Chapter 11

Troubleshooting Tips

The Top Ten Headaches Reported by Beginning Maple Users:

1. **Missing or Incorrect Punctuation**
2. **Unmatched Punctuation and Commands**
3. **Confusion between Functions and Expressions**
4. **Unexpected Current Values of Variables**
5. **Using On-Line Help**
6. **Failure to Plot**
7. **Confusing Exact and Approximate Calculations**
8. **Debugging Procedures**
9. **Forgetting to Save a Maple Session and Losing It**
10. **Trying to Get Maple to Do Too Much**

This list is based on the actual experiences of many new Maple users, mostly freshman and sophomore mathematics students and their instructors. They brought with them a wide range of computer experience. However, Maple has its own idiosyncracies, and certain problems seem to arise, repeatedly, for just about everyone. It is hoped that by having these common errors pointed out, along with examples of how they occur, how to spot them, and tips for avoiding them, you will more quickly and painlessly achieve proficiency with this powerful software tool.

Before considering specific examples of common errors, we mention that many times such errors can be prevented by simply changing the font size at the beginning of a Maple session. This sounds trivial, but it's amazing how many students don't think about it, and suffer eyestrain and wasted time correcting errors caused by reading difficulties. In the same vein, many find it helpful to insert blank spaces in Maple statements or break lines at reasonable places and thereby improve readability. However, before you print your worksheet, be sure to check the PRINT PREVIEW and, if necessary, set your font back to a reasonable size, and resize any plots.

11.1 Missing or Incorrect Punctuation

Missing Semicolon or Colon at the End of a Line: The single most common error, and the easiest to fix, is forgetting the semicolon or colon at the end of a statement. In the Classic interface, you get the following error message:

```
>  f := x^2 + 1

Warning, premature end of input
```

Just backspace (twice), add the semicolon and press ⟨ENTER⟩! In the Standard interface, Maple gives a warning, but then inserts the missing punctuation and produces output anyway. Unfortunately, the warning messages make for a messy, hard to read document. To remove the warning, just click at the end of the line, add the semicolon and press ⟨ENTER⟩.

Missing Parentheses: Parentheses are used to group symbols in mathematical expressions, and failure to insert them properly can result in completely different expressions and unintended results. However, in order to correct the error, one needs to *see* the expression that Maple is evaluating.

Three techniques that allow one to see an expression before Maple executes it are using *inert* operators, using the last output `%` operator and assigning names and using them.

Suppose you want Maple to calculate $\sum_{k=1}^{10} \frac{1}{k(k+2)}$, and you enter

```
>  sum(1/k*(k+2), k=1..10);
```

$$\frac{19981}{1260}$$

The summand is not written correctly, so the sum is wrong. You might have caught the error by noticing that the output is too large. (The sum of ten numbers less than one cannot be greater than ten.) A better way to detect the error is to use the *inert* `Sum` and `value`:

```
>  Sum(1/k*(k+2), k=1..10); value(%);
```

$$\sum_{k=1}^{10} \frac{k+2}{k}$$

$$\frac{19981}{1260}$$

This makes it easy to spot the missing parenthesis, so that you can enter the corrected statement.

```
>  Sum(1/(k*(k+2)), k=1..10); value(%);
```

$$\sum_{k=1}^{10} \frac{1}{k\,(k+2)}$$

$$\frac{175}{264}$$

Alternatively, you can use `%` to see how Maple interprets what you wrote, before it is evaluated in a `sum`:

```
>  1/(k*(k+1)); sum(%,k=1..10);
```

$$\frac{1}{k\,(k+1)}$$

$$\frac{10}{11}$$

or you can give the quantity a name and use it:

```
>  ak:=1/(k*(k+1)); sum(ak,k=1..10);
```

$$ak := \frac{1}{k\,(k+1)}$$

$$\frac{10}{11}$$

TIP: Blindly putting an expression that you have not seen into a Maple command, or putting one Maple command inside another Maple command is an invitation to disaster. Always assign the quantity a name, or use `%` or use a combination of an *inert* command, `Sum`, `Limit`, `Diff` or `Int`, to display the operation and then `value(%)` to compute it.

TIP: Try to get into the habit of pausing to examine the output from each line you enter, and asking yourself if it is reasonably close to what you were expecting. When writing a Maple procedure where you won't normally see the intermediate steps, either insert additional print statements or turn on tracing to help with debugging. (See Section 10.3.)

Incorrect Use of Brackets and Braces: Recall that in standard mathematical notation, square brackets and braces are often used like parentheses, as grouping symbols, especially when there are multiple levels of grouping. In Maple, however, square brackets and braces are *never* used as grouping symbols.

Square brackets define ordered lists, such as:

```
>  v:=[2,5,3,1,3,5];
```

$$v := [2, 5, 3, 1, 3, 5]$$

while a list of lists defines a matrix, such as:

```
>  A:=Matrix([[2,5,3,1,3,5],[4,1,2,6,2,1]]);
```

$$A := \begin{bmatrix} 2 & 5 & 3 & 1 & 3 & 5 \\ 4 & 1 & 2 & 6 & 2 & 1 \end{bmatrix}$$

On the other hand, braces define unordered lists or sets where the order of the entries is irrelevant and duplicate entries are ignored, such as

```
>  s:={2,5,3,1,3,5};
```

$$s := \{1,\ 2,\ 3,\ 5\}$$

If you don't use any delimiters, you get a sequence, such as

```
>  a:=2,5,3,1,3,5;
```

$$a := 2,\ 5,\ 3,\ 1,\ 3,\ 5$$

Sequences behave much like lists, except that a list counts as one argument in a function call, whereas a sequence of n expressions counts as n arguments in a function call.

Square brackets are also used to select entries, such as

```
>  v[4]; A[2,4]; s[2]; a[3];
```

$$1$$
$$6$$
$$2$$
$$3$$

In this context, square brackets can be used to produce subscripts, such as

```
>  u[2], b[n], g[1,3];
```

$$u_2,\ b_n,\ g_{1,\,3}$$

but Maple is actually trying to identify entries in undefined lists, sets, matrices or sequences.

However, we emphasize that square brackets and braces are *never* used as grouping symbols.

Missing $*$ for Multiplication: Forgetting an asterisk, `*`, for multiplication can lead to errors which are not obvious. Here are some examples:

```
>  y := 2(x+1) + x^2;
```

$$y := 2 + x^2$$

```
>  y := (2+3)(3+5);
```

$$y := 5$$

```
>  y := (3+y)(x+2);
```

$$y := 8$$

```
>  y := x(y+2); whattype(%);
```

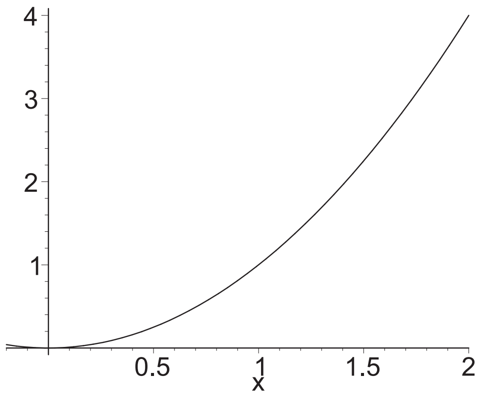
$$y := \mathrm{x}(10)$$
$$\textit{function}$$

In the first three examples, the second factor is apparently just ignored. In the fourth example, the second factor is treated as an argument of a function named x.

