Maple Tips & Tricks

Maple is a **Computer Algebra System** (CAS). A CAS can do everything a graphing calculator can do and a whole lot more. *Mathematica* is another popular CAS. In fact, the *Wolfram Alpha* website is essentially online Mathematica plus a natural language recognition interface. Maple and Mathematica are quite similar in their capabilities and (student) pricing. There are some things that Mathematica does better (generally it produces nicer looking graphs) while other things are handled better by Maple. Both CAS's are used here at Appalachian State, but the Department of Mathematical Sciences has mainly adopted Maple for its courses requiring a CAS. While each CAS uses different commands, it is comforting to know that once you have learned to use one CAS, it is much easier to learn any other CAS. There is also an open source CAS called *SAGE* whose syntax is built on the Python programming language. While SAGE is quite powerful, open, and free, it has a steeper learning curve than both Maple and Mathematica and (currently) is lousy at handling 3D graphics.

Of course there are many other mathematical software suites. Some of the most popular choices are focused on numerical computations (and thus not suitable for us since we need symbolic support). For example, MATLAB is widely used for numerical computations by both applied mathematicians and scientists of all kinds. Excel is not exactly mathematically sophisticated, but it is probably the most widely used computational software. R and SAS are among the most popular statistical softwares. Geometer's Sketchpad and Geogebra are used for geometry.

There are two main ways of interacting with Maple: worksheet mode or document mode. Documents look like and behave much like Microsoft Word documents. Worksheets are the older of the two interfaces. I will be using worksheets since they clearly delineate between text/description and computation. In a worksheet "`[>`" is a command prompt whereas (empty) text boxes look like "[".

- To get a new command prompt press the `[>` button. For a new text box press the T button.
  [Both are located along the top ribbon.]

- F5 toggles the input mode between math mode and text mode. You can also switch by clicking Text or Math.

  When you input in text mode `commands are in red text` and not pre-formatted. In math mode, Maple pre-formats expressions. For example, in text mode typing `x` then `^` then `2` yields `x^2` whereas in math mode we would see x^2.

  I prefer typing in text mode where what you see is what you typed. Whereas math mode easily hides errors. If you're code isn't working correctly, debugging math mode formatted commands is much much harder.

- Enter executes commands at a command prompt. Shift+Enter gives a new line at a command prompt (without executing the command). Generally when a command is executed, it makes a change to memory. Sometimes the command print some blue output to indicate what it did.

  Be careful: Commands are affected by what Maple currently has stored in Memory. If you jump around and execute commands out of order, you may get undesired results. Maple does not care about how you've ordered commands in your file. It just cares about the order of their execution!

- Lines of code end with either a colon ":" or a semicolon ";". A colon signals that Maple should suppress output (although it will still show certain warnings and errors) whereas a semicolon tells Maple to go ahead and display any relevant output (usually printed in blue below the command prompt).

- Executing `[> ? topic` will cause Maple to lookup help on `topic`. Of course, you can also use the Help menu. While Maple has extensive help, its help pages are rather uneven. Some pages give great explanations and examples while others make very little sense. Please keep in mind that Maple is quite popular, so you are likely to find what you need by Googling around. Better yet... go to my *Maple examples* page!

- `[> restart;` clears memory. I like to put this at the beginning of various sections of my worksheets.

- `[> with(PACKAGE):` loads `PACKAGE` into memory. Notice the colon. If you use a semicolon, Maple will list off all of the procedures you just loaded. Packages I use often:

  - `[> with(plots):` loads various plotting commands like the `display` command which allows one to put various plots together in one picture.
  - `[> with(VectorCalculus):` loads a bunch of multivariable calculus tools.
  - `[> with(LinearAlgebra):` loads various matrix and other linear algebra tools. You might run into the old linear algebra package `linalg`. You probably want to avoid using this old package since it's no longer fully supported.

- Assignment versus equations: `a = 5` is an equation and essentially asks "Is a equal to 5?" On the other hand, `a := 5` assigns the value 5 to the variable named `a`.

- **Important Warning:** Maple is case sensitive. For example, `A` and `a` are different variables. When someone is stuck and Maple isn't working, I often find that they have misspelled or miss-capitalized a command. For example, `ArcLength` computes the arc length of a vector valued function (this is part of the vector calculus package). If we typed `arclength` or `Arclength` this won't work.

