PLOTTING AND GRAPHICS OPTIONS IN MATHEMATICA

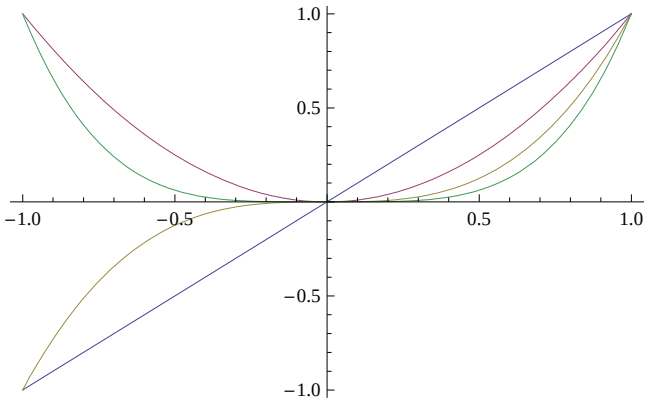
In addition to being a powerful programming tool, Mathematica allows a wide array of plotting and graphing options. We will look at a variety of these, starting with the Plot command.

The examples shown below merely scratch the surface of what you can do with *Mathematica*. I urge you to use the online Documentation Center (Doc Center as I refer to it throughout the write up) to see the range of possibilities with some very nice examples. This is intended to get you started; the best way to learn is through trial and error.

The Plot Command

We have already encountered some very simple versions of this. You have already learned how to plot several functions on a single graph :

```
Plot[{x, x^2, x^3, x^4}, {x, −1, 1}]
```



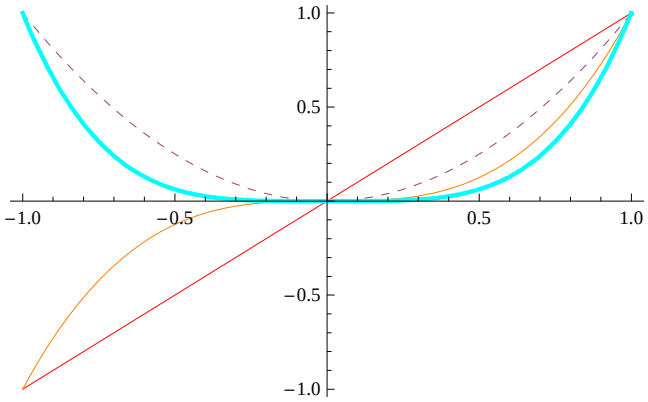
This plots the curves x, x^2, x^3, and x^4 between -1 and 1 on the same set of axes.

There are probably hundreds of options and styles we can apply to customize our diagram. The best way to learn about the various ways is to look up Plot in the online documentation center and then try out as many of the options as you can.

Make sure you also click on PlotStyle and see the range of choices that gives you. For instance,

suppose we want to customize the graph above by making the x curve a red line, x^2 curve a dashed line, the x^3 curve an orange line, and the x^4 curve a thick line, we would input:

```
Plot[{x, x^2, x^3, x^4}, {x, −1, 1}, PlotStyle → {Red, Dashed, Orange, Thick}]
```



And now we can readily tell one graph from another. In the PlotStyle command above, be aware of the following :

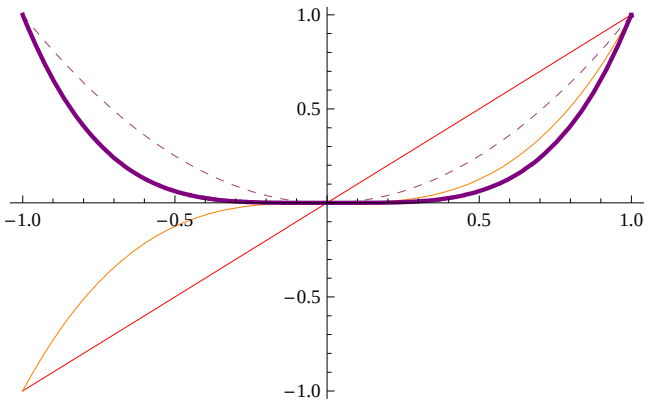
• The P and S are capitalized, so make sure you write PlotStyle

• In order to draw the arrow, your keystrokes should be the minus sign followed immediately (no space) by the greater than key. The first keystroke after inputting these (->) will generate the arrow you see.

• Notice that the styles Red, Dashed, Orange, Thick all start with capital letters, and the list of commands is contained in braces.

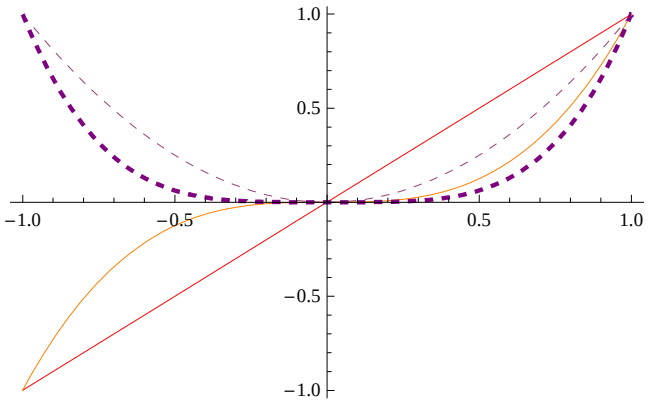
Suppose in the graph above that we want to change the color of the x^4 curve to purple (but keep the thickness of the line), we can nest commands:

```
Plot[{x, x^2, x^3, x^4}, {x, −1, 1}, PlotStyle → {Red, Dashed, Orange, {Thick, Purple}}]
```



So the fourth command is nested; if we want a thick, dashed, purple line, we can :

```
Plot[{x, x^2, x^3, x^4}, {x, −1, 1}, PlotStyle → {Red, Dashed, Orange, {Thick, Dashed, Purple}}]
```



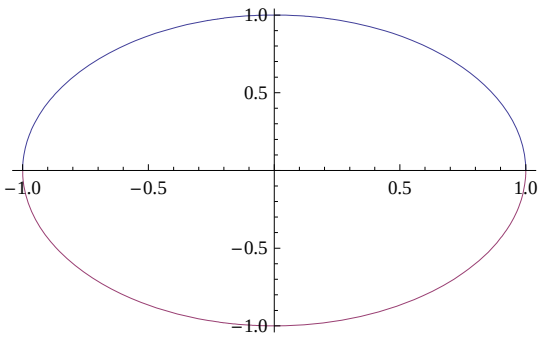
and you get the idea.

