle → {},
  GridLines → None, GridLinesStyle → {}, ImageMargins → 0., ImagePadding → All,
  ImageSize → Automatic, ImageSizeRaw → Automatic, LabelStyle → {},
  MaxRecursion → Automatic, Mesh → None, MeshFunctions → {#1 &, #2 &}, MeshShading → None,
  MeshStyle → Automatic, Method → Automatic, PerformanceGoal :→ $PerformanceGoal,
  PlotLabel → None, PlotLegends → None, PlotPoints → Automatic, PlotRange → Full,
  PlotRangeClipping → True, PlotRangePadding → Automatic, PlotRegion → Automatic,
  PlotStyle → Automatic, PlotTheme :→ $PlotTheme, PreserveImageOptions → Automatic,
  Prolog → {}, RotateLabel → True, TargetUnits → Automatic,
  TextureCoordinateFunction → Automatic, TextureCoordinateScaling → Automatic,
  Ticks → Automatic, TicksStyle → {}, WorkingPrecision → MachinePrecision}
```

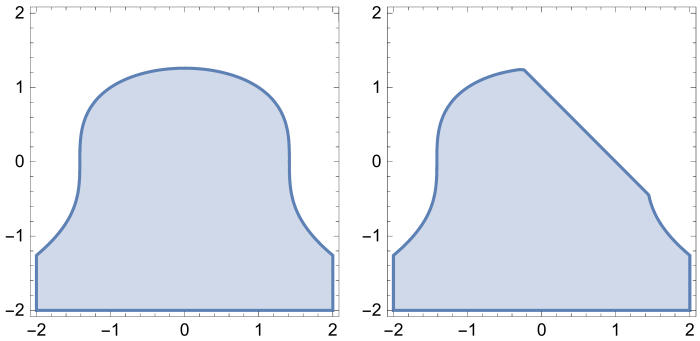
Plot a region defined by an inequality :

```
p1 = RegionPlot[x^2 + y^3 < 2, {x, -2, 2}, {y, -2, 2}];
```

Plot a region defined by logical combinations of inequalities :

```
p2 = RegionPlot[x^2 + y^3 < 2 && x + y < 1, {x, -2, 2}, {y, -2, 2}];
```

```
Show[GraphicsRow[{p1, p2}]]
```



Plot disconnected regions:

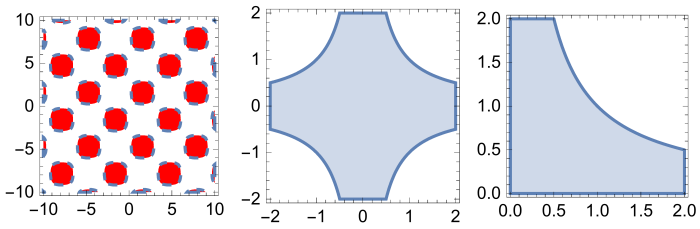
```
p1 = RegionPlot[Sin[x] Sin[y] > 1 / 4, {x, -10, 10},
    {y, -10, 10}, BoundaryStyle → Dashed, PlotStyle → Red];
```

Areas where the function is not real are excluded:

```
p2 = RegionPlot[Sqrt[Abs[x y]] < 1, {x, -2, 2}, {y, -2, 2}];
```

```
p3 = RegionPlot[Sqrt[Abs[x y]] < 1, {x, 0, 2}, {y, 0, 2}];
```

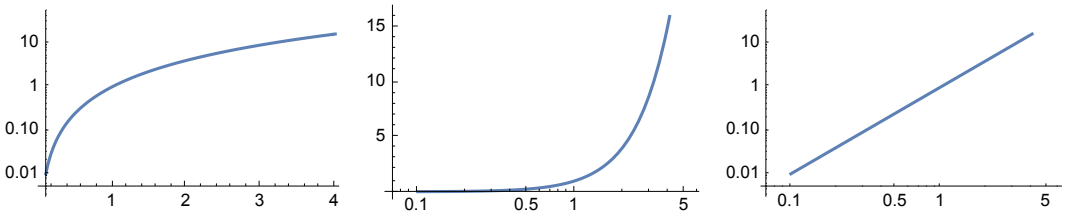
```
Show[GraphicsRow[{p1, p2, p3}]]
```



6.1.10 Plots with Logarithmic Scales

LogPlot[*f, {x,xmin,xmax}*]	Plot with linear scale for abscissa and logarithmic scale for ordinate of function ***f***
LogLinearPlot[*f, {x,xmin,xmax}*]	Plot with linear scale for ordinate and logarithmic scale for abscissa
LogLogPlot[*f, {x,xmin,xmax}*]	Plot with logarithmic scale for both scales
list = {{x1,y1},{x2,y2}, ...}	or
list = {y1,y2, ...}	this implies: ***x1 = 1, x2 = 2, ...***

```
p1 = LogPlot[x^2, {x, 0.1, 4}];
p2 = LogLinearPlot[x^2, {x, 0.1, 4}, PlotRange → All];
p3 = LogLogPlot[x^2, {x, 0.1, 4}];
Show[GraphicsRow[{p1, p2, p3}], ImageSize →  550]
```

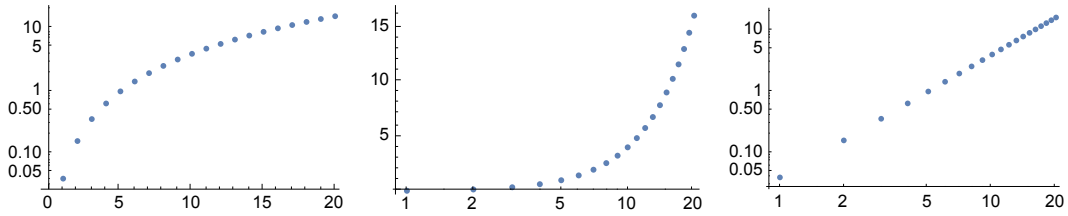


ListLogPlot[list, *{x,xmin,xmax}*]	Plot with linear scale for abscissa and logarithmic scale for ordinate of list li
LogLinearPlot[list, *{x,xmin,xmax}*]	Plot with linear scale for ordinate and logarithmic scale for abscissa
LogLogPlot[list, *{x,xmin,xmax}*]	Plot with logarithmic scale for both scales
list = {{x1,y1},{x2,y2}, ...}	or
list = {y1,y2, ...}	this implies: ***x1 = 1, x2 = 2, ...***

```
list = Table[x^2, {x, 0.2, 4, 0.2}]
```

```
{0.04, 0.16, 0.36, 0.64, 1., 1.44, 1.96, 2.56, 3.24, 4.,
 4.84, 5.76, 6.76, 7.84, 9., 10.24, 11.56, 12.96, 14.44, 16.}
```

```
p1 = ListLogPlot[list];
p2 = ListLogLinearPlot[list];
p3 = ListLogLogPlot[list];
Show[GraphicsRow[{p1, p2, p3}], ImageSize → 550]
```



6.1.11 Options for 2-Dimensional Plots

6.1.10.1 Options belonging to the various 2-dimensional plot commands

```
?Plot
```

Plot[f, {x, x_{min}, x_{max}}] generates a plot of f as a function of x from x_{min} to x_{max}.
Plot[{f_1, f_2, …}, {x, x_{min}, x_{max}}] plots several functions f_i.
Plot[…, {x} ∈ reg] takes the variable x to be in the geometric region reg. ≫

```
??Plot
```

Plot[f, {x, x_{min}, x_{max}}] generates a plot of f as a function of x from x_{min} to x_{max}.
Plot[{f_1, f_2, …}, {x, x_{min}, x_{max}}] plots several functions f_i.
Plot[…, {x} ∈ reg] takes the variable x to be in the geometric region reg. ≫

```
Attributes[Plot] = {HoldAll, Protected, ReadProtected}

Options[Plot] = {AlignmentPoint → Center, AspectRatio → 1/GoldenRatio, Axes → True,
  AxesLabel → None, AxesOrigin → Automatic, AxesStyle → {}, Background → None,
  BaselinePosition → Automatic, BaseStyle → {}, ClippingStyle → None,
  ColorFunction → Automatic, ColorFunctionScaling → True, ColorOutput → Automatic,
  ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
  DisplayFunction :→ $DisplayFunction, Epilog → {}, Evaluated → Automatic,
  EvaluationMonitor → None, Exclusions → Automatic, ExclusionsStyle → None,
  Filling → None, FillingStyle → Automatic, FormatType :→ TraditionalForm,
  Frame → False, FrameLabel → None, FrameStyle → {}, FrameTicks → Automatic,
  FrameTicksStyle → {}, GridLines → None, GridLinesStyle → {}, ImageMargins → 0.,
  ImagePadding → All, ImageSize → Automatic, ImageSizeRaw → Automatic,
  LabelStyle → {}, MaxRecursion → Automatic, Mesh → None, MeshFunctions → {#1 &},
  MeshShading → None, MeshStyle → Automatic, Method → Automatic,
  PerformanceGoal :→ $PerformanceGoal, PlotLabel → None, PlotLegends → None,
  PlotPoints → Automatic, PlotRange → {Full, Automatic}, PlotRangeClipping → True,
  PlotRangePadding → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,
  PlotTheme :→ $PlotTheme, PreserveImageOptions → Automatic, Prolog → {},
  RegionFunction → (True &), RotateLabel → True, TargetUnits → Automatic,
  Ticks → Automatic, TicksStyle → {}, WorkingPrecision → MachinePrecision}
```

```
?? ParametricPlot
```

ParametricPlot[{f_x, f_y}, {u, u_{min}, u_{max}}] generates
a parametric plot of a curve with x and y coordinates f_x and f_y as a function of u.
ParametricPlot[{{f_x, f_y}, {g_x, g_y}, ...}, {u, u_{min}, u_{max}}] plots several parametric curves
ParametricPlot[{f_x, f_y}, {u, u_{min}, u_{max}}, {v, v_{min}, v_{max}}] plots a parametric region
ParametricPlot[{{f_x, f_y}, {g_x, g_y}, ...}, {u, u_{min}, u_{max}}, {v, v_{min}, v_{max}}] plots several parametric regions
ParametricPlot[..., {u, v} ∈ reg] takes parameters {u, v} to be in the geometric region reg. >>

