# A Mathematica Reference Guide (for calculus students)

Written by Joseph Nakao
Last updated 31 August 2021

This is an introductory reference guide on basic functions and commands in Mathematica intended to help the typical calculus student. The reader should think of this guide as more of a “handbook,” rather than notes or a lab. I have included the functions most commonly seen in introductory calculus courses. This guide by no means exhaustive. More details on Mathematica can be found easily online (see the documentation), as well as in the phenomenal book “Mathematica: A Problem-Centered Approach” by Roozbeh Hazrat.

**The outline of these notes are as follows:**

1. General formatting of .nb notebooks/files and warnings
2. Getting started
3. How to type mathematical symbols in the “text” style format
4. How to type/use mathematical symbols in a cell
5. Functions
6. Limits and solving equations
7. Plotting functions
8. Differentiation
9. Integration (Single, Double, Triple Integrals)
10. Vectors and Vector Fields
11. Parameterizations, Line Integrals, Surface Integrals
12. Sum[] with applications to Riemann sums and convergent series
13. Lists and Loops
14. Tables
15. Manipulate[]

# 1. General formatting of .nb notebooks/files and warnings

Some general introductory pieces of advice that I explain more in detail are:
- “Enter/Return” changes to a new line, just like you would in Microsoft Word.

- To change cells, simply click the down arrow on your keyboard.
- "Shift + Enter" will command Mathematica to execute a cell.
- If you ever want to delete any part of a cell, click the vertical bar (see the right of the screen) and "DELETE."

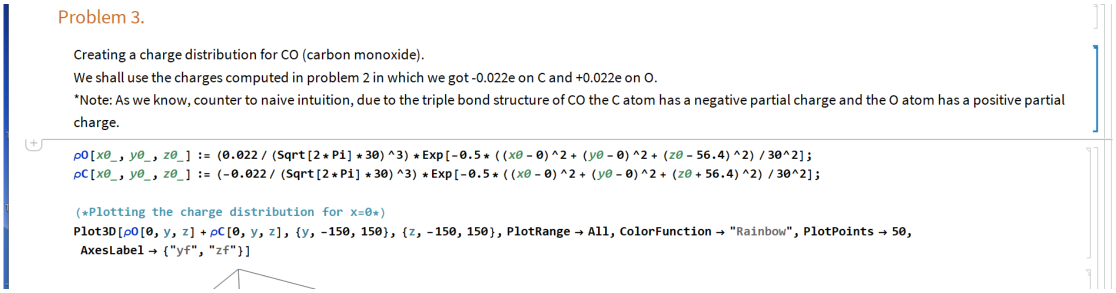
In my experience working with students, the most confusing things about (first) using Mathematica are:

**Q1. What's the difference between a cell and a text cell?**
**Q2. How do I delete unwanted cells and outputs in Mathematica?**
**Q3. HELP! Mathematica was working a second ago, but now it's broken!**

To answer these questions and concerns...

**A1. The difference between a cell and a text cell are that: a CELL is a "box" where you "do math" (i.e., whenever you're actually doing math); a TEXT CELL is a "box" where you want to just type text (e.g., how I am typing this very sentence).**

Mathematica automatically defaults you to a cell to start "doing math." If you ever want to put a CELL in between two (text) cells, simply click the space between them (see image below). Notice the line across the middle of the page where I have clicked the space between the blue highlighted (text) cell and the cell underneath.

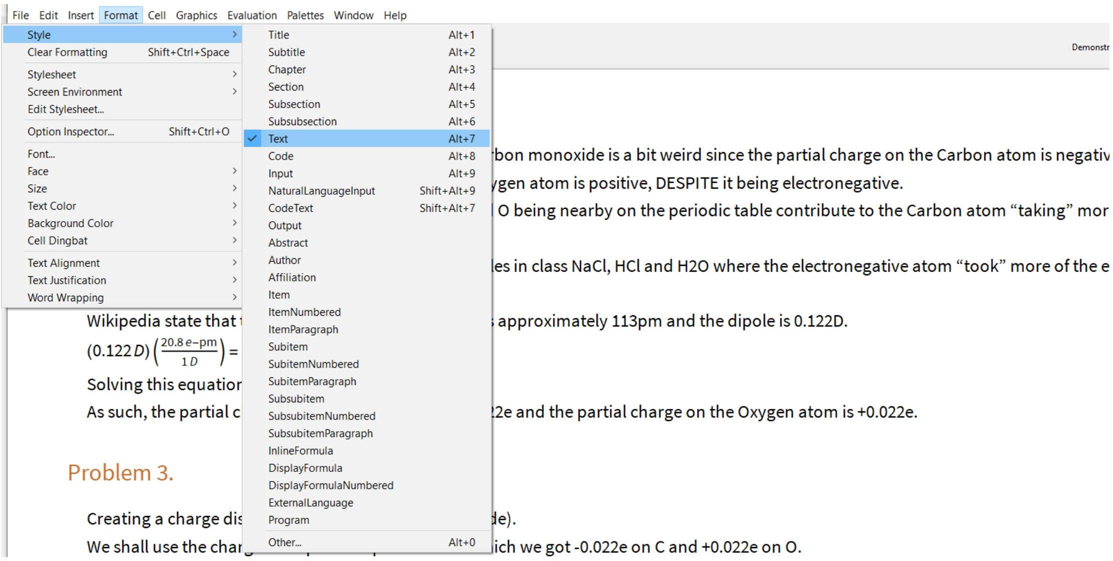


Problem 3.

Creating a charge distribution for CO (carbon monoxide).
We shall use the charges computed in problem 2 in which we got -0.022e on C and +0.022e on O.
*Note: As we know, counter to naive intuition, due to the triple bond structure of CO the C atom has a negative partial charge and the O atom has a positive partial charge.

```
ρO[x0_, y0_, z0_] := (0.022 / (Sqrt[2 * Pi] * 30) ^3) * Exp[-0.5 * ((x0 - 0) ^2 + (y0 - 0) ^2 + (z0 - 56.4) ^2) / 30^2];
ρC[x0_, y0_, z0_] := (-0.022 / (Sqrt[2 * Pi] * 30) ^3) * Exp[-0.5 * ((x0 - 0) ^2 + (y0 - 0) ^2 + (z0 + 56.4) ^2) / 30^2];

(*Plotting the charge distribution for x=0*)
Plot3D[ρO[0, y, z] + ρC[0, y, z], {y, -150, 150}, {z, -150, 150}, PlotRange → All, ColorFunction → "Rainbow", PlotPoints → 50,
 AxesLabel → {"yf", "zf"}]
```

To start a TEXT CELL, in the top bar (e.g., file, edit, insert, format, etc...), you'll want to go to "Format" -> "Style" -> "Title, text, etc..." for typing text. For a general TEXT CELL click "text." Chapter titles and such correspond to the other options. See the image below.



bon monoxide is a bit weird since the partial charge on the Carbon atom is negativ

ygen atom is positive, DESPITE it being electronegative.

O being nearby on the periodic table contribute to the Carbon atom “taking” mor

es in class NaCl, HCl and H2O where the electronegative atom “took” more of the e

Wikipedia state that t approximately 113pm and the dipole is 0.122D.

$(0.122\,D)\left(\frac{20.8\,e\text{−pm}}{1\,D}\right) =$

Solving this equatior

As such, the partial c 2e and the partial charge on the Oxygen atom is +0.022e.

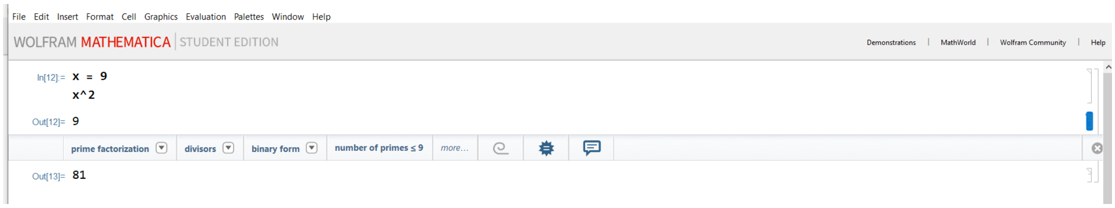
Problem 3.

Creating a charge dis de).

We shall use the char ich we got -0.022e on C and +0.022e on O.

**A2. If you ever want to delete a bunch of previous outputs, simply click the vertical bar on the right of the screen and then click the “delete” key on your keyboard (see the image below for an example). If I delete the blue bar highlighted, then Out[12]=9 will be deleted.**

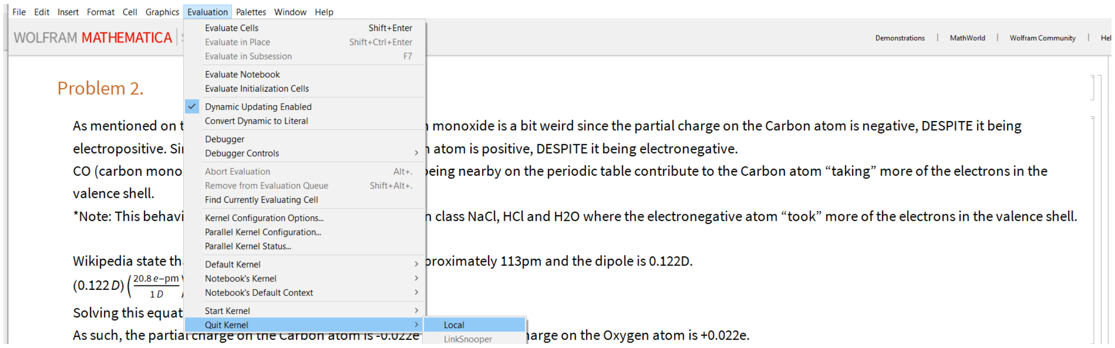
**WARNING!!!** If you delete a cell or output, then the variables will still be saved and might need to be cleared. In other words, let’s say you let $x = 9$ and computed $x^2$; this will output 81. But even if you delete this entire cell (inputs and outputs), $x$ will still be equal to 9.


File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
WOLFRAM MATHEMATICA | STUDENT EDITION
Demonstrations | MathWorld | Wolfram Community | Help
prime factorization
divisors
binary form
number of primes ≤ 9
more...

```
In[12]:= x = 9
         x^2
Out[12]= 9
Out[13]= 81
```

**A3. Sometimes Mathematica will just stop working. You might’ve plotted a function before, but suddenly now it won’t show. A common trick is to quit and restart the kernel. The kernel can be thought of as where the computer does all the work. Sometimes the kernel just gets a bit over-whelmed, so we give it a fresh start.**

As seen in the image below, go to “Evaluation” -> “Quit Kernel” -> “Local”, then click yes when the computer asks if you’re sure you want to do this. Repeat to restart but with “Start Kernel.”


File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
WOLFRAM MATHEMATICA
Evaluate Cells Shift+Enter
Evaluate in Place Shift+Ctrl+Enter
Evaluate in Subsession F7
Evaluate Notebook
Evaluate Initialization Cells
✓ Dynamic Updating Enabled
Convert Dynamic to Literal
Debugger
Debugger Controls ›
Abort Evaluation Alt+.
Remove from Evaluation Queue Shift+Alt+.
Find Currently Evaluating Cell
Kernel Configuration Options...
Parallel Kernel Configuration...
Parallel Kernel Status...
Default Kernel ›
Notebook's Kernel ›
Notebook's Default Context ›
Start Kernel ›
Quit Kernel ›
Local
LinkSnooper
Demonstrations | MathWorld | Wolfram Community | Hel

Problem 2.

As mentioned on t monoxide is a bit weird since the partial charge on the Carbon atom is negative, DESPITE it being electropositive. Si atom is positive, DESPITE it being electronegative.
CO (carbon mono being nearby on the periodic table contribute to the Carbon atom “taking” more of the electrons in the valence shell.
*Note: This behavi n class NaCl, HCl and H2O where the electronegative atom “took” more of the electrons in the valence shell.

Wikipedia state th proximately 113pm and the dipole is 0.122D.
$(0.122\,D)\left(\frac{20.8\,e\text{–pm}}{1\,D}\right.$
Solving this equat
As such, the partial charge on the Carbon atom is -0.022e arge on the Oxygen atom is +0.022e.

Some warnings/tips (WT) for coding in Mathematica:

**WT1 -- Standard mathematical objects and commands need to be capitalized and use brackets (NOT parentheses). For instance, if you ever want to use $\pi$ in a cell, use must type “Pi” and if you want the square root of 81 you type “Sqrt[81]”. I demonstrate below what happens when you do not do this.**

Notice that Mathematica doesn’t know what “pi” means; it’s literally just reading it as the two letters p and i put together. However, whenever Mathematica sees it as “Pi” it knows that that is $\pi$.

```
In[∘]:=  pi
         Pi

Out[∘]=  pi

Out[∘]=  π
```

Now notice below that just like with capitalizing “Pi” we must capitalize the function “Sqrt” and also use square brackets. Notice that only when we capitalize “Sqrt” and use square brackets does it do what we want. I talk more about this when I discuss functions.

```
In[∘]:=  sqrt (81)
         sqrt[81]
         Sqrt (81)
         Sqrt[81]

Out[∘]=  81 sqrt

Out[∘]=  sqrt[81]

Out[∘]=  81 Sqrt

Out[∘]=  9
```

**WT2 -- If you want to add stuff to an existing cell, you MUST click inside the cell. Be very careful. If you just click the “down arrow” key, you still start a new cell.**
Example -- say I already had a cell that found $x^2$ for $x = 4$. But, let’s say I realized an assignment wanted me to also compute $x^3$ for $x = 4$.
Below, I show what NOT to do.