AspectRatio

Let' s plot a circle of radius one; I will do this by plotting on the same axes the two solutions to the equation :

$$x^2 + y^2 = 1 \Rightarrow y = \pm\sqrt{1 - x^2}$$

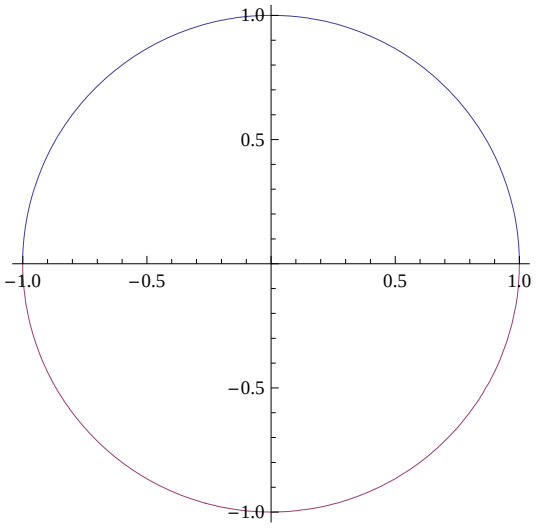
```
Plot[{Sqrt[1 − x^2], −Sqrt[1 − x^2]}, {x, −1, 1}]
```



Now, you know this is supposed to be a circle. It just doesn' t look much like one. But before you conclude either I or Mathematica have messed up, look carefully at this curve; this curve goes through the points (1, 0), (0, 1), (-1, 0) and (0, -1), just as a circle does. So why does it look like an ellipse?

The reason is that Mathematica' s plotting program assumes that the ratio of width to height is equal to 1/the golden ratio. If we want to plot this to look like a circle, we input :

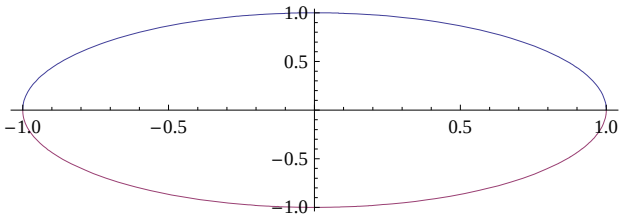
```
Plot[{Sqrt[1 - x^2], -Sqrt[1 - x^2]}, {x, -1, 1}, AspectRatio → Automatic]
```



voila.

Or, we can use the AspectRatio command to make an even more oblate shape (but the figure is still a circle):

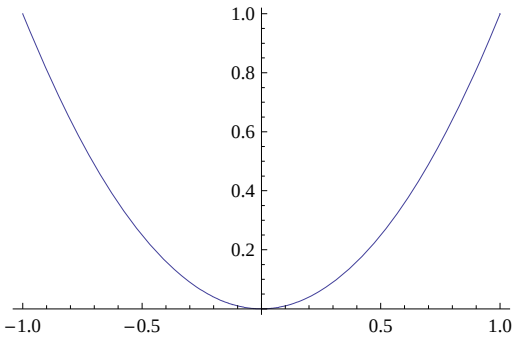
```
Plot[{Sqrt[1 - x^2], -Sqrt[1 - x^2]}, {x, -1, 1}, AspectRatio → 1 / 3]
```



Axes or no Axes :

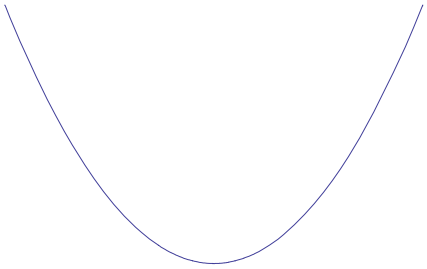
Mathematica assumes you want axes :

```
Plot[x^2, {x, -1, 1}]
```



But if you don' t :

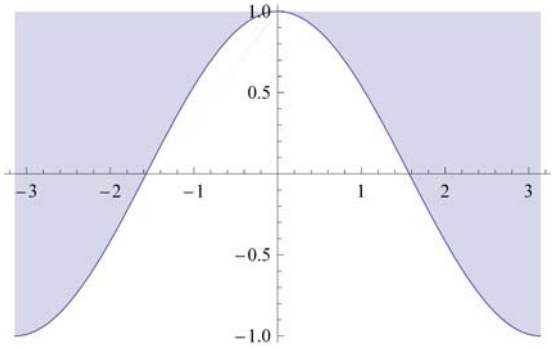
```
Plot[x^2, {x, −1, 1}, Axes → False]
```



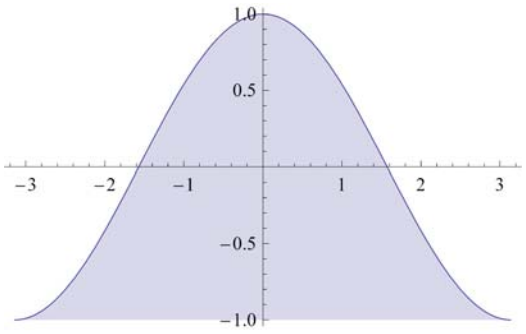
Filling

Let' s consider the graph of cos x :