```
Attributes[ParametricPlot] = {HoldAll, Protected, ReadProtected}
```

```
Options[ParametricPlot] = {AlignmentPoint → Center, AspectRatio → Automatic, Axes → True,
  AxesLabel → None, AxesOrigin → Automatic, AxesStyle → {}, Background → None,
  BaselinePosition → Automatic, BaseStyle → {}, BoundaryStyle → Automatic,
  ColorFunction → Automatic, ColorFunctionScaling → True, ColorOutput → Automatic,
  ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
  DisplayFunction :→ $DisplayFunction, Epilog → {}, Evaluated → Automatic,
  EvaluationMonitor → None, Exclusions → Automatic, ExclusionsStyle → None,
  FormatType :→ TraditionalForm, Frame → Automatic, FrameLabel → None,
  FrameStyle → {}, FrameTicks → Automatic, FrameTicksStyle → {}, GridLines → None,
  GridLinesStyle → {}, ImageMargins → 0., ImagePadding → All, ImageSize → Automatic,
  ImageSizeRaw → Automatic, LabelStyle → {}, MaxRecursion → Automatic, Mesh → Automatic,
  MeshFunctions → Automatic, MeshShading → None, MeshStyle → Automatic,
  Method → Automatic, PerformanceGoal :→ $PerformanceGoal, PlotLabel → None,
  PlotLegends → None, PlotPoints → Automatic, PlotRange → Automatic,
  PlotRangeClipping → True, PlotRangePadding → Automatic, PlotRegion → Automatic,
  PlotStyle → Automatic, PlotTheme :→ $PlotTheme, PreserveImageOptions → Automatic,
  Prolog → {}, RegionFunction → (True &), RotateLabel → True, TargetUnits → Automatic,
  TextureCoordinateFunction → Automatic, TextureCoordinateScaling → Automatic,
  Ticks → Automatic, TicksStyle → {}, WorkingPrecision → MachinePrecision}
```

```
?? ListPlot
```

ListPlot[{y_1, y_2, ...}] plots points corresponding to a list of values, assumed to correspond to x coordinates 1, 2, ...
ListPlot[{{x_1, y_1}, {x_2, y_2}, ...}] plots a list of points with specified x and y coordinates
ListPlot[{$list_1$, $list_2$, ...}] plots several lists of points >>

```
Attributes[ListPlot] = {Protected, ReadProtected}
```

```
Options[ListPlot] = {AlignmentPoint → Center, AspectRatio → 1/GoldenRatio, Axes → True,
  AxesLabel → None, AxesOrigin → Automatic, AxesStyle → {}, Background → None,
  BaselinePosition → Automatic, BaseStyle → {}, ClippingStyle → None,
  ColorFunction → Automatic, ColorFunctionScaling → True, ColorOutput → Automatic,
  ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
  DataRange → Automatic, DisplayFunction :→ $DisplayFunction, Epilog → {},
  Filling → None, FillingStyle → Automatic, FormatType :→ TraditionalForm, Frame → False,
  FrameLabel → None, FrameStyle → {}, FrameTicks → Automatic, FrameTicksStyle → {},
  GridLines → None, GridLinesStyle → {}, ImageMargins → 0., ImagePadding → All,
  ImageSize → Automatic, ImageSizeRaw → Automatic, InterpolationOrder → None,
  Joined → False, LabelStyle → {}, MaxPlotPoints → ∞, Mesh → None,
  MeshFunctions → {#1 &}, MeshShading → None, MeshStyle → Automatic, Method → Automatic,
  PerformanceGoal :→ $PerformanceGoal, PlotLabel → None, PlotLegends → None,
  PlotMarkers → None, PlotRange → Automatic, PlotRangeClipping → True,
  PlotRangePadding → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,
  PlotTheme :→ $PlotTheme, PreserveImageOptions → Automatic, Prolog → {},
  RotateLabel → True, TargetUnits → Automatic, Ticks → Automatic, TicksStyle → {}}
```

```
?? ListLinePlot
```

ListLinePlot[{y_1, y_2, ...}] plots a line through a list of values, assumed to correspond to x coordinates 1, 2, ...
ListLinePlot[{{x_1, y_1}, {x_2, y_2}, ...}] plots a line through specific x and y positions.
ListLinePlot[{$list_1$, $list_2$, ...}] plots several lines. >>

```
Attributes[ListLinePlot] = {Protected, ReadProtected}
```

```
Options[ListLinePlot] = {AlignmentPoint → Center, AspectRatio → 1/GoldenRatio, Axes → True,
  AxesLabel → None, AxesOrigin → Automatic, AxesStyle → {}, Background → None,
  BaselinePosition → Automatic, BaseStyle → {}, ClippingStyle → None,
  ColorFunction → Automatic, ColorFunctionScaling → True, ColorOutput → Automatic,
  ContentSelectable → Automatic, CoordinatesToolOptions → Automatic,
  DataRange → Automatic, DisplayFunction :→ $DisplayFunction, Epilog → {},
  Filling → None, FillingStyle → Automatic, FormatType :→ TraditionalForm, Frame → False,
  FrameLabel → None, FrameStyle → {}, FrameTicks → Automatic, FrameTicksStyle → {},
  GridLines → None, GridLinesStyle → {}, ImageMargins → 0., ImagePadding → All,
  ImageSize → Automatic, ImageSizeRaw → Automatic, InterpolationOrder → None,
  Joined → True, LabelStyle → {}, MaxPlotPoints → ∞, Mesh → None,
  MeshFunctions → {#1 &}, MeshShading → None, MeshStyle → Automatic, Method → Automatic,
  PerformanceGoal :→ $PerformanceGoal, PlotLabel → None, PlotLegends → None,
  PlotMarkers → None, PlotRange → Automatic, PlotRangeClipping → True,
  PlotRangePadding → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,
  PlotTheme :→ $PlotTheme, PreserveImageOptions → Automatic, Prolog → {},
  RotateLabel → True, TargetUnits → Automatic, Ticks → Automatic, TicksStyle → {}}
```

```
?? PolarPlot
```

PolarPlot[r, {θ, θ_{min}, θ_{max}}] generates a polar plot of a curve with radius r as a function of angle θ.
PolarPlot[{f_1, f_2, ...}, {θ, θ_{min}, θ_{max}}] makes a polar plot of curves with radius functions f_1, f_2, ... >>

```
Attributes[PolarPlot] = {Protected, ReadProtected}
```

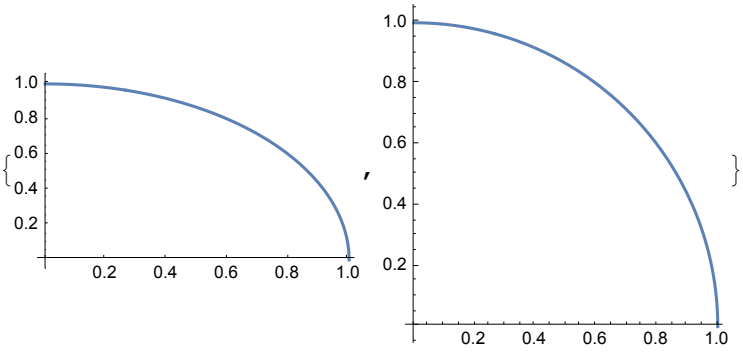
6.1.10.2 Discussion of various options

```
? AspectRatio
```

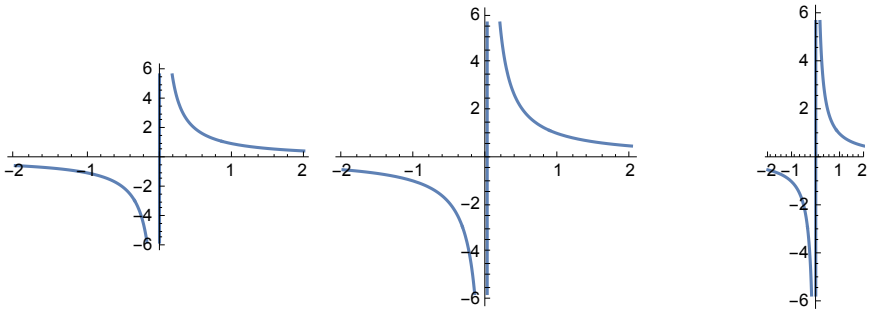
AspectRatio is an option for Graphics and related functions that specifies the ratio of height to width for a plot >>

AspectRatio -> Automatic determines the ratio of height to width from the actual coordinate values in the plot.

```
{Plot[Sqrt[1 - x^2], {x, 0, 1}],
 Plot[Sqrt[1 - x^2], {x, 0, 1}, AspectRatio → Automatic]}
```



```
p  = Plot[ 1/x, {x,-2,2}];
p1 = Show[p, AspectRatio -> 1];
p2 = Show[p, AspectRatio -> Automatic];
Show[GraphicsRow[{p,p1,p2}], ImageSize -> 500]
```



The default option is: **AspectRatio -> 1/GoldenRatio** .

```
? GoldenRatio
```

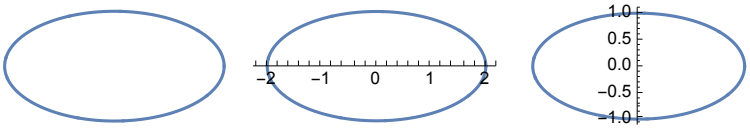
GoldenRatio is the golden ratio $\phi = \frac{1}{2}\left(\sqrt{5}+1\right)$, with numerical value $\simeq 1.61803$ >>

```
? Axes
```

Axes is an option for graphics functions that specifies whether axes should be drawn >>

Axes -> False	no axes shown
Axes -> {True,False}	x-axis only
Axes -> {False,True}	y-axis only

```
k = ParametricPlot[{2 Cos[ϕ], 1 Sin[ϕ]},{ϕ,0,2π}];
k1 = Show[k, Axes -> False];
k2 = Show[k, Axes -> {True,False} ];
k3 = Show[k, Axes -> {False,True} ];
Show[GraphicsRow[{k1,k2,k3}], ImageSize -> 400]
```



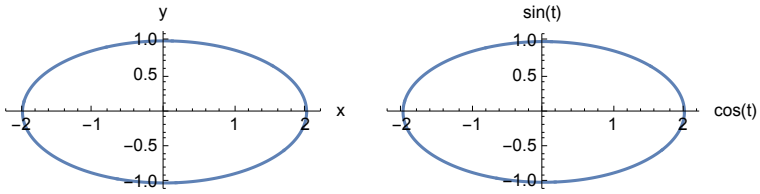

```
?AxesLabel
```

AxesLabel is an option for graphics functions that specifies labels for axes. >>

```
AxesLabel -> {"xtext", "ytext"}
```

The quotation marks specify the text as a string, which has no relation to all the expressions defined elsewhere;
while symbols may enter expressions not marked as strings by quotation marks.

