1.3.7 Changing Maple Commands

Sometimes you will need to change a command that you have already entered. To do so, move the Maple `|` cursor to the place where you want to make a change by clicking the left mouse button there (or by using the arrow keys). Then

the **Delete** key deletes the character to the *right* of the cursor

the **Backspace** key deletes the character to the *left* of the cursor

and you can insert new characters at the position of the cursor.

When you have changed a command and want to execute the changed command, make sure that the `|` cursor is (anywhere) on the line of that command and then press `<Enter>`. The result of the changed command should appear on the screen, replacing the previous result.

Note: If the command you changed uses `%` to refer to results of *previous* commands then you will have to go back and re-execute those commands (by moving the cursor to each of those command lines and pressing `<Enter>`) before you execute the command you changed. This is because `%` refers to the result of *the command most recently executed by Maple* which is NOT necessarily the one on the line above the cursor's position.

Also, if the previous results of the command you changed were used in any *subsequent* commands then these later commands will have to be executed again. To be safe, it is best to press `<Enter>` on *every* command line which comes after any command line that you have changed.

If you want to insert a whole new command among previous commands, move the cursor to the execution group immediately *above* where you want the new command to go. Then either click on the box `[>` in the tool bar or press `<Ctrl>-J` and a new line with a `[>` prompt will appear. On this line you can type your new command or insert a comment using one of the methods of section 1.3.6.
If you want to delete the whole region where the cursor is, press `<Ctrl>-<Delete>`.

Exercise If you have not already done so, start Maple. Enter the command

```
f := sin x;
```

You will get a message saying `missing operator or ';'` .
Correct the command to read

```
f := sin(x);
```

and then execute the corrected command. The result of this command is just an echo of the command. Now enter a second command

```
diff(f,x);
```

Then go back and insert a new prompt just after the result of the command `f := sin(x)`. At this prompt, insert the *comment* “ Differentiate f: ”.

1.4 Saving a Maple worksheet

A Maple worksheet can be saved and re-opened at a later time in much the same way as other kinds of documents such as word processer documents or spreadsheets. Saving a Maple worksheet preserves only what you see in the worksheet. When this worksheet is re-opened, Maple will not remember and variables you may have defined, packages loaded or the results of recently executed Maple commands. This will be become clearer once you have used Maple.

A Maple worksheet can also be exported into other formats such as html, i. e. as a webpage, or as a text file containing only the Maple commands.

It is strongly recommended that you save a Maple worksheet regularly while you are using it.

There are two ways to *save* your worksheet. You can either click on the Save Icon (picture of a floppy disk) or select either the `Save` or the `Save As ...` option from the `File` menu. If you select `Save` or click the Save Icon and the active worksheet has a name, it will be saved (*as a worksheet*) in a file with that name; if it has no name, you get the Save As dialogue box. So you can save an updated version of the same session in a file *with the same name* simply by clicking on the **Save icon**: If you wish to save it with a new name, you will need to select the `Save As ...` from the File menu.

Selecting `Save As ...` will always bring up the Save As Dialogue Box. At the bottom of the Save As Dialogue Box you will see a box which looks something similar to:

Maple Worksheet (.mw) ∇

If you are only using versions after Maple 9 you can ignore the options here; if you want to be able to open your worksheet using Maple 8 or earlier, your should save it as a "Maple Classic Worksheet (.mws)".

Move the mouse pointer into the blank box beside the word **File Name**, click the left mouse button in the box and type in the name of the file where you want to save the worksheet. Then click on the word `OK`, or press `<Enter>`.

Notes:

1. If the filename has no (or the wrong) extension then the extension `.mw` will be placed on the end.
2. If there is already a file with that name, you will be asked to confirm the name and if you do the old file will be overwritten.
3. These processes will *always* save as a Maple Worksheet, even if you try to save to a file with a different extension.

1.4.1 Exporting a Maple Input File

A Maple Input file contains only the Maple input regions (and text regions if any). This can be opened or **read** in — see section 2.23.

To create one of these, select `Export As...` from the `File` menu. The Export As Dialogue Box then appears. It is very similar to the Save As Dialogue Box, except that at the bottom you will see a box which looks something similar to:

HTML (,html, .htm) ∇

If you now click on this box, a menu will appear giving you seven choices for the **type** of file in which you can save your worksheet. The only one we mention here is the third: **Maple Input**). For some of the others see section 3.6. Select the option that says `Maple`

`Input (.mpl)` and then enter the name of the file in the **File Name** box. If the file does not have the `.mpl` extension that designates a Maple Input file then Maple will add one. If there is already a file with that name, you will be asked to confirm the name and if you do the old file will be overwritten.

1.5 Maple On-line Help

Maple has built in help that can be accessed from the `Help` menu. A menu appears looking like this:

Maple Help	Ctrl-F1
Take a Tour of Maple	
Quick Reference	Ctrl-F2
⋮	
Manuals, Dictionary, and more ▷	
On the Web	
About Maple...	

If you select Maple Help from this menu, a new window will appear. In the *right hand* section of this window is some text entitled **Maple Resources**. This window also has several *hyperlinks* indicated by underlined text.

The *left hand* section of this window is the Help Browser, which will show the Maple **Help Navigator** expanded as far as the help page visible in the right hand panel.

1.5.1 The Help Browser

To learn how to use the help browser, click on the Help and move your mouse down to the line "Manuals, Dictionary, and more". A second menu will appear; move over to it and click on the option "Using the Help System". The help browser will open and there are several links to pages telling you how to use Maple's help system, see figure 1.4.