In fact, in all four examples, the first factor is treated as a function and the second factor is treated as its argument. This is clear in the fourth example, but what about the other examples? They also become clear when you learn that Maple regards a number as a constant function whose constant value is that number. So for example $2(x) = 2$ for all x.

Using Decimal Points within Ranges: Suppose you want to plot the function x^2 on the interval $-.2 \le x \le .2$. *Notice the decimal points before the 2's.* If you type

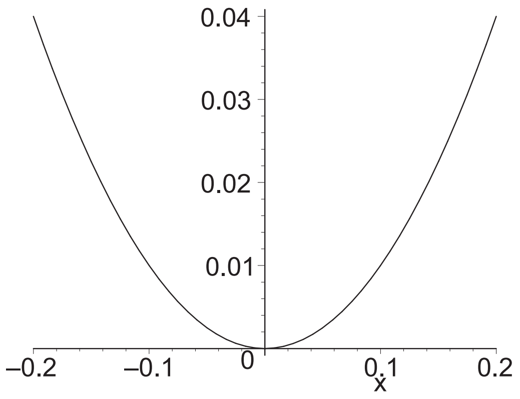
```
>  plot(x^2,x=-.2...2);
```



you get the plot on the interval $-.2 \le x \le 2$. The reason is that the symbol for a range is either `..` or `...` (or *any* number of consecutive periods) so that the last period is interpreted as part of the range symbol, rather than as a decimal point.

Consequently, the correct command to plot x^2 on the interval $-.2 \le x \le .2$ is (with zeros)

```
>  plot(x^2,x=-0.2..0.2);
```



or (with a space)

```
>  plot(x^2, x=-.2..  .2);
```

which produces the same plot.

11.2 Unmatched Punctuation and Commands

Unmatched Parentheses, Brackets and Braces: Maple makes unmatched parentheses fairly easy to find, as the following example shows.

```
>  x*sqrt(2*x+1) / (x^2+5)*(x+1) );

Error, `)` unexpected
```

Similarly, it finds unmatched square brackets and curly braces:

```
>  points:=[[1,2],[3,4,[5,6]];

Error, `;` unexpected

>  sets:=[{1,2},{3,4,{5,6}];

Error, `]` unexpected
```

By the way, it also finds a missing comma:

```
>  points:=[[1,2],[3,4][5,6]];

Error, invalid subscript selector
```

although the error message is cryptic.

Forgetting `end if`, `end do` and `end proc`: The pairs `if...end if`, `do...end do` and `proc...end proc` act like parentheses. If you leave off the closing `end if`, `end do` or `end proc`, Maple complains: For example, if you type the following and press ⟨Enter⟩,

```
>  a:=5;
```

$$a := 5$$

```
>  if a<1 then a^2 + 1 else 3*a;

Warning, premature end of input
```

you get a warning that something is incomplete. Just type `end if` and press ⟨Enter⟩:

```
>  if a<1 then a^2 + 1 else 3*a;
>  end if;
```

$$15$$

The warning goes away and the answer appears. The same happens with incomplete `do` or `proc` statements.

Tip: If you need more than one line to type your input, pressing ⟨Shift-Enter⟩ instead of ⟨Enter⟩ will give a new line without executing the Maple code. This is useful for entering multiline commands, such as `if`, `do` and `proc`. Then, when you press ⟨Enter⟩ without the ⟨Shift⟩, all of the lines will be executed.

Further, using ⟨Shift-Enter⟩ to put your commands on multiple lines not only delays execution until the command is complete, but also make it easier to read, and hence, to debug your Maple code.

What happens when these commands are nested? Consider:

```
>  h := proc(x)
>  if x<1 then x^2 + 1 else 3*x ;
>  end proc;
```

```
Error, reserved word `proc` unexpected
```

This says Maple received an `end proc` when it expected (and required) an `end if`. The remedy is to insert the `end if` and press ⟨ENTER⟩ again.

```
> h := proc(x)
> if x<1 then x^2 + 1 else 3*x end if;
> end proc;
```

$$h := \textbf{proc}(x)\,\textbf{if}\,x < 1\,\textbf{then}\,x^2 + 1\,\textbf{else}\,3 * x\,\textbf{end if end proc}$$

Notice that when a procedure is ended with a semicolon, one gets to see how Maple interprets the code. This is useful when you are debugging the code. However, once the procedure is complete and working, one should usually end the procedure with a colon to suppress the output, which is just a repetition of the input.

TIP: As soon as you type `if`, `do` or `proc` skip a few spaces or a few lines (by typing ⟨SHIFT-ENTER⟩) and type `end if`, `end do` or `end proc`; then back up and type what goes in between. This may help you to remember the `end if`, `end do` or `end proc`.

Unmatched Quotation Marks: Maple has three kinds of quotes, for three different purposes.

Name	Maple Symbol	Purpose
Double Quotes	`"..."`	for strings
Back Quotes	`` `...` ``	label for a variable name
Forward Quote	`'...'`	for delayed execution

Here are examples of the three kinds of quotes and the structures that they delimit:

Strings hold text you might want to display and are enclosed in double quotes (").

```
> "This is a string.";
```

"This is a string."

You can assign names to strings, manipulate them with Maple commands (for selecting substrings or concatenating strings) or print them with the `print` command. For more details, type `?string`. (Maple also has a `StringTools` package, if you want to get fancy.) For example:

```
> a:="notorbe";
```

$a :=$ "notorbe"

```
> b:=cat(substring(a,3..4), " ", substring(a,6..7),
> " ", substring(a,4..5), " ", substring(a,1..3),
> " ", substring(a,3..4), " ", substring(a,6..7)):
> print(b);
```

"to be or not to be"

Names for variables normally must begin with a letter or an underscore (_) followed by other characters. To create a name that starts with a number or contains spaces or special characters, one must surround it by single left quotes, (`).

```
>   `3 * 4 is not`:=7;
```

$$\mathit{3\ *\ 4\ is\ not} := 7$$

```
>   5+`3 * 4 is not`;
```

$$12$$

Forward single quotes (') delay the evaluation of a quantity for one execution cycle.

```
>   b,c:=2,3; a:='b+c'; a;
```

$$b,\, c := 2,\, 3$$

$$a := b + c$$

$$5$$

Use them when you want to display the name of a variable instead of its value.

```
>   'b'=b;
```

$$b = 2$$

but later still be able to evaluate it as its numeric value.

```
>   b;
```

$$2$$

Alternatively, one can evaluate the variable as a name:

```
>   evaln(b)=b;
```

$$b = 2$$

Here's another example where one might want to delay evaluation to improve readability:

```
>   p:=evalf(Pi,50):
>   Int(1/x,x=0..p);
```

$$\int_0^{3.1415926535897932384626433832795028841971693993751} \frac{1}{x}\, dx$$

If you want the integral to display the letter p but to compute with the value of π, then type:

```
>   Int(1/x,x=1..'p'); value(%);
```

$$\int_1^p \frac{1}{x}\, dx$$

$$1.144729886$$

Finally, since forward quotes delay execution, `'p'` is the name `p` instead of its value. Consequently, the statement

```
>   p:='p';
```

$$p := p$$

assigns to `p` the name `p` (and not its value), thereby unassigning `p`.