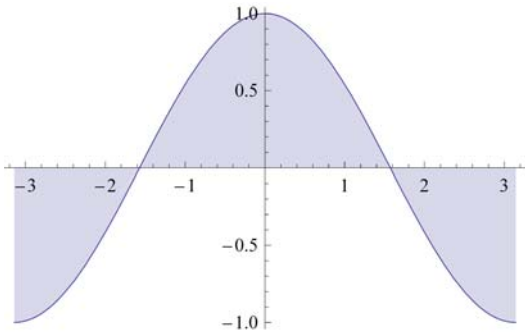
```
Plot[Cos[x], {x, −π, π}, Filling → Top]
```




```
Plot[Cos[x], {x, −π, π}, Filling → Bottom]
```

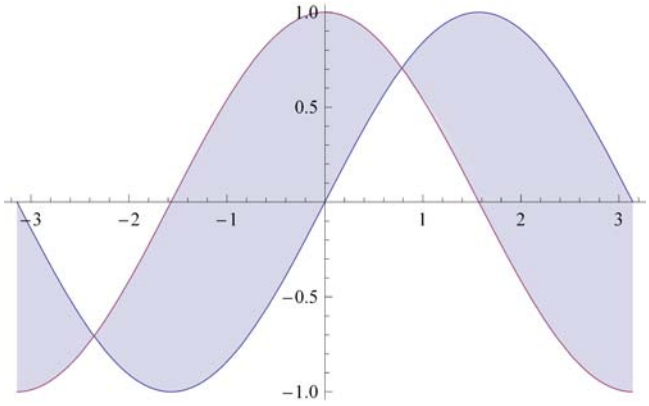


```
Plot[Cos[x], {x, −π, π}, Filling → Axis]
```



Let' s say you are writing a book on multivariable calculus and want to show the region defined between the curves sin x and cos x :

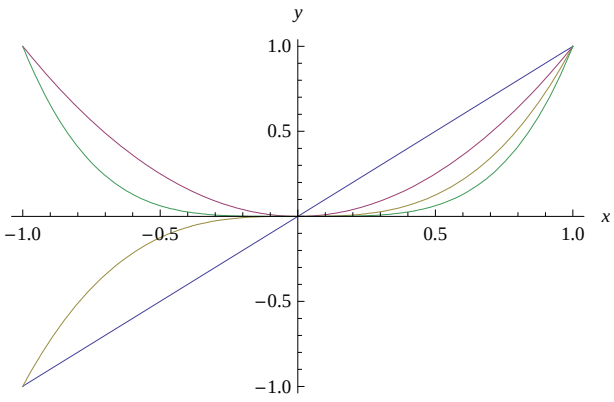
```
Plot[{Sin[x], Cos[x]}, {x, −π, π}, Filling → {1 → {2}}]
```



Labeling Axes and Plots :

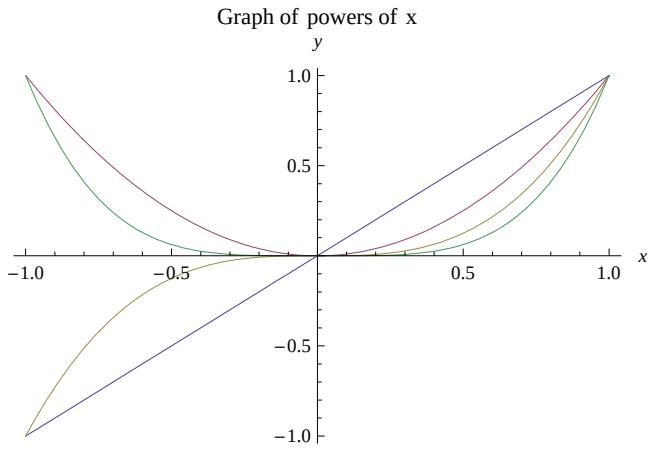
The first plot, with axes labelled :

```
Plot[{x, x^2, x^3, x^4}, {x, -1, 1}, AxesLabel → {x, y}]
```



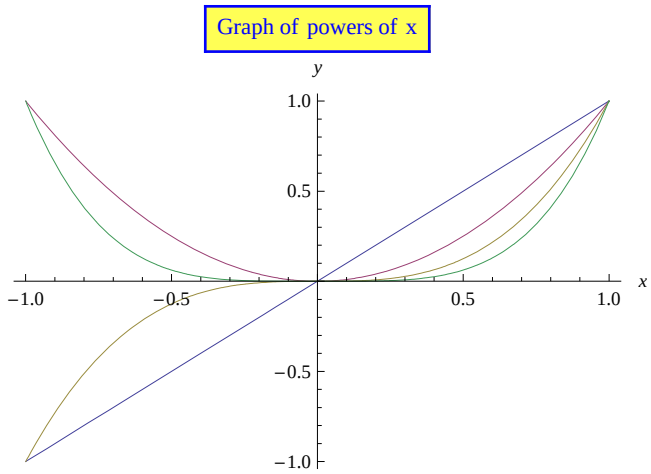
Now with axes labelled and a plot label :

```
Plot[{x, x^2, x^3, x^4}, {x, -1, 1}, AxesLabel → {x, y}, PlotLabel → "Graph of powers of x"]
```



Notice that text is put within quotes. Or to really jazz it up (this is an example on the Mathematica website) :

```
Plot[{x, x^2, x^3, x^4}, {x, -1, 1}, AxesLabel → {x, y},
 PlotLabel → Style[Framed["Graph of powers of x"], Blue, Background → Lighter[Yellow]]]
```