```
x = "cos(t)"; y = "sin(t)";
k1 = Show[k, AxesLabel -> {"x", "y"} ];
k2 = Show[k, AxesLabel -> { x ,  y } ];
Show[GraphicsRow[{k1,k2}], ImageSize -> 400]
```

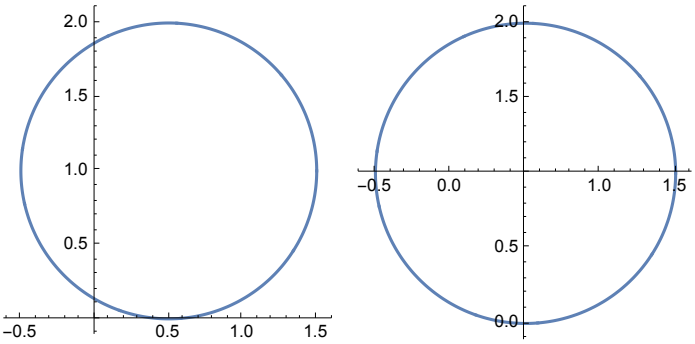



```
?AxesOrigin
```

AxesOrigin is an option for graphics functions that specifies where any axes drawn should cross >>

AxesOrigin -> {x0,y0}	Axes will cross at (x0,y0).

```
k1  = ParametricPlot[ {0.5 + Cos[t], 1 + Sin[t]}, {t,0,2Pi}];
k2 = Show[k1, AxesOrigin -> {0.5,1} ];
Show[GraphicsRow[{k1,k2}]]
```




```
?AxesStyle
```

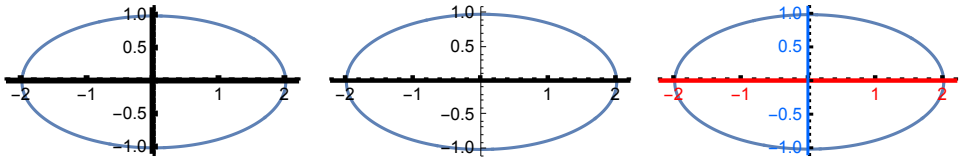
AxesStyle is an option for graphics functions that specifies how axes should be rendered >>

AxesStyle -> style

All axes rendered according to style directive, cf. § 6.5.2.

AxesStyle -> {xstyle, ystyle}

```
k1 = Show[k, AxesStyle -> Thickness[0.02] ];
k2 = Show[k, AxesStyle -> {{Thickness[.015]},{Thickness[.001]}} ];
k3 = Show[k, AxesStyle -> {{Thickness[.015],Hue[0]},{Thickness[.01],Hue[0.6]}} ];
Show[GraphicsRow[{k1,k2,k3}],ImageSize ->  500]
```



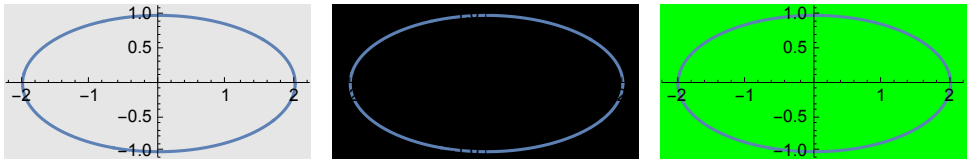
Note: **All the braces** given in the command for k2 are really **necessary** ! But they are needed in k3 !

```
?Background
```

Background is an option that specifies what background color to use. >>

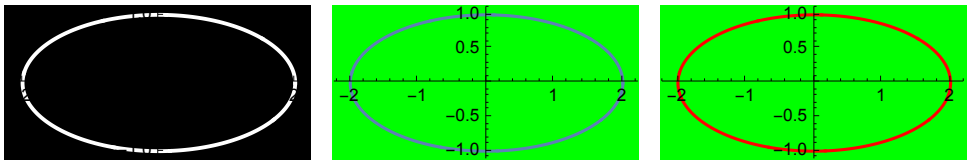
Background settings are: GrayLevel[], RGBColor[]; Hue[], see § 6.5.3 .

```
k1 = Show[k, Background -> GrayLevel[0.9] ];
k2 = Show[k, Background -> GrayLevel[0] ];
k3 = Show[k, Background -> RGBColor[.0, .999, 0.] ];
Show[GraphicsRow[{k1,k2,k3}], ImageSize -> 500]
```



On the screen the first drawing shows a black curve and black coordinate axes in front of a gray background. The second picture
is black, since the drawing's background are black. In the third one curve and axes are gray in front of a green background.
In some cases it may be recommended to use a style option for the curve, too. S. **PlotStyle .**

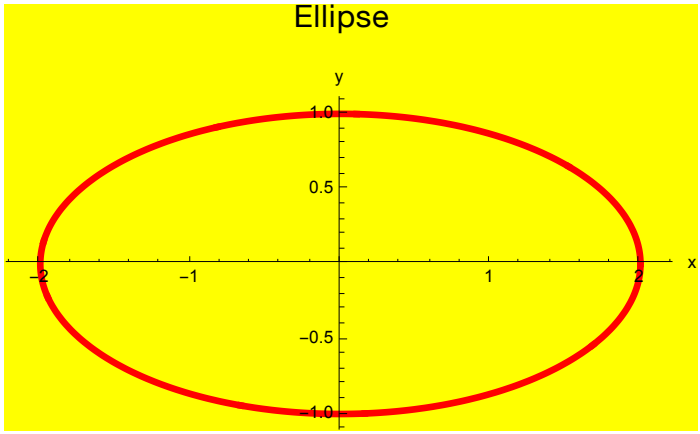
```
k1 = ParametricPlot[{2 Cos[ϕ], 1 Sin[ϕ]}, {ϕ, 0, 2 π},
    BaseStyle → RGBColor[0.9, 0, 0], Background -> RGBColor[.0, .999, 0.]];
k2 = ParametricPlot[{2 Cos[ϕ], 1 Sin[ϕ]}, {ϕ, 0, 2 π}, Background -> GrayLevel[0],
    PlotStyle -> {Thick, GrayLevel[1]}, BaseStyle → GrayLevel[1] ];
k3 = ParametricPlot[{2 Cos[ϕ], 1 Sin[ϕ]}, {ϕ, 0, 2 π},
    Background -> RGBColor[.0, .999, 0.], PlotStyle → {Red}];
Show[GraphicsRow[{k2, k1, k3}], ImageSize -> 500]
```



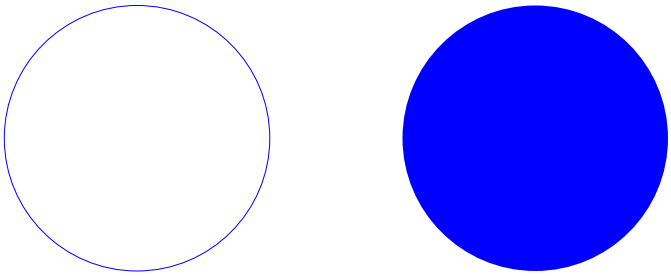

```
? BaseStyle
```

BaseStyle is an option for formatting and related constructs that specifies the base style to use for them >>

```
k4 = ParametricPlot[{2 Cos[ϕ], 1 Sin[ϕ]},{ϕ,0,2π},
PlotStyle -> {Hue[0],Thickness[0.01]}];
k1 = Show[k4,AxesLabel -> {"x","y"}, Background ->  Yellow,
BaseStyle ->  Magenta,
PlotLabel -> Style["Ellipse\n",FontSize -> 16, FontFamily -> Helvetica]]
```



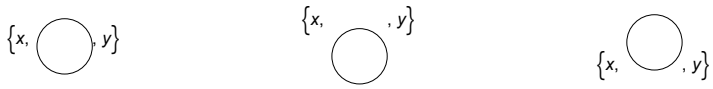
```
Graphics[{Circle[{0, 0}, 1], Disk[{3, 0}, 1]}, BaseStyle → Blue]
```



```
? BaselinePosition
```

BaselinePosition is an option that specifies where the baseline of an object is considered to be for purposes of alignment with surrounding text or other expressions. >>

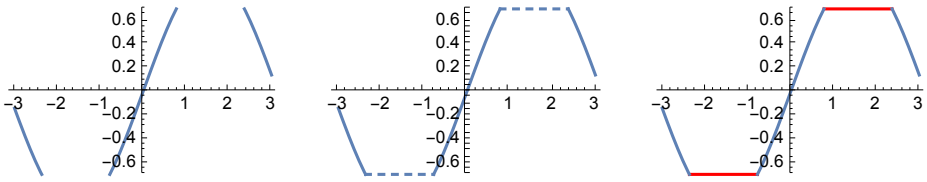
```
Clear[x, y]
p1 = {x, Graphics[Circle[], BaselinePosition → Center,
     BaselinePosition → Center, ImageSize → 30], y};
p2 = {x, Graphics[Circle[], BaselinePosition → Top,
     BaselinePosition → Center, ImageSize → 30], y};
p3 = {x, Graphics[Circle[], BaselinePosition → Bottom,
     BaselinePosition → Center, ImageSize → 30], y};
Show[GraphicsRow[{p1, p2, p3}], ImageSize → 450]
```



```
? ClippingStyle
```

ClippingStyle is an option for plotting functions that specifies the style of what should be drawn when curves or surfaces would extend beyond the plot range >>

```
SetOptions[Plot, ImageSize → 140];
pp = Table[Plot[Sin[x], {x, -3, 3}, PlotRange -> {-.7, .7}, ClippingStyle → cs],
           {cs, {None, Automatic, Red}}];
GraphicsRow[pp, Spacings → {Scaled[0.25], Scaled[1]}]
```

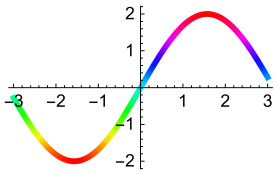


see also option **PlotRangeClipping .**

```
? ColorFunction
```

ColorFunction is an option for graphics functions that specifies a function to apply to determine colors of elements >>

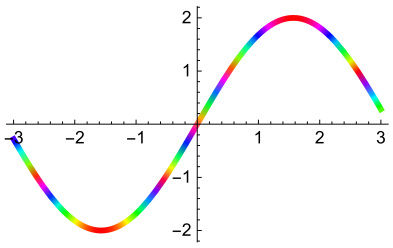
```
pp = Plot[2 Sin[x], {x, -3, 3},
  PlotStyle → AbsoluteThickness[3], ColorFunction → Hue]
```




```
? ColorFunctionScaling
```

ColorFunctionScaling is an option for graphics functions that specifies
whether arguments supplied to a color function should be scaled to lie between 0 and 1. >>

