A Mathematica Crib Sheet
(But It's OK, You'll Need One)

Mathematica is a full-featured programming language, with over 1500 commands built-in or in standard "packages". But it's like speaking English: an educated person probably knows a quarter of a million words, but only about a hundred and fifty different words are really needed to get through a whole week. Similarly, there are only a few commands that you'll need to run Mathematica well enough to learn calculus.

Think of this sheet as an emergency road-side manual, dedicated to getting you "up and running."

The Golden Rule: *Try It!*

Instead of asking, "What happens if I...", **do it!**

Experiment! Try commands out to **see** what they do. Try commands we haven't talked about. We promise: the computer won't catch fire, no matter how bad a mistake you make. (As long as it doesn't involve matches...)

Four Mistakes You'll Make Over and Over!

1. To end input you must hit "Enter" or "Shift-Return," not just "Return." Commands to Mathematica can span multiple lines, so if you just hit "Return" Mathematica thinks you haven't finished typing the command for that input cell, and waits for you to continue.
2. Mathematica functions and procedures use **square brackets** around their arguments, not parentheses!
3. Mathematica is very sensitive to the **case** a word is typed in. All Mathematica functions and procedures start with capital letters, but the rest of the letters must be lower-case, unless they effectively start a new word – for example, `FactorInteger`.
4. Ending an expression with a **semicolon** tells Mathematica to **suppress that output**. If you're not getting *any* answers at all, check for semicolons.

Getting Help from the Computer

There are two important tricks to keep in mind:

1. If you want to know something about a Mathematica function or procedure, just type a question mark followed by the name of the procedure. Example:

   ```
   ?FactorInteger
   ```

   After you press **`Enter`** Mathematica responds:

```
FactorInteger[n] gives a list of the prime factors of the integer n,
together with their exponents.
```

To get still more help, type **two** question marks in front of the name.

2. If you can't remember the full name of a Mathematica function or procedure, but can remember a few letters at the start, there is a "fill-in" capacity: after typing a few letters, type **`command-K`** (simultaneously press down the "open apple" key, located next to the space bar, with the letter "**`K`**"). Mathematica will either finish typing the name for you – if there is only one command which starts with those letters – or will present you with a "pop-up menu," a list of possible matching commnads. By clicking on one of these you select the complete command.

Some Common Mathematica Functions

`x^y`	x to the y-th power (exponentiation). Try `0^0`, `0^-1`.
`x*y`	x times y.
`x y`	x times y (one or more spaces separate the variables).
`2 x`	2 times x. Any numerical constant can replace 2. But note that `x2` doesn't mean x times 2, it's the name of a new variable.
`Sqrt[x]`	Square root of x.
`Exp[x]`	`E^x`. NOTE: the base of the natural logs is denoted by `E`, not `e`. Start with a capital letter!
`Log[x]`	Natural logarithm of x. Try `Ln[x]` just to see what happens.
`Log[b,x]`	Logarithm of x to the base b. Try `Log[-1,2]`. Can you figure this out?
`Sin[x], Cos[x], Tan[x], Csc[x], Sec[x], Cot[x]`	Trigonometric functions. NOT `Ctn[x]`.
`ArcSin[x], ArcCos[x], ArcTan[x], ArcCsc[x], ArcSec[x], ArcCot[x]`	Inverse trig functions. Remember to capitalize not only the first letter `A`, but also the first letter of the trig function you're inverting.
`n!`	Factorial of n. `Factorial[n]` also works.
`Abs[x]`	Absolute value of x (even if x is complex). Note that `|x|` doesn't work.
`Round[x]`	Closest integer to x.
`Floor[x]`	Largest integer <= x. `Floor[2.1]` = 2, but `Floor[-2.1]` = -3.
`Ceil[x]`	Smallest integer >= x.
`Mod[n,m]`	Remainder on division of n by m (for integers only; use `PolynomialRemainder[p,q,x]` if p and q are polynomials in x).
`Random[]`	Random number between 0 and 1.
`Max[x,y,...]`	Largest of x, y, ... If x and y are unassigned variables,

	Mathematica won't have any idea which is largest, and will return `Max` unevaluated.
`Min[x,y,...]`	Smallest of x, y, ... (duhhhh)

Some Mathematica Constants

`E`	Base of the natural logs, about 2.71828. Ask for `N[E,50]`.
`Pi`	About 3.14159... You'll forget at least once and write `pi`, which Mathematica will treat as a new variable, leading you to bewilderment.
`I`	The square root of -1. NOT `i`; remember, CAPITALIZE THE FIRST LETTER. You saw this guy when you asked for `Log[-1,2]`.
`Infinity`	Try `Infinity+Infinity`, `Infinity/Infinity`, then ask Mathematica what the devil `ComplexInfinity` is.

Using Previous Results
(*These save a lot of typing!*)

`%`	The result just generated.
`%%`	The result just before that.
`%%%`	The one just before *that*; and so on...
`%n`	The result generated on output line `Out[n]`.

Assigning Values

`x = value`	Assign a value to the variable x.
`x = y = value`	Assign a value (the same value) to BOTH x and y.
`Clear[x]`	Clear the value previously assigned to x (if any).
`x := value`	Assign the value to x, but don't do it right away; wait until x is actually used.
`x == y`	Tests whether x is equal to y. You may think it a perversion to take the innocent = sign, which is used in mathematics to denote a statement of fact ("x is equal to y"), and turn it into a verb ("x = y" becomes "set x equal to y"), after which you must invent a monster == to regain your statement of fact ("x is equal to y"), but there you have it – computer science majors will recognize "C".