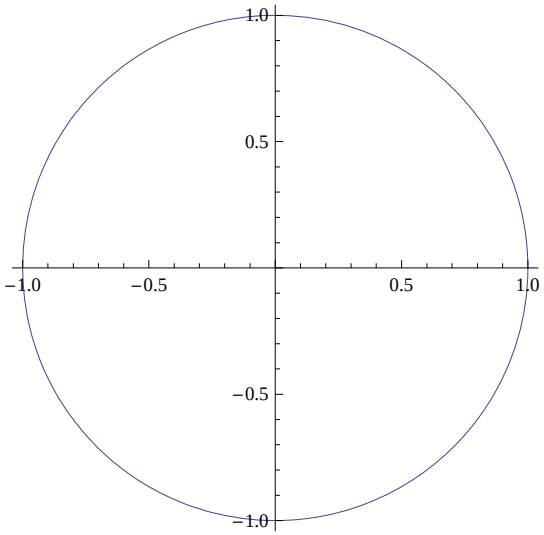


As you will see from even a cursory glance at the Doc Center, there are many, many ways you can spruce up your graphs. Now onto other topics.

PolarPlot

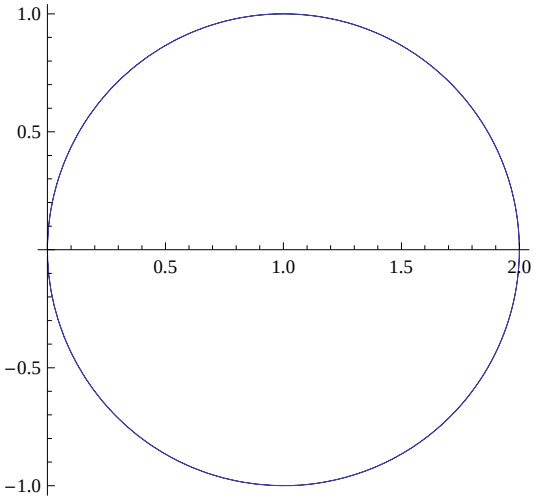
All of the functions we plotted above were written in Cartesian coordinates. Mathematica allows us to plot graphs using plane polar coordinates. (Read PolarPlot on the Doc Center). We could plot a circle of radius 1 by :

```
PolarPlot[1, {θ, 0, 2 π}]
```



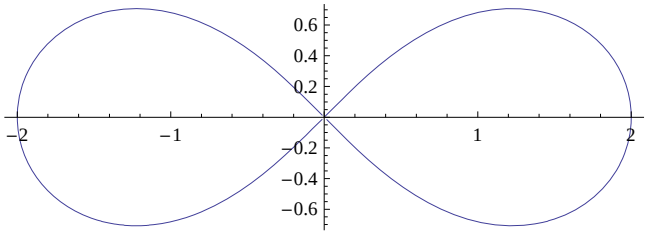
What does the curve r = 2 a cos θ look like?

```
PolarPlot[2 Cos[θ], {θ, 0, 2 π}]
```



Or the curve $\left(r^2\right) = a^2 \cos 2\theta$ (for a=2):

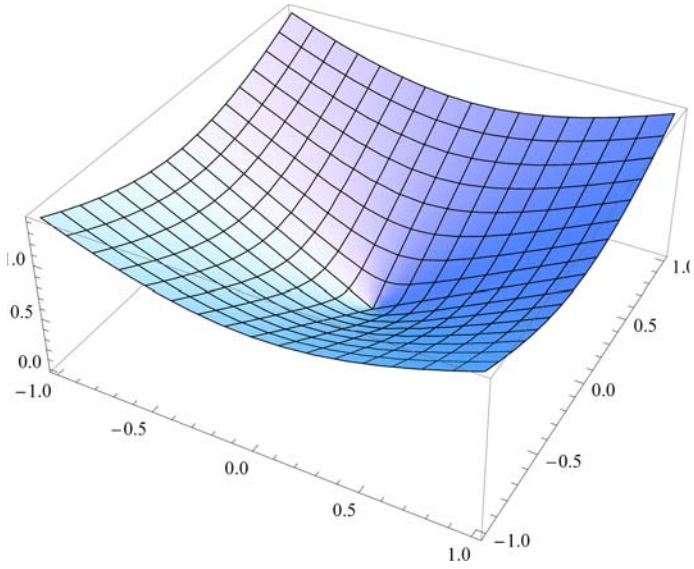
```
PolarPlot[2 Sqrt[Cos[2 θ]], {θ, 0, 2 π}]
```



Plotting in 3 Dimensions :

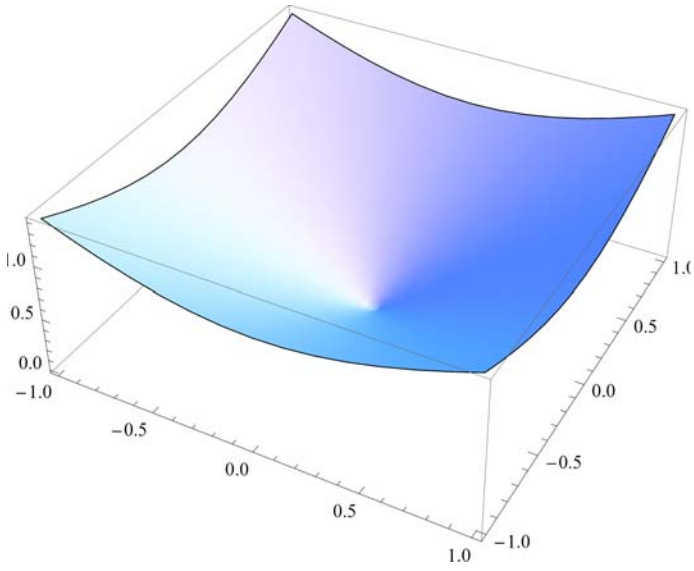
To plot a function z = f (x, y) :

```
Plot3D[Sqrt[x^2 + y^2], {x, -1, 1}, {y, -1, 1}]
```