```
pp = Plot[2 Sin[x], {x, -3, 3}, PlotStyle → AbsoluteThickness[3],
  ColorFunction → Hue, ColorFunctionScaling → False, ImageSize → 200]
```



ColorOutput has been superseded.

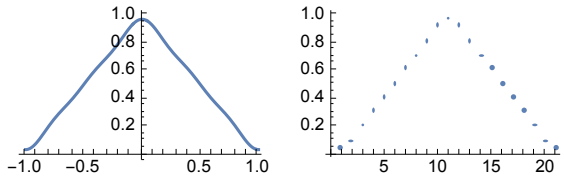
```
? DataRange
```

DataRange is an option for functions such as ListPlot and ListDensityPlot
that specifies what range of actual coordinates the data should be assumed to occupy >>

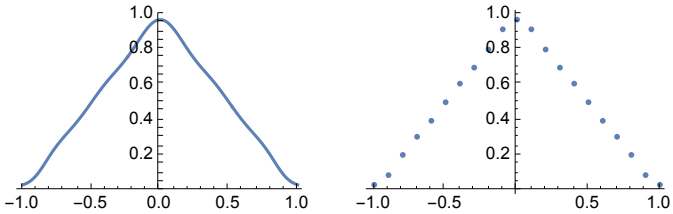
The range of the x-coordinates of points given in a list may be modified by **DataRange.**
This is shown in an example with a triangular funtion, which is approximated by the first three terms of the corresponding Fourier series:

```
Clear[f, t, lp, p1, p2]
f[t_] = 1/2 + 4/π^2 (Cos[(2 Pi * 1)/2 * t] + 1/9 Cos[(2 Pi * 3)/2 * t] + 1/25 Cos[(2 Pi * 5)/2 * t]);
```

```
p1 = Plot[f[t], {t, -1, +1}];
lp = Table[f[t], {t, -1, +1, 0.1}];
p2 = ListPlot[lp, PlotRange → All];
GraphicsRow[{p1, p2}]
```




```
p1 = ListPlot[lp, PlotRange → {0, 1},
    DataRange → {-1, 1}, Joined → True, InterpolationOrder → 5];
lp = Table[{t, f[t]} // N, {t, -1, 1, 1 / 10}] ;
p2 = ListPlot[lp];
GraphicsRow[{p1, p2}]
```




```
? DisplayFunction
```

DisplayFunction is an option for graphics and sound functions
that specifies a function to apply to graphics and sound objects before returning them >>

The option **DisplayFunction** had a meaning completly different from that shown below in earlier versions of *Mathematica* (*Mathematica* 5 and earlier versions}.

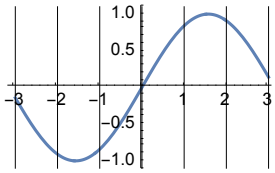
```
Show[Graphics[{Blue, Disk[{0, 0}, 1]}],
 DisplayFunction → (PopupWindow[Button["Click here"], #] &)]
```

Clickhere

```
? Epilog
```

Epilog is an option for graphics functions that gives a list of graphics primitives to be rendered after the main part of the graphics is rendered >>

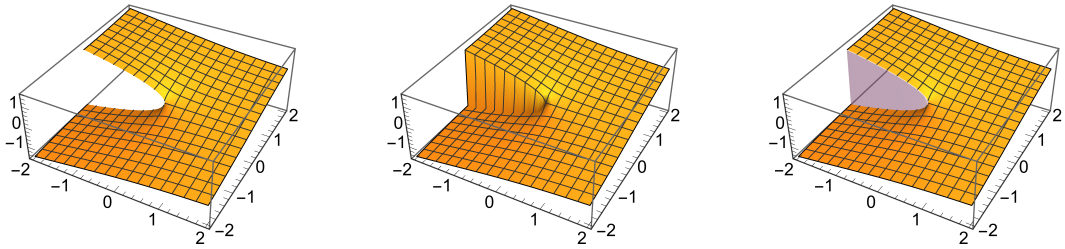
```
pp = Plot[ Sin[x], {x, -3, 3},
  Epilog → Map[Line, Table[{{n, 1.1}, {n, -1.1}}, {n, -3, 3}]]]
```



```
? Exclusions
```

Exclusions is an option that specifies where to exclude in regions used by functions like Plot, Plot3D, and NIntegrate. >>

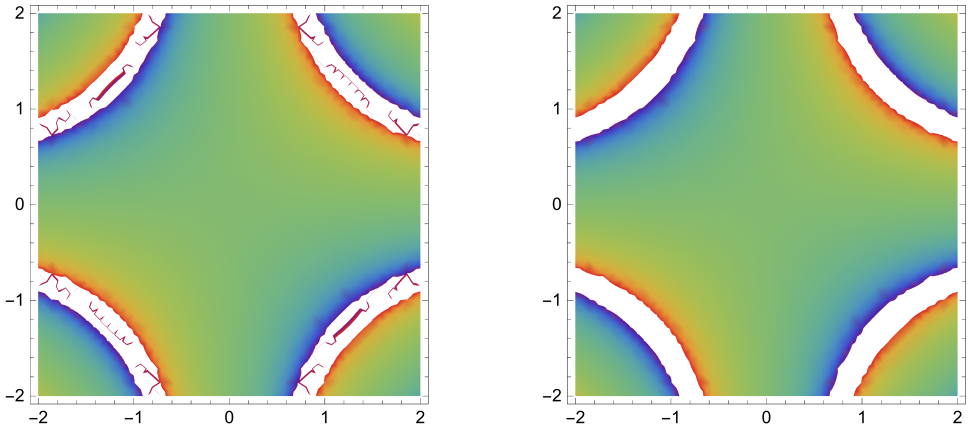
```
p1 = Plot3D[Im[Sqrt[(x + I y)]], {x, -2, 2}, {y, -2, 2}];
p2 = Plot3D[Im[Sqrt[(x + I y)]], {x, -2, 2}, {y, -2, 2}, Exclusions → None];
p3 = Plot3D[Im[Sqrt[x + I y]],
    {x, -2, 2}, {y, -2, 2}, ExclusionsStyle → Opacity[0.5]];
GraphicsRow[{p1, p2, p3}, Spacings → Scaled[0.4], ImageSize → 550]
```



```
? ExclusionsStyle
```

ExclusionsStyle is an option to plotting functions
that specifies how to render subregions excluded according to Exclusions. >>

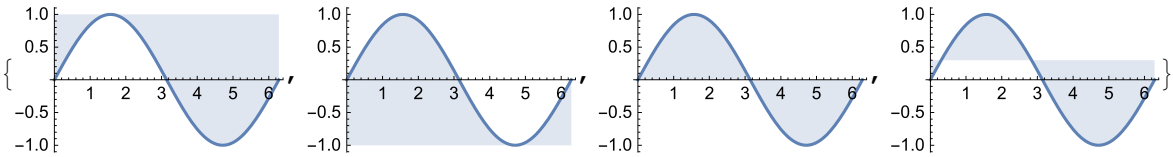
```
p1 = DensityPlot[Tan[x y], {x, -2, 2}, {y, -2, 2}, ColorFunction → "Rainbow"];
p2 = DensityPlot[Tan[x y], {x, -2, 2}, {y, -2, 2}, ColorFunction → "Rainbow",
      Exclusions → {Cos[x y] == 0}];
GraphicsRow[{p1, p2}, Spacings → Scaled[0.4], ImageSize → 500]
```



```
? Filling
```

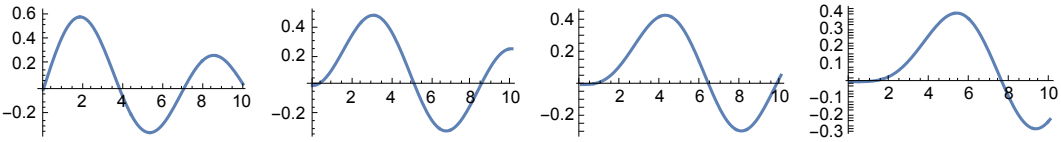
Filling is an option for ListPlot, Plot, Plot3D, and related
functions that specifies what filling to add under points, curves, and surfaces >>

```
Table[Plot[Sin[x], {x, 0, 2 Pi}, Filling → f], {f, {Top, Bottom, Axis, 0.3}}]
```

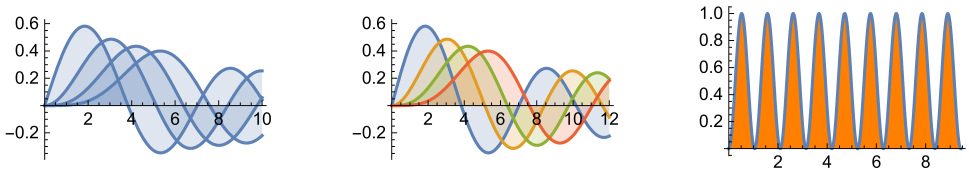


Filling multiple curves:

```
GraphicsRow[Table[Plot[BesselJ[n, x], {x, 0, 10}], {n, 4}], ImageSize → 550]
```



```
p1 = Plot[Table[BesselJ[n, x], {n, 4}], {x, 0, 10}, Filling → Axis];
p2 = Plot[Evaluate[Table[BesselJ[n, x], {n, 4}]], {x, 0, 12}, Filling → Axis];
p3 = Plot[Sin[3 x]^2, {x, 0, 3 π}, Filling → Axis, FillingStyle → Orange];
GraphicsRow[{p1, p2, p3}, Spacings → Scaled[0.4], ImageSize → 500]
```



```
? FillingStyle
```

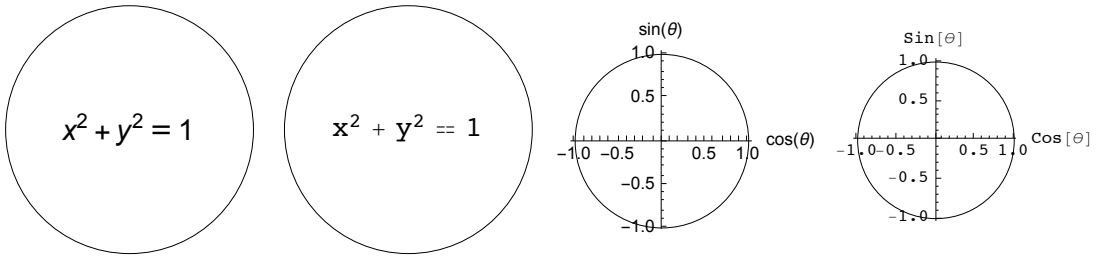
FillingStyle is an option for ListPlot, Plot, Plot3D,
and related functions that specifies the default style of filling to be used >>

(* see p3 above *)