(I already had a cell that found $x^2$ for $x = 4$)

```
In[ ]:= x  =  4
        x^2

Out[ ]= 4

Out[ ]= 16
```

(I now click inside the cell and click the “down arrow” key). Notice by looking at the bars on the right of the screen, $x^3$ is now in a different cell.

```
In[ ]:= x = 4
        x^2

In[ ]:= x^3

Out[ ]= 64
```

(I do the CORRECT thing, where I just click the “Enter/Return” key and type everything in A SINGLE CELL)

```
In[ ]:= x = 4
        x^2
        x^3

Out[ ]= 4

Out[ ]= 16

Out[ ]= 64
```

**WT3 -- Only type text in a TEXT CELL. If you type text in a CELL, then Mathematica is going to try to read it like code.**

As you might imagine, this will confuse Mathematica quite a lot. I show below what happens in this case. Notice that I ask to compute $x^2$ for $x = 4$, but nothing outputs.

```
In[ ]:= EXAMPLE - here I am typing text in a regular cell. Things are going to go awful if I try to do math,
        such as

        x = 4

        x^2
```

**WT4 -- If you want to type text in a CELL, you may do so as long as the text is in between (* and *). Shown below is an example.**

```
In[∘]:= (* I am typing text in a normal cell*)
        x = 4
        x^2
Out[∘]= 4
Out[∘]= 16
```

**WT5 -- If you ever want to suppress something in a cell (i.e., don’t want it to show up in the output), then end the line with a semicolon.**

```
        (* Without a semicolon, the line will show in the output *)
        x = 4
Out[∘]= 4

        (* With a semicolon, the line will not show in the output *)
        x = 4;
```

**WT6 -- (see the section on functions). If you use a letter to define a function but want to use it again, you MUST CLEAR that variable.**
For example, say you use “f” for f[x_]:=x^2. But later on you want to use “f” for f[x_]:=Sqrt[x] + 4. Then, you must Clear[f] before using it the second time; in fact, if you plan to use “f” more than once in a document, I recommend clearing “f” before each time.
I give an example below.

```
In[∘]:= Clear[f]
        f[x_] := x^2
        f[2]

        Clear[f]
        f[x_] := Sqrt[x] + 4
        f[2]
Out[∘]= 4
```

Out[∘]= $4 + \sqrt{2}$

**WT7 -- If you want to force an answer to be a decimal, try writing one of the numbers with a decimal. If you also want to further simplify an answer, use the command FullSimplify[].**
I give an example of these cases below.

```
In[∘]:= h[x_] := (1/(4*Pi))*((1/Sqrt[x^2-3])+(1/Sqrt[x^2+3]))+(1/Pi)
        h[5]
        h[5.0]
        FullSimplify[h[5]]
```

Out[∘]= $\frac{1}{\pi} + \frac{\frac{1}{2\sqrt{7}} + \frac{1}{\sqrt{22}}}{4\pi}$

```
Out[∘]= 0.350315
```

Out[∘]= $\frac{616 + 11\sqrt{7} + 7\sqrt{22}}{616\pi}$

# 2. Getting started

I shall demonstrate how to type and execute a cell.
***NOTE*** to execute a cell, you must click “SHIFT” + “ENTER/RETURN” (at least on a PC).

INSTRUCTIONS -- write a short code in a single cell that will compute the quadratic formula for any given $a$, $b$, $c$ to solve the equation $a x^2 + b x + c = 0$.

```
In[∘]:= (* Here, I want to declare what a, b, and c are *)
        a = 1;
        b = 0;
        c = -4;
        x1 = (-b+Sqrt[b^2-4*a*c])/(2*a)
        x2 = (-b+Sqrt[b^2-4*a*c])/(2*a)

Out[∘]= 2

Out[∘]= 2
```

# 3. How to type mathematical symbols in the “text” style format

If you want to type math in a text cell, click “CTRL+9,” and use “CTRL” with any special command (e.g., “CTRL+6” for an exponent/superscript).
Ex. The Pythagorean theorem is $a^2 + b^2 = c^2$.

Some commands that are nice to know (assuming you’ve already clicked CTRL+9 to type math) are:

“CTRL” + “-” for a subscript. Ex. $x_1$
“CTRL” + “7” to type above. Ex. $\tilde{x}$
“CTRL” + “/” for division. Ex. $\frac{2}{3}$

If you want to type mathematical symbols, we enter a word between esc and esc. For example, to type *π*, you type “esc pi esc.”

*π* -- esc pi esc
∇ -- esc del esc
× -- esc * esc (to be used for cross products)
· -- esc . esc (to be used for dot products)
*α* -- esc alpha esc
*β* -- esc beta esc
*γ* -- esc gamma esc
*δ* -- esc delta esc
Δ -- esc Delta esc
∇ -- esc del esc
∂ -- esc pd esc
≤ -- esc <= esc
≥ -- esc >= esc
≡ -- esc === esc

The list goes on...you can look online or in a book many of these commands. As for Greek letters, if you’re in a fraternity or sorority, you’ll probably be fine with figuring out the Greek alphabet commands!

# 4. How to type/use mathematical symbols in a cell

Typically, we use letters like *x*, *y* and *z* when doing math. However, sometimes we use Greek letters. You can do this by simply using the same commands as given in the previous section. I give the same quadratic formula example, but with Greek letters instead of *a*, *b*, *c*.

```
In[∘]:= α = 1;
        β = 0;
        γ = -4;
        x1 = (-β + Sqrt[β^2 - 4 * α * γ]) / (2 * α)
        x2 = (-β - Sqrt[β^2 - 4 * α * γ]) / (2 * α)

Out[∘]= 2

Out[∘]= -2
```

# 5. Functions

*****REMINDER*** most commands in Mathematica use closed brackets [] instead of open (). Be careful!**
**Q** -- what is a function in Mathematica?
**A** -- it’s sort of like instructions. You tell it what to take in (the input), and it will spit out the output.

There are two types of functions in Mathematica: (1) the built-in functions such as Plot[], Manipulate[], and Sum[]; (2) the functions that you build yourself.

As we know with basic functions from calculus (e.g., $f(x, y) = x^2 + y^2$), we have **INDEPENDENT** variables and a **DEPENDENT** variable.
**Here are some rules to follow when building functions:**

**1.** The dependent variable will be whatever you name the function.
**WARNING 1** -- Never name a function with a capital letter (e.g., F[x_]:=x^2) because Mathematica already has built-in functions that use some capital letters.
**WARNING 2** -- If you go back and change a function, then you’ll need to “SHIFT” + “ENTER” the cell, as well as all subsequent cells. This is because if you change the function, then everything that followed afterwards that used the function must be updated to this new function.
**2.** The independent variables must be followed by underscores when defining the function.
**3.** You must use brackets (not parentheses) when defining a function.
**4.** You must use a colon + equal sign to define the function. For example, f[x_]:=x^2 instead of f[x_]=x^2.

Below, I give a few examples.

## Example 1. $f(x) = x^2 + 1$

```
In[]:=  f[x_] := x^2 + 1;
        f[2]
Out[]=  5
```

```
In[]:=  x = 4
        f[x]
Out[]=  4

Out[]=  17
```

### Example 2. $g(x, y, z) = x^2(\sin(y) + e^{5z})$

```
In[∘]:= g[x_, y_, z_] := x^2 * (Sin[y] + Exp[5 * z])
        g[1, 2, 3]
Out[∘]= ⅇ^15 + Sin[2]
```

### Example 3. Solving a quadratic polynomial.

A function in Mathematica doesn’t only have to be literally a function like $f(x) = x^2$. As mentioned earlier, it is like a “set of instructions,” or rather think of it like a recipe. Say I want to make a recipe that will give me the solutions to the quadratic formula. The ingredients (i.e., inputs) need to be $a$, $b$, $c$.

```
In[∘]:= pythagorean1[a_, b_, c_] := (-b + Sqrt[b^2 - 4 * a * c]) / (2 * a)
        pythagorean2[a_, b_, c_] := (-b - Sqrt[b^2 - 4 * a * c]) / (2 * a)

        pythagorean1[1, 0, -4]
        pythagorean2[1, 0, -4]
Out[∘]= 2

Out[∘]= -2
```

# 6. Limits and solving equations

Oftentimes, we want to compute the limits of functions. The basic built-in function Limit[] in Mathematica is as follows:
(In 1D) Limit[f[x],x -> value]

When solving equations (e..g, solve $x^2 + y^2 = 1$), we use “double equal signs.” I show this in examples 4 and 5.

## Example 1. $\lim_{x\to 1} \frac{x^2-1}{x-1}$.

```
In[∘]:= (* METHOD 1 - hardcoding the function directly *)
        (* i.e., we simply just type the function into Limit[] *)

        Clear[x] (* note -- if you defined "x" earlier, you might need to Clear[x] *)
        Limit[(x^2 - 1) / (x - 1), x → 1]

        (* METHOD 2 - defining a function f[x] *)
        Clear[f] (* we used f earlier in these notes, so we Clear[f] *)
        f[x_] := (x^2 - 1) / (x - 1)
        Limit[f[x], x → 1]
        (* note -- we DO NOT use an underscore after defining the function *)

Out[∘]= 2

Out[∘]= 2
```

## Example 2. Show that $\lim_{(x,y)\to(0,0)} \frac{x^2-3\,y^2}{x^2+2\,y^2}$ DNE by showing the limit going to (0,0) along the x-axis is different that the limit going to (0,0) along the y-axis.

```
In[∘]:= Clear[f]
        f[x_, y_] := (x^2 - 3 * y^2) / (x^2 + 2 * y^2)
        Limit[f[x, y], {x, y} → {0, 0.001}] (*the limit on the y-axis close to the origin*)
        Limit[f[x, y], {x, y} → {0.001, 0}] (*the limit on the x-axis close to the origin*)
        Limit[f[x, y], {x, y} → {0, 0}] (*the actual limit at the origin*)

Out[∘]= -1.5

Out[∘]= 1.

Out[∘]= Indeterminate
```

## Example 3. Limit definition of the derivative.

Using the limit definition of the derivative, show the derivative of $f(x) = \sin(x)$ is $f'(x) = \cos(x)$.

```
In[∘]:= Clear[f]
        f[x_] := Sin[x]
        Limit[(f[x + h] - f[x]) / h, h → 0]

Out[∘]= Cos[x]
```

### Example 4. Solve $x^2 + 2x - 15 = 0$.

```
In[∘]:= Clear[f]
        f[x_] := x^2 + 2 * x - 15
        Solve[f[x] ⩵ 0, x]
        (* note -- to get rid of the outer set of parentheses,
        you can use the Flatten[] command *)

Out[∘]= {{x → -5}, {x → 3}}
```

```
In[∘]:= Flatten[Solve[f[x] ⩵ 0, x]]

Out[∘]= {x → -5, x → 3}
```

### Example 5. Solve $x^2 + y^2 - 2xy = 0$.

Note that this solution is going to be a function! As shown below, the solution to this equation is $y = x$. We know this because $x^2 - 2xy + y^2 = (x - y)^2$.

```
In[∘]:= Clear[f]
        f[x_, y_] := x^2 + y^2 - 2 * x * y
        Flatten[Solve[f[x, y] ⩵ 0, {x, y}]]

        ••• Solve: Equations may not give solutions for all "solve" variables.

Out[∘]= {y → x}
```

# 7a. Plotting functions ($y = f(x)$)

**Here, I talk about how to plot explicit functions of the form $y = f(x)$.**

The (basic) instructions for Mathematica’s built-in “Plot” function are **Plot[f[x],{x,a,b},Axes->True,AxesLabel->{“x”,”y”},PlotStyle -> Red]**

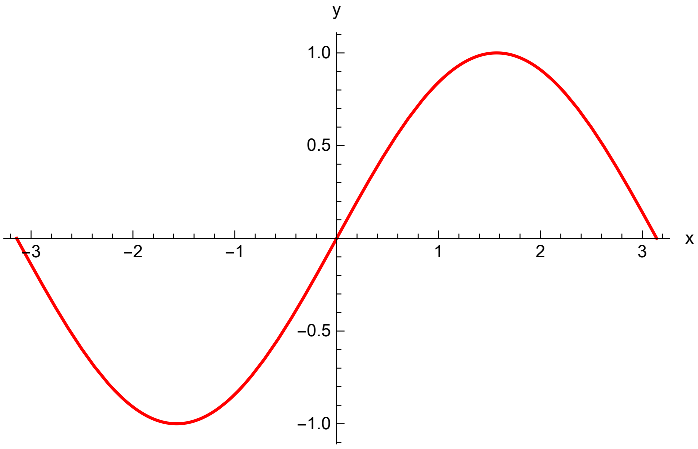
Let’s break down the Plot[] function into each component!
1. **f[x]** is the function you want to plot.
2. **{x,a,b}** is how you specify the bounds for *x*
3. **Axes->True** is how to decide whether or not you want axes to show. You can choose either True or False.
4. **PlotStyle->Red** is how to decide what color to make the plot. You could choose other colors instead of just red.

***NOTE*** There are MANY additional features you can add to a plot such as showing axes, titles, labels, shading, etc... You can look up online how to alter the Plot[] command to do such things. I given an example below where I use a few different features.

