Calculator Examples

```
# Mouse position, copy, paste work as expected.
# Some keys to use for typing:
#  ctrl-z       Undo last change. Repeat to undo previous changes.
#  ctrl-k       Insert a command line prompt, above.
#  ctrl-J       Insert a command line prompt, below.
#  Backspace    Delete character left.
#  Delete       Delete character right.
#  Return       Execute current group [blue printout]
#  Arrows       Move the cursor.
#  :=           Keys : and = assign a variable: x:=1;
#  : =          Common error. No space allowed.
#  ;            Key semicolon ends a line.
#  :            Key colon ends a line no echo (no blue print).



# Let's get started!


# Go ahead and type along in maple with these examples.


 2 + 2; 3*5; 6-2:

# All three computations were done, although only two results are shown
# (the colon : at the end of a command suppresses the output).  If you
# forget the semicolon, go ahead and put it on the next line:

 6-2
 ;

# Basic Math Operations

Addition +, subtraction -, and division / are standard, and parentheses
# are used as in algebra.  However, brackets [, ] and braces {, } are used
# for maple engine list and set delimiters, and not for math. An asterisk * means
# math multiplication and a caret ^ is used for powers. The dot (.) is used for
# decimals, ranges (double dot ..), dot product and matrix multiply. Format
# carefully when using a dot.

 (1 + 2) * (6 + 7) - 12 / 7;
 3^(2.1);

# Computer Algebra and Decimals

Maple by default computes exact quantities. Decimals will not appear
# in an answer unless they appeared in the problem (e.g., 2.1).


#  To give decimals in the answer (floating point), use Maple's evalf:
```

```
 Pi; # The constant 3.1415727... prints as a Greek letter. Must
     # be entered as uppercase P and lowercase i.
 pi; # Symbol, not 3.14. Prints as a Greek letter (confusing isn't it?)

 evalf(Pi);  # Print PI to 10 digits default
 exp(1);    # the number e=2.818... prints as lowercase italic e
 evalf(%); # The % sign stands for the most recently computed quantity.
 e; E;       # symbol e prints in lowercase italic
 evalf(e);  # decimal conversion of a symbol does nothing
 evalf(Pi, 50); # Compute Pi to 50 digits.
 Pi^(1/2);  # Print symbolic answer
 evalf(%);  # Print decimal answer 1.77245385
```

Upper and Lower Case Madness.

Maple code distinguishes upper-case letters from lower-case. Thus evalf(pi) is not the same as evalf(Pi).

#**Spacing**

```
# For the most part, spacing is unimportant in Maple.  In the code lines
# above, spaces could be omitted or added without causing any problems.
# Thoughtful use of spacing makes Maple code easier to read, easier to
# understand, and easier to edit.
```

Standard Mathematical Functions

```
# Maple uses naming conventions of computer languages Fortran and C. To
# find out a name, use maple help (?initialfunctions). A short list:

#  sin, cos, tan, csc, sec, cot
#  sinh, cosh, tanh, csch, sech, coth
#  arcsin, and so on. Use prefix arc on the previous for inverses.
#  sqrt, ln, log, log10, exp, round, trunc, ceil, floor, max, min

#  Re(z) and Im(z) for real and imaginary parts of a complex number a+b*I
#    I is a reserved symbol for the square root of minus one.

# Example.

# Let's compute the absolute value of -14 plus the sine of 1 minus the
# square root of 2 plus the base-e (natural logarithm base) power of
# cos(1.6 Pi) plus the arctangent of 3.

 abs(-14) + sin(1) - sqrt(2) + exp( cos(1.6*Pi) ) + arctan(3);
 evalf(%);
```

#**Degrees and radians**.

```
 tan(45);   # Surprised? Trig functions use radians only.
 tan(45*Pi/180); # Convert 45 degrees to radians

# Maple expression syntax can often be found by intelligent guessing. Thus
# tan(45) does indeed compute the tangent, and 20! computes a factorial.
# If your first guess doesn't work, then use Maple help or switch to a
```

```
# browser search engine, looking for sample code.

#Algebraic Variables

# Maple code uses variables and algebra.  Consider, for example,
# the square of the sum (a + b) with variables a,b.

(a + b)^2;
expand (%);
factor (%);
p := (a + b)^2; b := 1; p;j
# Expand gives the expanded form, and factor brings us back
# to our starting point. To make long computations easier and
# more intelligible, we can assign values to variables using ":="


# Other Variables.

# In the previous examples, variables store an expression or a number.  Variables
# can also store a list of points, a set, a string, an equation, a piece of text, or a function:

 pts := [ [1,2], [3,4] ]; # a double-list or list-of-lists
 eqn := 2*x - 3*y = 5;    # eqn abbreviates equation 2x+3y=5
 eqns := { 2*x - 3*y = 5, 5*x - 3*y = 1 }; # A set of two equations
 tag := "The nth partial sum is";  # string delimiter is a double quote
 print (pts, eqn, eqns, tag);   # check
 f := x -> x^2;         # Defines a function. Use 2 keys, MINUS and GREATER-THAN
 f(2);  f(3);           # Evaluate function f at x=2 and x=3
 g:=unapply(x^2,x);     # defines a function, with recursive symbol evaluation
 g(Pi); g(exp(1.1));    # Evaluate function g at x=3.14159 and x=exp(1.1)=3.004

# Assignment typos.

# Anything we can define or compute in Maple can be assigned to a variable
# for future reference using ":=".  The symbol = by itself is used to test
# equality. A space is NOT allowed between the : and the = in an
# assignment statement. Beware of using equal only when you meant
# colon-equal. Such typos are maddening to discover, because they generate
# no maple error message.

# Getting rid of variable definitions.

 b := 'b'; # same as unassign('b');  Removes b:=1; assignment made above to symbol b.
 p;        # re-execute formula for p, with b:=1 replaced by symbol b

# Similarly, clears the variables assigned above using unassign():

 unassign ( 'pts', 'eqn', 'eqns', 'tag');
 print (pts, eqn, eqns, tag);

 restart; # A drastic way to clear variables and computer memory
           # Also got rid of library loads for linalg and LinearAlgebra

#The restart command clears ALL variables and unloads all packages.
#So, if you need a package later, then you must reload it anew.
```

```
# Quotes.

# Pay special attention to the kind of quotes used in examples.  The
# possibilities are the single quote ', the left quote ` (back-quote),
# and the double quote ".

# Here is an extended example of how to use variables, quote and assignment
# statements:

 F := m*a;  # Newton's formula for force
 m := 2.1;  # set the mass
 a := 5;    # set the acceleration
 F;         # compute the force
 a := 21.9; # reset the acceleration
 F;         # recompute force
 a := 'a';  # clear a with single quotes
 s:="a";    # make a one-character string, no substitution of symbol a
 s:="m";    # one-character string, no substitution of symbol m (m equals 2.1)
 F;         # recompute F, symbol a was restored

# Substitutions.

# The subs command lets us make temporary substitutions in an expression
# as opposed to assigning values.  For example, try these examples:

 g := (a+1)^2 / (b-1)^3 + a / (b-1);
 simplify (g);
 subs( a=3, b=2, g);
 subs( a = x+y, b = x+1, g); # x,y can be symbols or := assigned values or constants
 simplify(%);    # Do all normal algebraic simplifications to last answer %
 a; b;           # The variables a and b were not permanently assigned a value.
```