- **Important Warning:** Maple *forgets everything* when it is shutdown. If you save your file, shutdown Maple, and then reopen your sheet, you will still see blue computation outputs from previous computations. This does not mean that Maple remembers these results. So just because you at some point in the past executed `[> a := 5;` and you see $a := 5$ displayed below it, does not mean Maple still remembers this. I like to refer to the blue output as ghosts of computations past.

  Probably more than 9 out of 10 problems I see with Maple are solved by going **back to the beginning of a worksheet and reexecuting everything**.

  On a related note, if we type `[> mycommand(1,2,3);` and Maple returns mycommand(1,2,3);, that is Maple's way of admitting it doesn't know what you're talking about (it just repeats back exactly what you told it). If this happens, double check your command's spelling/capitalization. Then make sure you reexecuted all `with` commands when you reopened your worksheet.

- Most arithmetical and standard functions have unsurprising symbols and names: `+ - * / ^` are add, subtract, multiply, divide, and exponentiate. `sin(x) cos(x) tan(x) sec(x) csc(x) arcsin(x) ln(x)` are sine, cosine, tangent, secant, cosecant, arcsine (i.e., inverse sine) and natural log.

  Important Warning: The exponential function e^x is `exp(x)`. If you type `e^x`, Maple will raise the variable `e` to the variable `x` power – almost certainly, this is something you'll never want to do. Accidentally typing `e^x` instead of `exp(x)` is one of the *most common mistakes* I see.

- **Important Warning:** The mathematical constant $\pi \approx 3.14159265$ is `Pi` in Maple. On the other hand, `pi` is the variable lowercase Greek letter π. Capitalization matters! Forgetting to capitalize pi is another *very common mistake.*

- Maple by default works with exact symbolic expressions. Commands like `evalf` tell Maple to give us a decimal approximation (the f in evalf stands for *floating point* approximation). For example, `[> evalf(sqrt(5));` will return an approximation of $\sqrt{5}$ with 10 digits of accuracy. Likewise, `[> evalf(Pi, 1000);` will give a 1,000 digits of π.

- Maple has context sensitive menus. In older versions, if you right-click on some blue output, Maple will pop-up a menu with a selection of suggested (hopefully relevant) commands. Starting in Maple 2019, if you right-click, the last option is "Open Context Panel for more...". This will open a side panel with a list of commands you can perform. For example, if I execute `[> x^2+3*x-6=0;`, right-click on the blue output: $x^2 + 3x - 6 = 0$, and open the context panel, then I get choices like "Differentiate" and "Solve".

- To solve an equation using a context panel you can use Solve→Solve, Solve→Numerically Solve, or Solve→Numericall Solve from point (among other options). If you give Maple an expression instead of an equation, it will try to solve that expression set equal to zero.

  Solve→Solve is the general solver. The associated command is `solve(EQUATION TO SOLVE)`. This solver will try to find all possible *exact symbolic* solutions. Note that sometimes solve returns nothing. This does not mean that there are no solutions. It just means Maple couldn't find any. It is good to stay a little skeptical about this command. If an equation or its solutions are too complicated, Maple can't/won't find them.

  Solve→Numerically Solve looks for approximate solutions. The associated command is `fsolve(EQUATION TO SOLVE)`. The "f" in fsolve stands for floating point. fsolve will try to find all solutions up to a certain number of digits of accuracy. Again, be skeptical. Sometimes fsolve will find no solution or only one solution when there are many solutions.

  A variant of numerically solving is Solve→Numerically Solve from point. This time Maple asks for a starting value (essentially a first guess). The associated command is `fsolve(EQUATION TO SOLVE, GUESS)`. If we are trying to solve really tricky equations and Maple isn't giving us the solution we want, this command allows us to help Maple out.

- `[> plot(x^2,x=-1..2);` creates a plot of $y = x^2$ with domain $-1 \le x \le 2$. If we wanted to adjust our plot, there are many plot options. For example, `[> plot(x^2,x=-1..2, color=green, scaling=constrained);` changes our function's plot to green and doesn't rescale axes (by default Maple will rescale so the horizontal and vertical axes are shown with the same length regardless of how long they actually are.