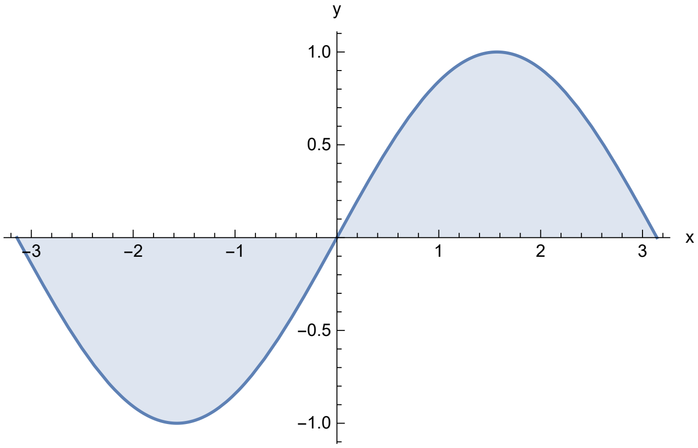
## Example 1. Plotting sin(*x*)

In[∘]:=
```
y[x_] := Sin[x]
Plot[y[x], {x, -Pi, Pi}, Axes → True, AxesLabel → {"x", "y"}, PlotStyle → Red]
(*adding labels*)

Plot[y[x], {x, -Pi, Pi}, Axes → True, AxesLabel → {"x", "y"},
 Filling → Axis, FillingStyle → Automatic] (*shading*)

Plot[y[x], {x, -Pi, Pi}, Axes → True, AxesLabel → {"x", "y"},
 Filling → Axis, ColorFunction → "Rainbow"] (*rainbow shading*)
```

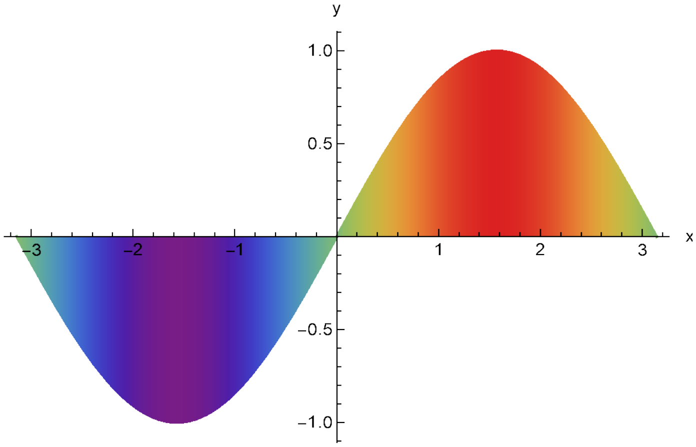
Out[∘]=



Out[∘]=



Out[∘]=



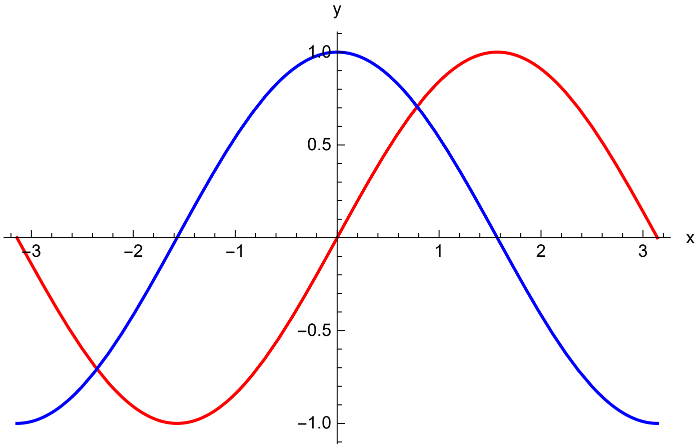
## Example 2. Plotting two (or more) functions on the same plot. Plot sin(*x*) and cos(*x*) on the same plot.

When plotting more than one function on the same plot, you use the same Plot[] function, but now you specify the two (or more) functions by putting them together in curly braces {f[x],g[x]}.
***NOTE*** when specifying the colors, be sure to put the corresponding colors in curly braces also.

In[∘]:=

```
Clear[f, g] (* we have already used f and g in these notes, so let's clear them *)
f[x_] := Sin[x]
g[x_] := Cos[x]
Plot[{f[x], g[x]}, {x, -Pi, Pi}, AxesLabel → {"x", "y"}, PlotStyle → {Red, Blue}]
```

Out[∘]=



# 7b. Plotting functions ($f(x, y) = k$)

**Here, I talk about how to plot implicit functions of the form *f*(*x*, *y*) = *k*.**

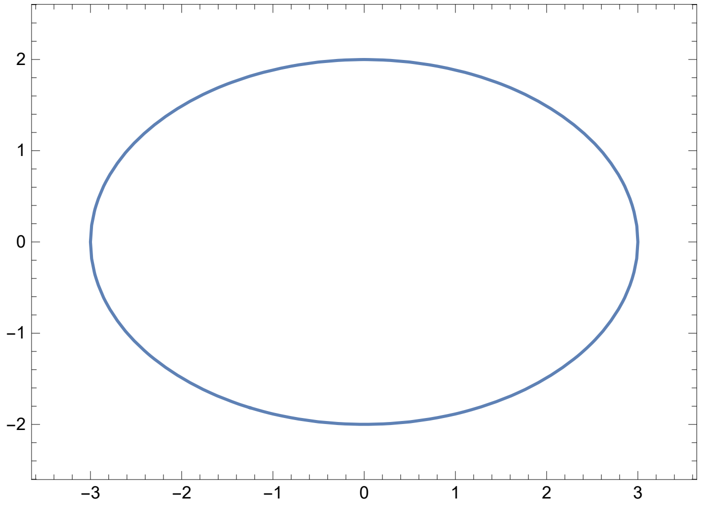
Instead of using the Plot[] function, we use ContourPlot[f[x,y]==k,{x,a,b}].
I give an example of a cardioid below.

One difference between ContourPlot[] and Plot[] is that we must specify the bounds for BOTH *x* and *y*.

### Example 3. Plot the ellipse $\frac{x^2}{9}+\frac{y^2}{4}=1$.

In[∘]:=
```
Clear[f]
f[x_, y_] := (x^2/9) + (y^2/4)
ContourPlot[f[x, y] == 1, {x, -3.5, 3.5}, {y, -2.5, 2.5}, AspectRatio → Automatic]
(* NOTE -- the AspectRatio→
 Automatic is to make the scalings of the x- and y-axes the same *)
```

Out[∘]=



# 7c. Plotting functions ($z = f(x, y)$)

**Here, I talk about how to plot explicit functions of the form $z = f(x, y)$.**

The function we use is similar to Plot[]. We use Plot3D[]. The basic instructions for Plot3D[] are Plot3D[f[x,y],{x,a,b},{y,c,d},AxesLabel->{“x”,“y”,“z”}]

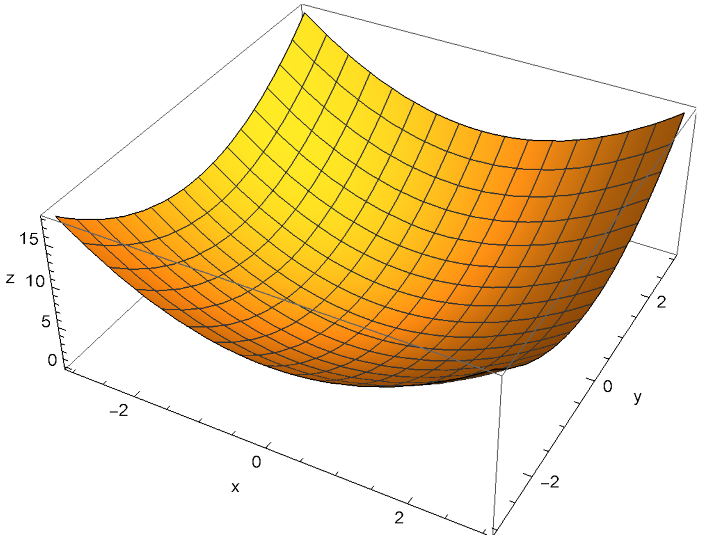
***NOTE*** As with ContourPlot[], we must specify the bounds for both *x* and *y*.

***NOTE*** We could also plot explicit functions like this using ContourPlot3D (see next section).

### Example 4. Plot the paraboloid $z = x^2 + y^2$.

```
In[]:= Clear[f]
       f[x_, y_] := x^2 + y^2
       Plot3D[f[x, y], {x, -3, 3}, {y, -3, 3}, AxesLabel → {"x", "y", "z"}]
```

Out[]=



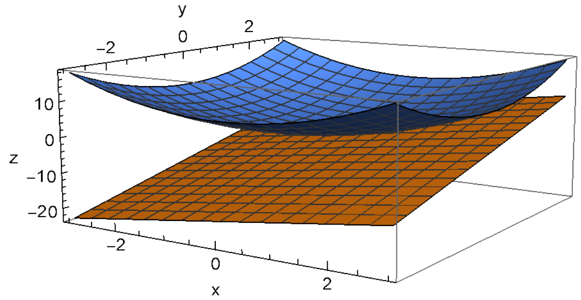
### Example 5. Plot the plane $2x + 3y - z = 8$ and the paraboloid $z = x^2 + y^2$ on the same plot.

Just like what we did with two function using Plot[], we do the same thing for Plot3D[]. We must specify both functions together in curly braces.

***NOTE*** We must first rewrite the plane as an explicit function to use Plot3D, $z = 2x + 3y - 8$.

```
In[]:= Clear[f, g]
       f[x_, y_] := 2 * x + 3 * y - 8
       g[x_, y_] := x^2 + y^2
       Plot3D[{f[x, y], g[x, y]}, {x, -3, 3}, {y, -3, 3}, AxesLabel → {"x", "y", "z"}]
```

Out[]=



# 7d. Plotting functions ($f(x, y, z) = k$)

**Here, I talk about plotting implicit functions of the form $f(x, y, z) = k$.**

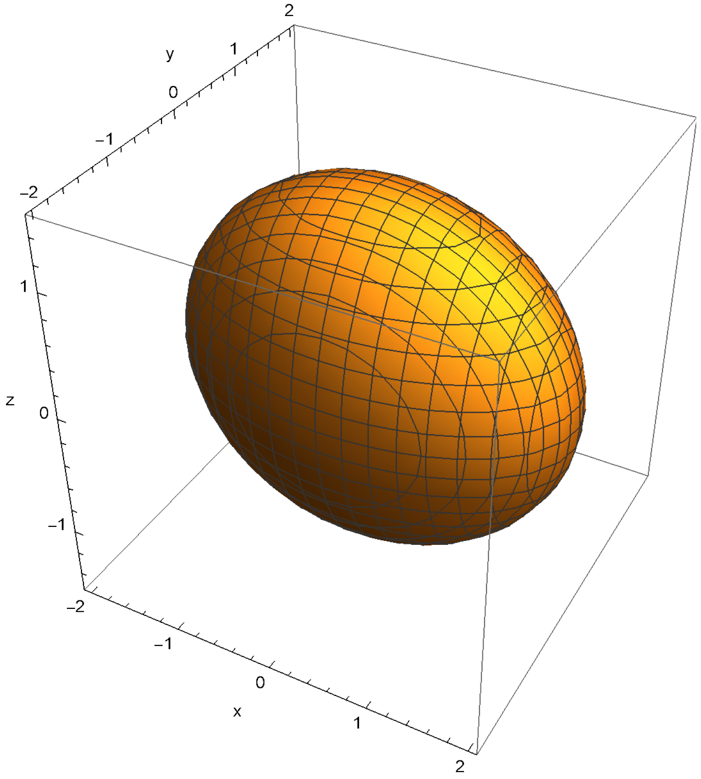
We use the function ContourPlot3D[]. The basic instructions for this function are ContourPlot3D[f[x,y,z]==k,{x,a,b},{y,c,d},{z,e,f},AxesLabel->{“x”,”y”,”z”}]

***NOTE*** We must specify the bounds for $x$, $y$, $z$.

## Example 6. Plot the ellipsoid $x^2 + 3\,y^2 + 2\,z^2 = 4$.

```
In[∘]:= Clear[f]
        f[x_, y_, z_] := x^2 + 3 * y^2 + 2 * z^2
        ContourPlot3D[f[x, y, z] == 4, {x, -2, 2},
         {y, -2, 2}, {z, -1.5, 1.5}, AxesLabel → {"x", "y", "z"}]
```

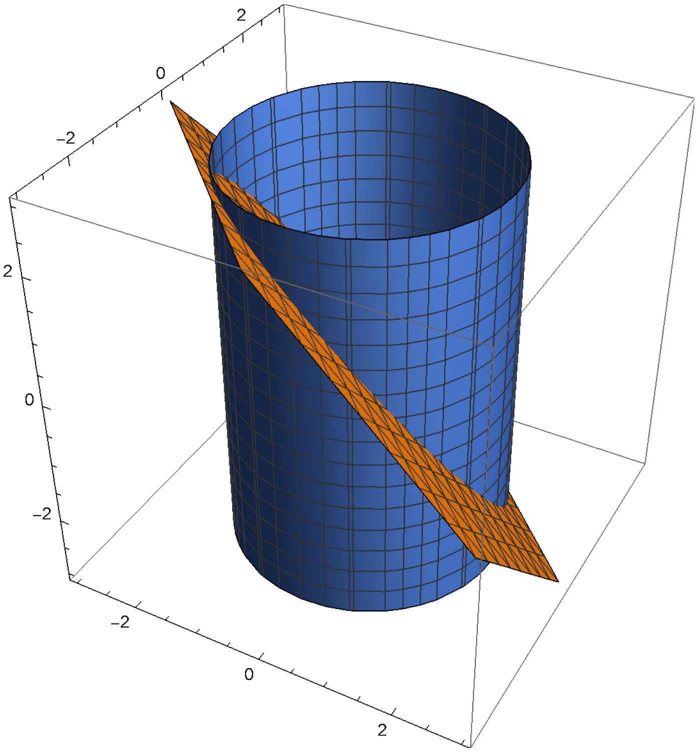
Out[∘]=



## Example 7. On the same set of axes, plot the plane $x + y + z = 4$ and the cylinder $4 = x^2 + y^2$.

```
In[•]:= Clear[f, g]
        f[x_, y_, z_] := x + y + z
        g[x_, y_, z_] := x^2 + y^2
        ContourPlot3D[{f[x, y, z] == 0, g[x, y, z] == 4}, {x, -3, 3}, {y, -3, 3}, {z, -3, 3}]
```

Out[•]=



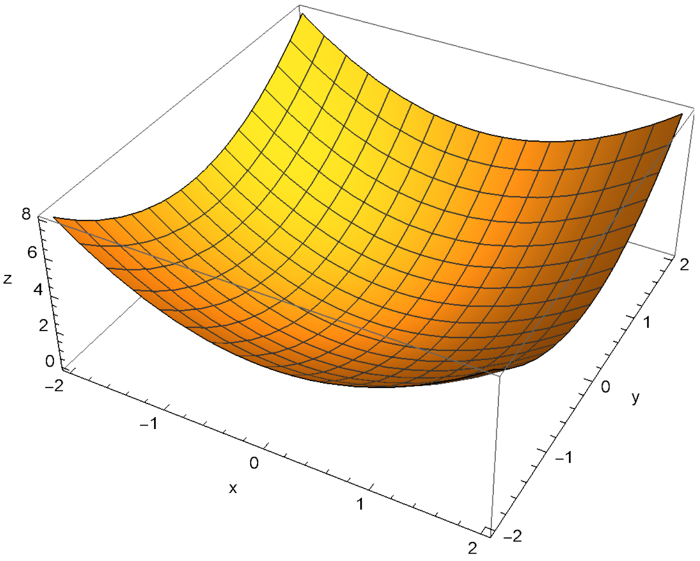
## Example 8. Plot the paraboloid $z = x^2 + y^2$ using (a) Plot3D[] and (b) ContourPlot3D[].