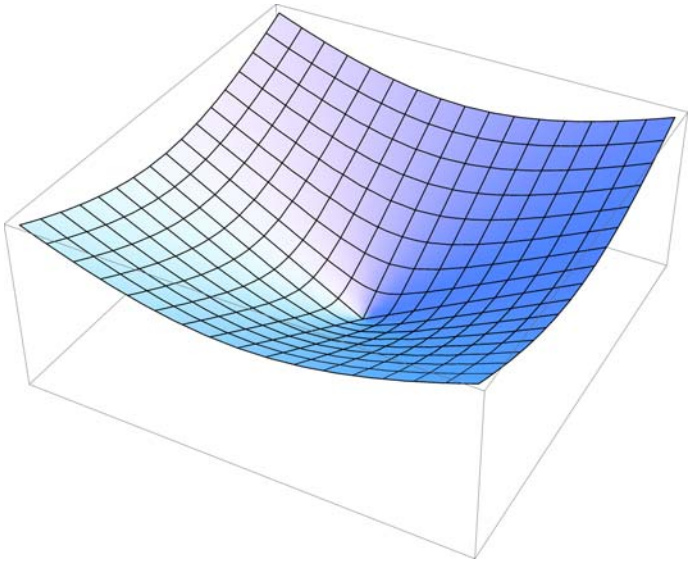


Some options :

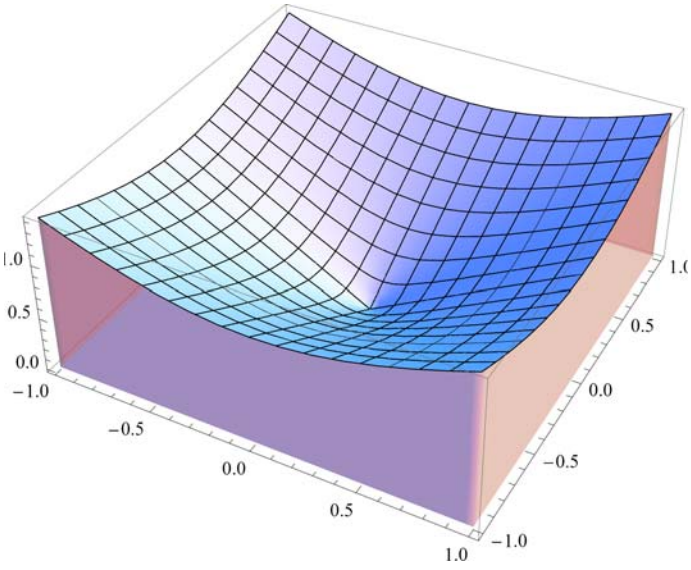
```
Plot3D[Sqrt[x^2 + y^2], {x, -1, 1}, {y, -1, 1}, Mesh → None]
```



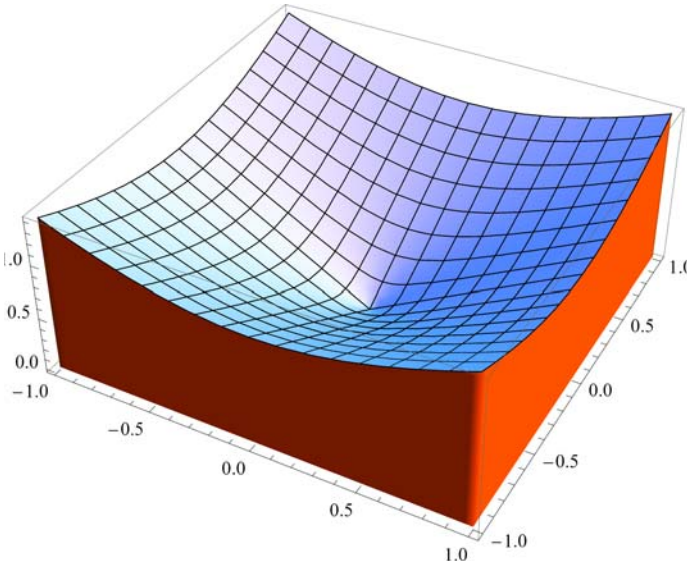
```
Plot3D[Sqrt[x^2 + y^2], {x, -1, 1}, {y, -1, 1}, Axes → False]
```



```
Plot3D[Sqrt[x^2 + y^2], {x, -1, 1}, {y, -1, 1}, Filling → Bottom]
```



```
Plot3D[Sqrt[x^2 + y^2], {x, -1, 1}, {y, -1, 1}, Filling → Bottom, FillingStyle → Orange]
```