- **Functions vs. Expressions:** If we want to define the function $f(x) = x^2$, we use `[> f := x -> x^2;`. Breaking this down: `x^2` is the expression x^2, `x -> x^2` is the mapping that sends x to x^2. Then `f := x -> x^2;` assigns the name f to the mapping $x \mapsto x^2$.

On the other hand, `[> g := x^2;` just assigns the expression x^2 to g. If we executed both of these commands, f is a function and g is an expression. This means that if we execute `[> f(2);` we should expect the output 4. On the other hand, to plug $x = 2$ into g, we need a substitute command like `[> subs(x=2,g);`. In my example pages you will often see things like,

```
[> f := x -> x^2:
   'f(x)' = f(x);
```

If executed, these commands return $f(x) = x^2$. Notice that the first command ends in a colon so its output is suppressed. This command is the one that actually defines the function. The second command merely prints out our pretty $f(x) = x^2$. Notice the quotes: `'f(x)'` this tells Maple not to actually plug x into f but instead just to print out $f(x)$.

- Differentiation is done with `diff` and integration is done with `int`. For example, `[> diff(sin(x), x);` computes the derivative of $\sin(x)$. Likewise, `[> diff(sin(x), x,x,x);` computes the third derivative of $\sin(x)$. Similarly, `[> int(sin(x), x);` computes an antiderivative of $\sin(x)$. Note: Maple does not add in the arbitrary constant – be careful. Finally, `[> int(sin(x), x=-Pi..2*Pi);` computes the definite integral $\int_{-\pi}^{2\pi} \sin(x)\,dx$.

  Note on inert commands: Many commands have an *inert* companion: `diff` vs. `Diff`. The only difference between these commands is that Maple actually executes diff while is holds off final execution of Diff. I will often use this to make Maple display integrals. For example, `[> Int(sin(x), x=-Pi..2*Pi) = int(sin(x), x=-Pi..2*Pi);` when executed displays $\int_{-\pi}^{2\pi} \sin(x)\,dx = -2$.

- **Vector Calculus:** Vectors are defined as we might expect from class. `[> v := <1,2,3>;` defines $\mathbf{v} = \langle 1, 2, 3\rangle$. Maple displays $(1)e_x + (2)e_y + (3)e_z$. Here $e_x = \mathbf{i} = \langle 1, 0, 0\rangle$, $e_y = \mathbf{j} = \langle 0, 1, 0\rangle$, and $e_z = \mathbf{k} = \langle 0, 0, 1\rangle$.

  The dot product is just a period: `<1,2,3>.<4,5,6>` computes $(1)4 + (2)5 + (3)6 = 32$. So `sqrt(v.v)` computes $\sqrt{\mathbf{v} \bullet \mathbf{v}} = |\mathbf{v}|$. You can also compute the length of a vector using `Norm(v)`. **Be very careful:** `Norm`, `norm`, and `length` all return a number but only the `Norm` returns the value we want! There is also a command to normalize called `Normalize`. So `Normalize(v)` and `v/Norm(v)` do the same thing.

  The cross product is denoted by `&x`. So `<1,0,0> &x <0,1,0>` returns e_z since $\mathbf{i} \times \mathbf{j} = \mathbf{k} = e_z$.

- We will introduce several other useful vector calculus commands in homework assignments. I will just mention two commands for 3D plotting.

  First, `plot3d` plots surfaces. For example, `plot3d(x^2+y^2,x=-1..1,y=-1..1);` plots the paraboloid $z = x^2 + y^2$ over the domain $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$. This plot won't look quite how we envision a circular paraboloid. We usually plot this *circular* paraboloid over a circular domain. Notice that the region $x^2 + y^2 \leq 1$ can be described by $-\sqrt{1-x^2} \leq y \leq \sqrt{1-x^2}$ and $-1 \leq x \leq 1$ (lower to upper semicircle and negative to positive radius). Thus the command, `plot3d(x^2+y^2,y=-sqrt(1-x^2)..sqrt(1-x^2),x=-1..1);` plots the paraboloid over a disk and looks much better.

  Second, to plot curves in 3-space we use `spacecurve`. For example, `spacecurve(<t,t^2,t^3>,t=-1..2);` plots the twisted cubic $\mathbf{r}(t) = \langle t, t^2, t^3\rangle$ when $-1 \leq t \leq 2$.