This is meant to demonstrate that you can plot any explicit function as an implicit function. For part (b), we must rewrite the equation as $z - x^2 - y^2 = 0$.

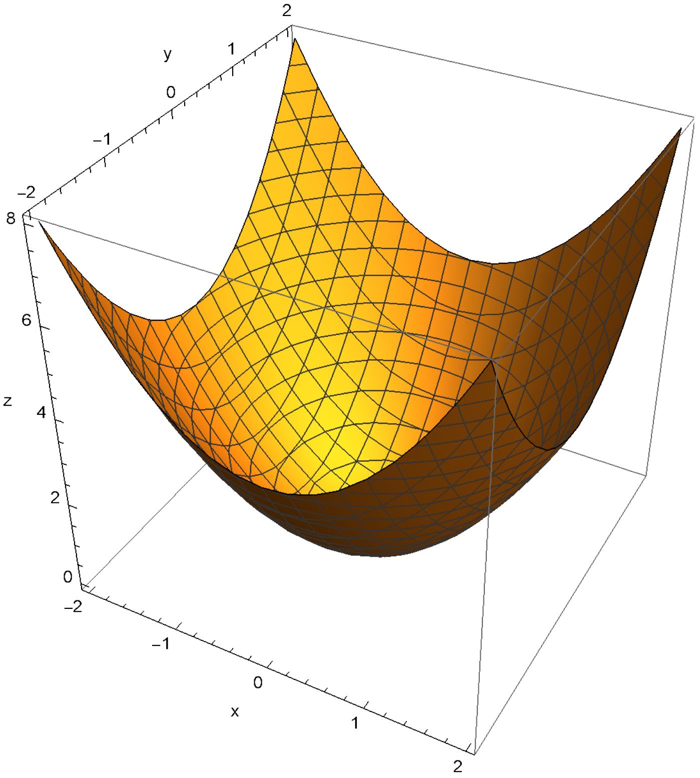
In[∘]:=

```
(* a *)
Clear[f]
f[x_, y_] := x^2 + y^2
Plot3D[f[x, y], {x, -2, 2}, {y, -2, 2}, AxesLabel → {"x", "y", "z"}]

(* b *)
Clear[f]
f[x_, y_, z_] := z - x^2 - y^2
ContourPlot3D[f[x, y, z] == 0, {x, -2, 2}, {y, -2, 2}, {z, 0, 8}, AxesLabel → {"x", "y", "z"}]
```

Out[∘]=



Out[∘]=



# 8a. Differentiation ($y = f(x)$)

There are many ways to differentiate a function of the form $y = f(x)$ in Mathematica.

**If your function is of one variable, i.e., $y = f(x)$, you have two options:**
1. Literally just write f’[x] as you normally would with pen and paper.
2. Use the built-in function D[f[x],x]
***NOTE*** If you wanted to take three derivatives, you would instead use
1. f’’’[x]
2. D[f[x],x,x,x]

### Example 1. Find the velocity and acceleration of a ball whose distance is given by $s(t) = -4.9\,t^2 + 20\,t + 5$ feet. Compute the velocity at time $t = 3$ seconds.

***NOTE*** For the sake of demonstrating how to use BOTH methods of computing the derivative in Mathematica, I shall use the first method for the velocity and the second method for the acceleration.

```
In[•]:= Clear[s, v, a] (* we are going to use s,v,a *)
        s[t_] := -4.9 * t^2 + 20 * t + 5
        v[t_] := s'[t] (*define the velocity as the derivative of distance*)
        a[t_] := D[s[t], t, t]
        (*define the acceleration as the second derivative of distance*)

        s[t] (*distance*)
        v[t] (*velocity*)
        a[t] (*acceleration*)

        v[3]

Out[•]= 5 + 20 t - 4.9 t^2

Out[•]= 20 - 9.8 t

Out[•]= -9.8

Out[•]= -9.4
```

# 8b. Differentiation ($z = f(x, y)$)

Unlike a function of a single variable, we cannot simply use f’[x,y] because we wouldn’t know whether ’ stood for a partial derivative with respect to $x$, or a partial derivative with respect to $y$.

**The two ways to compute partial derivatives are:**

**1. D[f[x,y],x,y,x,x], where you specify the order of partial derivatives.**
**2. Derivative[nx,ny,nz][f][x,y,z] will take a partial derivative with respect to x (nx-times), then with respect to y (ny-times), then with respect to z (nz-times).**

### Example 2. Compute $f_{xy}$ and $f_{yx}$ for $f(x, y) = \sin(x)\, e^y$. By Clairaut's theorem, we expect these to be identical.

```
In[◦]:= Clear[f]
        f[x_, y_] := Sin[x] * Exp[y]
        D[f[x, y], x, y] (* f_xy *)
        D[f[x, y], y, x] (* f_yx *)

Out[◦]= ⅇ^y Cos[x]

Out[◦]= ⅇ^y Cos[x]
```

### Example 3. Compute $f_{xxyyzzz}$ for $f(x, y, z) = \sin(x) \sin(y) \sin(z)$. Compute this derivative at the point (1,2,3).

```
In[◦]:= Clear[f]
        f[x_, y_, z_] := Sin[x] * Sin[y] * Sin[z]
        Derivative[2, 2, 3][f][x, y, z]
        Derivative[2, 2, 3][f][1, 2, 3]
Out[◦]= -Cos[z] Sin[x] Sin[y]

Out[◦]= -Cos[3] Sin[1] Sin[2]
```

## 9a. Integration ($y = f(x)$)

Integrating functions of a single variable $y = f(x)$ is just as straightforward as derivatives for functions of a single variable.

**The Mathematica function you will use for INDEFINITE integrals is**
**Integrate[f[x],x].**
**The Mathematica function you will use for DEFINITE integrals is**
**Integrate[f[x],{x,a,b}] OR NIntegrate[f[x],{x,a,b}]**

***NOTE*** The difference between Integrate and NIntegrate is that NIntegrate solves the integral numerically (e.g., trapezoid rule, Simpson's rule, Gaussian quadratures, etc...). **For the most part in your calculus courses, you'll just use Integrate[].** The function NIntegrate isn't really necessary until you get into upper-level mathematics courses.

### Example 1. Compute the indefinite integral $\int x^{-1/2}$ dx. Then, compute the

indefinite integral $\int_{[0,3]} x^{-1/2}$ dx.

```
In[ ]:= Clear[f]
        f[x_] := x^(-1/2)
        Integrate[f[x], x]
        Integrate[f[x], {x, 0, 3}]

Out[ ]= 2 √x

Out[ ]= 2 √3
```

# 9b. Integration ($z = f(x, y)$)

**Here, we talk about how to compute (definite) integrals for explicit functions of the form $z = f(x, y)$.**

The Mathematica function you’ll want to use is the same as before, Integrate[]. However, you will need to specify the bounds for $x$ and for $y$.
Expressing and implementing the bounds/limits of integration is complicated and is different for Cartesian coordinates vs. polar coordinates. I give examples for each coordinate system.

### Example 2. **(Cartesian coordinates on a rectangle)** Compute the volume under the surface (i.e., the integral of the function) $z = f(x, y) = x^2 + y^2 + x$ over the rectangular domain $[0, 3] \times [0, 2]$.

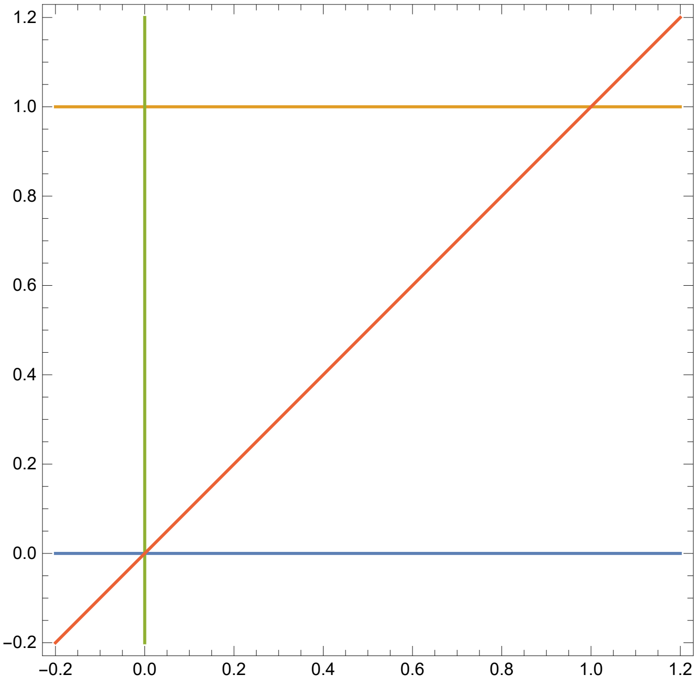
```
In[ ]:= Clear[f]
        f[x_, y_] := x^2 + y^2 + x
        Integrate[f[x, y], {x, 0, 3}, {y, 0, 2}]

Out[ ]= 35
```

### Example 3. **(Cartesian coordinates on a triangle)** Compute the volume under the surface (i.e., the integral of the function) $z = f(x, y) = x^2 + y^2 + x$ over the triangular domain bounded by $0 \le y \le 1$, $0 \le x \le y$.

For the sake of seeing the problem geometrically, I first plot the DOMAIN below. **This is not needed for computing the actual integral itself!!!** This is just to help the reader. The region of integration is the triangle bounded by the three (technically four) lines.

```
In[•]:= Clear[x, y]
        ContourPlot[{y == 0, y == 1, x == 0, x == y}, {x, -0.2, 1.2}, {y, -0.2, 1.2}]
```

Out[•]=



Now to actually compute the integral.

*****NOTE*** Although perhaps counter-intuitive, you must specify the constant bounds FIRST, followed by the function bounds SECOND!**

For this example, I mean we must specify {y,0,1} BEFORE {x,0,y}. See the calculation below.

```
In[•]:= Clear[f]
        f[x_, y_] := x^2 + y^2 + x
        Integrate[f[x, y], {y, 0, 1}, {x, 0, y}]
```

Out[•]= $\frac{1}{2}$

### Example 4. **(Polar coordinates on a half circle)** Compute the volume under the surface $z = f(x, y) = x^2\, e^{x^2+y^2}$ over the semicircle of radius 2 residing in the upper half plane.

As we know, we must convert to polar coordinates!

$f(r,\ \theta) = (r\cos(\theta))^2\, e^{r^2}$. Thus, $\int\int_D x^2\, e^{x^2+y^2}\ \mathrm{dA} = \int\int_D (r\cos(\theta))^2\, e^{r^2}\, r\, \mathrm{drd}\theta$.

In my opinion, there are two ways to code this in polar coordinates.

1. You know the bounds for $r$ and $\theta$, so you can just compute it as we did in the last two examples.
2. You can change coordinates using $x = r\cos(\theta)$ and $y = r\sin(\theta)$ in the integration. Just be sure to remember the extra $r$ from dxdy = rdrd$\theta$.

I shall demonstrate both methods.

```
In[∘]:= (* Method 1 *)
        Clear[f]
        f[r_, θ_] := (r * Cos[θ])^2 * Exp[r^2] * r
        Integrate[f[r, θ], {r, 0, 2}, {θ, 0, Pi}]
```

Out[∘]= $\frac{1}{4}\left(1+3\,e^{4}\right)\pi$

```
In[∘]:= (* Method 2 *)
        Clear[f]
        f[x_, y_] := x^2 * Exp[x^2 + y^2]
        Integrate[f[r * Cos[θ], r * Sin[θ]] * r, {r, 0, 2}, {θ, 0, Pi}]
```

Out[∘]= $\frac{1}{4}\left(1+3\,e^{4}\right)\pi$

# 9c. Integration ($w = f(x, y, z)$)

**Here, we talk about how to compute (definite) integrals for explicit functions of the form $w = f(x, y, z)$.**

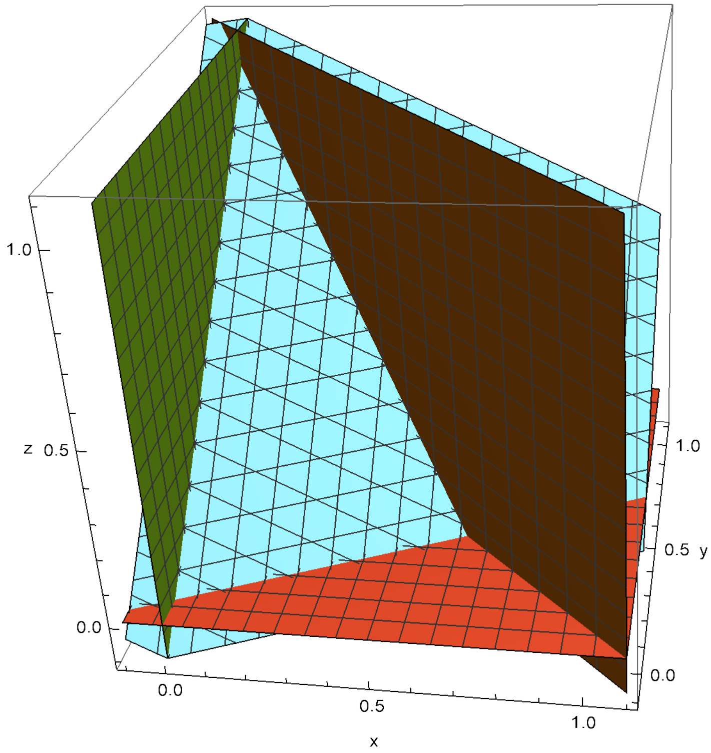
The Mathematica function you’ll want to use is the same as before, Integrate[]. However, you will need to specify the bounds for $x$, $y$, $z$.
Expressing and implementing the bounds/limits of integration is complicated and is different for Cartesian coordinates vs. cylindrical coordinates vs. spherical coordinates. I give examples for each coordinate system.

