A Brief Maple Tutorial
Part 1: Arithmetic and Simple Algebra

Michael Penna, Indiana University – Purdue University, Indianapolis

Objective

In this project we discuss some of the arithmetic and algebraic capabilities of Maple. Since Maple can be run on different types of computers and under different operating systems, and since the choice of computer and operating system used for this and future projects is yours, we make minimal platform-specific comments. Questions regarding specific platforms — including how to get Maple up and running, how to save, edit, and print — are left to you to resolve.

Narrative

Maple combines the features of a computer algebra system with those of graphing software in one package. In this project you'll learn about some of Maple's basic algebraic features; in the next project you'll learn about some of its graphing features. To complete this project you must be able to get Maple up and running, edit, save, and print.

Task

Once you have Maple up and running, type the command lines in the left-hand column below into Maple in the order in which they are listed. The effect of each command is described in the right-hand column for your reference; *do not* type what is written in the right-hand column into Maple! If your output does not agree with what is written in the right-hand column, then you've probably made a mistake; in this case, check what you've written: you must go back and correct your mistake before going on! A message such as "`Error ...`" also means that you've made a mistake; it, too, must be corrected before going on! Your lab report will be a hard copy of your typed input and Maple's responses.

`> # Your name and today´s date`	This is a comment. See Comment 2 at the end of this project.
`> # Arithmetic and Simple Algebra`	This is the project title.
`> restart;`	Clear Maple's memory.
`> 6+7;`	Add 6 and 7.
`> 6+7:`	Maple performs the addition, but the colon suppresses any output.
`> 9-3;`	Subtract 3 from 9.
`> 4*3;`	Multiply 4 by 3.
`> 24/2;`	Divide 24 by 2.
`> 1/2+1/3;`	Add 1/2 and 1/3.
`> evalf(%);`	Express the previous value as a decimal.
`> Digits := 20;`	Henceforth express decimals to 20 places.
`> evalf(%%%);`	Express the third-last value as a floating point number.
`> 1/(2+1/3);`	Note that this *isn't* $1/2 + 1/3$! The moral: watch your parentheses!
`> Pi;`	Write π.
`> evalf(Pi);`	Write π as a decimal to 20 places.
`> pi;`	Write π.
`> evalf(pi);`	Try to write π as a decimal to 20 places. You shouldn't get a number here: Maple is "case sensitive": it sees `pi` and `Pi` as two different quantities.

`> x := 5;`	Let x be 5.
`> x^3;`	Find x^3.
`> x^(1/2);`	Find $x^{1/2}$.
`> evalf(x^(1/2));`	Write $x^{1/2}$ as a floating point number.
`> x := ´x´;`	Reestablish x as a variable.
`> x^2;`	Find x^2.
`> 17!;`	Find 17!.
`> p := (x-4)*(3*x+2);`	Let $p = (x-4)(3x+2)$.
`> expand(p);`	Expand p.
`> factor(p);`	Factor $p = 3x^2 - 10x - 8$.
`> solve(p=0,x);`	Solve $p = 3x^2 - 10x - 8 = 0$ for x *symbolically.*
`> fsolve(p=0,x);`	Solve $p = 3x^2 - 10x - 8 = 0$ for x *numerically.*
`> subs(x=5,p);`	Substitute 5 for x in p.
`> subs(x=x+h,p);`	Substitute $x + h$ for x in p.
`> expand(%);`	Note that if we replace $x + h$ by x in $3x^2 - 10x - 8$, we *don't* get $3x^2 - 10x - 8 + 3h^2 - 10h - 8$.
`> expand((x+2)^8);`	Expand the expression $(x + 2)^8$.
`> eqns := {x-y-z=0,6*x+4*z=12,3*y-4*z=-3};`	Let `eqns` denote the system of equations $x - y - z = 0, 6x + 4z = 12, 3y - 4z = -3$.
`> sols := solve(eqns,{x,y,z});`	Solve the system `eqns` for x, y, and z.

At this time, make a hard-copy of your typed input and Maple's responses; this hard-copy will be your lab report.

Comments

1. Observe that when using Maple you must be careful and explicit when using mathematical notation. For example, not only must you be careful when using parentheses as indicated above, but you must also be careful when using operations: for example, you must write `3*x+2`, being explicit about multiplication by using the multiplication sign $*$, rather than writing $3x + 2$ which you might write when working by hand.

2. As you work through this and subsequent projects, think about what you're doing: think about what you're typing, why you're typing it, and what you are — or should be — getting as output. Errors often arise from simple typographical mistakes, and one of the best indicators that you've made a typographical mistake is that you're getting output that is not what it should be.

3. Comments (such as the first two lines of code in this project) clarify your code but they do not affect computation. Comments always begin with a "#". It is wise, for many reasons, to document your code with at least your name, the date of your work, and the project title. You will, in fact, be asked to do this throughout these projects.