```
? FormatType
```

FormatType is an option for output streams, graphics, and functions
such as Text that specifies the default format type to use when outputting expressions. >>

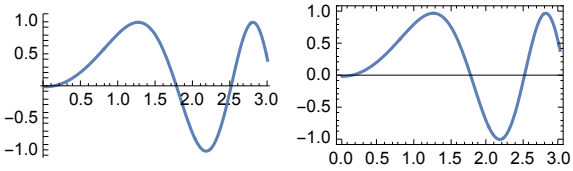
```
Clear[x, y]
p1 = Graphics[{Circle[], Text[Style[x^2 + y^2 == 1, 15]]}];
p2 = Graphics[
   {Circle[], Text[Style[x^2 + y^2 == 1, 15]]}, FormatType → StandardForm];
p3 = Graphics[Circle[], Axes → True, AxesLabel → {Cos[θ], Sin[θ]}];
p4 = Graphics[Circle[], Axes → True,
   AxesLabel → {Cos[θ], Sin[θ]}, FormatType → StandardForm];
GraphicsRow[{p1, p2, p3, p4}, ImageSize → 570]
```



```
? Frame
```

Frame is an option for Graphics, Grid, and other constructs that specifies whether to include a frame ≫

```
p1 = Plot[ Sin[x^2],  {x,0,3}];
p2 = Show[p1, Frame -> True];
GraphicsRow[{p1,p2}]
```



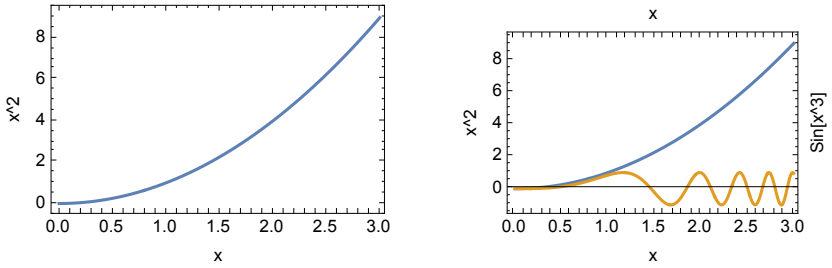

```
?FrameLabel
```

FrameLabel is an option for Graphics, Manipulate, and related functions that specifies labels to be placed on the edges of a frame ≫

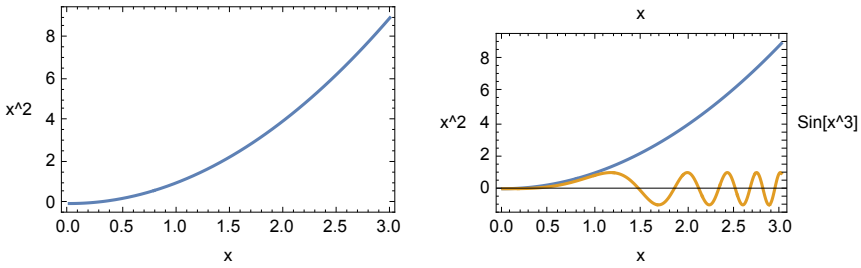
```
FrameLabel -> {xtext, ytext}
FrameLabel -> {xltext, yltext,xutext,yrtext}
```

Edges and labels are counted clockwise starting from the bottom edge.

```
p1 = Plot[ x^2, {x,0,3}, Frame -> True, FrameLabel -> {"x", "x^2"}];
p2 = Plot[ {x^2, Sin[x^3]}, {x,0,3}, Frame -> True,
           FrameLabel -> {"x","x^2","x","Sin[x^3]" }];
GraphicsRow[{p1,p2},ImageSize ->  450]
```




```
p3 = Show[p1, RotateLabel -> False]; p4 = Show[p2, RotateLabel -> False];
GraphicsRow[{p3, p4}, ImageSize ->  450]
```



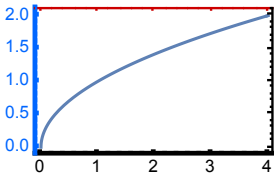

```
?FrameStyle
```

FrameStyle is an option for Graphics, Grid, and other constructs that specifies the style in which to draw frames ≫

Directives are (cf. § 6.5.2 and § 6.5.3) :
AbsoluteDashing, AbsoluteThickness[], Dashing[], Thickness[];
RGBColor[], CMYKColor[]; Hue[].

Edges are counted clockwise from the bottom edge. In the example below all braces are necessary to get the wanted result shown in the plot below at right.

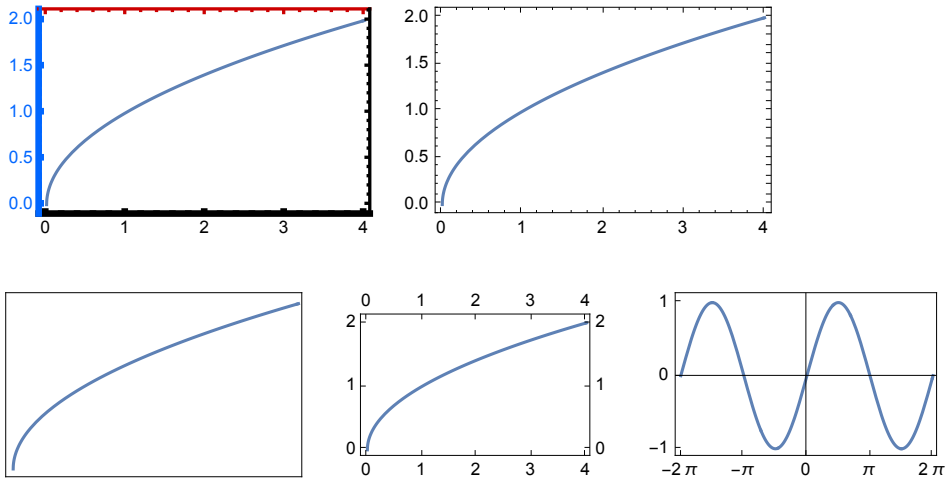
```
p2 = Plot[ Sqrt[x], {x,0,4}, Frame -> True,
     FrameStyle -> { {Thickness[0.02]},{Thickness[0.02],Hue[0.6]},
        {Thickness[0.01],RGBColor[.8,0,0]},{Thickness[0.01]}  }]
```




```
?FrameTicks
```

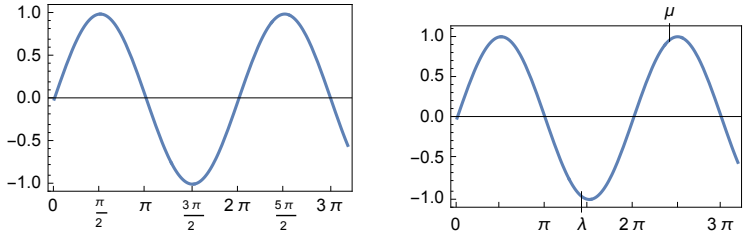
FrameTicks is an option for 2D graphics functions that specifies tick marks for the edges of a frame ≫

```
pr = Plot[ Sqrt[x], {x,0,4}, Frame -> True];
GraphicsRow[{p2,pr}, ImageSize -> 400]
p1 = Show[pr, FrameTicks -> None];
p2 = Show[pr, FrameTicks -> {{0,1,2,3,4}, {0,1,2}}];
p3 = Plot[Sin[x],{x,-2Pi,2Pi}, Frame → True, FrameTicks→{Pi {-2,-1,0,1,2},{-1,0,1},{},
GraphicsRow[{p1,p2,p3}, ImageSize -> 490]
```



The length of any tick may be chosen at will; see below the labels λ and μ.

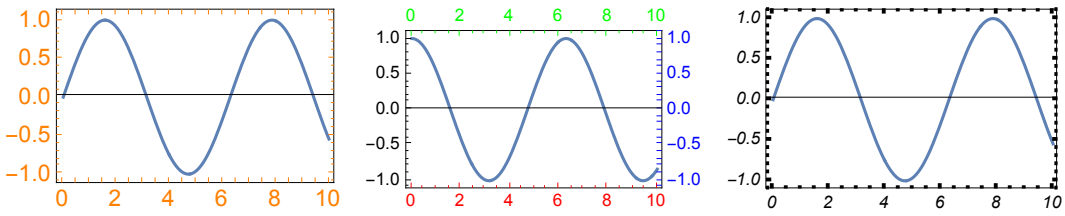
```
bp1 = Plot[Sin[x], {x, 0, 10}, Frame → True,
    FrameTicks → {π Range[0, 6] / 2, Automatic, None, None}];
bp2 = Plot[Sin[x], {x, 0, 10}, Frame → True,
    FrameTicks → {{0, {π / 2, ""}, π, {3 π / 2, ""}, {4.5, "λ", {0.05, 0.02}}, 2 π,
       {5 π / 2, ""}, 3 π}, Automatic, {{7.6, "μ", {0.05, 0.01}}}, None}];
GraphicsRow[{bp1, bp2}, ImageSize → 400]
```



```
? FrameTicksStyle
```

FrameTicksStyle is an option for 2D graphics functions that specifies how frameticks should be rendered >>

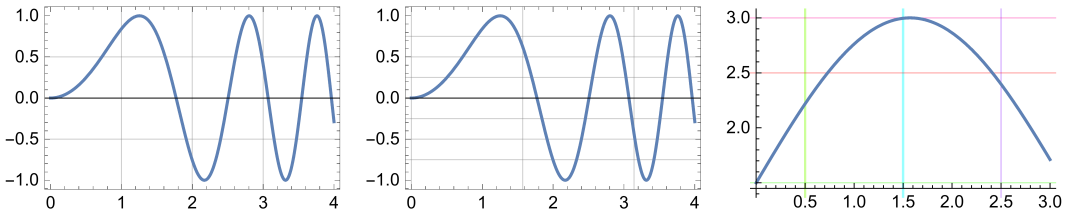
```
p1 = Plot[Sin[x], {x, 0, 10}, Frame → True,
    FrameTicks → Automatic, FrameTicksStyle → Directive[Orange, 12]];
p2 = Plot[Cos[x], {x, 0, 10}, Frame → True, FrameTicks → All,
    FrameTicksStyle → {{Black, Blue}, {Red, Green}}];
p3 = Plot[Sin[x], {x, 0, 10}, Frame → True,
    FrameTicksStyle → Directive[Thick, Italic]];
Show[GraphicsRow[{p1, p2, p3}], ImageSize → 550]
```