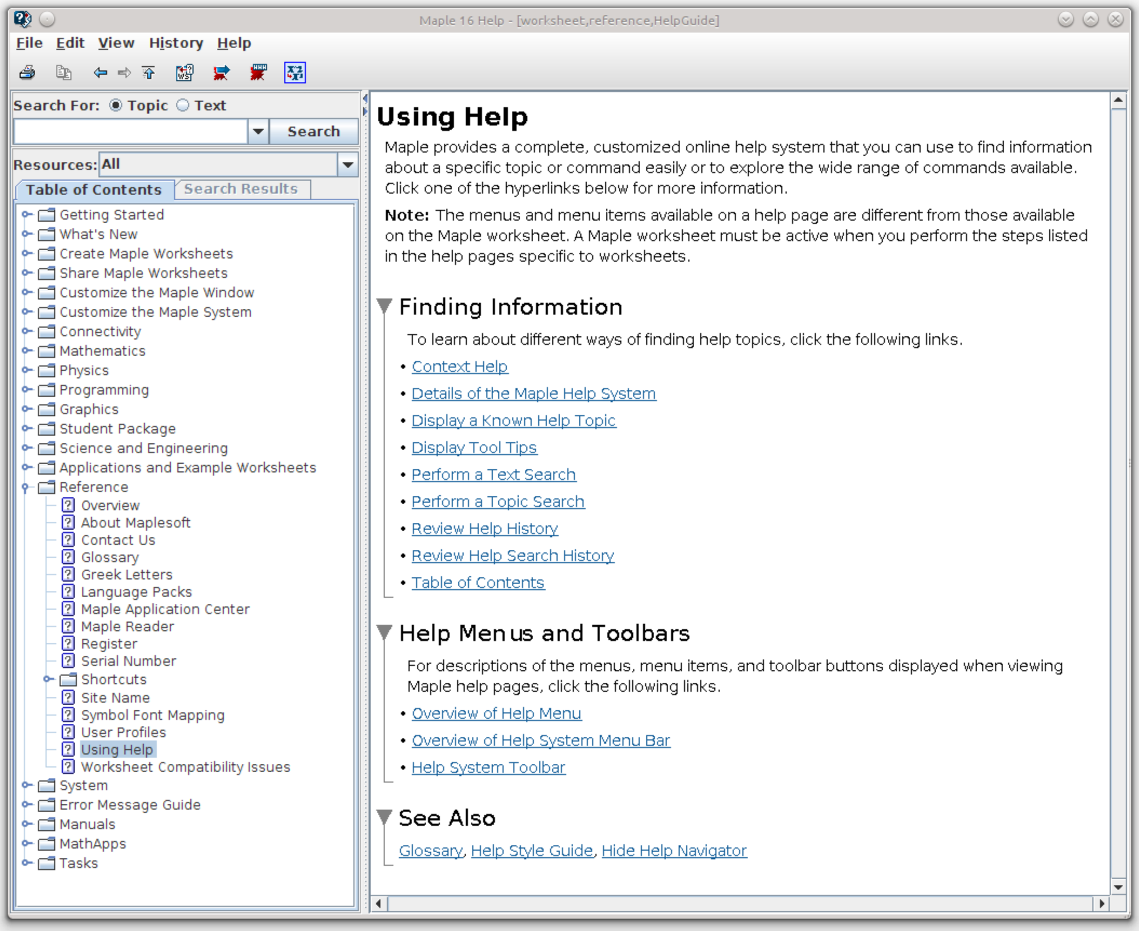
Note: you can go back to a previous page in the Help Browser by clicking on the large left pointing arrow in the top menu bar (it is "greyed out" in figure 1.4).

At the top of the left panel you will see a search box. You can use this to search for help on a general topic (such as "integration") or on a specific function. Below the search box is a small drop down menu that allows you to be more specific in your search — you can restrict to actual help pages on functions or have the search check through all of Maple's help pages, examples, tutorials, dictionary etc.

The results appear in the main part of the left panel, under two tabs. The "Search Results" tab will be a list of the possible matches to your search, and the "Table of Contents" is the Help Navigator. Clicking on one of the listed pages (under either tab) will open that help page (or definition, or worksheet). The one at the top of the Search Results list is automatically opened.

The Help Navigator shows how Maple's help pages are linked together in a large tree, and you can use this to find pages on similar functions to the one you serached for, which can be useful if the search has not quite given you what you want.

You can also choose to search on "Topic" or "Text" with the buttons above the search box. The difference between them is that "Text" looks though the pages for the appearance of the search text, while "Topic" looks for pages about the search text.



Using Help

Maple provides a complete, customized online help system that you can use to find information about a specific topic or command easily or to explore the wide range of commands available. Click one of the hyperlinks below for more information.

Note: The menus and menu items available on a help page are different from those available on the Maple worksheet. A Maple worksheet must be active when you perform the steps listed in the help pages specific to worksheets.

Finding Information

To learn about different ways of finding help topics, click the following links.

- Context Help
- Details of the Maple Help System
- Display a Known Help Topic
- Display Tool Tips
- Perform a Text Search
- Perform a Topic Search
- Review Help History
- Review Help Search History
- Table of Contents

Help Menus and Toolbars

For descriptions of the menus, menu items, and toolbar buttons displayed when viewing Maple help pages, click the following links.

- Overview of Help Menu
- Overview of Help System Menu Bar
- Help System Toolbar

See Also

Glossary, Help Style Guide, Hide Help Navigator

Figure 1.4: Maple Help Browser

For example, with "Text" selected, a search on "differentiation" gives you a lot of pages; using "Topic" just two help pages and the dictionary definition page.

If you had selected a help page using the **?** (see below) then the "Search Results" tab is open, and lists all the results of a search on your input text.

You can change the relative width of the two panels in the help page by dragging the left boundary of the right panel with the mouse.

1.5.2 Using the Results

The Help Navigator and Search Result panels use different icons to tell you about the pages. An icon shaped like a folder tells you that there are subpages (or subfolders) collected under that heading: clicking on a line with a folder icon opens up the subfolders and pages.