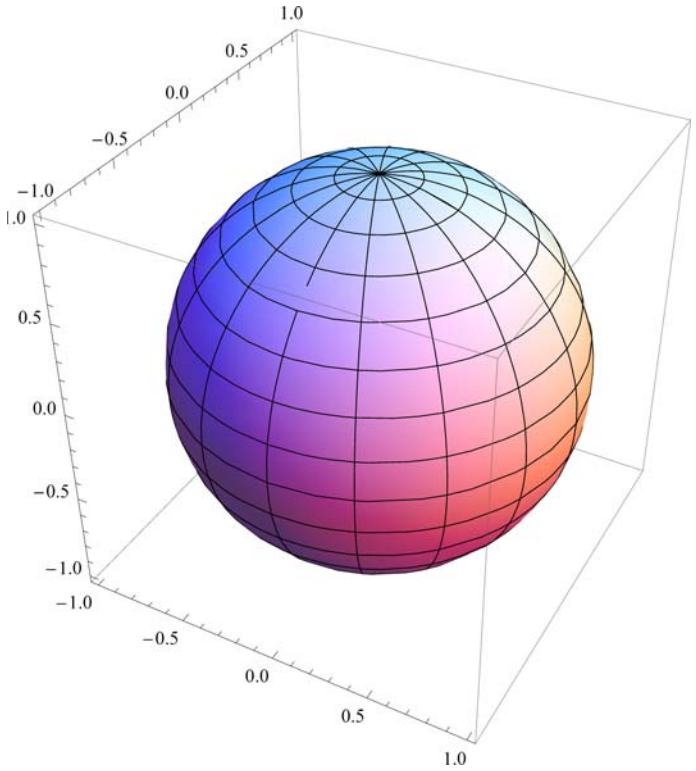


SphericalPlot3D

To plot in three dimensional spherical coordinates, use the SphericalPlot3D command. It will take a function of r, θ, ϕ and plot it in spherical coordinates.

First, a sphere of radius 1 is plotted as :

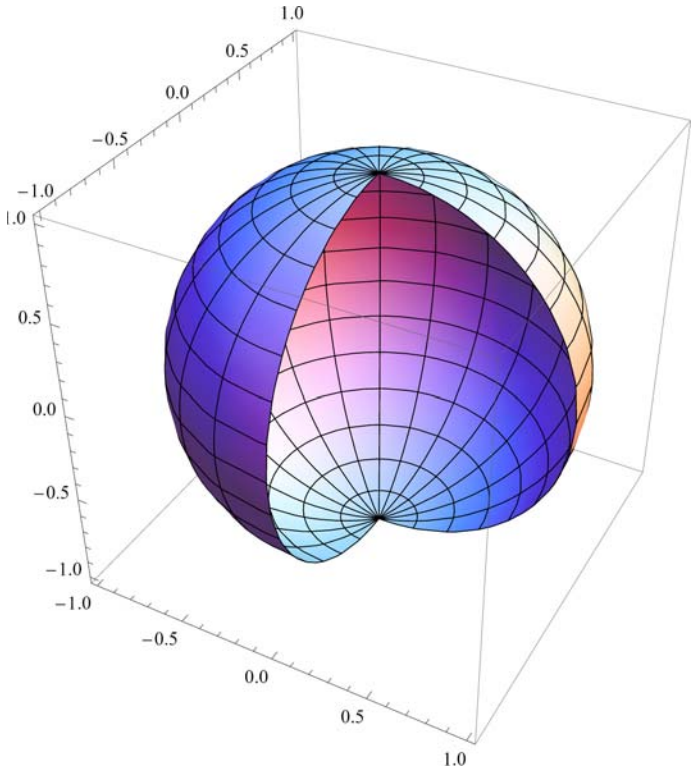
SphericalPlot3D[1, {θ, 0, π}, {ϕ, 0, 2 π}]



The two sets of braces mean you are plotting over the polar angle, θ, from 0 to π, and over the azimuthal angle, ϕ, from 0 to 2π.

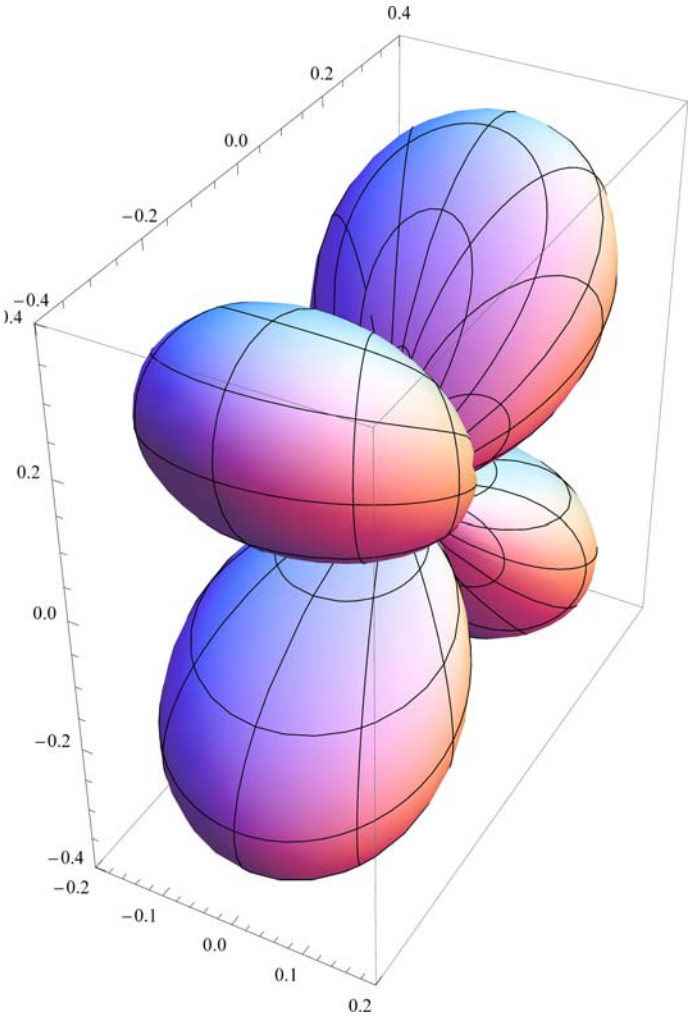
We can peer into the sphere by plotting over only 270 degrees of azimuth.

```
SphericalPlot3D[1, {θ, 0, π}, {ϕ, 0, 3 π / 2}]
```



