

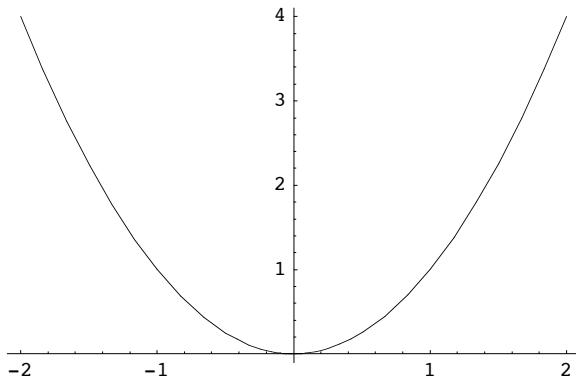
Introduction to *Mathematica*: Graphics in 2 Dimensions

Bob Bradshaw, Ohlone College

Simple Plots

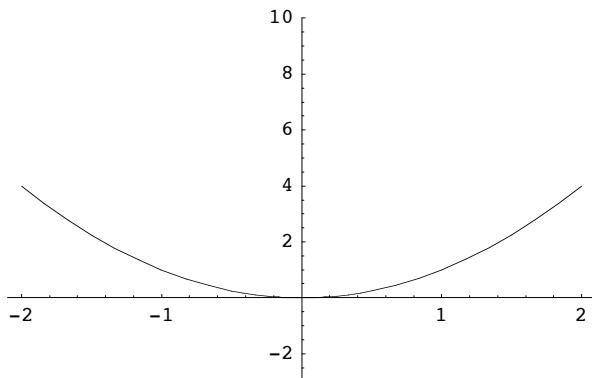
The standard **Plot** command has the form **Plot[expression, {x, start, end}, PlotStyle->{settings}]**

In[8]:= Plot[x^2, {x, -2, 2}];



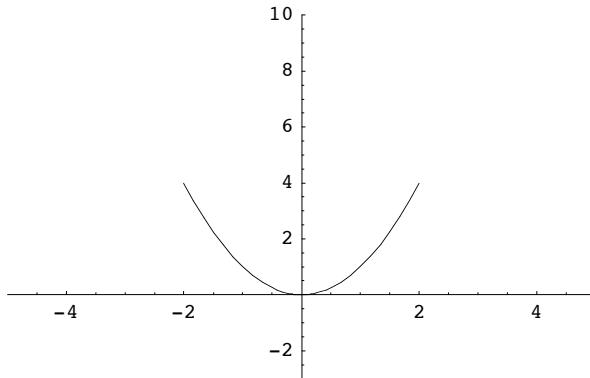
You can also set the y values using the **PlotRange** option.

In[9]:= Plot[x^2, {x, -2, 2}, PlotRange -> {-3, 10}];



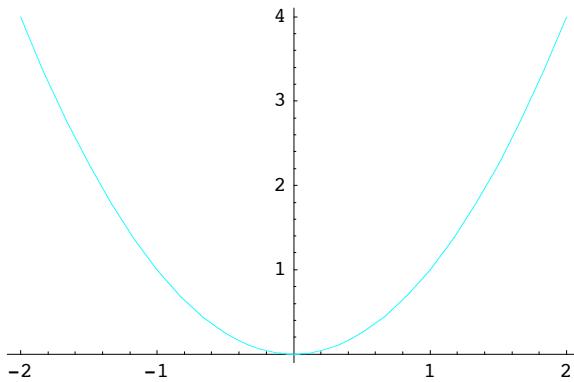
The **PlotRange** option also allows you to show a graph on a large region than is actually used in the plot.

```
In[10]:= Plot[x^2, {x, -2, 2}, PlotRange -> {{-5, 5}, {-3, 10}}];
```



Unlike *Maple*, *Mathematica* requires you to enter the options within the command, not after the graph is drawn.

```
In[11]:= Plot[x^2, {x, -2, 2}, PlotStyle -> {Hue[0.5]}];
```



Colors

Colors can be declared either by the `Hue[#]` or `RGB[#, #, #]` commands. Alternatively, loading the following package allows you to use the name of the color.

```
In[12]:= << Graphics`Colors`
```