If any of these three types of quotes are unmatched when you press ⟨ENTER⟩, Maple will complain.

```
>  "This is a string.;
```

```
Warning, incomplete string;  use " to end the string
```

```
>  `This is a name.;
```

```
Warning, incomplete quoted name;  use ` to end the name
```

```
>  '3*p;
```

```
Error, `;` unexpected
```

Likewise, you may have problems if you use the wrong kind of quotes.

This command is correct: a label is assigned as a name for a string.

```
>  `This is a name.`:="This is a string.";
```

$$\textit{This is a name.} := \text{“This is a string.”}$$

However, a string cannot be used as a label.

```
>  "This is a name.":=3;
```

```
Error, invalid left hand side of assignment
```

```
>  "b":=b;
```

```
Error, invalid left hand side of assignment
```

11.3 Confusion between Functions and Expressions

Definitions: Maple has two different mathematical constructs, expressions and (arrow-defined) functions. The natural one to work with is expressions. However, there are times when functions are more convenient.

An expression is simply a formula for computing something, for example:

```
>  f:=x^2+1;
```

$$f := x^2 + 1$$

On the other hand, a function is a rule which assigns a given output to a given input, for example:

```
>  g:=x -> x^2+1;
```

$$g := x \to x^2 + 1$$

As pointed out in Section 10.3, a function is actually a shorthand for a certain procedure. Thus, `g` above is totally equivalent to

```
>  g0:=proc(x) x^2+1 end proc:
```

Evaluation: Variables in expressions are given algebraic or numerical values with the `subs` or `eval` command:

```
>  subs(x=2,f); eval(f,x=2);
```

$$5$$
$$5$$

Functions are evaluated by putting the value of the argument in parentheses:

```
> g(2);
```

$$5$$

Errors occur when the syntax appropriate for a function is used with an expression or vice versa:

```
> f(x);
```

$$\mathrm{x}(x)^2+1$$

```
> f(1);
```

$$\mathrm{x}(1)^2+1$$

```
> subs(x=1,g);
```

$$g$$

Conversion: It is usually easy to convert expressions into functions, and vice versa:

```
> f1 := unapply(f,x);
```

$$f1 := x \to x^2+1$$

```
> g1 := g(x);
```

$$g1 := x^2+1$$

However, you should not use an arrow to convert an expression to a function; it will not evaluate properly. Recall that:

```
> f;
```

$$x^2+1$$

If we define the function

```
> F:=x->f;
```

$$F := x \to f$$

we might expect that `F(2)` would be 5. Instead we get:

```
> F(2);
```

$$x^2+1$$

Alternatively, you *can* use "copy and paste" to produce the function

```
> F:=x->x^2+1;
```

$$F := x \to x^2+1$$

However, there are times when you would not want to use "copy and paste", for example, when the formula is a very large result of a very long computation. You should also read Section 11.4 on the dangers in Using Copy and Paste.

Conversely, it may even be impossible to convert some more complcated functions into expressions. For example, if the function involves an `if` statement, such as

```
> h:=x -> if x<1 then x^2+1 else 3*x fi:
```

then it will not convert properly

```
> h(x);
```

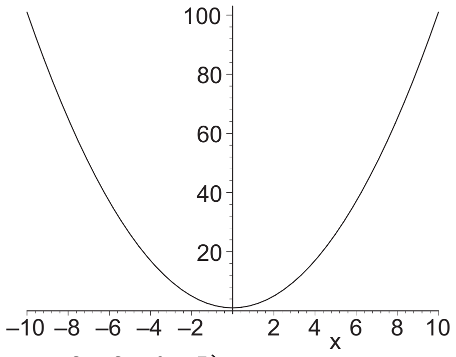
```
Error, (in h) cannot determine if this expression is true or false: x
< 1
```

because Maple cannot determine if the variable x, not yet defined, satisfies the condition $x < 1$.

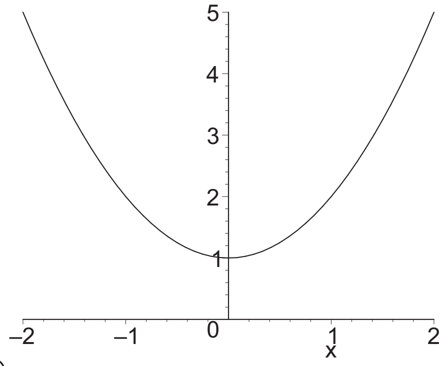
Plots: Maple treats functions and expressions differently in plots. With an expression in a plot command, the range *must* contain the plotting variable: `x=-10..10`. With an arrow-defined function or other procedure, the plotting variable x *must* be omitted: `-10..10`.

The following examples show correct ways of obtaining plots of the expression `f` and the function `g`, on both the default interval $-10 \le x \le 10$ and the interval $-2 \le x \le 2$. (We only show each plot once.)

```
> plot(f, x);
```



```
> plot(f, x=-2..2, 0..5);
```



```
> plot(g);
```

(Plot not shown.)

```
> plot(g, -2..2, 0..5);
```

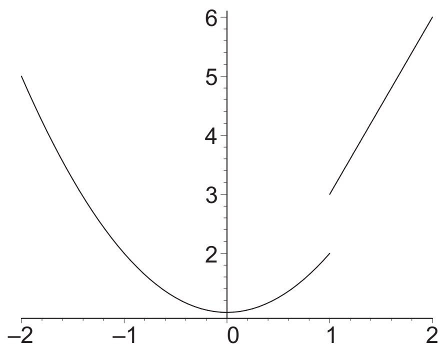
(Plot not shown.) For `g`, you can also use

```
> plot(g(x), x=-2..2, 0..5);
```

(Plot not shown.) since `g(x)` is an expression.

For `h`, which is a function and also has a discontinuity, you should use

```
> plot(h,-2..2, discont=true);
```



However, as pointed out above, the following shows that we can't plot `h` by converting it to the expression `h(x)`, since `h(x)` isn't an expression.

```
>  plot(h(x), x=-2..2, discont=true);

   Error, (in h) cannot determine if this expression is true or false: x
   < 1
```

On the other hand, if you delay the evaluation of `h(x)` by surrounding it in single quotes, then it will execute because `h(x)` is not evaluated until after x is chosen:

```
>  plot('h(x)', x=-2..2, discont=true);
```

(Plot not shown.)