A question mark designates a help page: clicking on that opens an actual help page in the right hand side. A letter "D" in a yellow square links to a dictionary definition page. Both of these open in the right panel of the Help Browser.

A "WS" in a light blue box is a Maple worksheet: this opens in a new window.

The help pages start with a formal statement of the 'syntax' of the command (i.e. details of how to enter it). This may be hard to understand, but at the end of the entry will be some examples of usage. Use the scroll bar to move through the entry until you get to the examples.

Exercise Use a topic search to find the help entry for the integration command `int` and read through it trying to understand it. Use the mouse to highlight one of the example command lines from the end of this entry and press < **Ctrl** >**-C** to copy the command. Then activate a Maple window and paste the command into it using < **Ctrl** >**-V** . Press **`<Enter>`** and see that the result of the command agrees with the result shown in the help window.

To close this Help window, open the `File` menu for the Help window and then select the **Close Help** option.
An alternative to using the Topic tab in Maple's Help window when you know the *exact* word is to type a **?** at a Maple prompt in a worksheet followed by the command name, and then press **`<Enter>`**. For example,

```
?int
```

will open a Help window (if necessary) with the help on **`int`** in the right hand panel.

1.6 Maple and Moodle

In the First Year Moodle modules there are several lessons on Maple. These lessons cover a large proportion of the material as chapter 2 of these notes, but as well as missing out some detail, include a little extra material and examples.

Each First Year Course has access to the lessons appropriate for that course. To see these lessons, follow the “Maple note, lessons, assessments” link in the “Computing Component” section on the Moodle homepage of the Moodle module for your course.

Chapter 2
MAPLE COMMANDS AND LANGUAGE.

This chapter, which contains details of the Maple commands and language, is quite long and looks formidable. However, most of the Maple commands are fairly obvious once you get the hang of things. All you have to worry about is the exact syntax (way of writing them), which you can get from Maple's on-line help (see section 1.5).

The best way to use this chapter is first to glance through it to get an idea of what Maple can do (actually it can do far more than what we have described here), bearing in mind that many of the things in this chapter refer to mathematical ideas and processes which may not yet have been covered in lectures (you may skip those bits until your lecturer comes to them). You should particularly look at section 2.22 on common mistakes in Maple: knowledge of these would probably give more than 50% of students an average of 2 extra marks in the tests.

Later, when you are solving a specific problem, read through the relevant sections of this chapter (and possibly look at some of the Moodle Learning Modules, see section 1.6) before preparing a list of Maple commands to solve that problem. Then, when you are entering these commands, use Appendix A (which contains a list of most of the useful Maple commands) and Maple's on-line help for their exact syntax.

2.1 Arithmetic.

The usual arithmetical operations are available in Maple and you should use the following notation to enter them in commands.

addition	`+`
subtraction	`-`
multiplication	`*`
division	`/`
exponentiation	`^`

So `a^b` means a to the power b (i.e. a^b).

These follow the usual order of evaluation, i.e. anything in brackets, then powers, then multiplication or division, then addition or subtraction.

If you want to use a different order then you will have to insert brackets '`(`' and '`)`' in the appropriate places. For example `-1^(1/2)` means `-(1^(1/2))` (i.e. -1), whereas `(-1)^(1/2)` means $\sqrt{-1}$ (i.e. the imaginary number i, which is denoted `I` in Maple) and `-1^1/2` gives $-\frac{1}{2}$.

Note that you cannot write `a^b^c` in Maple because it is imprecise (and grammatically incorrect). Use either `(a^b)^c` or `a^(b^c)` as required. Also you cannot use two operators next to one another as in `a*-b` — you should use `a*(-b)` instead.

There is another arithmetical operator, the `!` symbol, which comes after a number and denotes the factorial (so $5! = 1.2.3.4.5 = 120$). Most calculators cannot find exact factorials past about 12! and cannot even approximate them past about 70!. But on

Maple you can easily find 1000! because Maple can handle integers of almost any size. Try finding 1000!, but do not get carried away with calculating factorials because they get very large very quickly and can easily lead you to exceed your time and memory limits.

Unlike most calculators and most computer programming languages, Maple does all arithmetic EXACTLY, i.e. as rational numbers (fractions with numerator and denominator having as many digits as is necessary) or as surds or as roots of equations. The only exception is when you deliberately enter numbers as decimals.

If you want to evaluate a fraction as a decimal number, use the command `evalf` ('evaluate as floating point'). This will normally display the answer to 10 significant digits (although it uses more than 10 digits internally when doing its calculations). If you want to use a different number of digits for all your displays of decimal numbers, use the command `Digits` to set the required number. For example, enter

```
Digits := 50;
```

to tell Maple that you want all floating point results displayed to 50 significant digits. If you only want to display one number to a different number of digits (without changing the number of digits for all displays), you can include the number of digits in the `evalf` command itself. For example,

```
evalf(1/17,50);
```

will evaluate $1/17$ to 50 significant digits.

There are several ways to enter a decimal or 'floating point' number. For example, 67.2319 can be entered as `67.2319` or `0.672319*10^2` or `672319*10^(-4)` etc, or in the form `Float(672319,-4)` which stands for 672319×10^{-4}. Note that this always has the form

`Float(`*integer*`,`*integer*`);`

There are limits to the size of the second integer (which specifies the exponent) and the number of digits is governed by the value of the variable `Digits`.

Anything which is entered as a floating point number will stay as a floating point number and will not be converted to a fraction unless you specifically ask Maple to convert it into a fraction. To do that, use the `convert` command as in

```
convert(%,fraction);
```

Arithmetic done on floating point numbers will always give a floating point answer.