In[13]:= **AllColors**

Out[13]= {AliceBlue, AlizarinCrimson, Antique, Apricot, Aquamarine, AureolineYellow, Azure, Banana, Beige, Bisque, Black, BlanchedAlmond, Blue, BlueViolet, Brick, Brown, BrownMadder, BrownOchre, Burlywood, BurntSienna, BurntUmber, CadetBlue, CadmiumLemon, CadmiumOrange, CadmiumYellow, Carrot, Cerulean, Chartreuse, Chocolate, ChromeOxideGreen, CinnabarGreen, Cobalt, CobaltGreen, ColdGray, Coral, CornflowerBlue, Cornsilk, Cyan, CyanWhite, DarkGoldenrod, DarkGreen, DarkKhaki, DarkOliveGreen, DarkOrange, DarkOrchid, DarkSeaGreen, DarkSlateBlue, DarkSlateGray, DarkTurquoise, DarkViolet, DeepCadmiumRed, DeepCobaltViolet, DeepMadderLake, DeepNaplesYellow, DeepOchre, DeepPink, DeepSkyBlue, DimGray, DodgerBlue, Eggshell, EmeraldGreen, EnglishRed, Firebrick, Floral, ForestGreen, Gainsboro, GeraniumLake, Ghost, Gold, Goldenrod, GoldOchre, Gray, Green, GreenishUmber, GreenYellow, Honeydew, HotPink, IndianRed, Indigo, Ivory, IvoryBlack, Khaki, LampBlack, Lavender, LavenderBlush, LawnGreen, LemonChiffon, LightBeige, LightBlue, LightCadmiumRed, LightCadmiumYellow, LightCoral, LightGoldenrod, LightGray, LightPink, LightSalmon, LightSeaGreen, LightSkyBlue, LightSlateBlue, LightSlateGray, LightSteelBlue, LightViridian, LightYellow, LimeGreen, Linen, Magenta, ManganeseBlue, Maroon, MarsOrange, MarsYellow, MediumAquamarine, MediumBlue, MediumOrchid, MediumPurple, MediumSeaGreen, MediumSlateBlue, MediumSpringGreen, MediumTurquoise, MediumVioletRed, Melon, MidnightBlue, Mint, MintCream, MistyRose, Moccasin, Navajo, Navy, NavyBlue, Oak, OldLace, Olive, OliveDrab, Orange, OrangeRed, Orchid, PaleGoldenrod, PaleGreen, PaleTurquoise, PaleVioletRed, PapayaWhip, Peach, PeachPuff, Peacock, PermanentGreen, PermanentRedViolet, Peru, Pink, Plum, PowderBlue, PrussianBlue, Purple, Raspberry, RawSienna, RawUmber, Red, RoseMadder, RosyBrown, RoyalBlue, SaddleBrown, Salmon, SandyBrown, SapGreen, SeaGreen, Seashell, Sepia, Sienna, SkyBlue, SlateBlue, SlateGray, Smoke, Snow, SpringGreen, SteelBlue, TerreVerte, Thistle, Titanium, Tomato, Turquoise, TurquoiseBlue, Ultramarine, UltramarineViolet, VanDykeBrown, VenetianRed, Violet, VioletRed, WarmGray, Wheat, White, Yellow, YellowBrown, YellowGreen, YellowOchre, Zinc}

To see what these colors are, see the colors.nb notebook <http://online.ohlone.edu/math/bbradshaw/math111/colors.nb>

Options

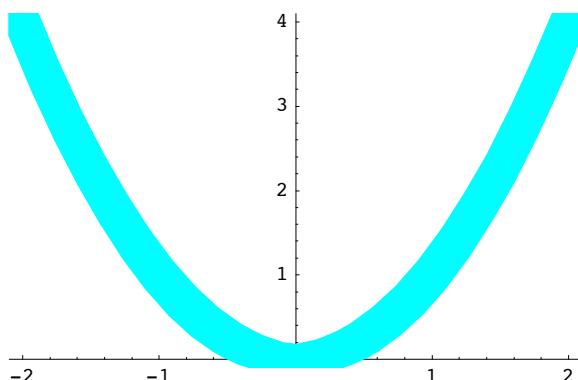
Mathematica has a full set of options for controlling every characteristic of your graph. For full details, see **plot_options.nb** (http://online.ohlone.edu/math/bbradshaw/math111/plot_options.nb) and **plot_style.nb** (http://online.ohlone.edu/math/bbradshaw/math111/plot_style.nb)

In general, **PlotStyle** controls the appearance of the curve while **PlotOptions** controls the appearance of the entire graph.