Here are some more examples which are incorrect, since they mix the syntax for functions and expressions.

```
>  plot(g(x), -2..2);

   Warning, unable to evaluate the function to numeric values in the
   region; see the plotting command's help page to ensure the calling
   sequence is correct

   Plotting error, empty plot

>  plot(g, x=-2..2);

   Error, (in plot) expected a range but received x = -2 .. 2

>  plot(f(x), -2..2);

   Warning, unable to evaluate the function to numeric values in the
   region; see the plotting command's help page to ensure the calling
   sequence is correct

   Plotting error, empty plot
```

Finally, the following shouldn't work, but it does anyway.

```
>  plot(f(x), x=-2..2);
```

(Plot not shown.)

11.4 Unexpected Current Values of Variables

Using the *!!!* Icon: When one first re-opens a Maple worksheet after it has been saved, the input and output shows in the worksheet, but none of it is in Maple's memory. To fix this you need to press ⟨ENTER⟩ on each line to re-execute it and put the result in memory. As a shortcut, you can click on the *!!!* icon in the toolbar, which automatically re-executes the entire worksheet in order from top to bottom.

A major problem with re-executing the worksheet can occur if you did not enter the commands in a linear order. In particular, suppose at some point in your computations you revise a portion of the worksheet near the top, based on values of a variable you computed near the bottom. Then you save, close and reopen the worksheet and re-execute the worksheet using the *!!!* icon. When the revised commands near the top are re-executed, they will not work properly because Maple does not yet know the values of the variables computed at the bottom. (A specific example is given below in the subsection on Re-executing Statements to Avoid Retyping.)

TIP: If you need to reuse some code from earlier in the worksheet with variables redefined later in the worksheet, copy and paste the code from earlier in the worksheet to down below where the variables have been redefined. Then re-execute them.

The *!!!* icon is also useful when Maple appears to misbehave in the middle of a session. In particular, the symbol `%` does not indicate the output of the command immediately preceding a given command, but instead, the output of the last command executed, regardless of where that command might lie. If one has moved about the worksheet to make corrections, it may well be the case that the last output is not the command above the command containing the `%`. Putting a `restart;` at the beginning of the worksheet and use of *!!!* will almost magically cure the resulting problem

TIP: Proper Maple procedure is always, if possible, to put the command containing the `%` on the same line (or in the same execution group) as the statement whose output it refers to. Students often neglect this important requirement.

Using COPY and PASTE: COPY and PASTE are available from the EDIT menu or from the keyboard as ⟨CTRL-C⟩ and ⟨CTRL-V⟩.

There is nothing wrong with using copy and paste to copy a region of a worksheet to another part of the worksheet, edit it and execute the modified code. The problem arises when you copy and paste the result of a computation instead of giving it a name and using the name, or using `%` to refer to the last output. Here is an example:

Suppose problem 4 in your book says to find the square of each solution of the equation, $x^2 - 7x + 4 = 0$. So you enter the equation, solve the equation and square each result:

```
>  eq:=x^2-7*x+4=0;
```

$$eq := x^2 - 7\,x + 4 = 0$$

```
> fsolve(eq,x);
```

$$0.6277186767, 6.372281323$$

```
> a:=(.6277186767)^2; b:=(6.372281323)^2;
```

$$a := 0.3940307371$$
$$b := 40.60596926$$

Notice we used copy and paste to copy each solution to the next line. At that point, you suddenly realize you did the wrong problem. You were supposed to do problem 5 which says to square each solution of the equation, $x^2-8x+3=0$. So you change the equation and re-execute:

```
> eq:=x^2-8*x+3=0;
```

$$eq := x^2 - 8\,x + 3 = 0$$

```
> fsolve(eq,x);
```

$$0.3944487245, 7.605551275$$

```
> a:=(.6277186767)^2; b:=(6.372281323)^2;
```

$$a := 0.3940307371$$
$$b := 40.60596926$$

Unfortunately, you now have the wrong answer! Copying and pasting results is dangerous! The better way to solve the problem is to give the answer a name and use it:

```
> eq:=x^2-7*x+4=0;
```

$$eq := x^2 - 7\,x + 4 = 0$$

```
> sol:=fsolve(eq,x);
```

$$sol := 0.6277186767, 6.372281323$$

```
> a:=(sol[1])^2; b:=(sol[2])^2;
```

$$a := 0.3940307371$$
$$b := 40.60596926$$

That way, if you change the equation and re-execute, the other commands will automatically update.

```
> eq:=x^2-8*x+3=0;
```

$$eq := x^2 - 8\,x + 3 = 0$$

```
> sol:=fsolve(eq,x);
```

$$sol := 0.3944487245, 7.605551275$$

```
> a:=(sol[1])^2; b:=(sol[2])^2;
```

$$a := 0.1555897963$$
$$b := 57.84441020$$

Re-executing Statements to Avoid Retyping: Re-executing statements can lead to errors because Maple variables may not be what you expect. For example, suppose you calculate the area of a circle of radius 2 by entering

```
> r:=2;
```

$$r := 2$$

```
> A:=Pi*r^2;
```

$$A := 4\,\pi$$

```
> evalf(%);
```

$$12.56637062$$

Now suppose you want the area of a circle of radius 3. You enter

```
> r:=3;
```

$$r := 3$$

```
> evalf(A);
```

$$12.56637062$$

and get the wrong answer. The reason is that `A` hasn't been recalculated. If you now scroll up, click on the definition of `A` and press ⟨ENTER⟩, its output becomes

```
> A:=Pi*r^2;
```

$$A := 9\,\pi$$

Now when you re-execute

```
> evalf(A);
```

$$28.27433389$$

its output is correct.

However, there is still a problem. If you ever re-execute the worksheet, (Say you stopped in the middle one day and continued the next.) then the output would change to the wrong value. You *must* retype the definition of `A`.

TIP: To prevent re-execution errors, put several statements on one line. This ensures that they will all be re-executed when the line is re-entered.

```
> r:=2; A:=Pi*%^2; evalf(%);
```

$$r := 2$$
$$A := 4\,\pi$$
$$12.56637062$$

TIP: To save time, use COPY and PASTE to copy the previous line to a new line, change the `2` to a `3` and press ⟨ENTER⟩.

```
> r:=3; A:=Pi*r^2; evalf(%);
```

$$r := 3$$
$$A := 9\,\pi$$
$$28.27433389$$

Forgetting to Unassign Variables: Maple has a memory like an elephant.

As another example of Maple variables having unexpected values, supppose you have (unwisely) assigned a value to the letter `k`.

```
> k:=3;
```

$$k := 3$$

Later, you want to calculate a sum and try to execute.

```
> Sum(1/(k*(k+2)), k=1..10); value(%);
```

$$\sum_{3=1}^{10} \frac{1}{15}$$

```
Error, (in sum) summation variable previously assigned, second
argument evaluates to 3 = 1 .. 10
```

The problem is that the index of summation already has a value. Even worse you might just execute

```
>  sum(1/(k*(k+2)), k=1..10);
```

```
Error, (in sum) summation variable previously assigned, second
argument evaluates to 3 = 1 .. 10
```

and not see the error in the sum. This error is easily corrected by unassigning `k` and making it a free variable again.

```
>  k := 'k';
```

$$k := k$$

```
>  Sum(1/(k*(k+2)), k=1..10); value(%);
```

$$\sum_{k=1}^{10} \frac{1}{k\,(k+2)}$$

$$\frac{175}{264}$$

Of course, if you have executed the assignment statement three problems before the one that you are working on now, it might be rather hard to recall that fact.