### Example 5. **(Cartesian coordinates over a triangular wedge)** Compute the volume (i.e., $w \equiv 1$ in triple integrals) of the region in the first octant bounded by the planes $x + y = 1$, $z = y - 0.5\,x$, $x = 0$ and $z = 0$.

Again, purely for the sake of geometrically visualizing the region of integration, I shall plot the planes.

```
In[•]:= ContourPlot3D[{x + y == 1, z == y - 0.5 x, x == 0, z == 0}, {x, -0.1, 1.1},
          {y, -0.1, 1.1}, {z, -0.1, 1.1}, Axes → True, AxesLabel → {"x", "y", "z"}]
```

Out[•]=



**Now, we compute the triple integral (in Cartesian coordinates). Just like with double integrals, we must FIRST specify the constant bounds of integration, SECOND specify the bounds dependent on one variables, TIHRD specify the bounds dependent on two variables.**

See the code below.

As we can see from the bounds (as well as the plot), $0 \leq x \leq 2/3$, $0 \leq y \leq 1 - x$, and $0 \leq z \leq y - 0.5\,x$.

```
In[•]:= Clear[w]
        w[x_, y_, z_] := 1 (* only when we ONLY want the volume via a triple integral *)
        Integrate[w[x, y, z], {x, 0, 2/3}, {y, 0, 1 - x}, {z, 0, y - 0.5 * x}]

Out[•]= 0.0987654
```

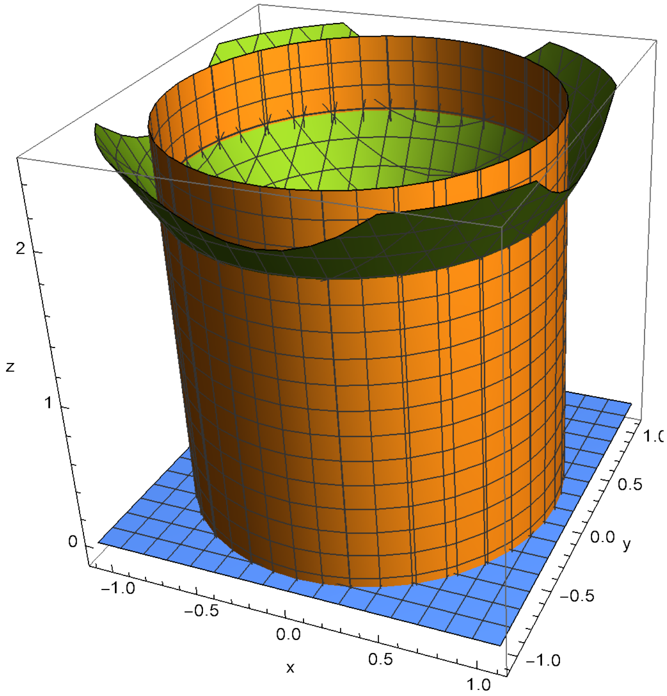
## Example 6. **(Cylindrical coordinates)** Compute the triple integral of $w = f(x, y, z) = x^2$ over the volumetric region $E$ that lies within the cylinder $x^2 + y^2 = 1$, above the plane $z = 0$, and below the cone $z^2 = 4\left(x^2 + y^2\right)$.

Again, I shall plot the domain for a geometric visualization.

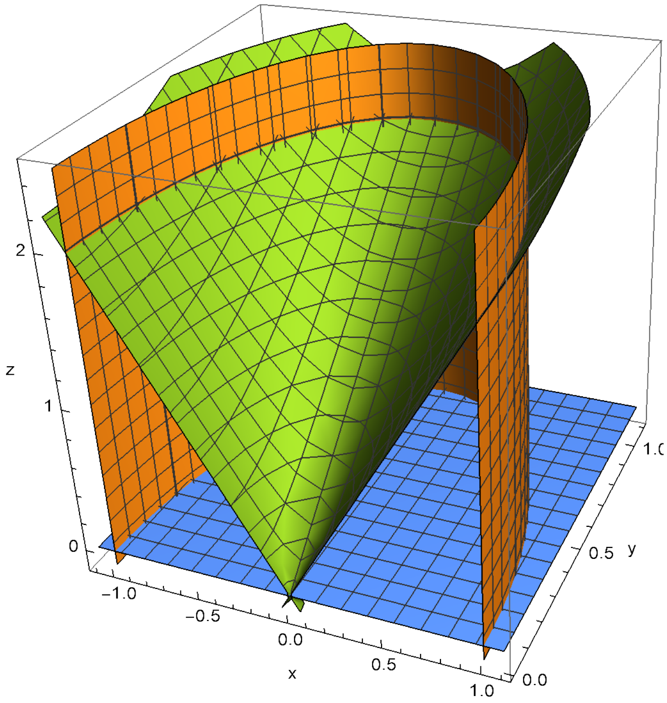
***NOTE*** I plot both the entire domain, as well as a cross-sectional plot.

```
In[]:= ContourPlot3D[{x^2 + y^2 == 1, z == 0, z^2 == 4 * (x^2 + y^2)},
         {x, -1.1, 1.1}, {y, -1.1, 1.1}, {z, -0.1, 2.5}, AxesLabel → {"x", "y", "z"}]
       ContourPlot3D[{x^2 + y^2 == 1, z == 0, z^2 == 4 * (x^2 + y^2)},
         {x, -1.1, 1.1}, {y, 0, 1.1}, {z, -0.1, 2.5}, AxesLabel → {"x", "y", "z"}]
```

Out[]=



Out[]=



Now to compute the triple integral. It can be found that $0 \le \theta \le 2\pi$, $0 \le r \le 1$ and $0 \le z \le 2r$. The polar coordinate change is $x = r\cos(\theta)$ and $y = r\sin(\theta)$, and we keep $z$ the same.

Be sure to not forget that dxdydz = rdzdrd$\theta$

```
In[∘]:= Clear[f]
        f[x_, y_, z_] := x^2
        Integrate[f[r * Cos[θ], r * Sin[θ], z] * r, {θ, 0, 2 * Pi}, {r, 0, 1}, {z, 0, 2 * r}]
```

Out[∘]= $\frac{2\pi}{5}$

### Example 7. **(Spherical coordinates)** Compute the triple integral of $w = f(x, y, z) = xe^{x^2+y^2+z^2}$ over the volumetric region $E$ given as the portion of the unit ball $x^2 + y^2 + z^2 \leq 1$ that lies in the first octant.

It can be easily verified that $0 \leq \rho \leq 1$, $0 \leq \theta \leq \pi/2$ and $0 \leq \phi \leq \pi/2$. Given that $x = \rho\cos(\theta)\sin(\phi)$, $y = \rho\sin(\theta)\sin(\phi)$, $z = \rho\cos(\phi)$ and dxdydz = $\rho^2\sin(\phi)\,\mathrm{d}\rho\mathrm{d}\theta\mathrm{d}\phi$, we can compute the integral quite easily in Mathematica.

```
In[∘]:= Clear[f]
        f[x_, y_, z_] := x * Exp[x^2 + y^2 + z^2]
        Integrate[f[ρ * Cos[θ] * Sin[ϕ], ρ * Sin[θ] * Sin[ϕ], ρ * Cos[ϕ]] * ρ^2 * Sin[ϕ],
         {ρ, 0, 1}, {θ, 0, Pi / 2}, {ϕ, 0, Pi / 2}]
```

Out[∘]= $\frac{\pi}{8}$

# 10. Vectors and Vector Fields

Vectors are made in Mathematica using curly braces {a,b,c}. Note that points and vectors are both denoted with {a,b,c}.

In this section, the functions that you will need are shown through the many examples.

### Example 1. **(Vector addition, scalar multiplication, and norms)** Given the vectors $\vec{a} = \langle 1, 3, -2\rangle$ and $\vec{b} = \langle 0, 2, 4\rangle$, compute the vector $4\vec{a} - \vec{b}$. Compute the norm of $4\vec{a} - \vec{b}$.

Vector addition and scalar multiplication are pretty intuitive in Mathematica.

***NOTE*** Notice that below, we don’t need to use := when defining the vectors because they are NOT functions of some other variables. Only when the vectors are dependent on some variable (such as time *t*) do we need to define them as functions.

**As for the norm, we can use Mathematica’s built-in function Norm[].**

```
In[∘]:= a = {1, 3, -2}; (* vector a *)
        b = {0, 2, 4}; (* vector b *)
        4 * a - b
        Norm[4 * a - b] (* norm *)

Out[∘]= {4, 10, -12}

Out[∘]= 2 √65
```

## Example 2. **(Dot and cross products)** Given the vectors $\vec{a} = \langle 1, 3, -2 \rangle$ and $\vec{b} = \langle 0, 2, 4 \rangle$, compute the dot product $\vec{a} \cdot \vec{b}$ and the cross product $\vec{a} \times \vec{b}$.

**The dot product between vectors a and b can be done with the built-in function Dot[a,b].**

**The cross product between vectors a and b can be done with the built-in function Cross[a,b].**

```
In[∘]:= Clear[a, b]
        a = {1, 3, -2};
        b = {0, 2, 4};
        Dot[a, b]
        Cross[a, b]

Out[∘]= -2

Out[∘]= {16, -4, 2}
```

## Example 3. **(Gradients, Directional Derivatives, Divergence, Curl)** Given the function $f(x, y, z) = x^3 \cos(y) \sin(z)$ and the vector field $\vec{F}(x, y, z) = \langle x^2, y\cos(z), y + z \rangle$, do the following:

(a) Compute the gradient of $f$, $\nabla f$.

(b) Compute the directional derivative $D_{\vec{u}}\, f = \nabla f \cdot \vec{u}$ in the direction $\vec{v} = \langle 1, 1, 1 \rangle$.

(c) Compute the divergence of $\vec{F}$, $\nabla \cdot \vec{F}$.

(d) Compute the curl of $\vec{F}$, $\nabla \times \vec{F}$.

**(a) As we know, the gradient is defined by $\nabla f = \langle f_x, f_y, f_z \rangle$. The built-in Mathematica function for the gradient is Grad[f[x,y,z],{x,y,z}].**

**It is important that you have the {x,y,z} in Grad[] because this tells you to compute the gradient in Cartesian coordinates!!! In other words, you want to take partial derivatives with respect to *x*, *y*, *z*.**

**(b) Now, simply use what we’ve learned so far about dot products and gradients in Mathematica.**

**(c) Similar to Grad[], the divergence of a vector field $\vec{F}$ can be computed using the built-in function Div[F[x,y,z],{x,y,z}], where again {x,y,z} tells Mathematica that we want to compute the divergence in Cartesian coordinates!!!**

**(d) Similar to Grad[] and Div[], the curl of a vector field $\vec{F}$ can be computed using the built-in function Curl[F[x,y,z],{x,y,z}], where again {x,y,z} tells Mathematica that we want to compute the divergence in Cartesian coordinates!!!**

In[∘]:=
```
Clear[f, F, u, v]
(* a *)
f[x_, y_, z_] := x^3 * Cos[y] * Sin[z]
Grad[f[x, y, z], {x, y, z}]

(* b *)
v = {1, 1, 1};
u = v / Norm[v]; (* need to make unit vector *)
Duf = Dot[Grad[f[x, y, z], {x, y, z}], u]

(* b *)
F[x_, y_, z_] := {x^2, y * Cos[z], y + z}
Div[F[x, y, z], {x, y, z}]

(* d *)
Curl[F[x, y, z], {x, y, z}]
```

Out[∘]= $\left\{3\,x^2\,\mathrm{Cos}[y]\,\mathrm{Sin}[z],\ -x^3\,\mathrm{Sin}[y]\,\mathrm{Sin}[z],\ x^3\,\mathrm{Cos}[y]\,\mathrm{Cos}[z]\right\}$

Out[∘]= $\frac{x^3\,\mathrm{Cos}[y]\,\mathrm{Cos}[z]}{\sqrt{3}}+\sqrt{3}\,x^2\,\mathrm{Cos}[y]\,\mathrm{Sin}[z]-\frac{x^3\,\mathrm{Sin}[y]\,\mathrm{Sin}[z]}{\sqrt{3}}$

Out[∘]= $1+2\,x+\mathrm{Cos}[z]$

Out[∘]= $\{1+y\,\mathrm{Sin}[z],\ 0,\ 0\}$

## Example 4. **(Plotting 2D Vectors)**

**Q** -- What information do we need to plot a vector (in 2D)?
**A** -- We need a **starting point** and an **ending point**, or equivalently we need a point and a vector.

The command in Mathematica to plot a vector is
**Graphics[ {Color,Thickness[Large],Arrow[{startingpoint,endingpoint}]} , Axes->True , AxesLabel->{“x”,“y”} , AxesOrigin->{0,0} , PlotRange→{{a,b},{c,d}} ]**

There are four main components to this Mathematica function that can be broken down as such:
1. **{Color,Thickness[Large],Arrow[{startingpoint,endingpoint}]}**
You need to choose a color, the thickness of the arrows (choose from Small, Medium, Large), and the starting and ending points of the vector (e.g., {0,0} and {1,2}).
2. **Axes->True**
Do you want axes to show on your plot? Choose True or False.
3. **AxesLabel->{“x”,”y”}**

This shouldn’t be new to you.

4. **AxesOrigin->{0,0}**

This just defines where you want to “origin” to be on the axes.

5. **PlotRange→{{a,b},{c,d}}**

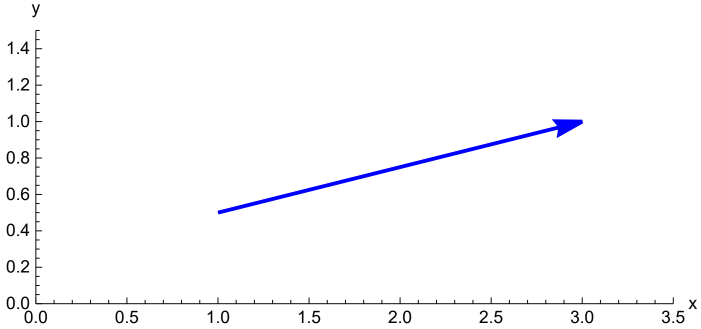
Specify the bounds for *x* and *y*.

***NOTE*** We can also plot more than one vector on the same plot using the exact same methodology as we do when plotting.