For example, we can change the thickness of the curve using either the **Thickness** or **AbsoluteThickness** options in **PlotStyle**.

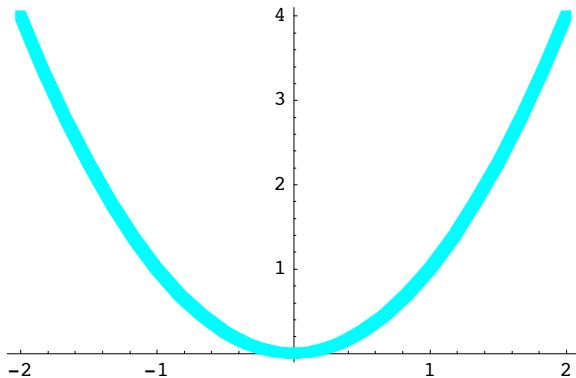
The thickness of a curve is set as a percentage of the width of the domain.

In[14]:= **Plot[x^2, {x, -2, 2}, PlotStyle -> {Hue[0.5], Thickness[0.05]}];**



You can also set the thickness of a curve in terms of points (12 points per inch.)

```
In[15]:= Plot[x^2, {x, -2, 2}, PlotStyle -> {Hue[0.5], AbsoluteThickness[5]}];
```

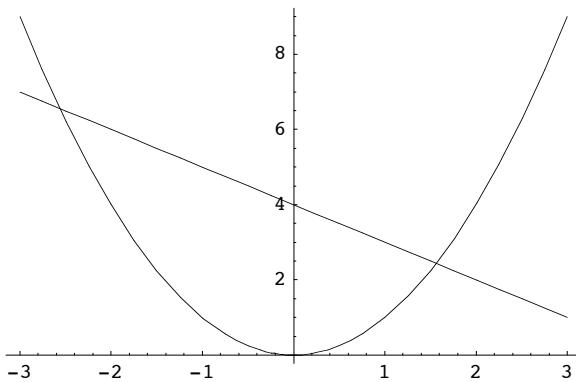


Combining Plots

Two graphs can be combined either by placing them in one statement or by graphing separately and then combining the results.

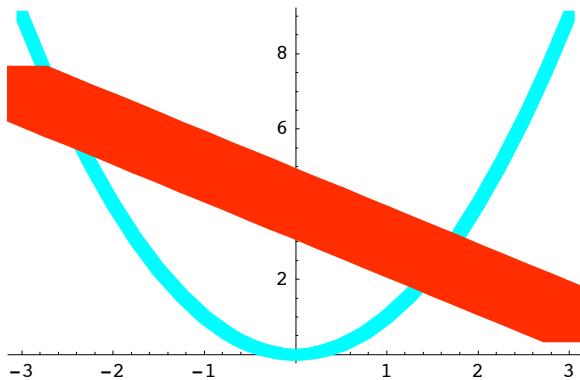
To graph two curves simultaneously, use braces {} around the functions.

```
In[16]:= Plot[{x^2, 4 - x}, {x, -3, 3}];
```



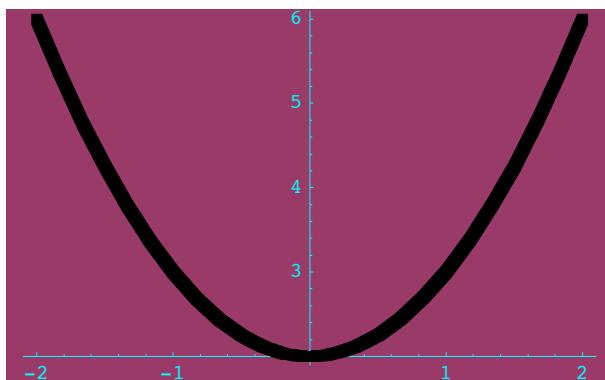
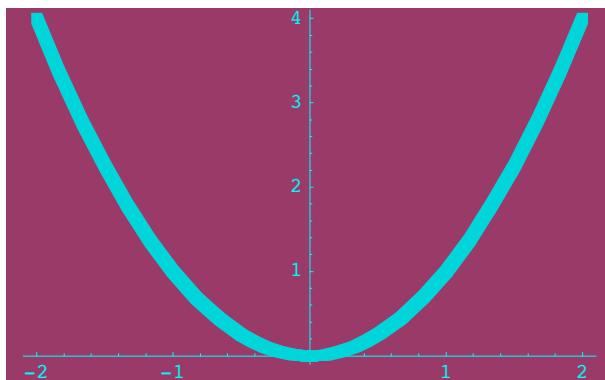
To include options, each function has its own set of options in **PlotStyle**, again enclosed in braces.