```
?GridLines
```

GridLines is an option for two-dimensional graphics functions that specifies gridlines >>

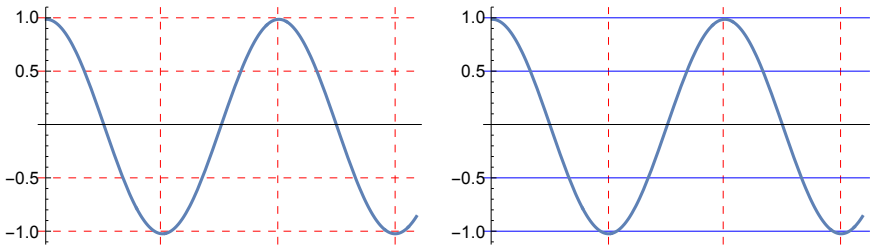
```
p1 = Plot[ Sin[x^2], {x,0,4}, Frame -> True,  GridLines -> Automatic];
p2 = Show[p1,GridLines -> {{0,Pi/2,Pi},{-.75,-.5,-.25,0,.25,.5,.75}}];
th = Thickness[.0075];
p3 = Plot[1.5 (1 + Sin[x]),{x,0,3},PlotRange → All,
  GridLines→{{{.5,{Hue[.25],th}},{1.5,{Hue[.5],th}},
            {2.5,{Hue[.75]}}},{{1.5,{Hue[.3]}},{2.5,{Hue[0]}},{3,{Hue[.9]}}}}];
Show[GraphicsRow[{p1,p2,p3}], ImageSize -> 550]
```




```
? GridLinesStyle
```

GridLinesStyle is an option for 2D graphics functions that specifies how gridlines should be rendered >>

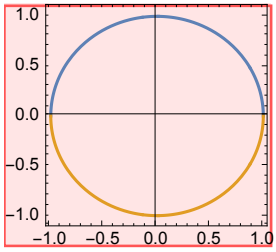
```
p1 = Plot[Cos[x], {x, 0, 10}, Ticks → {None, Automatic},
    GridLines → {{Pi, 2 Pi, 3 Pi}, {-1, -.5, .5, 1}},
    GridLinesStyle → Directive[Red, Dashed]];
p2 = Plot[Cos[x], {x, 0, 10}, Ticks → {None, Automatic},
    GridLines → {{Pi, 2 Pi, 3 Pi}, {-1, -.5, .5, 1}},
    GridLinesStyle → {{Red, Dashed}, {Blue}}];
GraphicsRow[{p1, p2}, ImageSize → 450]
```



```
? ImageMargins
```

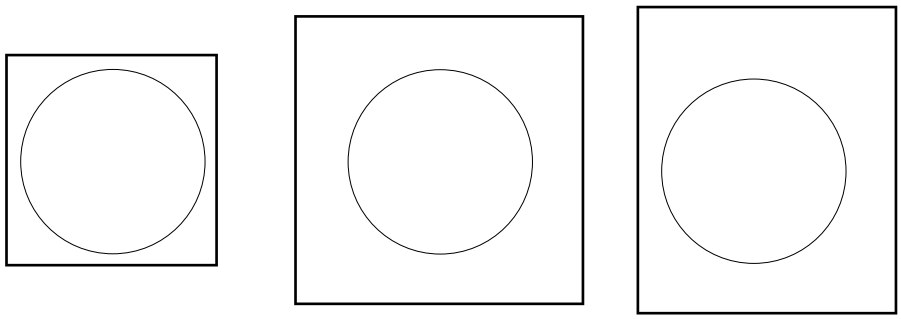
ImageMargins is an option that specifies the absolute margins to leave around the image displayed for an object. >>

```
Plot[{Sqrt[1 - x^2], -Sqrt[1 - x^2]}, {x, -1, 1}, Frame → True, AspectRatio → 1,
 ImageMargins → {5, 10}]
```



There is an error message: "The specified setting for the option GraphicsBoxOption, ImageMargins cannot be used."
One can proceed as follows:

```
p1 = Framed[Graphics[Circle[{0, 0}, 1], ImageSize → 100]];
p2 = Framed[Graphics[Circle[{0, 0}, 1], ImageMargins → 20, ImageSize → 100]];
p3 = Framed[Graphics[Circle[{0, 0}, 1],
     ImageMargins -> {{5, 20}, {20, 30}}, ImageSize → 100]];
GraphicsRow[{p1, p2, p3}, ImageSize → 500, PlotRange → All]
```



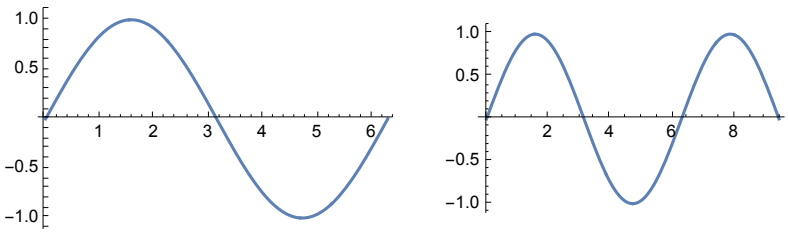
However, if the ImageSize is 150 in place of 100, the GraphicsRow is not correct !

```
? ImageSize
```

ImageSize is an option that specifies the overall size of an image to display for an object. >>

The option for ImageSize determines the widths of the drawing; it is given in printer's points (1 pt = 0.54 mm).
The default value is 288. In the graphics below **xcm** is the wanted width in cm.

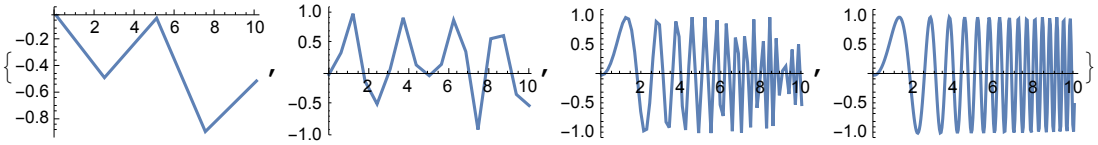
```
p1 = Plot[Sin[x], {x, 0, 2 π}, ImageSize → 200];
pp := Plot[Sin[x], {x, 0, 3 π}, ImageSize → 29 xcm]
xcm = 6; GraphicsRow[{p1, pp}]
```



```
? MaxRecursion
```

MaxRecursion is an option for functions like
NIntegrate and Plot that specifies how many recursive subdivisions can be made >>

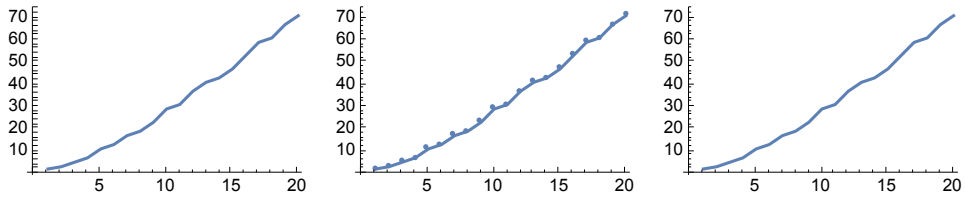
```
SetOptions[Plot, ImageSize → 130];
Table[Plot[Sin[x^2], {x, 0, 10}, PlotPoints → 5, MaxRecursion → mr],
 {mr, {0, 2, 4, 6}}]
```



```
? Mesh
```

Mesh is an option for Plot3D, DensityPlot, and other plotting functions that specifies what mesh should be drawn ≫

```
p1 = ListLinePlot[Table[Prime[n], {n, 20}]];
p2 = ListLinePlot[Table[Prime[n], {n, 20}], Mesh -> All];
p3 = ListLinePlot[Table[Prime[n], {n, 20}], MeshStyle → Hue[0]];
Show[GraphicsRow[{p1, p2, p3}], ImageSize -> 500]
```




```
? MeshStyle
```

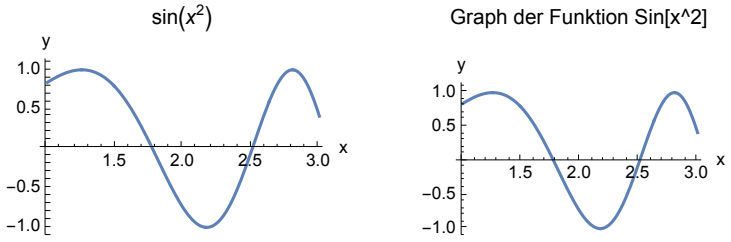
MeshStyle is an option for Plot3D, DensityPlot
and other plotting functions that specifies the style in which to draw a mesh ≫

(* see preceeding p3 *)

```
?PlotLabel
```

PlotLabel is an option for graphics functions that specifies an overall label for a plot ≫