TIP: The best procedure is to perform a `restart:` command at the beginning of each new problem. This approach has the effect of clearing *all* previous assignment operations, and it limits how far back one might need to look for such assignments. (The one downside is that packages must be reentered, but that is a small price to pay.)

The following error is similar, but not as easy to diagnose. Suppose `x` is given a value

```
>  x:=1;
```

$$x := 1$$

Then, at some later time, you decide to compute an integral:

```
>  int(x^2, x);
```

```
Error, (in int) integration range or variable must be specified in the
second argument, got 1
```

The error message is very cryptic. Maple does not say the integration variable is previously assigned. The first step to understanding the problem is to switch to the *inert* `Int` command:

```
>  Int(x^2, x); value(%);
```

$$\text{Int}(1,\, 1)$$

```
Error, (in int) integration range or variable must be specified in the
second argument, got 1
```

You can now see the problem is that x is replaced by a 1. (Moreover, you can check this by entering `x;` and observing the output.) Again, the remedy is to unassign `x`.

```
>  x:='x':  Int(x^2,x); value(%);
```

$$\int x^2\,dx$$

$$\frac{x^3}{3}$$

TIP: Avoid assigning anything to single letter variables, especially the variables `x, y, z, t, i, j, k, m,` or `n` so that they can be used for free variables in functions, plots, sums and integrals. Rather use variables such as `x0, x1, i0, i1,` etc. for values of the variables. (Alternatively, the `subs` command can frequently be used instead of assigning a value to the variable.)

11.5 Using On-Line Help

Maple has an excelent on-line help facility. Frustration with the on-line help is a common complaint of new users, even to the point that they stop trying to use it. This is unfortunate, because the on-line help is an extremely valuable resource in Maple, whether used as reference for looking up some forgotten syntax, or as a vehicle for exploring the myriad commands and mathematics Maple puts at your disposal.

Maple Help can be accessed either from the menubar, from the command line, by pressing ⟨F1⟩ or by pressing ⟨CTRL-F1⟩. The behavior is slightly different depending on whether you are using the Classic or Standard interface. The Standard interface is significantly better.

In the Classic interface, if you click on the HELP menu, you can access (i) the Maple Help Browser by clicking on INTRODUCTION, (ii) context help (i.e. help on the word currently containing the cursor), or (iii) the Topic Search or Full Text Search windows. You can also get to the context help directly by pressing ⟨F1⟩ or ⟨CTRL-F1⟩.

In the Standard interface, if you click on the HELP menu, you can access the Maple Help Browser by clicking on MAPLE HELP. This browser contains a box for a Topic Search or a Full Text Search. You access the context help by pressing ⟨CTRL-F1⟩. If you press ⟨F1⟩, you will get QUICK HELP on a very small number of common tasks.

In either interface, you can also access the help system from a prompt by typing 1, 2 or 3 question marks and the name of the command. (The number of question marks determines which part of the help page is emphasized.) In the Classic interface, this only works if you know the precise spelling of the command. In the Standard interface, this takes you to the Topic Search.

Once you are at the Topic Search box, if you type a word, Maple will offer a list of commands that deal with that word, including a definition from its dictionary. For example, if you type "implicit," then Maple will list commands like `implicitdiff` and `implicitplot, plots`, which you may click on for further information.

Once you are on a help page, the punch line of Maple Help, like the punch line of a joke, always occurs at the end, where there are always a collection of examples. (You can access the examples directly by typing 3 question marks and the name of the command.)

TIP: Always skip down to the examples. After you have studied the examples, you can copy and paste the commands from the example into your worksheet and then replace the sample data with your own data.

We will look at some typical situations.

Accessing Help from a Prompt: Suppose you want to plot two graphs on one coordinate system (you have seen it done before, so you know it's possible), but you don't remember the exact syntax. To see examples of the `plot` command, you enter

```
> ???plot
```

It shouldn't take you very long to find the following example.

```
> plot([sin(x), x-x^3/6], x=0..2, color=[red,blue],
> style=[point,line]);
```

If you need more explanation of the command, enter

```
> ?plot
```

for the full on-line help entry on `plot`, or else

```
> ??plot
```

for an abbreviated entry just on the syntax of applying the command.

Use Help on a Need-to-Know Basis: Sometimes the explanations contain so much jargon and unfamiliar terminology that you're overwhelmed. It may be best to just ignore what you don't understand and not worry about it; you'll either learn it later, or else find you don't really need to know it. For example, suppose you want to find out about the `do` command, and you enter

```
> ?do
```

You immediately encounter terms like "statement sequence" and "boolean expression", which may not be too meaningful to you, so you skip to the examples (which you could have gone to immediately with the `???do` statement), and find that these are quite sufficient for your purposes. In fact, if you're so inclined, you can use the examples to help you understand the explanation and its unfamiliar terms.

Similarly, Maple on-line help generally doesn't attempt to teach the mathematics that you may not understand, so you have to learn to separate the mathematics topics from the Maple topics. For example, the entry

```
> ?int
```

will tell you about Maple's `int` and `Int` commands for evaluating *definite* and *indefinite integrals*, but it expects you to already know what these mathematical terms mean. In fact, the Standard interface now has a dictionary in which you can look up the definition of integral. You access it from the Topic Search under "integral, Definition".)

Sometimes something unexpected comes up, but can safely be ignored, as in the following example. You want to find a floating point decimal solution to the equation $e^x + x - 2 = 0$, and so you enter the following

```
> solve(exp(x)+x-2=0,x);
```

$$-\mathrm{LambertW}(e^2) + 2$$

"What in the world is LambertW?" you say to yourself. To find out, you decide to enter

```
> ?LambertW
```

and see a screen full of information about something called Lambert's W function. Since you really only need a numerical approximation to the solution, you recall that you could have used `fsolve` instead of `solve`, and so you modify the above command and enter

```
> fsolve(exp(x)+x-2=0,x);
```

$$0.4428544010$$

Alternatively, you could simply append `evalf(%)` after the `solve`. The point is that with so much information available in the on-line help, you should learn to selectively ignore what you don't need, while hunting down what you do need.

Help on Packages: Some Maple commands reside in packages that must be loaded into a Maple session before they can be used. To see a listing of the available packages, enter

```
> ?index,package
```

Sometimes you may remember a command and its syntax, but have forgotten that it is part of a package. If you try to use the command before loading the package, not only won't it work, but the reason may not be readily apparent.