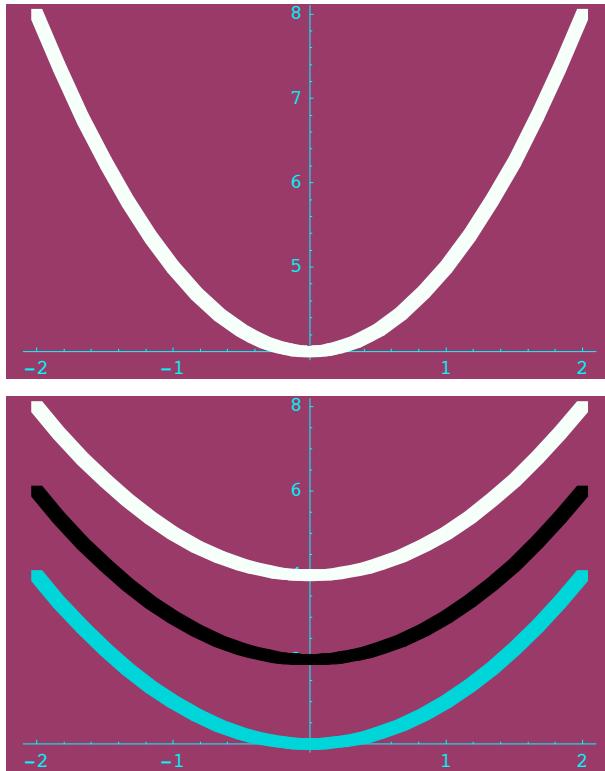
```
In[17]:= Plot[{x^2, 4 - x}, {x, -3, 3},
  PlotStyle -> {{Hue[0.5], Thickness[0.02]}, {Hue[0.01], Thickness[0.09]}}];
```



The second way to combine plots is by naming separate plots, and then combining them by using the **Show** [] command.

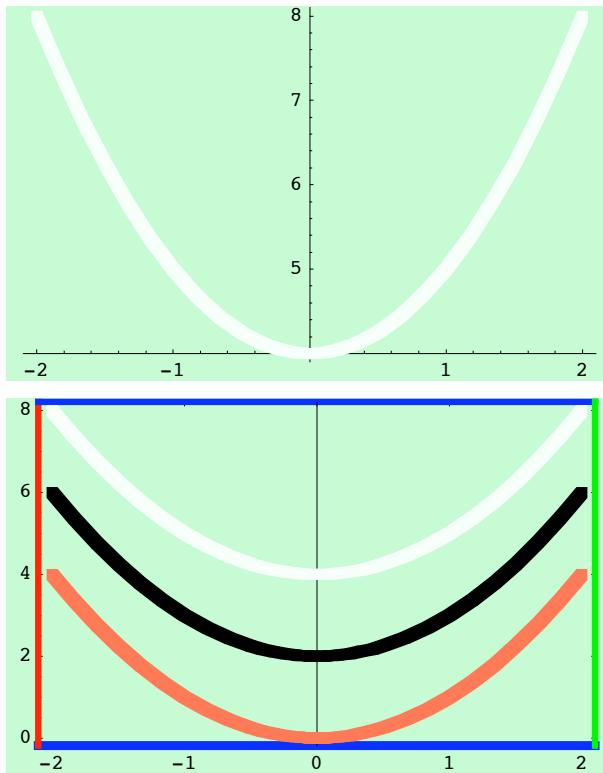
```
In[18]:= p1 = Plot[x^2, {x, -2, 2}, Background -> Raspberry,
  PlotStyle -> {DarkTurquoise, Thickness[0.02]}];
p2 = Plot[x^2 + 2, {x, -2, 2}, Background -> Raspberry,
  PlotStyle -> {Black, Thickness[0.02]}];
p3 = Plot[x^2 + 4, {x, -2, 2}, Background -> Raspberry,
  PlotStyle -> {MintCream, Thickness[0.02]}];
Show[
  p1,
  p2,
  p3];
```