2.2 Variables: Assignment and Unassignment.

2.2.1 Assigning

You can assign any expression to a variable for further use, as was done in section 1.3 with the command

```
f := sin(x);
```

This assigns the current 'value' of the expression `sin(x)` to the variable `f`. If `x` is an unknown, as was the case in Chapter 6, then `f` stands for the expression $\sin x$ and we can, for example, differentiate this expression with respect to x. But if `x` had already been assigned a value then that value will be used to assign a value to `f`. For example, if `x` already had the value 0, then the above assignment would give `f` the value 0 (since $\sin 0 = 0$) and if `x` had the value `a+2` then `f` would be given the value `sin(a+2)`. Notice that the use of the word 'value' is not being restricted just to numerical values. Possible 'values' which can be assigned to a variable include sets and lists and even equations.

What we have been doing is called **assigning a value to a variable** and the general format for doing it is

variable_name `:=` *expression*`;`

(Notice carefully that it is `:=` and not just `=` that is being used here and there must NOT be a space between the `:` and the `=` in `:=`.) After you have given an assignment command, Maple will replace the named variable with its assigned value wherever that variable name occurs in the future.

A variable that has not been assigned a value is an **unassigned variable** and can be used just like any mathematical variable or unknown.

2.2.2 Variable Names

Variable names must start with a letter, or the underline character `_`, and the intial letter can be followed by letters, digits and the underline character. Note that when Maple creates an "arbitrary constant" it usually begins the name with the underline character, so you should avoid starting your own variables with this character.

There is effectively no limit to the length of a name. Upper and lower case letters are treated as *different* in names. Also, any string of characters surrounded by back-quotes (i.e. ` which is not the same as the forward-quote ' or the double-quote ") is considered a name, although such names are not useful as variables. Here are some examples

There are a number of variable names (such as `Digits`) which Maple uses for its own purposes and you should not use these **reserved names** for your own variables. You can get a list of most (but not all) of these reserved names by typing `? ininames` at a Maple prompt — the Help Browser will then open at the list of initially known names.

If Maple behaves unpredictably, you might try changing the names you have used for variables, just in case you have used a reserved name.

Three names stand for constants that are important for us, namely,

`Pi`	$\pi = 3.141592\ldots$	(note capital P, small i)
`I`	$i = \sqrt{-1}$	(note capital I)
`infinity`	∞ (used with limits)	

Note that when Maple displays *results* it shows `Pi` as π and `I` as I and `infinity` as ∞. It is legally possible for you to give the name `pi` (with a lower case `p`) to a variable, but do not do it because it will cause confusion (unfortunately it will be shown as π in Maple displays, but it will not evaluate to $3.141592\ldots$).

You may come across other named constants, like `gamma` which stands for Euler's constant

$$\gamma = \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k} - \ln(n)\right) = 0.5772156649\ldots.$$

2.2.3 Unassigning

It will sometimes happen that after assigning a specific value to a variable, say `x := 2` (which we do not recommend: see below), you want to go back to using that variable as an unspecified unknown. For example, you might want to evaluate the expression `x^3-5*x+3` for `x` equal to 2 and then differentiate the expression with respect to the unknown `x`.

This process is called **unassigning** a variable and you can do it by a command of the form

variablename`:= '`*variablename*`'`
(For example, `x := 'x'` to unassign the variable `x`.) Notice that both the quote symbols here are *forward* quote `'` symbols.

If you want to unassign *several* variables at the same time, use the `unassign` command. For example

```
unassign('a','b','fred');
```

will unassign the three variables `a`, `b` and `fred`. The quotes are necessary here: otherwise you will get an error message like

```
Error, (in unassign) cannot unassign '3' (argument must be assignable)
```

because the *names* will be replaced by their *values*. The term **argument** that Maple uses here refers to the terms inside the brackets.

If you wish to unassign *all* variables at once, restart your Maple session by typing the command

```
restart;
```

or by clicking the **Restart icon** (shape of a loop with an arrow).

WARNINGS:

1. If you try to restart your Maple session by selecting New from the File menu in Maple, you will get a new worksheet window. None of the variables you have assigned will have values. Thus if you have assigned `x` the value of 2 in a worksheet called 'Untitled(1)' and select New, you will get a new worksheet called 'Untitled(2)' where the value of `x` will be `x`. However, this behaviour can be altered (see 3.11.1 on changing the Kernel Mode).
2. If you define an expression `f` in terms of `x` *at a time when a value has already been assigned to* `x` then a subsequent unassignment of `x` will NOT change `f` to being an expression in the unknown `x`. For example, the sequence of commands

   ```
   x := Pi/2;
   f := sin(x);
   x := 'x';
   diff(f,x);
   ```

   will give the answer 0 because `f` is the constant 1, whereas the sequence

   ```
   f := sin(x);
   x := Pi/2;
   x := 'x';
   diff(f,x);
   ```

   will give the answer $\cos(x)$.
3. You should NEVER assign values to commonly used variables such as `x`, because you are likely to want to use them later as unknowns or you may go back to change an earlier command in which one of these variables was used as an unknown. Note that if you go back to change and re-execute a command then the value of any variable in the command is the *most recent* value that you gave it in your Maple session and this is NOT necessarily the value it had when you originally executed the command that you are changing. You can get strange errors if you use assigned variables as though they were unassigned. If you do assign to `x` or `t` or other one letter variables, remember to unassign to them immediately after you have finished using them with a value.

2.3 Expressions and Functions.

It is important in Maple that you distinguish between **expressions** and **functions**. For example, `sin` is a function (note the absence of any variable such as x), whereas $x^2 - x$ is an expression. Thus the command

```
y := x^2-x;
```

assigns to the (dependent) variable `y` the value of the *expression* $x^2 - x$, involving the (independent) variable `x`.