```
Clear[x]
p1 =
  Plot[ Sin[x^2], {x, 1, 3}, AxesLabel -> {"x", "y"}, PlotLabel -> Sin[x^2] ];
p2 = Plot[ Sin[x^2], {x, 1, 3}, AxesLabel -> {"x", "y"}, PlotLabel ->
    "Graph der Funktion Sin[x^2]\n"];
Show[GraphicsRow[{p1, p2}], ImageSize -> 400]
```



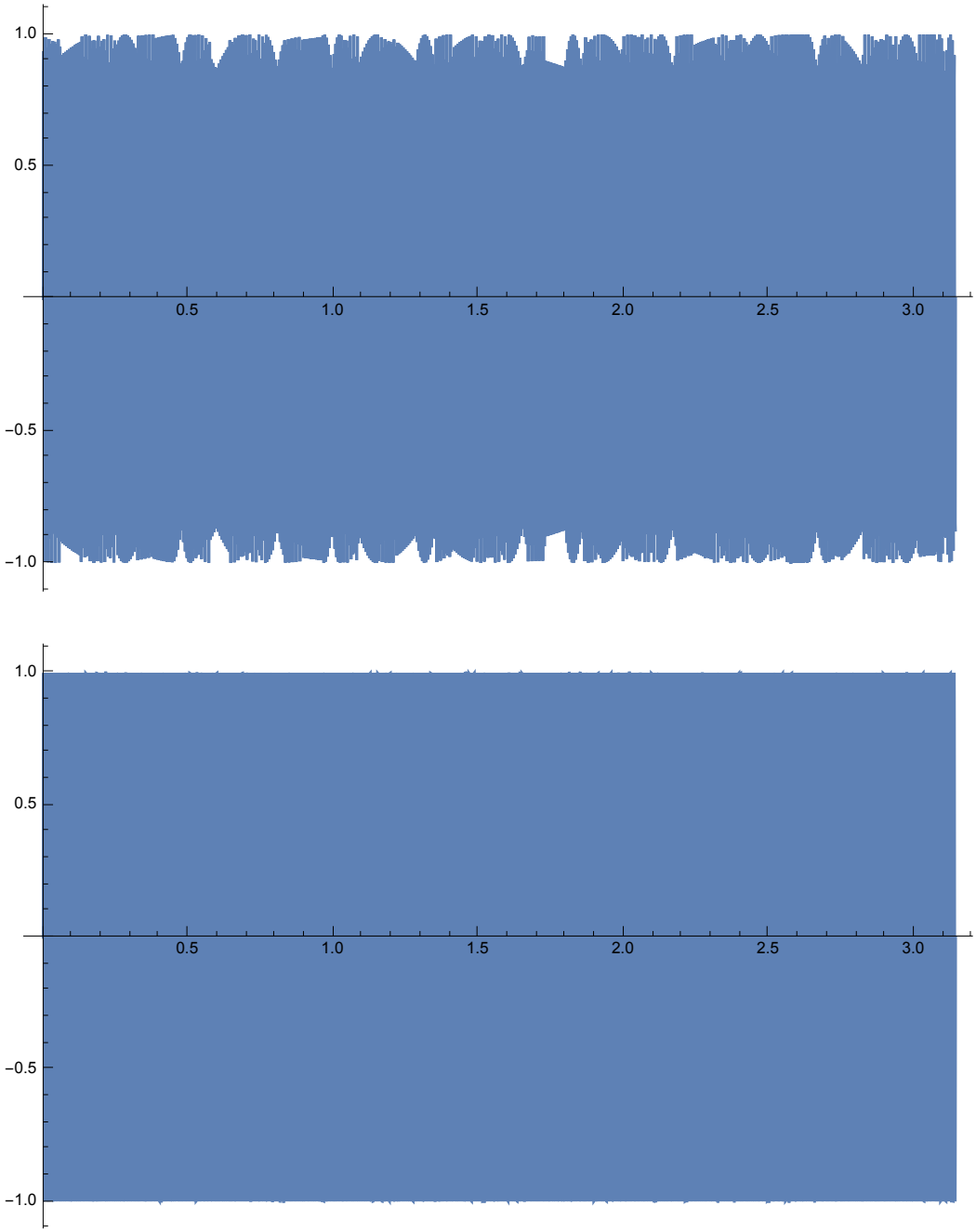
The symbol `\n` in the directive of `PlotLabel` induces a line break. An empty line may be used to
shift upwards a headline in collission with other inscriptions as, for example, the label of the ordinate.

```
?? PlotPoints
```

PlotPoints is an option for plotting functions that specifies how many initial sample points to use. >>

```
Attributes[PlotPoints] = {Protected}
```

```
p1 = Plot[Sin[1000 x], {x,0,π}];
p2 = Plot[Sin[1000 x], {x,0,π}, PlotPoints -> 800 ];
Show[GraphicsGrid[{{p1},{p2}}],ImageSize → 620]
```



```
? PlotRange
```

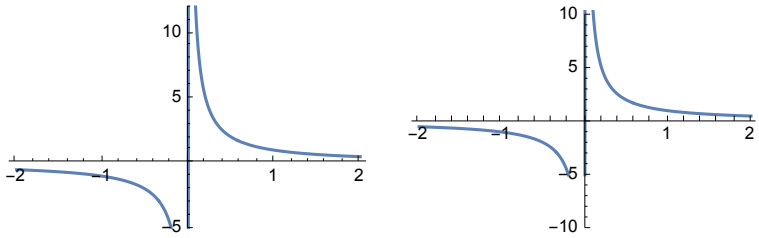
PlotRange is an option for graphics functions that specifies what range of coordinates to include in a plot ≫

```
PlotRange -> ymax
```

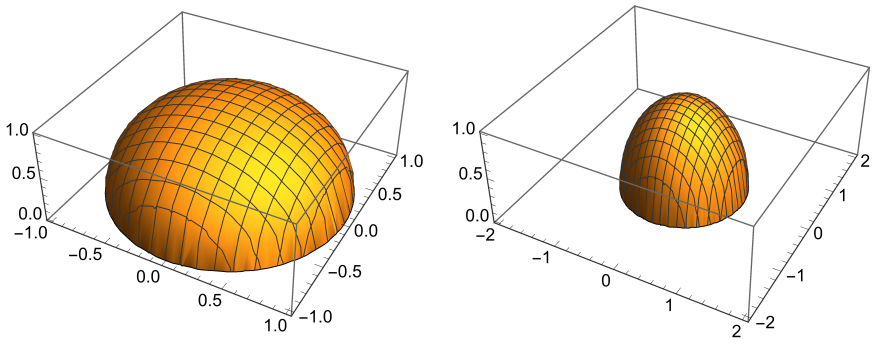
PlotRange → ymax

```
PlotRange -> {ymin, ymax}                    range for ordinate
PlotRange -> {{xmin, xmax}, {ymin, ymax}}
      range          for abscissa,          for ordinate
PlotRange -> All                plot everything
PlotRange -> Full               plot the full range of the variables
```

```
p1 = Plot[1/x, {x,-2,2}, PlotRange -> {-5,12}];
p2 = Show[p1, PlotRange -> {{-2,2}, {-10,10}} ];
Show[GraphicsArray[{p1,p2}], ImageSize -> 400]
```




```
p1 = Plot3D[Sqrt[1 - x^2 - y^2], {x, -2, 2}, {y, -2, 2}, PlotRange → All];
p2 = Plot3D[Sqrt[1 - x^2 - y^2], {x, -2, 2}, {y, -2, 2}, PlotRange → Full];
GraphicsRow[{p1, p2}, ImageSize → 450]
```



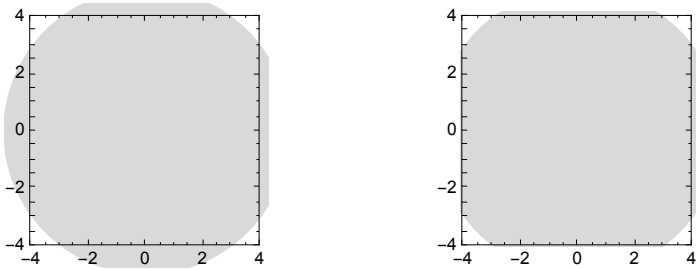

```
?? PlotRangeClipping
```

PlotRangeClipping is an option for graphics functions that specifies whether graphics objects should be clipped at the edge of the region defined by PlotRange, or should be allowed to extend to the actual edge of the image ≫

```
Attributes[PlotRangeClipping] = {Protected}
```

```
g1 = Graphics[{GrayLevel[0.85], Disk[{0, 0}, 5]},
    Frame → True, PlotRange → 4, PlotRangeClipping → False];

g2 = Graphics[{GrayLevel[0.85], Disk[{0, 0}, 5]},
    Frame → True, PlotRange → 4, PlotRangeClipping → True];
```

```
GraphicsRow[{g1, g2}, Spacings → {Scaled[1], Scaled[1]}]
```



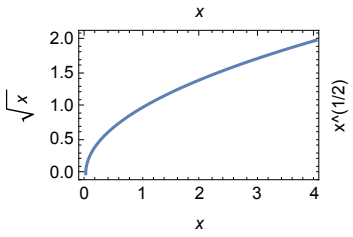

```
?PlotRegion
```

PlotRegion is an option for graphics functions that specifies what region of the final display area a plot should fill >>

This option was useful in cases where the drawing fills the alloted region completely and parts of it are destroyed in attempts of increasing or decreasing the size.of the drawing. Click into the picture below to see the effect of this option.

```
PlotRegion -> {{0 <= xmin, xmax <=1 },{0 <= ymin, ymax <= 1 }}
```

```
pr = Plot[ Sqrt[x], {x, 0, 4},  Frame -> True, FrameLabel -> {x, Sqrt[x], x, "x^(1/2)"}
          PlotRegion -> {{.2,.8},{.2,.8}}, ImageSize -> 300 ]
```



```
?PlotStyle
```

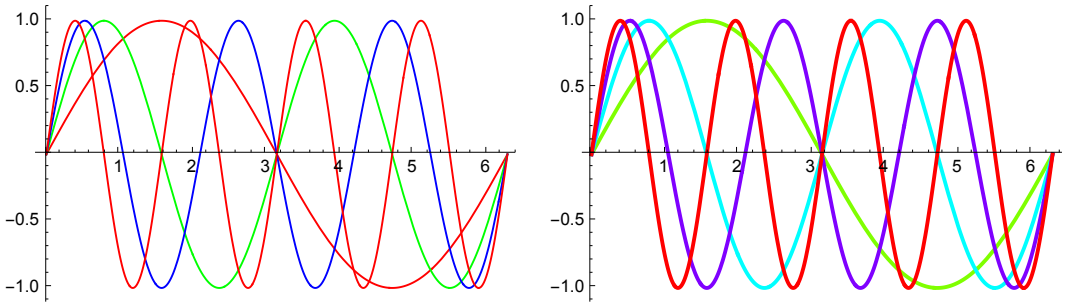
PlotStyle is an option for plotting and related functions that specifies styles in which objects are to be drawn >>

PlotStyle -> style specifies that all lines or points are to be generated with the specified graphics directive, or list of graphics directives. -

PlotStyle -> {{style1}, {style2},...} specifies that successive lines generated should use graphics directives style1, style2, ...
The styles must be included in lists, perhaps of length one. The style i directives are used cyclically.

Styles can be specified using graphics directives such as (cf.§ 6.5.2)
Dashing[], AbsoluteDashing[], Thickness[], AbsoluteThickness[], Thick; RGBColor[], CMYKColor[]; Hue[].

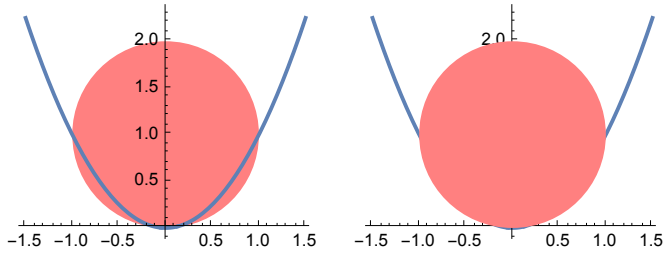
```
th = AbsoluteThickness[1];
p1 = Plot[{Sin[x], Sin[2 x], Sin[3 x], Sin[4x]}, {x,0,2π}, PlotStyle ->
   {{th, RGBColor[1,0,0]},{th, RGBColor[0,1,0]}, {th,RGBColor[0,0,1]}}];
pi = Table[Plot[Sin[k x],{x,0,2π}, PlotStyle -> {Thick,Hue[k 0.25]}],{k,4}];
p2 = Show[pi]; GraphicsRow[{p1,p2},ImageSize -> 550]
```



```
? Prolog
```

Prolog is an option for graphics functions which gives a list of graphics primitives to be rendered before the main part of the graphics is rendered >>

```
p1 = Plot[x^2, {x, -1.5, 1.5}, Prolog → {Pink, Disk[{0, 1}, 1]},
    PlotStyle → Thick, AspectRatio → Automatic];
p2 = Plot[x^2, {x, -1.5, 1.5}, Epilog → {Pink, Disk[{0, 1}, 1]},
    PlotStyle → Thick, AspectRatio → Automatic];
GraphicsRow[{p1, p2}, ImageSize → 350]
```



```
? RotateLabel
```

RotateLabel is an option for graphics and related functions
that specifies whether labels on vertical frame axes should be rotated to be vertical >>

See picture **p3** of **FrameLabel**.