Notice that ending the **Plot** command with a semicolon only suppressed the "Graphics" output, not the graph itself.
To suppress the graph, use **DisplayFunction->Identity**.
To resume displaying a graph, use **DisplayFunction->\$DisplayFunction**

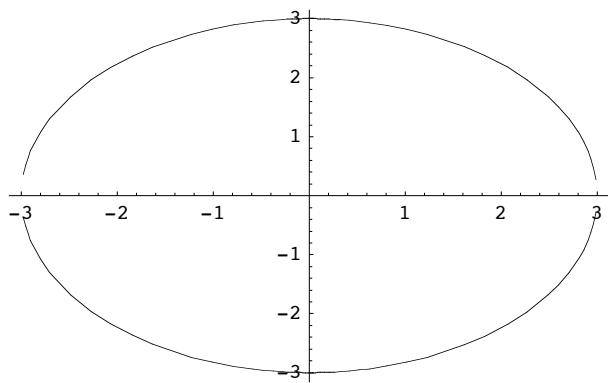
```
In[22]:= p1 = Plot[x^2, {x, -2, 2},
  Background -> Mint,
  PlotStyle -> {Tomato, Thickness[0.02]},
  DisplayFunction -> Identity];
p2 = Plot[x^2 + 2, {x, -2, 2},
  Background -> Mint,
  PlotStyle -> {Black, Thickness[0.02]},
  DisplayFunction -> Identity];
p3 = Plot[x^2 + 4, {x, -2, 2},
  Background -> Mint,
  PlotStyle -> {MintCream, Thickness[0.02]}];
Show[p1, p2, p3,
  DisplayFunction -> $DisplayFunction,
  Frame -> True,
  FrameStyle -> {{Blue, Thickness[0.017]}, {Red, Thickness[0.012]}, {Blue, Thickness[0.012]}, {Green, Thickness[0.012]}}];
```



Typical Plotting Problems

The plotting routines are not perfect. This is supposed to be the graph of a circle.

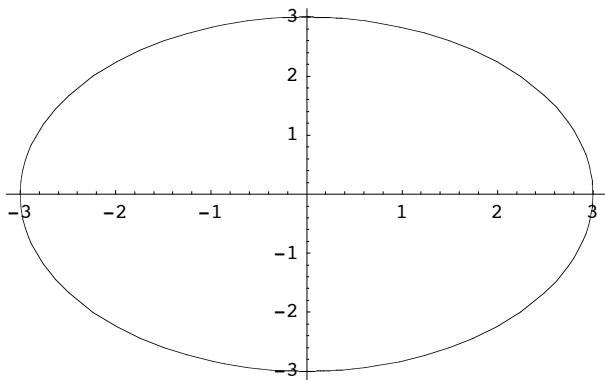
```
In[26]:= y3 = Sqrt[9 - x^2];
y4 = -y3;
Plot[{y3, y4}, {x, -10, 10}];
```



The error messages are due to the function being undefined at $x = -10$, etc. However, there are also gaps in the graph and the graph is oval, not circular.

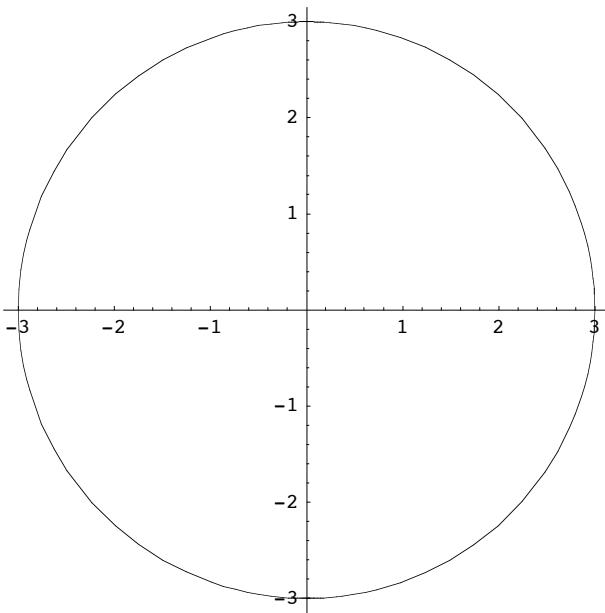
We can fix the errors and the gaps by changing the domain.

```
In[29]:= Plot[{y3, y4}, {x, -3, 3}];
```



We can fix the oval by using the **AspectRatio** option.