Note that if `f` is a function then `f(x)` is an expression which depends on `x`, and expressions can be built up using functions in this way. For example, you can define y to be the expression $1 - \sqrt{|\sin x|}$ by entering

```
y := 1 - sqrt( abs(sin(x)) );
```

Methods of defining new mathematical functions in Maple will be discussed later. However, at this point, we WARN you that you CANNOT define a function `f` by a command such as `f(x) := ...` because this DOES NOT make `f` a function.

2.3.1 Built-in Functions.

Although we will not discuss the creation of new functions until section 2.21, we will be *using* functions in the next few sections, and so we will need some functions which have already been defined. Maple has an enormous number of 'initially-known' mathematical functions (i.e. ones which are already there when you start Maple). These include the trigonometric functions

`sin, cos, tan, csc` (i.e. cosec)`, sec, cot`

and their inverse functions

`arcsin, arccos, arctan, arccsc, arcsec, arccot`

and the hyperbolic functions

`sinh, cosh, tanh, csch` (i.e. cosech)`, sech, coth`

and their inverse functions

`arcsinh, arccosh, arctanh, arccsch, arcsech, arccoth` as well as, for example:

Function	Description	Example
`abs`	absolute value	`abs(-2);`
`sqrt`	square root	`sqrt(4);`
`ifactor`	factorise integers (can take a long time)	`ifactor(12);`
`igcd`	greatest common divisor of integers	`igcd(6,8);`
`ilcm`	least common multiple of integers	`ilcm(6,8);`
`max`	maximum of a sequence of numbers	`max(132,129,66,120);`
`min`	minimum of a sequence of numbers	`min(132,129,66,120);`
`binomial`	binomial coefficient	`binomial(4,2);`
`round`	round (up/down) to an integer	`round(3.5);`
`trunc`	truncate (towards zero) to an integer	`trunc(3.5);`
`floor`	round down to an integer	`floor(-3.1);`
`ceil`	round up to an integer	`ceil(-3.1);`
`frac`	fractional part	`frac(3.5);`
`exp`	exponential	`exp(1);`
`log` or `ln`	natural logarithm	`log( exp(2) );`
`log10`	logarithm to base 10	`log10(100);`

For a complete list of the initially-known Maple functions, get help on `inifcns` (either by typing `?inifcns` or using the Maple Help Browser).

Not all Maple functions are initially-known. Some exist in **packages** and must be **loaded** before you can use them. A package is a collection of functions which are loaded using the command `with`. For example, most linear algebra functions are in the package `LinearAlgebra`. They can be loaded using the command

```
with(LinearAlgebra):
```

Note the use of a colon here to supress unnecesary output — you should almost always use a colon when loading a package. If you use a semi-colon you will get a (long) list of all the functions in `LinearAlgebra`.

Once a package has been loaded, all the functions in it remain available until you end your Maple session. If a function belongs to a package this will be shown in the results of a help search. For example, when you get help on the function `Rank` in the `LinearAlgebra` package, the help page is titled `LinearAlgebra[Rank]`

The main packages you will be using are `student` (for calculus), `geom3d` (for 3-dimensional geometry) and `LinearAlgebra` (for linear algebra), which are described in sections 2.9, 2.13.2 and 2.12 respectively.

Exercise Load the package `student`. Get the help entry for the function `completesquare` in this package. Use `completesquare` to write $4x^2 + 12x$ as the difference of two squares.

2.3.2 Evaluating a Function and Substituting in an Expression.

If a function `f` has already been defined (in particular, if `f` is one of Maple's built-in functions), you can use the usual notation `f(x)` for the value of `f` at `x`. For example, `sin(Pi)` gives the value of sine at π and `sqrt(a+2)` stands for $\sqrt{a+2}$.

However, if `f` has been defined to be an *expression* in the variable `x` (for example by `f := x^2-x;`), you CANNOT get the value of `f` when `x` is 3 by writing `f(3)` *since* `f` *is not a function*. In this situation there are two methods which you can use to evaluate `f` for a particular value of `x`, the first being the preferred one:

1. Use the command `subs` to substitute for `x` as in

   ```
   subs(x=3,f);
   ```

   Note that this does NOT change the value of `x` or `f` — it simply displays the value that `f` would have if `x` were equal to 3. In this case `x` remains an unassigned variable and `f` remains an expression dependent on `x`.

   In general, the command

   `subs(`*expression1* `=` *expression2*`,` *expression3*`);`

   will substitute *expression2* for *expression1* everywhere that *expression1* appears EXPLICITLY in *expression3*. For example,

   ```
   subs(m=e/c^2,f=m*a);
   ```

   gives the result $f = ae/c^2$.

   Several substitutions can be done in the one command. For example,

   ```
   subs(a=2*d,d=b/c,a*b*c);
   ```

   will substitute `2*d` for `a` and *then* substitute `b/c` for `d`, giving the result $2b^2$.

2. Assign the desired value to `x` and then ask Maple to display `f`. For example, the sequence of commands

   ```
   f := x^2-x;
   x := 3;
   f;
   ```

   will finally display the value 6, which is the value of $x^2 - x$ when $x = 3$. If you want the value of `f` at a second value of `x`, just assign the second value to `x` and then display `f` again. If you want to return `f` to being an expression in the unknown `x` then you will have to unassign `x` by one of the methods described in section 2.2.3.

2.3.3 Simplifying an Expression.

Maple often leaves an expression in a complicated form rather than in its ‘simplest’ form. To remedy this situation, Maple provides a number of procedures which you can use in an attempt to get an expression into a form which suits you better. Nevertheless, you have to bear in mind that factorising and simplifying expressions (other than very easy ones such as those in high school) is a *very* difficult process, both for humans and for the computer, and it is not always clear what ‘simplify’ means. Consequently, you may have trouble getting Maple to produce what *you* consider to be the nicest form of an expression. This is probably the most frustrating part of using a computer algebra package.