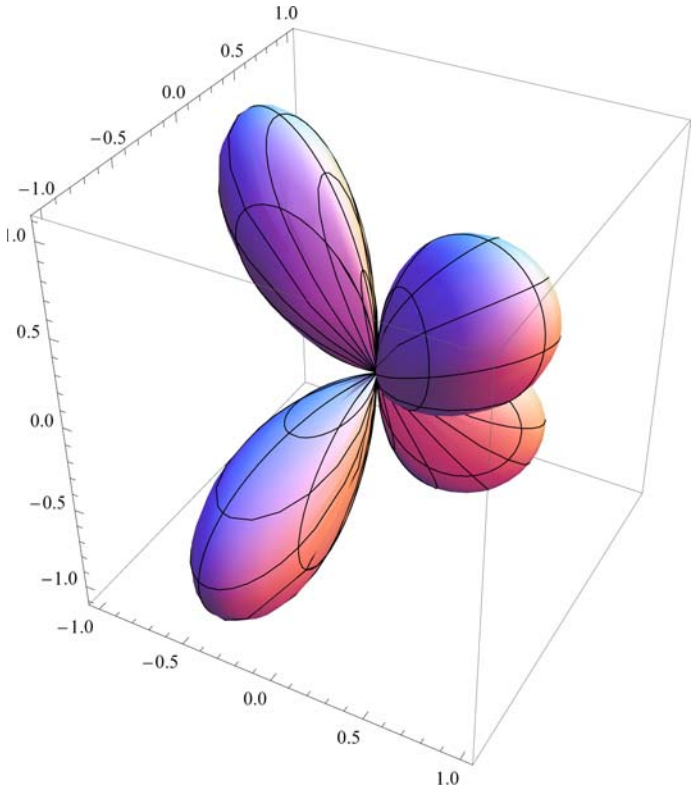
For all you chemists out there:we plot one of the d orbitals :

```
SphericalPlot3D[Sin[θ] Cos[θ] Sin[ϕ], {θ, 0, π}, {ϕ, 0, 2 π}]
```



And another d orbital :

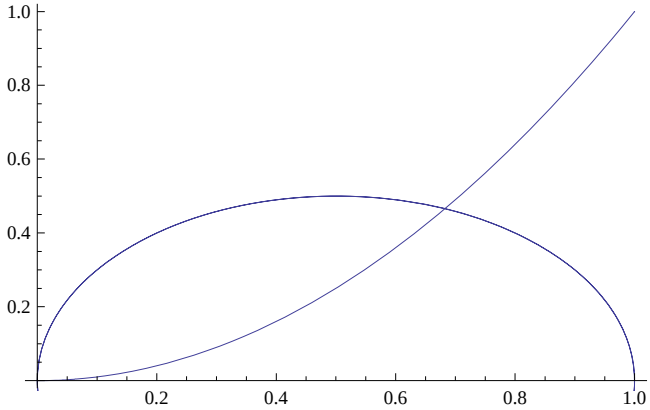
```
SphericalPlot3D[Sin[2 θ] (Cos[2 ϕ] - Sin[2 ϕ]), {θ, 0, π}, {ϕ, 0, 2 π}]
```



The Show Command

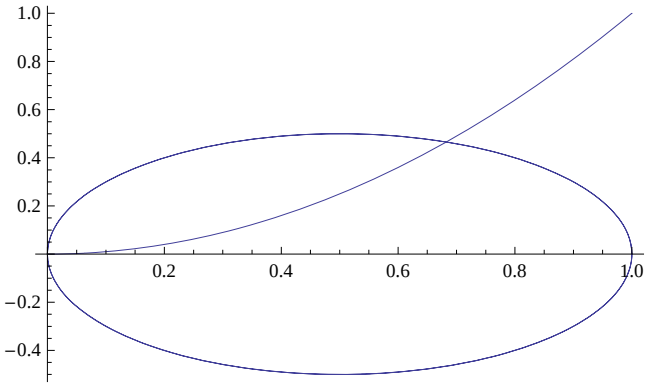
Suppose we want to plot two different types of plots on the same set of axes; for instance suppose we want to overlay the plots of y = $x^2$and r = cos θ.

```
Show[Plot[x^2, {x, 0, 1}], PolarPlot[Cos[θ], {θ, 0, 2 π}]]
```



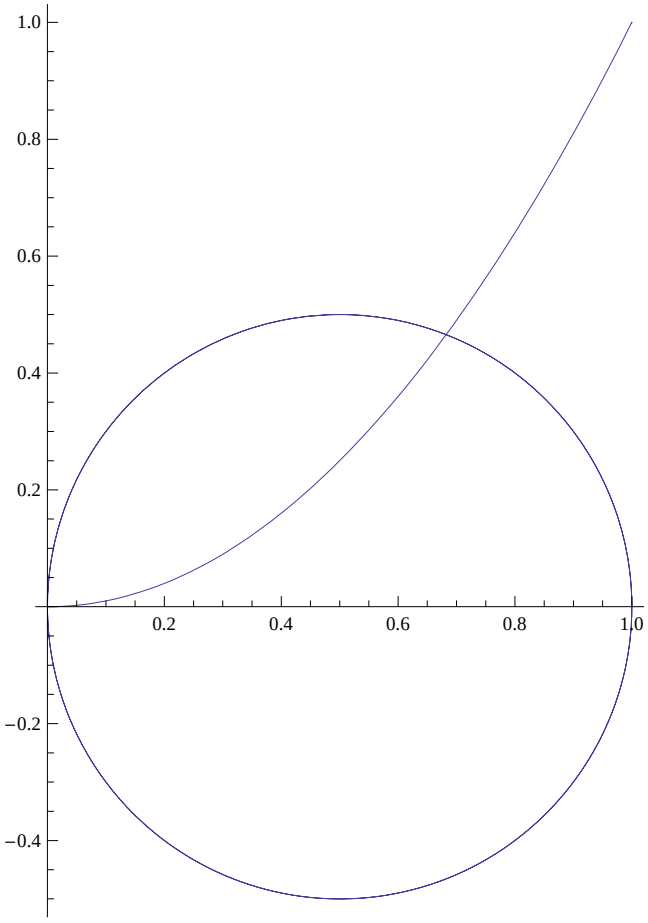
We use the Show command to combine two different sets of graphics. Look what happens when we add the PlotRange option :

```
Show[Plot[x^2, {x, 0, 1}], PolarPlot[Cos[θ], {θ, 0, 2 π}], PlotRange → All]
```



And remembering the AspectRatio issue :

```
Show[Plot[x^2, {x, 0, 1}], PolarPlot[Cos[θ], {θ, 0, 2 π}],
 PlotRange → All, AspectRatio → Automatic]
```



and our plot looks like a circle as we expect.