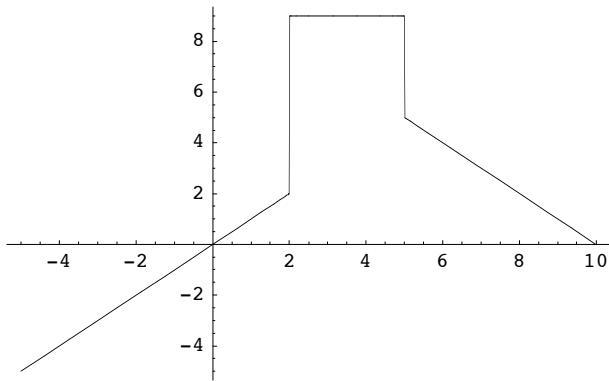
```
In[30]:= Plot[{y3, y4}, {x, -3, 3}, AspectRatio -> Automatic];
```



Piecewise Functions

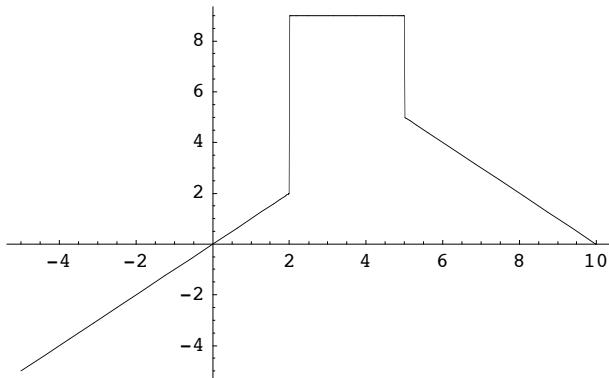
Mathematica can create and graph piecewise continuous functions. To create a piecewise function, use the `/;` notation to specify the domain. Notice that the resulting graph is incorrect since it is shown as being continuous.

```
In[31]:= f[x_] := x /; x < 2
f[x_] := 9 /; 2 ≤ x ≤ 5
f[x_] := 10 - x /; x > 5
Plot[f[x], {x, -5, 10}];
```



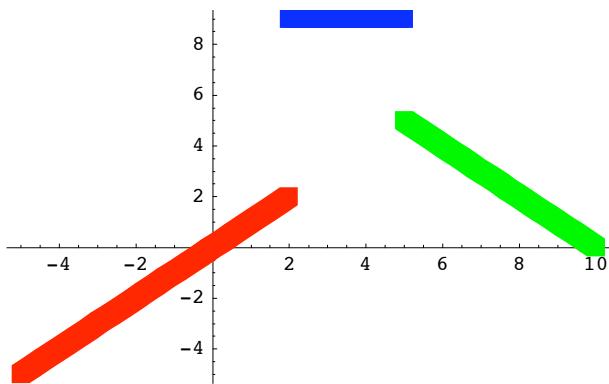
A more compact but harder to read version of the same function can be created using the **Which** command.

```
In[35]:= h[x_] := Which[x < 2, x, 2 ≤ x ≤ 5, 9, x > 5, 10 - x]
In[36]:= Plot[h[x], {x, -5, 10}];
```



In order to create a discontinuous graph, plot the portions separately and then combine using the **Show** command.

```
In[37]:= p1 :=
  Plot[f[x], {x, -5, 2}, DisplayFunction → Identity, PlotStyle → {Red, Thickness[0.03]}];
p2 := Plot[f[x], {x, 2, 5}, DisplayFunction → Identity,
  PlotStyle → {Blue, Thickness[0.03]}];
p3 := Plot[f[x], {x, 5, 10}, DisplayFunction → Identity,
  PlotStyle → {Green, Thickness[0.03]}];
Show[p1, p2, p3, DisplayFunction → $DisplayFunction];
```