***NOTE*** We can do the same thing for vectors in 3D. Simply define starting and ending points in three coordinates (e.g., {0,0,0}) and define the PlotRange for the ranges of *x*, *y*, *z*.

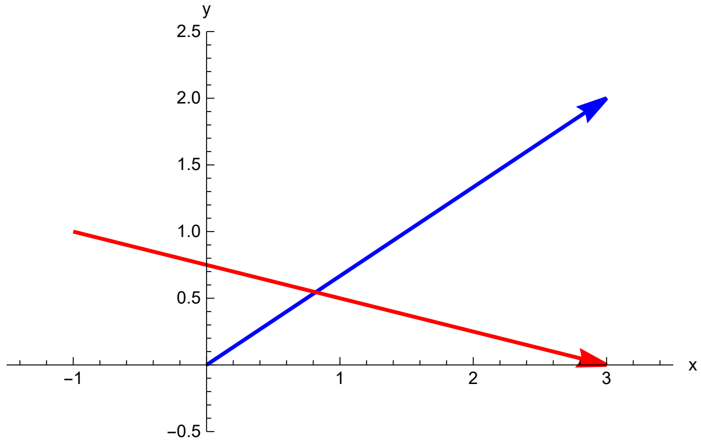
```
start = {1, 0.5};
end = {3, 1};
Graphics[{Blue, Thickness[Large], Arrow[{start, end}]}, Axes → True,
 AxesLabel → {"x", "y"}, AxesOrigin → {0, 0}, PlotRange → {{0, 3.5}, {0, 1.5}}]
```

Out[∘]=



```
In[∘]:= start1 = {0, 0};
        end1 = {3, 2};
        start2 = {-1, 1};
        end2 = {3, 0};
        Graphics[{{Blue, Thickness[Large], Arrow[{start1, end1}]},
           {Red, Thickness[Large], Arrow[{start2, end2}]}}, Axes → True,
         AxesLabel → {"x", "y"}, AxesOrigin → {0, 0}, PlotRange → {{-1.5, 3.5}, {-0.5, 2.5}}]
```

Out[∘]=



***NOTE*** Although completely optional, you could also define/create a function that does all of this in a much cleaner fashion. Two functions that we could make are

**Vector[ {start:{_,_},end:{_,_}} ,color_ _] :=Graphics[{color,Thickness[Large],Arrow[{start,end}]}]**

**Vector[ {ip:{_,_},vec:{_,_}} ,color_ _] :=Graphics[{color,Thickness[Large],Arrow[{ip,ip+vec}]}]**

The way we first defined Vector[], you give it a **starting point** and an **ending point**.
The way we second defined Vector[], you give it a **starting point** and a **vector** for the direction.

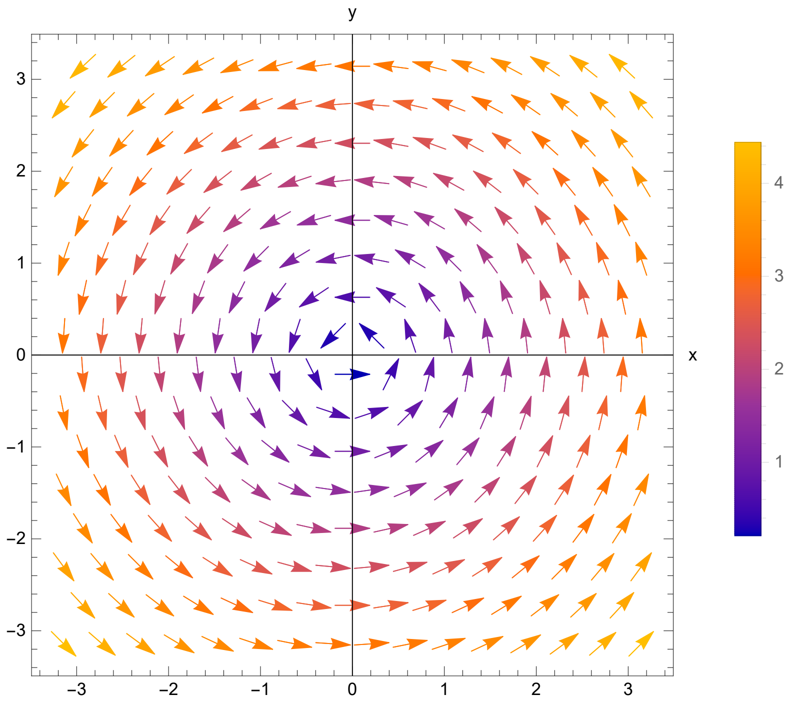
## Example 5. **(Plotting Vector Fields in 2D)**

Given a vector field $\vec{F}(x, y)$, you plot it using the built-in function **VectorPlot[F[x,y],{x,a,b},{y,c,d},Axes->True,AxesLabel->{“x”,“y”},VectorMarkers->“Arrow”,PlotLegends->Automatic]**

In the example below, I plot the vector field $\vec{F}(x, y) = \langle -y, x\rangle$.
***NOTE*** I named my vector field fvec[x_,y_] because we shouldn’t use capital letters when defining variables and functions.

```
In[ ]:= Clear[fvec]
        fvec[x_, y_] := {-y, x}
        VectorPlot[fvec[x, y], {x, -Pi, Pi}, {y, -Pi, Pi}, Axes → True,
         AxesLabel → {"x", "y"}, VectorMarkers → "Arrow", PlotLegends → Automatic]
```

Out[ ]=



### Example 6. **(Plotting Vector Fields in 3D)**

This is near identical to plotting vector fields in 2D.
Given a vector field $\vec{F}(x, y, z)$, you plot it using the built-in function
**VectorPlot3D[F[x,y,z],{x,a,b},{y,c,d},{z,e,f},Axes->True,AxesLabel->{“x”,“y”,“z”},VectorMarkers->“Arrow”,PlotLegends->Automatic]**

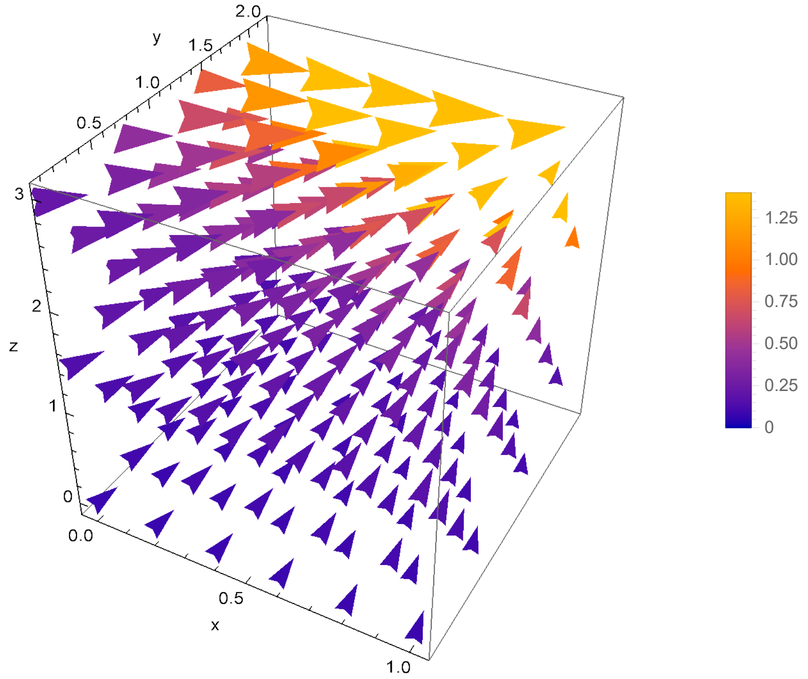
In the example below, I plot the vector field

$$\vec{F}(x, y, z) = \left\langle (1-x)\Big/\left((x-1)^2+(y-2)^2+(z-3)^2\right)^{1.5},\right.$$
$$\left. (2-y)\Big/\left((x-1)^2+(y-2)^2+(z-3)^2\right)^{1.5}, (3-z)\Big/\left((x-1)^2+(y-2)^2+(z-3)^2\right)^{1.5}\right\rangle$$

***NOTE*** I named my vector field fvec[x_,y_,z_] because we shouldn’t use capital letters when defining variables and functions.

```
In[∘]:= Clear[fvec]
fvec[x_, y_, z_] := {(1 - x) / ((x - 1) ^2 + (y - 2) ^2 + (z - 3) ^2) ^1.5,
   (2 - y) / ((x - 1) ^2 + (y - 2) ^2 + (z - 3) ^2) ^1.5,
   (3 - z) / ((x - 1) ^2 + (y - 2) ^2 + (z - 3) ^2) ^1.5}
VectorPlot3D[fvec[x, y, z], {x, 0, 1}, {y, 0, 2}, {z, 0, 3}, Axes → True,
 AxesLabel → {"x", "y", "z"}, VectorMarkers → "Arrow", PlotLegends → Automatic]
```

Out[∘]=



# 11. Parameterizations, Line Integrals, Surface Integrals

If you refer to the sections in these notes on functions, integration, and vectors, you should be able to naturally extend those ideas to parameterizations, line integrals, and surface integrals since they utilize all the previous topics.

Just like the section on vectors, I shall demonstrate these topics through many examples.

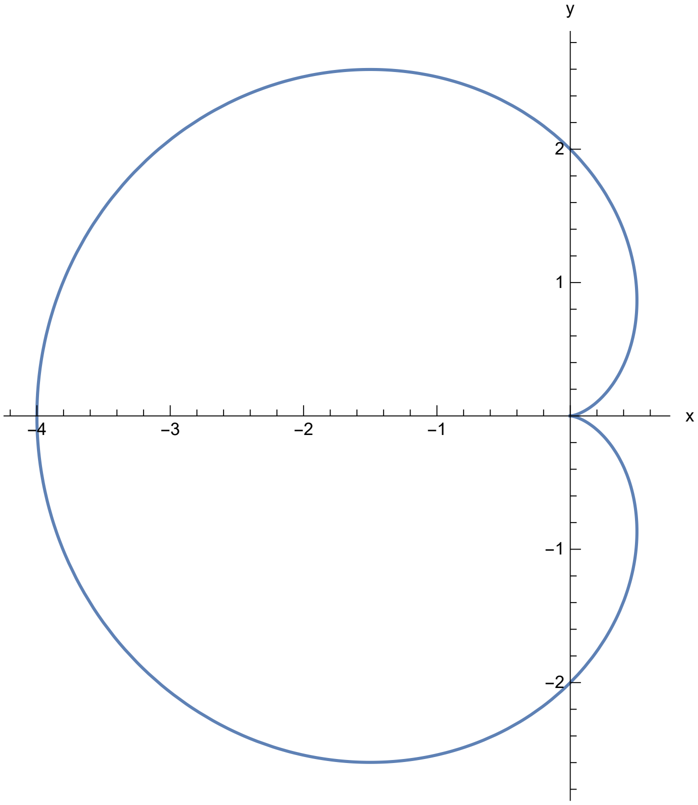
### Example 1. **(Parameterizing *x*, *y*)** Given the parameterization $x(t) = 2\cos(t)\,(1 - \cos(t))$ and $y(t) = 2\sin(t)\,(1 - \cos(t))$, plot the cardioid for $0 \leq t \leq 2\pi$.

For the parameterization, *x* and *y* are functions of *t*.
First, we need to define the parameterization x[t_]:=2cos(t)(1-cos(t)) and y[t_]:=2sin(t)(1-cos(t)).
Second, we plot using
**ParametricPlot[{x[t],y[t]},{t,a,b},Axes->True,AxesLabel->{“x”,“y”}]**
Notice that this is very similar to the Plot[] function. The main difference is that we need to input the position vector function $\langle x(t), y(t)\rangle$.

***NOTE*** Another way to do this is to define the **position vector $\vec{r}(t) = \langle x(t), y(t)\rangle$** and use
**ParametricPlot[r[t],{t,a,b},Axes->True,AxesLabel->{“x”,“y”}]**

```
In[∘]:= Clear[x, y]
        x[t_] := 2 * Cos[t] * (1 - Cos[t])
        y[t_] := 2 * Sin[t] * (1 - Cos[t])
        ParametricPlot[{x[t], y[t]}, {t, 0, 2 * Pi}, Axes → True, AxesLabel → {"x", "y"}]
```

Out[∘]=



## Example 2. **(Parameterizing *x*, *y*, *z*)** Consider the parametric curve $\vec{r}(t) = \langle \sin(t), 1 - \sin(16\,t), \cos(t)\rangle$, $0 \le t \le 2\,\pi$. Plot this curve.