For example, suppose you want to plot a circle, and you remember using the `circle` command with arguments being the center and radius. So you enter:

```
> circle([3,2],2);
```

$$\mathrm{circle}([3, 2], 2)$$

Maple just repeats what you typed. This means the `circle` command is undefined, probably because it belongs to some package. So you look at its help.

```
> ?circle
```

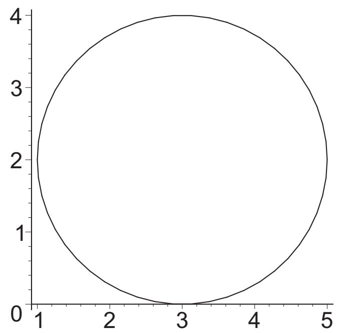
You see many choices of help pages, but select `circle, plottools` to see its help and also conclude that `circle` is in the `plottools` package. So you modify your input to read:

```
> with(plottools):
> circle([3,2],2);
```

$$\mathrm{CURVES}([[5., 2.], [4.984229403, 2.250666467], \ldots$$

The result is just "CURVES" and a a lot of numbers. (NOTE: We deleted most of the output to save space.) So you look back at the examples on the help page and see the `circle` command requires the `display` command. To complete the problem, you modify your input to read:

```
> with(plots):
> display(circle([3,2],2), scaling=constrained);
```



TIP: By the way, you only need to install a package once, not each time you use one of its commands. So if you use a package regularly, get in the habit of executing the `with` command at the top of the worksheet.

Spelling Problems: A similar problem arrises if you misspell a command. Maple will just repeat what you typed.

TIP: Remember that Maple is case sensitive. Some commands are all lower case, some are all upper case, some have the first letter capitalized, while others are in CamelCase (where words are run together, with capital letters to mark the start of each word). So it is easy to incorrectly capitalize a command. Spelling errors are even worse.

For example, suppose you remember that there is a command that will pop up a Tutor (a Maplet window) and allow you to play with approximate integration techniques. So you try

```
> ApproximateIntegralTutor(x^2,x=-2..2);
```

$$\text{ApproximateIntegralTutor}(x^2,\, x = -2..2)$$

Maple just repeats the command. So you try help:

```
> ?ApproximateIntegralTutor
```

And Maple can't find any help. So you try doing a Topic Search on ApproximateIntegralTutor or Approximate Integral Tutor or Integral Approximate to no avail. Finally you try doing a Full Text Search on Approximate Integral and find it. The correct name is ApproximateIntTutor and it is in the Student[Calculus1] package. So you enter

```
> with(Student[Calculus1]):
> ApproximateIntTutor(x^2,x=-2..2);
```

and get what you want.

TIP: When you are unsure of the spelling, try a Full Text Search which is available on the HELP menu in the Classic interface and at the top left of the Help Browser in the Standard interface with a radio button labeled TEXT.

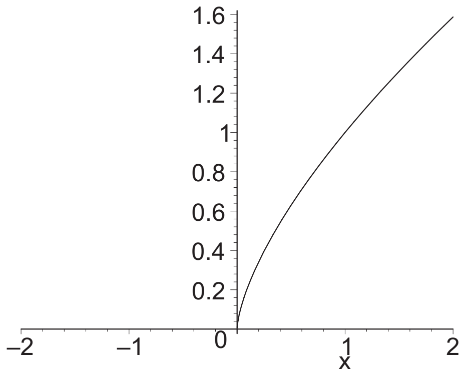
TIP: Whenever Maple *parrots back* a command that you have typed, rather than executing it, you should check for missing packages and spelling errors.

11.6 Failure to Plot

Roots of Odd Degree: Students are often confused when they try to plot expressions containing roots of odd degree, like the cube root. For example, an excellent example of an expression that fails to have a derivative at the origin is the "Seagull" given by $y = x^{2/3}$. However, when one attempts to plot this expression a difficulty arises, and only half the plot is shown.

```
>  x^(2/3); plot(%,x=-2..2);
```

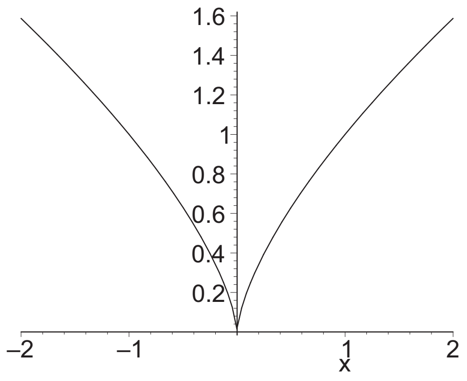
$$x^{(2/3)}$$



The reason that the left hand side fails to appear is that Maple uses complex numbers. All real numbers have three cube roots, one real and two complex. When Maple computes the cube root of a positive number, it picks the real cube root. However, when it computes the cube root of a negative real number, it picks the first complex number counterclockwise from the positive real axis. Since Maple cannot plot the resulting complex number, no plot appears.

There are three easy solutions to this problem. One is to rewrite $x^{2/3}$ as $\left(x^2\right)^{1/3}$. A second is to replace the variable x by its absolute value `abs(x)`. And the third is to use the `surd` command. (See `?surd`.) Any of the three yields the desired full graph:

```
>  plot((x^2)^(1/3),x=-2..2);
```

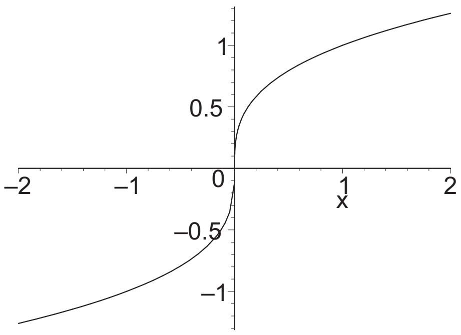


```
>  plot(abs(x)^(2/3),x=-2..2);
```

```
>  plot(surd(x^2,3),x=-2..2);
```

Unfortunately, for the very similar function $x^{1/3}$ you either need to use `surd` or plot the function in two parts:

```
>  plot(surd(x,3),x=-2..2);
```



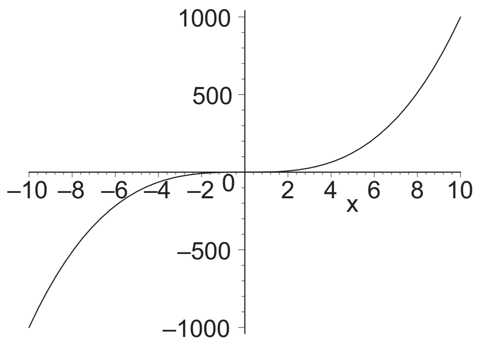
```
>  plot([x^(1/3), -(-x)^(1/3)],x=-2..2);
```

Notice that the function $-(-x)^{1/3}$ is equivalent to $x^{1/3}$ but when x is negative, you are taking the cube root of the positive number $-x$.

Combining Plots with `display`: Yet another situation that students often encounter in which plots fail to appear is when one attempts to display two or more plots on the same pair of axes. One first creates individual plots, which are labeled `a`, `b`, etc, and ends the command with a colon, rather than a semicolon, to suppressing output. One can then enter `with(plots);` and the `display([a,b])` comand. However, it is not uncommon for one or more of the plots to fail to appear. Frequently the problem is that students often *create* the labeled plots, but never *check* that the plot actually *exists*. And in many cases it doesn't.