Values you assign remain assigned until you change them!! If you set `x = 5` and then type

```
Solve[x^2 - 2x + 1 == 0, x]
```

(asking Mathematica to solve the quadratic equation for x), Mathematica returns an empty solution set! You have asked it to solve the equation `5^2 + 2*5 + 1 == 0`, and Mathematica realizes this has no solution.

Moral: to be safe, before using a variable such as x you should always type **`Clear[x]`**.

Replacements

`expr /. x->value`	Replace every occurrence of x in the expression expr with value. (Read: "expr, **given that x** is changed to value.")
`expr /. {x->xval,y->yval}`	Perform several replacements simultaneously.

For example, suppose we want to check that Mathematica got the correct roots of the quadratic equation `x^2 - 7x - 3 == 0`. Remembering the fiasco where we had set `x = 5`, we begin with

```
Clear[x]; Solve[x^2-7x-3==0,x]
```

Mathematica replies

```
Out[5]=
       7 - Sqrt[61]          7 + Sqrt[61]
{{x -> ------------}, {x -> ------------}}
            2                     2
```

The result is a *list of transformation rules*. Now we type

```
x^2-7x-3/.%5[[1]]
```

that is, we ask for the value of `x^2 - 7x - 3`, given that the first transformation rule in `%5` is applied. To understand the `[[1]]`, read the next section on **Lists and Iterators**.

Lists and Iterators

`{a,b,c}`	A *list* (in this case, of the three objects called a, b and c).
`{}`	A *list* (of **no** objects!).
`a[[n]]`	The n-th item in the list a. Note the **double brackets!!**
`Sum[a[[n]],{n,3,7}]`	The sum `a[[3]] + a[[4]] + a[[5]] + a[[6]]`. Here `{n,3,7}` is what is called an *iterator*, that is, a statement which says to repeat something; in this case, it means the sum of `a[[n]]` for n = 3 to 7, inclusive.
`Sum[a[[n]],{n,1,7,2}]`	The sum `a[[1]] + a[[3]] + a[[5]] + a[[7]]`. This form of the iterator says to let n run from 3 to 7, inclusive, *incrementing n by 2* at each step. (That's an easy way to get the sum of all the odd terms, or all the even terms.)
`Plot[f[x],{x,a,b}]`	Plot the graph of the expression `f[x]`, from x = a to x = b, inclusive. `{x,a,b}` is still called an iterator, because x takes on values between a and b, although of course not *all* real values between a and b.

Transforming Algebraic Expressions

`Expand[expr]`	Multiples out products and powers.
`ExpandAll[expr]`	Apply "Expand" everywhere throughout in expr.
`Factor[poly]`	Factors a polynomial over the integers.
`FactorTerms[poly]`	Pull out common factors that appear in each term of a polynomial.
`Together[expr]`	Put all terms over a common denominator.
`Apart[expr]`	Separate into terms with simpler denominators.
`Cancel[expr]`	Cancel common terms between numerators and denominators.
`Collect[expr,x]`	Group together powers of x.
`Simplify[expr]`	Try a sequence of algebraic transformations and give the simplest form of expr found.
`Numerator[expr]`	The numerator of expr.
`Denominator[expr]`	The denominator of expr.

Defining a Function

`f[x_] := x^3-x`	Define a new function f. **Note the underscore on the x on the left side of the statement!** It *must* be there; it tells Mathematica to treat x as a *pattern*; thereafter, when you type something like `f[a+b]`, Mathematica will then immediately consider that to be the same as `(a+b)^3-(a+b)`.

Derivatives and Integrals

`D[y,x]`	The derivative of the expression y with respect to x.
`f'[x]`	The derivative of `f[x]` with respect to x (if f has previously been defined as a function).
`D[y,x,x], f''[x]`	Second derivative with respect to x.
`Integrate[f[x],x]`	The indefinite integral of `f[x]` with respect to x. Mathematica will try to evaluate the integral; if it cannot, it will return your expression unchanged.
`Integrate[f[x],{x,a,b}]`	The definite integral of `f[x]` with respect to x on the interval [a,b]. If Mathematica cannot evaluate the integral, it will return your expression unchanged.

Miscellaneous Useful Procedures

`Solve[x^2+x==1,x]`	Solve the given equation x^2 + x = 1 for x. Note that we must use the double-equals sign. Also, only an *exact* solution will satisfy Mathematica. You can also solve *systems* of equations, e.g. `Solve[{x+y==1,x^2+y^2==1/2},{x,y}]` solves those two

	equations for x and y. (Can you do this by hand?)
`Roots[f[x]==0,x]`	Finds the roots of a *polynomial* equation. Only exact solutions will be reported.
`NRoots[f[x]==0,x]`	Finds the roots of a *polynomial* equation, but numerically, i.e. approximately.
`FindRoot[f[x]==0,{x,a}]`	Find roots of `f[x]` = 0 *numerically*, starting at x = a. Works for non-polynomials, but may fail to find the roots, or all of the roots, even when they exist.

Yes, Virginia, There *Is* a Limit

`Limit[f[x],x->a]`	The limit of `f[x]` as x approaches a. a **can be** `Infinity` or `-Infinity`. You won't be asked to use this very much, for fear of corrupting your morals.