The following is a list of some of the commands you can try if you want to ‘simplify’ an expression. The descriptions given here are only brief, and in each case you should use Maple’s Help to find out more about these commands.

normal tries to simplify expressions involving **rational functions** (i.e. ratios of polynomials). In particular, it will cancel common factors in a rational function and will bring sums of rational functions to a common denominator. For example,

```
normal( 1/(x-1) - 1/(x+1) );
```

gives the result

$$\frac{2}{(x-1)(x+1)}.$$

simplify tries to simplify *any* expression. For example

```
simplify( cos(x)^2+sin(x)^2 + 2^(5/2) );
```

will give you the result $1+4\sqrt{2}$.

This is more general than `normal`, and it may not do what you want for rational functions. Also, you may sometimes have to tell Maple what type of simplification you expect. For example, if x is positive (see section 2.18)

```
simplify( ln(x^2) + ln(x^3),ln );
```

will give you the answer $5\ln(x)$: the option `ln` is telling Maple how to simplify.

radsimp will simplify radicals in an expression. It may not rationalise denominators without using the `ratdenom` keyword.

expand is used to expand a product of sums of terms. For example,

```
expand( (x+1)^2 );
```

More ambitiously, you could try

```
expand( (x+a)^13*(2*x-3*b)^7 );
```

Also `expand` can do some expansions with other functions. For example,

```
expand( cos(a+b) );
```

gives the result $\cos(a)\cos(b)-\sin(a)\sin(b)$.

combine is almost the reverse of `expand`. It tries to combine terms into a single term and applies the trig identities in the reverse direction. It also tries to combine sums of unevaluated integrals into single integrals.

Sometimes you need to specify which type of combining is required. For example

```
combine( 3*ln(2)-2*ln(3), ln );
```

will produce $\ln(\frac{8}{9})$.

There are other commands for manipulating polynomials, such as **factor** (which factorises it into factors with *rational* coefficients), and **collect** (which collects coefficients of like powers). For more details of these, and other commands for manipulating expressions, follow the links

Mathematics. . . **Algebra**. . . **Expression Manipulation**. . .

in the Maple Help Navigator.

2.3.4 Defining functions with the arrow operator.

The arrow operator is used to define functions. For example, the function f which acts as $f(x) = x^2 - x$ is defined by

```
f := x -> x^2-x;
```

Here we use an arrow (typed as a `-` immediately followed by `>`) to show that the function replaces the (dummy) variable `x` by the expression `x^2-x`. Then the command

```
f(3);
```

will display 6, which is the value of `f` when `x` is equal to 3, and you can write `f(2*a-1)` for the value of `f` when `x` is equal to $2a - 1$.

Note that the `x` which occurs in the definition of `f` is a **dummy variable** and has *no relation to any variable* `x` which might occur anywhere else in your Maple session. It just provides a way of specifying a formula for the function.

You can use `f(x)` in any situation where Maple will accept an expression in the unknown `x`. You can also use `f` on its own to represent the function in some situations, but you must make sure that the situation is one in which this applies. For example, you can differentiate a function using the `D` operator (see section 2.4.2) or use a function to define a Matrix or Vector (see section 2.10.4).

In MATH1231/1241 you will study functions of more than one variable. These can also be defined with the arrow operator. For example,

```
d := (x,y,z) -> sqrt(x^2+y^2+z^2);
```

is a definition of the 3-dimensional distance function $d(x, y, z) = \sqrt{x^2 + y^2 + z^2}$.

Exercise Enter the above definition of the function `f`.
What happens when you now enter

```
f;
eval(f);
```

2.4 Elementary Calculus.

2.4.1 Limits

To find the limit of an expression as a variable tends to a value, use

`limit(`*expression*`,`*variable*`=`*value*`);`

For example, to find the limit of $(\sin x)/x$ as $x \to 0$, type

```
limit( sin(x)/x, x=0 );
```

You can use `infinity` as a value if you want to find the limit as $x \to \infty$.
For example, to find the limit of $(1 + 1/x)^x$ as $x \to \infty$, type

```
limit( (1+1/x)^x, x=infinity );
```

You can ask for a one-sided limit by inserting `left` or `right`. For example, to get the limit of $1/x$ as $x \to 0^+$, try using

```
limit( 1/x, x=0, right );
```

Note that the result is ∞. In this case the lefthand limit gives the result $-\infty$ and the two-sided limit gives the result *undefined.*

2.4.2 First Derivatives

To differentiate an *expression* with respect to a variable, use

`diff(`*expression*`,`*variable*`);`

For example, to differentiate $e^x \sin x^2$ with respect to x, you can enter

```
    y := exp(x)*sin(x^2);    diff(y,x);
```

or just

```
    diff( exp(x)*sin(x^2),x );
```

Remember that you should only differentiate with respect to an *unassigned* variable. There are two ways to differentiate a *function* `f`:

1. Put the *expression* `f(x)` in the `diff` command. For example
   ```
   diff( sin(x),x );
   ```
2. Use the operator `D`. In this case *do not* mention any variable in the command. For example,
   ```
   D(sqrt);
   ```
   $$\frac{1}{2\,\mathrm{sqrt}}$$

It is often neater to use `D` because you can use it to *differentiate and evaluate* in one step. For example, to find the derivative of $\sin x$ at $x = \pi/2$ you can use

```
    D(sin)(Pi/2);
```

which is neater than the alternative

```
    simplify( subs( x=Pi/2, diff(sin(x),x) ) );
```

You can use the `D` operator to differentiate a function that you have defined using the arrow operator. For example, if you have made the definition

```
   f := x -> x^2-x;
```

then

```
    D(f);
```

will give $x \to 2x - 1$ because the derivative of $x^2 - x$ is $2x - 1$. You can evaluate this function for a specific value just like any function, for example `D(f)(1);` returns 1, the value of $x^2 - x$ at $x = 1$.