```
? Style
```

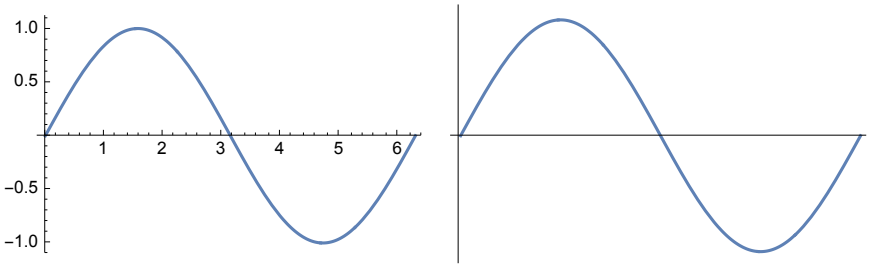
Style[*expr*, *options*] displays with *expr* formatted using the specified option settings
Style[*expr*, "*style*"] uses the option settings for the specified style in the current notebook
Style[*expr*, *color*] displays using the specified color
Style[*expr*, Bold] displays with fonts made bold
Style[*expr*, Italic] displays with fonts made italic
Style[*expr*, Underlined] displays with fonts underlined
Style[*expr*, Larger] displays with fonts made larger
Style[*expr*, Smaller] displays with fonts made smaller
Style[*expr*, *n*] displays with font size *n*.
Style[*expr*, Tiny], Style[*expr*, Small], etc. display with fonts that are tiny, small, etc. >>

see, for example, Math6-3y.nb 6.5.6

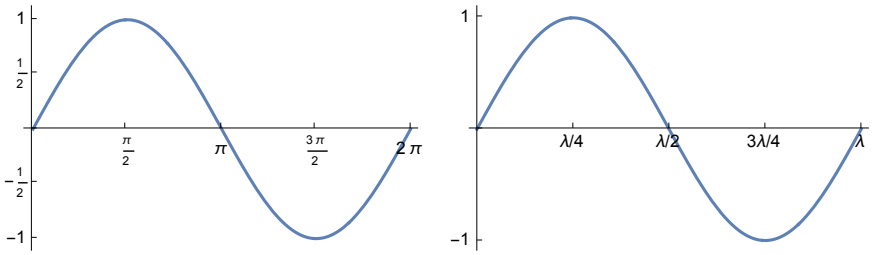
```
?Ticks
```

Ticks is an option for graphics functions that specifies tick marks for axes. >>

```
s  = Plot[  Sin[t] , {t,0,2Pi}];
p1 = Show[s, Ticks -> None];
Show[GraphicsRow[{s,p1}], ImageSize -> 450]
```




```
p1 = Show[s, Ticks -> {π Range[4]/2, Range[-1,1,1/2]} ];
p2 = Show[s, Ticks -> { {{π/2,"λ/4"},{π,"λ/2"},{3π/2,"3λ/4"},{2π,"λ"}},
                        {-1, 0, 1} } ];
Show[GraphicsRow[{p1,p2}], ImageSize -> 450]
```

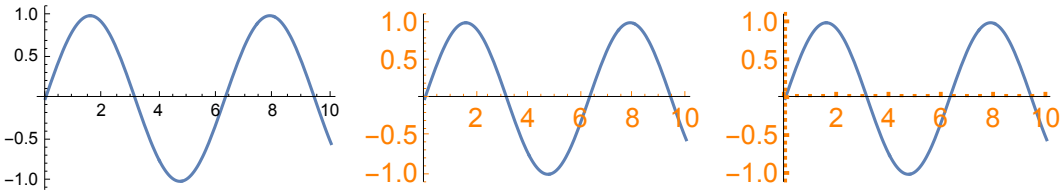



```
? TicksStyle
```

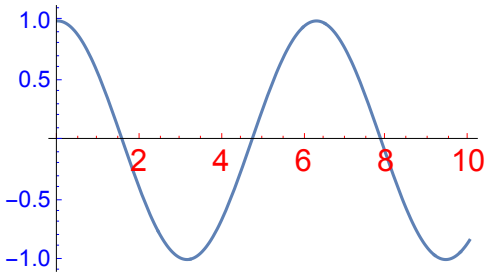
TicksStyle is an option for graphics functions which specifies how ticks should be rendered >>

TicksStyle affects tickswidth (but not their length !) and tick labels

```
p1 = Plot[Sin[x], {x, 0, 10}];
p2 = Plot[Sin[x], {x, 0, 10}, TicksStyle → Directive[Orange, 14]];
p3 = Plot[Sin[x], {x, 0, 10}, TicksStyle → Directive[Thick, Orange, 14]];
GraphicsRow[{p1, p2, p3}, ImageSize → 550]
```

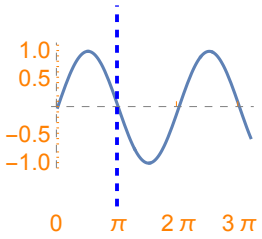


```
Plot[Cos[x], {x, 0, 10}, TicksStyle → {{16, Red}, {12, Blue}}, ImageSize →  250]
```



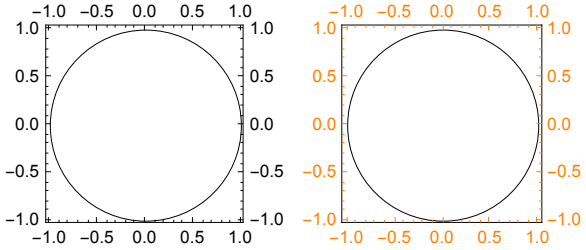
Individually styled ticks can be used with other styles, and have higher priority:

```
Plot[Sin[x], {x, 0, 10},
 Ticks → {{0, {Pi, Pi, 1, Directive[Blue, Thick]}, 2 Pi, 3 Pi}, Automatic},
 AxesStyle → Directive[Gray, Dashed], TicksStyle → Directive[Orange, 12]]
```



Frame ticks are not affected by **TicksStyle**, use **FrameTicksStyle** instead:

```
p1 = Graphics[Circle[], Frame → True, FrameTicks → All, TicksStyle → Orange];
p2 = Graphics[Circle[], Frame → True, FrameTicks → All, FrameTicksStyle → Orange];
GraphicsRow[{p1, p2}, ImageSize → 300]
```



6.1.11 Exercises

2D Graphics

Not all exercises are obligatory; Obligatory ones have a cross (+) after the number.

6.1 Plot the function cos (x) - sin (x) for the intervall [-2 π, 2 π]. From the plot find approximate values for the roots, from these accurate roots (6 figures). Give a list containing the values of the function for all this roots.

6.2 Treat the transcendental equation x = tg (x) in the following way :
Plot the left - hand and right - hand sides of this eq. for - 5 π <= x <= 7 π.
Compute a list containing all roots in the given intervall; compute another list with the values of tg (x) - x.

6.3 + Compute analytically (if possible) and numerically the roots of the equations given below. Plot these in the complex plane.
6.3.1: x7 = 1. 6.3.2: x7 + 3 x3 = 1.
6.3.3: 1/2 x ex = 1 (Verify that there is only one root within the square {{1, i}, {-1, -i}}.)

6.4 Plot the cardioid, r = 1 + cos ϕ, in polar coordinates (r, ϕ) and in rectangular coordinates.

6.5+ Plot the curve f(x) = $x^4 - 2x^3 + x^2 + 3x$ in the intevall [-1,1]. The curve shoud be blue in green background; the axes, ticks and the other notations yellow. The heading shoud give the polynomial. The coordinate axes are labelled as x and f(x).

6.6 Plot the curve $9x^2 - 12yx - 16x + 4y^2 + 8y - 27 = 0$ in two ways.

6.7 The period T of a mathematical pendulum is a function of the amplitude (maximum angle) α. T_0 ist the period for very small amplitudes. The ratio of these periods is given by: T/T_0 = $(2/\pi)$ K(k = sin(α/2)). K(k) is the complete elliptical integral of the first kind as a function of the modulus k. Mathematica computes this function as a function of the parameter m = k^2 by the command **EllipticK[m].**
This function becomes logarithmically infinite for m = 1, i.e. for $\alpha = \pi$.
Plot T/T0 versus α . The drawing should have a frame and gridlines. The axes should have labels α and T/T0 . The figure should have the heading: Mathematical Pendulum.
The ordinate should extend from 0 to 7.
There should be ticks for every half integer, the integers 1, 2, 3,....7 should be written and horizontal gridlines drawn.
The abscissa should have ticks for every ten degrees, numbers and gridlines are required for every 30 degrees.

6.8 Plot both branches of the hyperbola y = k1 k2/x (with k1 = 3, k2 = 2) for - k1 ≤ x < 0, 0 < x ≤ k1. Axes have labels x, y respectively. There are only ticks and corresponding labels for ± k1 and ± k2.
The parts of the branches belonging to |x| < k1 should be thicker. The two points ± (k1, k2) should be marked by thick dots.
Heading is: Branch cuts of square root.
Prepare the same drawing with red curve for |x| < k1 and yellow curves for |x| > k1.

6.9 Plot the curve

$$V(r) = r^2 \quad \text{for } 0 \le r \le 1,$$
$$= 1/r \quad \text{for } 1 \le r \le 3.$$

Abscissa has label r, ordinate V(r). Continue the two pieces making up V(r) in their complementary ranges (1 ≤ r ≤ 3, 0 ≤ r ≤ 1) in dashed form.

6.10+ Draw a regular pentagon; its edges should be about 1.5 mm thick independent of the size of the drawing. Thin circular arcs should be drawn at the outside form a thin continuation of the preceeding edge to the next edge. These arcs should be labelled by Greek letters α, β, χ, δ, ϵ. The corners should be labelled by letters A, B, C, D, E located in the interior.

6.11+ Plot the curves x and x^2/2, both with the AspectRatio = 1 and display them in a row.