TIP: It is good Maple practice to show the plot at the same time that one creates and labels it. (This is frequently long before the `display` command.) This should be done by using the following syntax:

```
>  a:=plot(x^3,x):  a;
```



11.7 Confusing Exact and Approximate Calculations

Many commands work very differently depending on whether Maple is using exact arithmetic or floating point decimals. A decimal point in a single number is enough to cause Maple to use approximate arithmetic (decimals, rather than rational numbers). Consider the following examples.

Polynomials of degree four or less can be solved exactly using `solve`: (However, you may be less than happy with the result.).

```
>   solve(x^4+x-2=0, x);
```

$$1, -\frac{(188+12\sqrt{249})^{(1/3)}}{6} + \frac{4}{3\,(188+12\sqrt{249})^{(1/3)}} - \frac{1}{3},$$

$$\frac{(188+12\sqrt{249})^{(1/3)}}{12} - \frac{2}{3\,(188+12\sqrt{249})^{(1/3)}} - \frac{1}{3}$$

$$+\frac{1}{2}\,I\,\sqrt{3}\left(-\frac{(188+12\sqrt{249})^{(1/3)}}{6} - \frac{4}{3\,(188+12\sqrt{249})^{(1/3)}}\right),$$

$$\frac{(188+12\sqrt{249})^{(1/3)}}{12} - \frac{2}{3\,(188+12\sqrt{249})^{(1/3)}} - \frac{1}{3}$$

$$-\frac{1}{2}\,I\,\sqrt{3}\left(-\frac{(188+12\sqrt{249})^{(1/3)}}{6} - \frac{4}{3\,(188+12\sqrt{249})^{(1/3)}}\right)$$

or approximately by including a decimal point:

```
>   solve(x^4.+x-2=0, x);
```

$$1., 0.1766049821 + 1.202820819\,I, -1.353209964, 0.1766049821 - 1.202820819\,I$$

Notice this is different from `fsolve` which returns only the real roots.

```
>   fsolve(x^4+x-2=0, x);
```

$$-1.353209964, 1.$$

On the other hand, polynomials of degree five or higher can't in every case be solved exactly, and here's how Maple responds when asked for exact solutions to such equations. First an easy one:

```
>   solve(x^5-x=0, x);
```

$$0, 1, -1, I, -I$$

and now a hard one:

```
>   solve(x^5+x-3=0, x);
```

$$\mathrm{RootOf}(\_Z^5 + \_Z - 3, \mathit{index} = 1), \mathrm{RootOf}(\_Z^5 + \_Z - 3, \mathit{index} = 2),$$
$$\mathrm{RootOf}(\_Z^5 + \_Z - 3, \mathit{index} = 3), \mathrm{RootOf}(\_Z^5 + \_Z - 3, \mathit{index} = 4),$$
$$\mathrm{RootOf}(\_Z^5 + \_Z - 3, \mathit{index} = 5)$$

Maple knows there are 5 roots (counting multiplicities) but it can't find them exactly using rational numbers. So it names them using the `RootOf` notation. However, Maple has a variety of ways for finding approximate roots, some of which we give below. In particular, the `evalf` command can be used to give a decimal approximation to a `RootOf` symbol.

```
>  evalf(%);
```

$1.132997566,\ 0.4753807567 + 1.129701725\,I,\ -1.041879540 + 0.8228703381\,I,$
$-1.041879540 - 0.8228703381\,I,\ 0.4753807567 - 1.129701725\,I$

You can get directly to these solutions by including a decimal point (on the 3.)

```
>  solve(x^5+x-3.=0, x);
```

$1.132997566,\ 0.4753807567 + 1.129701725\,I,\ -1.041879540 + 0.8228703381\,I,$
$-1.041879540 - 0.8228703381\,I,\ 0.4753807567 - 1.129701725\,I$

or you can get the real one(s) using `fsolve`:

```
>  fsolve(x^5+x-3=0, x);
```

$$1.132997566$$

If all the roots of a polynomial — complex as well as real — are desired, the following will always work.

```
>  fsolve(x^5+x-3=0, x, complex);
```

$-1.041879540 - 0.8228703381\,I,\ -1.041879540 + 0.8228703381\,I,$
$0.4753807567 - 1.129701725\,I,\ 0.4753807567 + 1.129701725\,I,\ 1.132997566$

Finally, we recall the example from Section 11.5, which also exhibits the difference between exact and approximate calculations.

```
>  solve(exp(x)+x-2=0, x); evalf(%);
```

$$-\mathrm{LambertW}(e^2) + 2$$
$$0.442854401$$

11.8 Debugging Procedures

Maple procedures were discussed in Section 10.3. However, as with all programming, people make mistakes and need to debug their programs. We'll look at three types of errors: typing, syntax and math errors.

Example 1: Write a program to find all critical points of a polynomial.

Solution: You decide to write the polynomial as an expression and enter:

```
>  critpts:=proc(p)
>  local eq,x,xsol;
>  eq:=diff(p,x)=0;
>  xsol:=fslove(eq,x);
>  for x in [xsol] do
>  print([x,subs(x,p)]);
>  end do;
>  end proc:
```

(We did not display the output because it is exactly a repeat of the input.) This procedure uses an alternate version of the `do` loop from that discussed in Section 10.2. Here, the loop is repeated with `x` taking on the value of each entry in the list `[xsol]`. (See `?do`.) So it should print out the x- and y-coordinates of each critical point.

Now you try it on a polynomial you know has 4 zeros, hence 3 critical points.

```
>  (x-2)*(x-3)*(x-4)*(x-5); q:=expand(%);
```

$$(x-2)\,(x-3)\,(x-4)\,(x-5)$$

$$q := x^4 - 14\,x^3 + 71\,x^2 - 154\,x + 120$$

```
>  critpts(q);
```

```
Error, (in critpts) invalid input: subs received fslove(0 = 0,x),
which is not valid for its 1st argument
```

Something went wrong. However, you look at the output and immediately notice that you misspelled `fsolve`. So you correct it. Now you have:

```
>  critpts:=proc(p)
>  local eq,x,xsol;
>  eq:=diff(p,x)=0;
>  xsol:=fsolve(eq,x);
>  for x in [xsol] do
>  print([x,subs(x,p)]);
>  end do;
>  end proc:
```

```
>  critpts(q);
```

$$[x = x,\, x^4 - 14\,x^3 + 71\,x^2 - 154\,x + 120]$$

There is still something wrong with the `fsolve` command. This time there is confusion between the indeterminate x in the polynomial p and the variable name x in the procedure. By using a different variable name s instead of x, you now have

```
>  critpts:=proc(p)
>  local eq,s,xsol;
>  eq:=diff(p,x)=0;
>  xsol:=fsolve(eq,x);
>  for s in [xsol] do
>  print([s,subs(x=s,p)]);
>  end do;
>  end proc:
```

```
>  critpts(q);
```

$$[2.381966011,\ -1.0000000]$$

$$[3.500000000,\ 0.5625000]$$

$$[4.618033989,\ -1.0000003]$$

It worked!

Example 2: Write a procedure to sort three numbers in ascending order.