2.4.3 Unevaluated Derivatives

If y is an *unassigned* variable, then the command

```
    diff(y,x);
```

gives the answer 0 (since as far as Maple is concerned, y does not depend on x). You can force Maple consider y to be a function of x by using $y(x)$ in place of y. For example,

```
    diff( y(x),x );
```

gives the answer $\frac{d}{dx}\,\mathrm{y}(x)$. This will be needed when solving differential equations (see section 2.15).

2.4.4 Higher Order Derivatives

To find the second derivative of $\sin 5x$, you can use

```
    diff( sin(5*x), x, x );
```

or alternatively

```
    diff( sin(5*x), x$2 );
```

This second form is an example of the fact that (almost) anywhere in Maple you can use

expression `$` *number*

(with or without spaces before and after the `$`) to stand for the sequence consisting of *expression* repeated *number* times. To differentiate an expression n times with respect to x, you can enter

```
diff( expression,x$n );
```

To find a higher derivative, for example the third derivative, of a function f of one variable using the `D` operator use

```
D[1,1,1](f);
```

This notation looks a little odd, but will make more sense once you learn about partial derivatives in MATH1231/1241 (see 2.14).

2.4.5 Implicit Differentiation

If y related to x by an equation that defines y as a function of x, then Maple can use **implicit differentiation**, to find the derivative of y with respect to x with the Maple command `implicitdiff`.
For example, to find the slope of (a tangent to) the circle $x^2 + y^2 = 1$, use

```
implicitdiff(x^2+y^2=1,y,x);
```

2.4.6 Maxima and Minima

To find the global minimum value over the whole real line for an expression in one unknown `x`, use

```
minimize(expression);
```

Use `maximize` to find the global maximum.

You can also use these commands to find global maximum and minimum values for expressions in several unknowns. For example, to find the smallest value (over all real values of x and y) of $x^2 + 2x + y^2$, try using

```
minimize( x^2 + 2*x + y^2 );
```

(You should get the answer -1, as you can see by completing the square.)

2.4.7 Integration

To find an indefinite integral of an expression with respect to a variable, use

```
int(expression,variable);
```

For example,

```
f := x^2*sin(x);
int(f,x);
```

or just

```
int( x^2*sin(x),x );
```

Note that Maple does NOT show an arbitrary constant C in an indefinite integral.

If `f` is a *function*, then you need to put the *expression* `f(x)` in the `int` command, not just `f`.

You can find a *definite* integral

$$\int_a^b f(x)\,dx$$

by denoting the limits by the *range* `x=a..b` in the `int` command. For example,

```
int( sin(x), x=0..Pi );
```

Remember to put exactly *two* full stops in the range. If you want to do an integral from a to ∞, just put `infinity` for the upper limit. For example, try

```
int( 1/(1+x^2), x=0..infinity );
```

Not all 'elementary' functions have an indefinite integral which can be expressed as an 'elementary' function. When Maple cannot find an elementary expression for an indefinite integral, it may use non-elementary functions (not covered in first year mathematics) to express its answer or, if it is really stuck, it will just display the integral expression unchanged as its answer. As examples, try the following:

```
int( sin(x^2),x );
int( exp(-x^2),x );
int( exp(sin(x)),x );
```

If Maple will not give you an exact value for a *definite* integral, you can get an *approximate* numerical value by means of `evalf(%)`.

If you are using integration to find **the area between two curves** which intersect several times, you might be tempted to ask Maple to integrate the absolute value of the difference between the functions defining the two curves. Unfortunately, this does not always work. You may have to find the points of intersection of the curves and integrate appropriately over each interval between consecutive points of intersection.

2.4.8 Partial Fractions

The standard method for integrating a rational function is to expand the integrand in partial fractions. You can expand a rational function (i.e. a polynomial divided by a polynomial) in partial fractions by the command

`convert(`*expression*`,parfrac,`*variable*`);`

(The variable has to be specified because Maple can do expansions with respect to one variable in expressions involving several unknowns.)

For example, try expanding $1/(x^2-1)$ in partial fractions and then integrating it by

```
convert( 1/(x^2-1),parfrac,x );
int(%,x);
```

Notice that you get a different-looking (but equivalent) answer to what you would get by simply saying

```
int( 1/(x^2-1),x );
```

2.5 Collections of Expressions, etc.

2.5.1 Sequences

A *sequence* in Maple is two or more *expressions* separated by commas, e.g.

```
2, 8, Pi, x^2, x*exp(x)
```

You can type a sequence and assign it to a variable, or the result of a Maple command may be a sequence. For example, sequences are often produced by Maple when it solves an equation that has more than one solution. If you try the following maple command that finds the solutions to a cubic,

```
solve( x^3 - 4*x^2 + 5*x - 2 = 0 );
```

you will see that Maple responds with the *sequence* $1,\ 1,\ 2$, that is, some values separated by commas. If you assign a sequence to a variable, the order of the values will be preserved.

If you have a variable whose value is a *sequence*, take care that you do not use it as the argument of a Maple function that expects a single object. For example, if you apply `evalf` to a *sequence* with more than 2 values, you will get an error.

If you want to give Maple a sequence that can be described by a formula, use the command `seq`. For example, the command

```
    seq( n*(n+1), n=1..5);
```

generates the finite sequence $2, 6, 12, 20, 30$.

Sequences are used to construction several other types of objects in Maple, for example, sets and lists. We will also use them to enter Vectors and Matrices (section 2.10).

2.5.2 Sets and Lists

A Maple *list* is a maple sequence enclosed in square brackets and a *set* is a sequence enclosed in a curly brackets. For example, `[1,2,3]` is a list and `{1,2,3}` is a set. Maple treats a list or a set as a single object.