Similar to using ParametricPlot[], we now use the function

**ParametricPlot3D[{x[t],y[t],z[t]},{t,a,b},Axes->True,AxesLabel->{“x”,“y”,“z”}]**

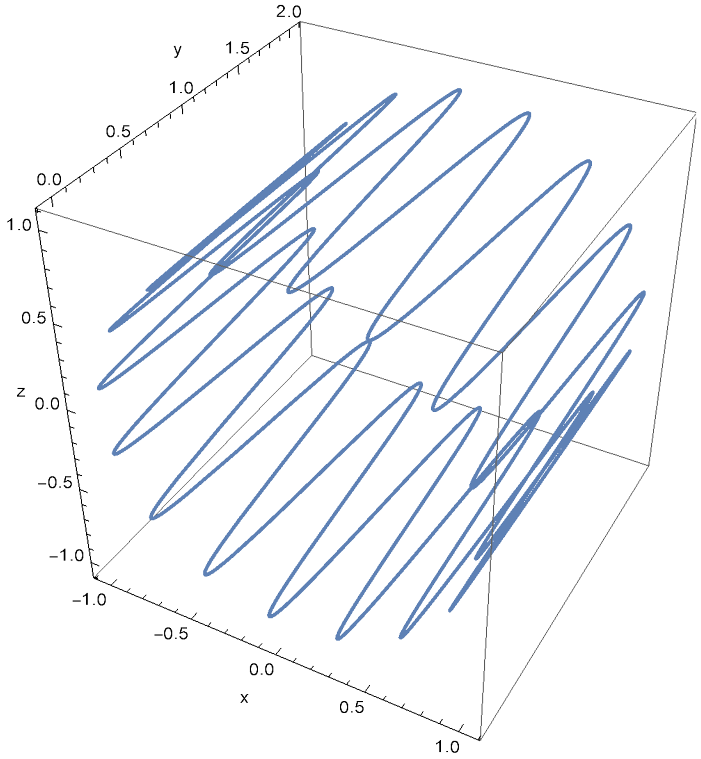
or

**ParametricPlot3D[r[t],{t,a,b},Axes->True,AxesLabel->{“x”,“y”,“z”}]**

I shall use the latter in this example.

```
In[•]:= Clear[r]
        r[t_] := {Sin[t], 1 - Sin[16 * t], Cos[t]}
        ParametricPlot3D[r[t], {t, 0, 2 * Pi}, Axes → True, AxesLabel → {"x", "y", "z"}]
```

Out[•]=

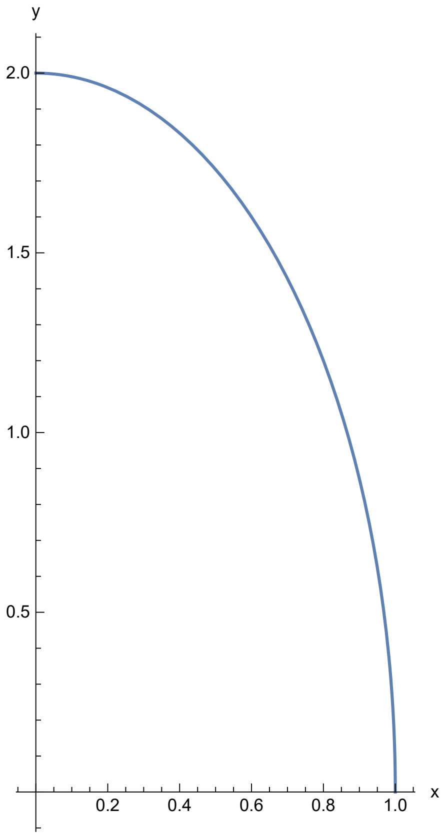


Example 3. **(Arc length in *x*, *y*)** The curve *C* is given by the parameterization $x(t) = \cos(t)$, $y(t) = 2\sin(t)$, $0 \le t \le \pi/2$. Plot the function and then compute its arc length, given that the arc length formula for a parameterization is $\int_{t=a}^{t=b} ||\vec{r}\,'(t)||\,dt \;=\; \int_{t=a}^{t=b} \sqrt{\left(\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right)}\,dt.$

***NOTE*** In your calculus course you might've seen the arc length formula (WITHOUT parameterization) given by $\int_{x=a}^{x=b} \sqrt{\left(1 + \left(\frac{dy}{dx}\right)^2\right)}\,dx$. This is the exact same thing. In our case, if you're looking at a function $y = f(x)$, we simply define the parameterization as $\langle x(t), y(t)\rangle = \langle t, f(t)\rangle$.

```
In[]:= Clear[r]
       r[t_] := {Cos[t], 2 * Sin[t]}
       ParametricPlot[r[t], {t, 0, Pi / 2}, Axes → True, AxesLabel → {"x", "y"}]
       Integrate[Norm[r'[t]], {t, 0, Pi / 2}]
       (* Since the answer to the integration is kind of gross,
       we replace 0 with a decimal 0.0. If you give Mathematica a decimal,
       it will usually give you a decimal in return *)
       Integrate[Norm[r'[t]], {t, 0.0, Pi / 2}]
```

Out[]=



```
Out[]= 2 EllipticE[3/4]
```

```
Out[•]= 2.42211
```

Example 4. **(Arc length in *x*, *y*, *z*)** Using the arc length formula given in the last example, compute the arc length for the curve *C* given by the parameterization $\vec{r}(t) = \langle 2\cos(t), 0.2t, 2\sin(t)\rangle$, $0 \le t \le 2\pi$.

```
In[•]:= Clear[r]
        r[t_] := {2 * Cos[t], 0.2 * t, 2 * Sin[t]}
        Integrate[Norm[r'[t]], {t, 0, 2 * Pi}]
Out[•]= 12.629
```

Example 5. **(Line integrals for scalar fields and vector fields)**

(a) Compute the line integral $\int_C x e^{yz}\, ds$, where *C* is the line segment from (0,0,0) to (1,2,3).

***NOTE*** the parameterization is $\vec{r}(t) = \langle t, 2t, 3t\rangle$ with $0 \le t \le 1$.

(b) Compute the line integral $\int_C \vec{F} \cdot d\vec{r}$, where $\vec{F}(x, y) = e^{x-1}\,\hat{i} + x\,y\,\hat{j}$ and *C* is given by $\vec{r}(t) = t^2\,\hat{i} + t^3\,\hat{j}$ for $0 \le t \le 1$.

Recall that the equations for the line integrals of scalar and vector fields are respectively given by

$\int_C f(x, y, z)\, ds = \int_{[a,b]} f(\vec{r}(t))\, \|\vec{r}\,'(t)\|\, dt$

and $\int_C \vec{F} \cdot d\vec{r} = \int_{[a,b]} \vec{F}(\vec{r}) \cdot \vec{r}\,'(t)\, dt$.

```
In[•]:= (* a *)
        Clear[f]
        f[x_, y_, z_] := x * Exp[y * z]
        x[t_] := t
        y[t_] := 2 * t
        z[t_] := 3 * t
        r[t_] := {x[t], y[t], z[t]}
        Integrate[f[x[t], y[t], z[t]] * Norm[r'[t]], {t, 0, 1}]
```

Out[•]= $\frac{1}{6}\sqrt{\frac{7}{2}}\left(-1 + e^{6}\right)$

```
In[•]:= (* b *)
        Clear[fvec]
        fvec[x_, y_] := {Exp[x - 1], x * y}
        x[t_] := t^2
        y[t_] := t^3
        r[t_] := {x[t], y[t]}
        Integrate[Dot[fvec[x[t], y[t]], r'[t]], {t, 0, 1}]
```

Out[•]= $\frac{11}{8} - \frac{1}{e}$

Example 6. **(Surface Integrals for scalar fields and vector fields)**

(a) Compute the surface integral $\iint_S (x+y+z)\, dS$, where $S$ is the parallelogram with parametric equations $x = u+v$, $y = u-v$, and $z = 1+2u+v$, where $0 \le u \le 2$ and $0 \le v \le 1$.
(b) Compute the surface integral $\iint_S \vec{F} \cdot d\vec{S}$, where $S$ is the cylinder $x^2 + z^2 = 4$ with $0 \le y \le 1$. Let $\vec{F}(x, y, z) = \langle -x/2, 0, -z/2 \rangle$.
***NOTE*** A natural parameterization is $x = 2\cos(u)$, $z = 2\sin(u)$, $y = v$, with $0 \le u \le 2\pi$ and $0 \le v \le 1$.

Recall that the equations for the surface integrals of scalar and vector fields are respectively given by
$\iint_S f(x, y, z)\, dS = \iint_D f(\vec{r}(u, v)) \, \| \vec{r}_u \times \vec{r}_v \| \, dA$
and $\iint_S \vec{F} \cdot d\vec{S} = \iint_D \vec{F}(\vec{r}(u, v)) \cdot (\vec{r}_u \times \vec{r}_v)\, dA$.

```
In[∘]:= (* a *)
        Clear[f, x, y, z, r]
        f[x_, y_, z_] := x + y + z
        x[u_, v_] := u + v
        y[u_, v_] := u - v
        z[u_, v_] := 1 + 2 * u + v
        r[u_, v_] := {x[u, v], y[u, v], z[u, v]}
        Integrate[f[x[u, v], y[u, v], z[u, v]] * Norm[Cross[D[r[u, v], u], D[r[u, v], v]]],
         {u, 0, 2}, {v, 0, 1}]

Out[∘]= 11 √14
```

```
In[∘]:= (* b *)
        Clear[fvec, x, y, z, r]
        fvec[x_, y_, z_] := {-x / 2, 0, -z / 2}
        x[u_, v_] := 2 * Cos[u]
        y[u_, v_] := v
        z[u_, v_] := 2 * Sin[u]
        r[u_, v_] := {x[u, v], y[u, v], z[u, v]}
        Integrate[Dot[fvec[x[u, v], y[u, v], z[u, v]], Cross[D[r[u, v], u], D[r[u, v], v]]],
         {u, 0, 2 * Pi}, {v, 0, 1}]

Out[∘]= 4 π
```

# 12. Sum[] with applications to Riemann sums and convergent series

Recall the use of sigma (Σ) notation. For instance, $\sum_{i=1}^{n} f_i = f_1 + f_2 + \ldots + f_n$. In Mathematica, we can use the function **Sum[f[i],{i,a,b}].**
In the function Sum[], we must specify the (indexed) values we are summing up, as well as the range for the indices.

## Example 1. Sum up the values $i^2$ for $i = 1, 2, \ldots, 20$.

```
In[•]:= Sum[i^2, {i, 1, 20}]

Out[•]= 2870
```

## Example 2. **(Riemann sums)** Compute the Riemann sum using $n = 50$ midpoint rectangles to approximate the integral $\int_1^3 \sin(x)\,dx$. Compare with the exact answer.

As we know from Riemann sums, $\int_a^b f(x)\,dx \approx \frac{b-a}{n} \sum_{i=1}^n f(x_i)$, where $x_i$ is any point in the $i^{\text{th}}$ subinterval. We shall use the **midpoint of each subinterval**. Note that in this case, if $\Delta x = \frac{b-a}{n}$, then $x_i = a + (i - 0.5)\,\Delta x$.

```
In[•]:= Clear[f]
        f[x_] := Sin[x]
        dx = (3 - 1) / 50;
        dx * Sum[f[1 + (i - 0.5) * dx], {i, 1, 50}]
        Integrate[f[x], {x, 1.0, 3.0}]

Out[•]= 1.5304

Out[•]= 1.53029
```

## Example 3. **(Convergent series)** As we know, the series $\sum_{k=0}^{\infty} a\,r^k$ converges if and only if $|r| < 1$. Moreover, if this series converges, then it converges to $\frac{a}{1-r}$. Verify this for $a = 1$ and $r = 0.8$.

```
In[•]:= Sum[1 * 0.8^k, {k, 0, Infinity}]
        1 / (1 - 0.8)

Out[•]= 5.

Out[•]= 5.
```

## Example 4. **(Taylor Series)** Compute the truncated Taylor series for $f(x) = \sin(x)$ at the point where $x = 1$, up through the first 8 terms (i.e., up to and including the $x^7$ term). Compare with the exact answer.

We can compute Taylor series two different ways in Mathematica.
1. We can do it the hard way, building it by hand.
2. We can use Mathematica's built-in function **Series[f[x],{x,a,n}]**, where $a$ is the value at which we want to center the series solution and $n$ is the polynomial order we want up to (e.g., $n = 7$ means we want up to and including the $x^7$ term).

We shall use both ways here centering about $a = 0$. Recall that the Taylor series representation is

$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$

```
In[•]:= Clear[f]
        f[x_] := Sin[x]

        (* Method 1 *)
        Sum[(Derivative[n][f][0]) * (1.0 - 0)^n / (n!), {n, 0, 7}]
        (* Method 2 *)
        Series[f[x], {x, 0, 7}]
        Limit[Series[f[x], {x, 0, 7}], {x → 1.0}]
        (* Exact solution *)
        Sin[1.0]

Out[•]= 0.841468

             3     5      7
            x     x      x          8
Out[•]= x - ── + ─── - ──── + O[x]
            6    120   5040

Out[•]= 0.841468

Out[•]= 0.841471
```

# 13. Lists and Loops

Most students come into calculus without any prior experience coding in any language. As such, the idea of “loops” is a foreign concept. We shall take a moment to go over what loops are. There are really two different types of loops that we care about: **“for loops”** and **“while loops.”**

**“For loops”** can be thought of as follows -- **“For certain values, I want to perform some action.”**
**“While loops”** can be thought of as follows -- **“Until a certain goal is met, I will continue to update a certain action.”**

Some examples of “for loops” are:
1. For $n = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$, I want to compute $n^2$.
2. Consider a set/list of functions, {sin(x), sin(2x), sin(3x), sin(4x), ...}. For each function $f(x)$ in this set, I want to compute $f'(1)$.
3. Newton’s method is a simple way to find the root of a function $f(x)$, i.e., solving $f(x) = 0$. If we pick a reasonable starting value $x_0$, then we will possibly converge to a root of $f(x)$ using the recursive relationship $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ for $n = 1, 2, 3, \ldots$

An example of a “while loop” is:
We know that as you take more and more terms in a (convergent) series, we will get closer and closer to the exact value. So, let’s keep adding more and more terms (one at a time) until the series approximation is 5% of the exact solution.

A **“list”** can be thought of as a set of objects (these objects could be anything! functions, numbers, species of animals, etc...)

**In Mathematica, we define a list using curly braces, such as typing list = {1,2,3,6,7,8}.**

**You can add/append an object to the list by using the command list = Append[list,newobject].**

**To access the *n*th object in a list, you must use double brackets, e.g., list[[4]] will give you the fourth item.**

## Example 1. **(Appending lists without using loops)** Consider a list containing the objects (i.e., numbers) 1,2,3,6,7,8. Append the values 10,11,12,13. After doing this, what is the fourth object/number in the list?

```
In[∘]:= Clear[list]
        list = {1, 2, 3, 6, 7, 8};
        list = Append[list, 10];
        list = Append[list, 11];
        list = Append[list, 12];
        list = Append[list, 13]
        list[[4]]

Out[∘]= {1, 2, 3, 6, 7, 8, 10, 11, 12, 13}

Out[∘]= 6
```

In Mathematica, we code **“for loops”** using a format similar to the following example:

**For[i=1,i<4,i++,Print[i]]**

Let’s break down what this is saying.

“For *i* starting at 1; print the value of *i*; updated *i* to *i* + 1 (***NOTE*** that is what *i*++ means); and continue only for *i* < 4.”