Solution: The basic ideas were discussed at the end of Section 10.1. Here's the procedure and an execution:

```
> ascend:=proc(a,b,c)
> local m,n;
> if a>b
> then m,n:=a,b;
> else m,n:=b,a;
> end if;
> if c<m
> then c,m,n;
> elif c<n
> then m,c,n;
> else m,n,c;
> end if;
> end proc:
```

```
> ascend(4,7,-2);
```

$$-2, 7, 4$$

Something went wrong. This is probably a math error, and those are much harder to find.

Maple has a tool to help with debugging. At the beginning of the procedure, add the statement `option trace`.

```
> ascend:=proc(a,b,c)
> option trace;
> local m,n;
> if a>b
> then m,n:=a,b;
> else m,n:=b,a;
> end if;
> if c<m
> then c,m,n;
> elif c<n
> then m,c,n;
> else m,n,c;
> end if;
> end proc:
```

This will cause Maple to print out all the intermediate results as the procedure is executed:

```
> ascend(4,7,-2);
```

```
{--> enter ascend, args = 4, 7, -2
```

$$m, n := 7, 4$$

$$-2, 7, 4$$

```
<-- exit ascend (now at top level) = -2, 7, 4}
```

$$-2, 7, 4$$

You should immediately notice that `m` and `n` are backwards. So you look at the code and discover that `a>b` should be `a<b`. So you fix it and re-execute: (Don't forget to remove the trace once the procedure is working.)

```
>  ascend:=proc(a,b,c)
>  local m,n;
>  if a<b
>  then m,n:=a,b;
>  else m,n:=b,a;
>  end if;
>  if c<m
>  then c,m,n;
>  elif c<n
>  then m,c,n;
>  else m,n,c;
>  end if;
>  end proc:
```

```
>  ascend(4,7,-2);
```

$$-2, 4, 7$$

Another trick for debugging your programs is to add extra print statements along the way to print out intermediate results and let you know how far the program gets before the error occurs.

11.9 Forgetting to Save a Maple Session and Losing It

It sometimes happens that a Maple session will lock up and you have to kill it, or some other program you are running will die and you need to reboot your computer. If you are unable to save the session, you may lose a lot of work or be forced to start all over. For this reason, it's good practice to regularly save the session by clicking on FILE and SAVE. (There is also a SAVE icon that looks like a floppy disk on the toolbar. This button often works when clicking on the menu fails.)

Better yet, you can have Maple automatically save your file. See Appendix A for information on Maple's AutoSave feature.

11.10 Trying to Get Maple to Do Too Much

How to Interrupt a Computation: Whether by accident, miscalculation or poor judgement, a Maple computation sometimes takes longer than you expect. Suppose you want to compute $\sum_{n=0}^{200} n!$, but you mistakenly enter

```
>  Sum(n!, n=0..200000); value(%);
```

Maple goes on forever. Try not to worry. Look at the tool bar and click on the red STOP sign (with a hand in the Standard interface). Maple should stop and give you the following warning.

```
>  Sum(n!, n=0..200000); value(%);
```

$$\sum_{n=0}^{200000} n!$$

```
Warning, computation interrupted
```

Now you can correct your error and re-execute the command. Unfortunately, the stop sign doesn't always work. Let's look at another example.

Students often get carried away and ask Maple to do more than is reasonable. For example, you may have noticed that earlier in this chapter, we computed the first 50 digits of π by typing

```
>  evalf(Pi,50);
```

$$3.1415926535897932384626433832795028841971693993751$$

So you might ask if Maple can compute 100 or 500 or 5000 digits by just changing the `50`. It can! Try it. But then you try 500,000 digits and Maple goes on forever!

```
>  evalf(Pi,500000);
```

You try the STOP sign; and several things may happen:

- The computation stops. You get the warning message and you go merrily on your way.
- The computation does not stop. Further, the FILE muenu and SAVE icon are not available. Your only option is to close Maple by clicking on the X in the title bar or killing the task or process. At this point you may or may not be asked if you want to save your work. If this choice does not appear, then you have lost all the work you have not saved! That's why you need to *save your work often!*
- The computation does not stop. But worse, you get a curt pop-up window which says "mserver.exe has encountered a problem and needs to close." At this point you may or may not be asked if you want to save your work. If not, then you have lost all the work you have not saved! Again *save your work often!*

TIP: If you think a computation may take a long time, *save your work before executing the statement!*

TIP: You can set Maple to automatically save your files. See Appendix A.

Maple Cannot and Does Not Need to Do Everything: We finally come to the section on what Maple cannot and should not do.

It frequently happens that, when students are given an assignment to use Maple on a problem, they proceed under the assumption that they must use Maple for every step in the problem. This can lead to the absurd situation of having to painstakingly cajole Maple into performing steps that are completely obvious, and should just be done by hand.

In addition, Maple cannot do the basic setup of the problem. Nor can it interpret the answer and judge if it is reasonable. Those are jobs for a human! Consider the following example:

Example: A cylindrical aluminum can is to hold 500 cm^3 of soda. What are the dimensions (height and radius) of such a can which has the least surface area?

Solution: Maple does not know that the volume of a cylinder is $V = \pi r^2 h$. Maple does not know that the surface area of a can is $A = 2\pi rh + 2\pi r^2$, where the first term is from the sides and the second term is from the two ends. Maple certainly cannot know that you need to include the two ends of the can in the surface area. In addition you have to tell Maple how to set up the problem and what steps to follow to solve it. Maple cannot read English nor interpret it as math. (Maybe sometime in the future, but not now.) That is your job.

So you proceed to enter the volume and set it equal to 500 and solve for h:

```
> Veq:=Pi*r^2*h=500;
```

$$Veq := \pi\, r^2\, h = 500$$

```
> h1:=solve(Veq,h);
```

$$h1 := \frac{500}{\pi\, r^2}$$

These two steps are so obvious that you might just as well have done them in your head and entered:

```
> h1:=500/(Pi*r^2);
```

$$h1 := \frac{500}{\pi\, r^2}$$

Now you enter the area and substitute `h1` for `h`:

```
> A:=2*Pi*r*h + 2*Pi*r^2;
```

$$A := 2\,\pi\, r\, h + 2\,\pi\, r^2$$

```
> A1:=subs(h=h1,A);
```

$$A1 := \frac{1000}{r} + 2\,\pi\, r^2$$

Next you set the derivative equal to zero and solve for `r`.

```
> DA:=diff(A1,r);
```

$$DA := -\frac{1000}{r^2} + 4\,\pi\, r$$

```
> r_sol:=fsolve(DA=0,r);
```

$$r\_sol := 4.301270069$$

Last you substitute `r_sol` back into `h1`.

```
>  subs(r=r_sol,h1); hsol:=evalf(%);
```

$$\frac{27.02567690}{\pi}$$

$$hsol := 8.602540136$$

Finally, you interpret your solution: The optimal can has height 8.6 cm and radius 4.3 cm.

The point is Maple cannot set up the problem and does not know the steps to solve the problem, but it can do the algebra. In addition Maple does not need to do every step, if you can do it in your head. Finally, Maple cannot interpret the answer.