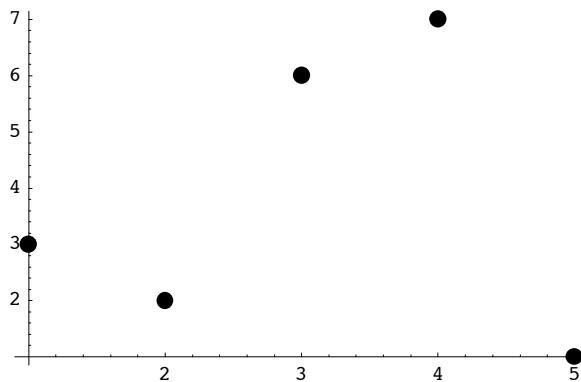


Plotting a List of Points

Mathematica can create a graph of list of points in a couple of ways. One way is to create a list of points and then use the **ListPlot** command.

```
In[41]:= mypoints = {{1, 3}, {2, 2}, {3, 6}, {4, 7}, {5, 1}}
ListPlot[mypoints, PlotStyle → PointSize[0.03]];
```

```
Out[41]= {{1, 3}, {2, 2}, {3, 6}, {4, 7}, {5, 1}}
```

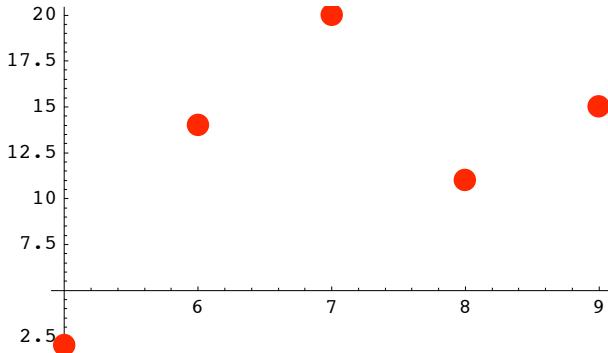


If you have a large number of data, entering the data in this form can be tedious. Instead, you can use the table editor. Enter the name of your data and then the "=" sign. Next choose **Input->Create Table/Matrix/Palette** (or **SHIFT** **CTRL** **C**) and choose the Table option. Enter the number of rows and columns, close the dialog box and then enter the data in the table. You can use the **TAB** key to move between cells.

```
In[43]:= mypoints2 = {{5, 2}, {6, 14}, {7, 20}, {8, 11}, {9, 15}}
```

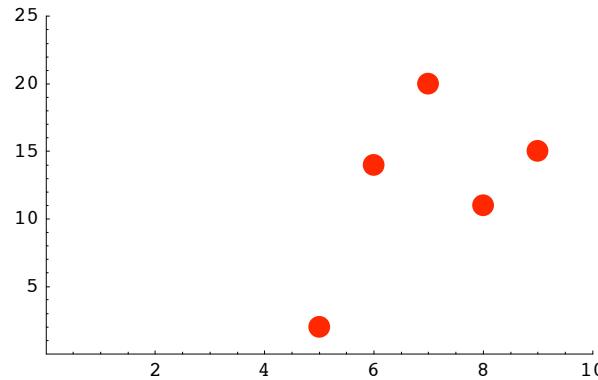
```
Out[43]= {{5, 2}, {6, 14}, {7, 20}, {8, 11}, {9, 15}}
```

```
In[44]:= ListPlot[mypoints2, PlotStyle -> {Red, PointSize[0.04]}];
```



For some reason I don't understand, the axis is above the last point. This can be fixed using the option **PlotRange**.

```
In[45]:= ListPlot[mypoints2, PlotRange -> {{0, 10}, {0, 25}}, PlotStyle -> {Red, PointSize[0.04]}];
```



You Try It!

1. Graph the following equation. Adjust the x and y values to create an appropriate viewing rectangle.
 $y = 58x^4 - x^3 + 67x^2 - x + 1$
2. Adjust the above graph so that the axes are in a large font. Also create a title for the graph.
3. Create a single graph containing the graphs of the following functions on the interval $-2\pi < x < 2\pi$ using three different colors and line thicknesses.
 $y = \sin(2x)$, $y = 2\sin(2x)$, and $y = 3\sin(2x)$
4. Create a single graph containing the graphs of the following functions, using three different symbols. The graphs should consist of points, not a smooth curve.
 $y = \ln(x)$, $y = \ln(2x)$, and $y = \ln(3x)$
5. Create the graph of a piecewise function of your choice using both methods listed above.