**!!!WARNING!!! It is VERY IMPORTANT that you include the (i++) in your for loop!!! This is telling Mathematica to “updated” your index/value. If were to not include (i++) in your for loops, then *i* would never be updated and Mathematica would run your loop literally forever. Your computer would crash and you would be forced to restart your computer...no fun.**

## Example 2. **(Appending lists using loops)** Repeat example 1, but use a for loop.

```
In[∘]:= Clear[list1, list2]
        list1 = {1, 2, 3, 6, 7, 8};
        list2 = {10, 11, 12, 13};
        (* NOTE -- the number of items in list2 is 4, so let i≤4 in the for loop *)
        For[i = 1, i ≤ 4, i++, list1 = Append[list1, list2[[i]]]]
        list1
        list1[[4]]

Out[∘]= {1, 2, 3, 6, 7, 8, 10, 11, 12, 13}

Out[∘]= 6
```

## Example 3. **(Appending lists using loops -- Newton’s method)** Use Newton’s method to approximately solve $x^2 - 4x + 1 = 0$ using $n = 10$ iterations and starting at the value $x_0 = -1$. Compare with the exact answer.

Before getting going, let’s make a plan of action. We will need to do 10 iterations. We shall make a list of “updated solutions” from each iteration. After each iteration, we want to add the new solution to the list. Notice that by Newton’s method, newx = oldx - f(oldx)/f’(oldx).

```
In[∘]:= Clear[f, xlist]
        f[x_] := x^2 - 4 * x + 1
        xlist = {-1.0}; (* approximate solution starting at -1 *)
        For[n = 1, n ≤ 10, n++, xlist = Append[xlist, xlist[[n]] - f[xlist[[n]]] / f'[xlist[[n]]]]]
        (* Approximate solution *)
        xlist[[11]]
        (* Exact solution *)
        Solve[f[x] == 0.0, {x}]

Out[∘]= 0.267949

Out[∘]= {{x → 0.267949}, {x → 3.73205}}
```

In Mathematica, we code “while loops” are formatted similar to this example:
**n=1; While[n<4, Print[n]; n++]**
If we are to pick this apart, it reads “start with n=1. While n is less than 4; print n; and then update n = n+1.”

**!!!WARNING!!! Again, it is VERY IMPORTANT that you have the (n++) so that Mathematica doesn’t continue in an endless loop forever!!!**

## Example 4. **(Using a while loop for convergence of a series)** Compute the exact value of sin(1). How many terms are needed in the Taylor series approximation of sin(x) centered about $a = 0$ to approximate the exact value within a tolerance of 0.0001?

Let’s map out a plan of action. Start with a list that will hold our updated values. We want to know -- start with n=0 (i.e., the first term in the Taylor series). While |approx - exact|>=0.0001; update the list and then updated n = n+1.

As can be seen in the results below, there are only 7 items in the list approxvals. This means that the n=8 get an error within our desired tolerance! Note that the 8th item did not get appended to the list approxvals because the while loop had already ended.

```
In[∘]:= Clear[f, approxvals]
        f[x_] = Sin[x];
        exact = Sin[1.0];
        approxvals = {};
        n = 0; While[Abs[Limit[Series[f[x], {x, 0, n}], {x → 1.0}] - exact] >= 0.0001,
          approxvals = Append[approxvals, Limit[Series[f[x], {x, 0, n}], {x → 1.0}]];
          n++]
        Abs[approxvals - exact]
Out[∘]= {0.841471, 0.158529, 0.158529, 0.00813765, 0.00813765, 0.000195682, 0.000195682}
```

# 14. Tables

As we’ve seen when dealing with lists, it is oftentimes visually convenient to display them in a nice table. This is the entire point of this section. Two examples of the command to turn a list into a table (with two columns) in Mathematica are:
**Table[{col1,col2},{n}]//TableForm** to make n copies of {col1,col2}
**Table[{col1,col2},{j,jmin,jmax}]//TableForm** to output {col1,col2} starting from j=jmin to j=jmax; here {col1,col2} are dependent on the index j.

Note that the first command is if you have two lists of the same that are not dependent on an index. The second command is if each list is dependent on an index.

## Example 1. **(Table of two lists)**

```
In[∘]:= col1 = {1, 2, 3, 4, 5, 6};
        col2 = {4, 8, 3, 1, 7, 9};
        Table[{col1, col2}, {1}] // TableForm
Out[∘]//TableForm=
        1    4
        2    8
        3    3
        4    1
        5    7
        6    9
```

## Example 2. **(Table of two lists dependent on index)** Repeat example 3 and 4 from the Loops Section and put the results into tables.

```
(* Example 3 *)
```

```
In[∘]:= Clear[f, xlist]
        f[x_] := x^2 - 4 * x + 1
        xlist = {-1.0}; (* approximate solution starting at -1 *)
        For[n = 1, n ≤ 10, n++, xlist = Append[xlist, xlist[[n]] - f[xlist[[n]]] / f'[xlist[[n]]]]]
        Table[{n, xlist[[n]]}, {n, 1, 10}] // TableForm
Out[∘]//TableForm=
        1     -1.
        2     0.
        3     0.25
        4     0.267857
        5     0.267949
        6     0.267949
        7     0.267949
        8     0.267949
        9     0.267949
        10    0.267949
```

```
In[∘]:= (* Example 4 *)
        Clear[f, approxvals]
        f[x_] = Sin[x];
        exact = Sin[1.0];
        approxvals = {};
        n = 0; While[Abs[Limit[Series[f[x], {x, 0, n}], {x → 1.0}] - exact] >= 0.0001,
         approxvals = Append[approxvals, Limit[Series[f[x], {x, 0, n}], {x → 1.0}]];
         n++]

        numberitems = approxvals // Length; (* number of items in the list approxvals *)
        Table[{n, approxvals[[n]]}, {n, 0, numberitems}] // TableForm
Out[∘]//TableForm=
        0    List
        1    0
        2    1.
        3    1.
        4    0.833333
        5    0.833333
        6    0.841667
        7    0.841667
```

# 15. Manipulate[]

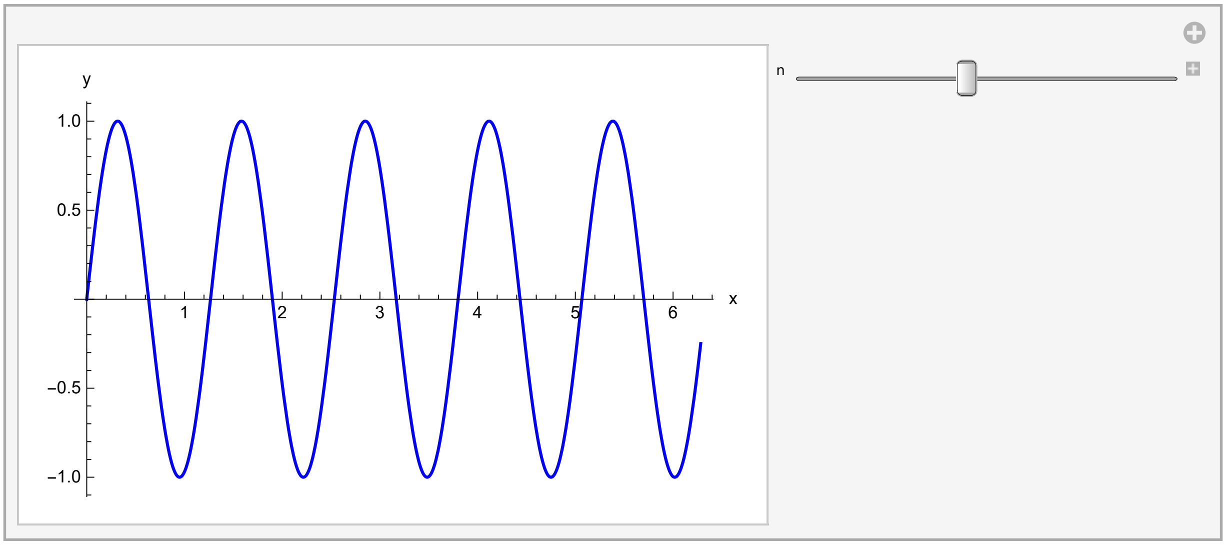
A convenient function that's built into Mathematica is the Manipulate[] function. It allows you to change any parameter and see how the result differs. This can be applied to equations, integrals, plots, etc...
The general format of this function is
**Manipulate[whatever you want,{parameter1,a,b},{parameter2,c,d},...]**
In other words, you tack on the end whatever range you want to test for each parameter.
I think this is better shown through examples, so please read below.

An easy way to think about Manipulate[] is to simply type out the Mathematica code as you normally would. Then, put that ALL inside Manipulate[] and just tack on the range of each parameter you are varying.

## Example 1. Plotting the function sin(nx) for *n* ranging from 1 to 10.

```
In[∘]:= Manipulate[Plot[Sin[n * x], {x, 0, 2 * Pi},
          Axes → True, AxesLabel → {"x", "y"}, PlotStyle → Blue], {n, 1, 10}]
```

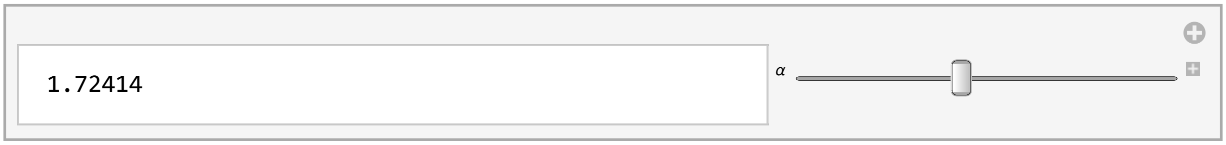
Out[∘]=



## Example 2. Evaluating the integral of $f(x) = \frac{1}{x^{\alpha}}$ from 0 to 1 where we change the parameter $\alpha$ from 0 to 0.99.

```
In[∘]:= Manipulate[Integrate[1 / x^α, {x, 0, 1}], {α, 0, 0.99}]
```

Out[∘]=

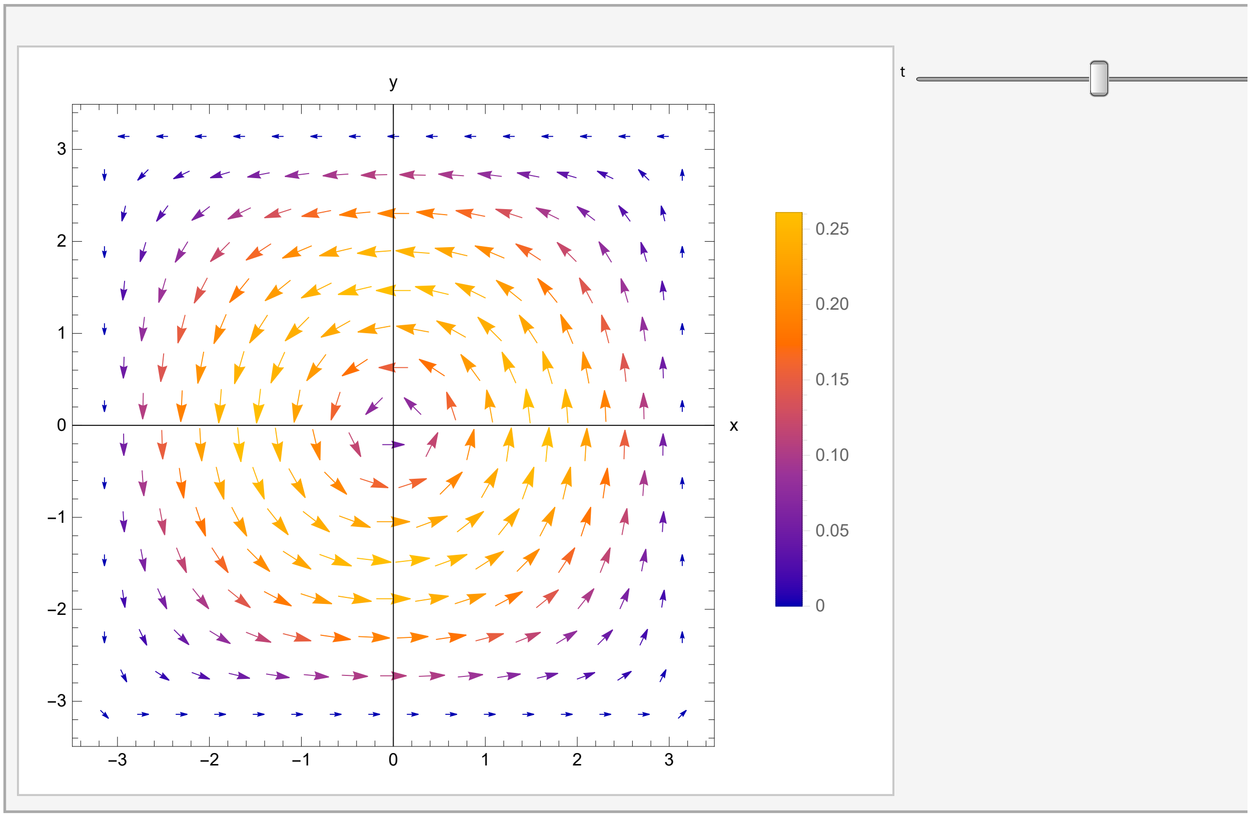


```
1.72414
```

## Example 3. Plotting the vector field $\vec{F}(x, y, t) = \langle \pi\cos^2(\frac{x}{2})\sin(y)\cos(\frac{\pi t}{3}), \pi\cos^2(\frac{y}{2})\sin(x)\cos(\frac{\pi t}{3})\rangle$ for $0 \le t \le 3$ on the domain $-\pi \le x, y \le \pi$.

```
In[185]:= Clear[fvec]
          fvec[x_, y_, t_] :=
           {-Pi * (Cos[x / 2]) ^2 * Sin[y] * Cos[Pi * t / 3], Pi * (Cos[y / 2]) ^2 * Sin[x] * Cos[Pi * t / 3]}
          Manipulate[VectorPlot[fvec[x, y, t], {x, -Pi, Pi}, {y, -Pi, Pi}, Axes → True,
            AxesLabel → {"x", "y"}, VectorScaling → Automatic, PlotLegends → Automatic], {t, 0, 3}]
```

Out[187]=


t
y
3
2
1
0
−1
−2
−3
x
−3
−2
−1
0
1
2
3
0.25
0.20
0.15
0.10
0.05
0