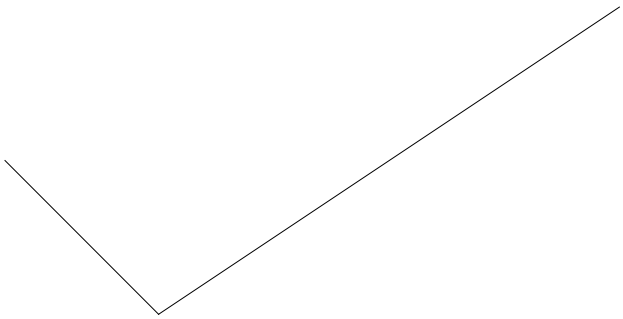
Graphics

Just as the Plot and PlotStyle commands give you incredible sets of options for processing plots, the Graphics command allows you to draw almost any figure you wish. Almost any figure, now matter how complicated, can be constructed by combining different *primitives.*

A primitive is a basic geometric form, like circle, line, point, and so on (look up Graphics on the Doc Center for much, much more).

Let's show how Graphics works with a few examples. Suppose I want to draw a line from (0,0) to (3,2); and on the same diagram a line from (0,0) to (-1,1). We input:

```
g1 = Graphics[Line[{{0, 0}, {3, 2}}]];
g2 = Graphics[Line[{{0, 0}, {-1, 1}}]];
Show[g1, g2]
```

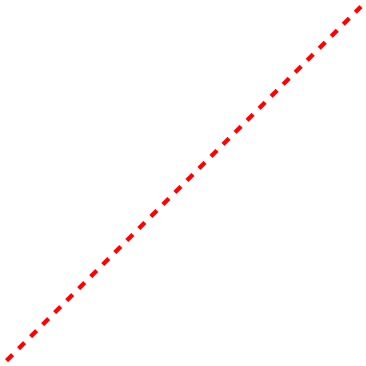


Notice that we define each Graphics statement, and use the Show command to display both statements. The semicolons at the end of each line keeps the lines from being displayed individually. Notice also the structure of the command; we first call Graphics; the we call Line. The elements of Line are the coordinate positions of the beginning and endpoints.

Be sure you recognize why there are nested braces inside the Line command. Each coordinate position represents a set of numbers, so is written in braces as {0, 0}. However, by defining two points, we have a set of sets, so we use braces to designate the set of sets, leading to the construction :
Line[{{0, 0}, {3, 2}}].

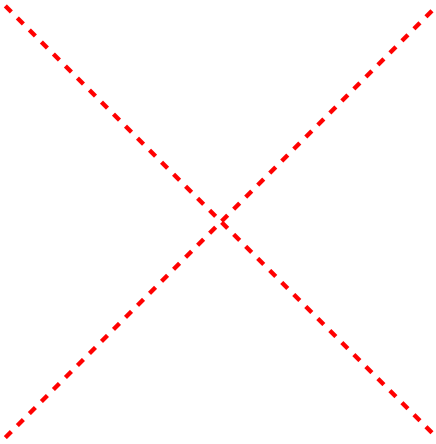
We can also add *directives* to Graphics commands; for instance:

```
Graphics[{Red, Thick, Dashed, Line[{{0, 0}, {1, 1}}]}]
```



If we want intersecting lines, we could either define two lines and us the Show command, or put two lines into one Graphics command :

```
Graphics[{Red, Thick, Dashed, {Line[{{0, 0}, {1, 1}}], Line[{{0, 1}, {1, 0}}]}}]
```



There are countless ways to construct graphs. My suggestion is that you try to learn the way I did; by starting with simple diagrams and building them up. I profited greatly by using the online documentation center. To show you just some of what is possible, here is the code (and output) I wrote for a very common physics situation, the force diagram for a block on a plane :

```
g1 = Graphics[Line[{{0, 0}, {20, 0}}]];
g2 = Graphics[Line[{{0, 0}, {15, 15}}]];
g3 = Graphics[
   {Opacity[0.2], Blue, Rotate[Rectangle[{8, 8}, {12, 12}], 45 Degree, {Left, Bottom}]}];
g4 = Graphics[{Blue, Thickness[0.01], Arrow[{{8, 10.8}, {8, 2.8}}]}];
g5 = Graphics[{Cyan, Thickness[0.01], Arrow[{{8, 10.8}, {3.8, 15}}]}];
g6 = Graphics[{Orange, Thickness[0.01], Arrow[{{2.4, 5.2}, {6.6, 9.4}}]}];
g7 = Graphics[Text[
    StyleForm["Weight", FontSize → 14, FontWeight → "Bold"], {9, 6.8}, {0, 1}, {0, -1}]];
g8 = Graphics[Text[StyleForm["Normal", FontSize → 14, FontWeight → "Bold"],
    {6, 13}, {0, -1}, {1, -1}]];
g9 = Graphics[Text[StyleForm["Friction", FontSize → 14, FontWeight → "Bold"],
    {4.5, 7.2}, {0, 1}, {1, 1}]];
Show[g1, g2, g3, g4, g5, g6, g7, g8, g9]
```