The important differences between sets and lists are

- The *order* in which the things appear is *not* significant for sets but it *is* significant for lists. Thus the sets `{1,2,3}` and `{3,2,1}` are treated as the *same* set but the lists `[1,2,3]` and `[3,2,1]` are different. Of course the contents of a set will be printed in some particular order when Maple displays it, but this order is decided by Maple and may not be the same as the order you entered. Maple keeps the order of display of a set constant in any one session, but if you enter the same set in another Maple session then *the order of display may be different*.

- Repetition of an item in a set is ignored, but this is not so for lists. Thus the sets `{1,2,3}` and `{1,2,3,1}` are treated as the *same* set but the lists `[1,2,3]` and `[1,2,3,1]` are different.

Note that the things in a set or list do not have to be numbers. For example, you could define a list

```
    a := [ 1, x, x^2+x+1, y^2+y=0, {1, 2, 3} ];
```

in which the entries are a number, an variable, an expression, an equation and a set.

2.5.3 Converting Structures

You can turn a *sequence* into a *set* or a *list* by putting it inside square or curly brackets respectively. For example, try

```
    Sq := 1,2;
    St := {Sq};
    L := [Sq];
```

and check that the values assigned to the variables `St` and `L` are the *set* `{1,2}` and the *list* `[1,2]` respectively. This is useful because you cannot apply `evalf` to a sequence but you can apply it to a list.

To convert a set into a list or vice versa, use the command `convert`. For example

```
    convert( St,list );
```

gives a list with the same value as `L`. This is useful if the things in the set are numbers and you want to sort them into numerical order. You cannot apply the command `sort` in section 2.5.5 to a set or a sequence, but you can apply it to a list.

To convert a set or list into a sequence, use the command `op`. For example

```
    op(L);
```
gives a sequence with the same values as `Sq`. This is useful if you want to add someting at the end of the list. For example, you can change the value of the variable `L` from the list `[1,2]` to the list `[1,2,3]` by
```
    L := [ op(L), 3 ];
```

The command `op` will also convert any expression into a sequence. This is because algebraically any expression consists of an **operator** acting on a sequence of 'parts' or **operands**, and `op` extracts the sequence of those operands. For example, the command
```
    op(6*(x+y)/z);
```
gives the sequence $6,\ x+y\,,\ 1/z$ since $6(x+y)/z$ is the *product* of 6, $(x+y)$ and $1/z$. There is a problem in that it is not always easy to tell what Maple will regard as the operator. For example, Maple regards `1/z` as the *power* of `z` to the `-1`, and so
```
    op(1/z);
```
gives the sequence $z\,,\ -1$.

2.5.4 Selecting Operands

The command `op` is actually quite a powerful general procedure for selecting operands in many contexts, but its full description is a bit too advanced for inclusion here (for full details, use Maple's Help). One use is that if `n` has a positive integer value then

`op(n,`*expression*`);`

selects the `n`th operand of the expression. For example, the command
```
    op(2,1/z);
```
gives -1 (see the example above).

In special cases there are synonyms for `op` such as:

`numer` and `denom` give the numerator and denominator of a quotient expression. For example,

```
    numer( sin(x)/(1+cos(x)) );
```

gives the result $\sin(x)$.

`coeff` picks out the coefficient of a specified power of a variable in a polynomial. For example,

`coeff(` *polynomial*`,x,5 );`

gives you the coefficient of `x^5` in the polynomial. (If like powers have not been gathered together in the polynomial, use `collect` first.)

`lhs` and `rhs` pick out the lefthand and righthand sides of an equation.

2.5.5 Sorting

If the entries in a *list* have values which are *real numbers*, you can use the command `sort` to sort the entries in the list into *increasing* numerical order, but you CANNOT do this for a *sequence* or a *set*. For example
```
    sort([-1,2,-4,9]);
```
gives the result $[-4,-1,2,9]$.

`sort` can also be used to put the terms of a polynomial into *decreasing* power order. For example,

```
    sort(1+x+x^6-x^3);
```

will give the result $x^6 - x^3 + x + 1$.

This command is useful for selecting one of the roots of an equation *whose roots are all real.* For example, if you want to select the second smallest root of the equation $x^4 - 6x^3 - 33x^2 + 46x + 72 = 0$, the best way to do it is

```
  soln := solve( x^4 - 6*x^3 - 33*x^2 + 46*x + 72 = 0 );
  soln := [soln];    # to convert to a list
  soln := sort(soln);
  soln[2];
```

However, if some of the roots are *not* real, they may not be able to be sorted by size.

2.5.6 Substituting into a Structure

If `S` is a set or list whose entries are expressions involving a variable `x`, then you can use the command

```
    subs(x=2, S)
```

to substitute the value 2 for `x` in `S`.

The Maple command `subs` can be used in a similar way with Vectors and Matrices that will be discussed in section 2.10.

2.5.7 Applying Functions to each Entry of a Structure

You cannot generally apply a function to each entry of a structure `S` just by giving a command of the form *function*`(S)`. To do this, use `map` in the form `map(`*function*`, S)`. For example, to apply the function `sin` to each entry of a set S, use

```
    map(sin, S);
```

This will replace each entry of the set `S` by its sine. Similarly, if `L` is a list whose entries are expressions involving numbers, then you can apply `evalf` to each entry of `L` using `map`

```
    map(evalf, L);
```

to evaluate all the entries as floating point numbers. For example

```
    L := [Pi, sqrt(2), sin(1)]:
    map(evalf, L);
```

yields $[3.141592654 \quad 1.414213562 \quad .8414709848]$. Some Maple functions automatically are 'mapped' to the entries of a structure and `evalf` is one such function. So the same result could have been achieved by `evalf(L)`.

The Maple command `map` can be used in a similar way with Vectors and Matrices that will be discussed in section 2.10.

2.5.8 Sums and Products

If `f` is an expression in the unknown `k`, you can use the command

```
    sum(f,k=m..n);
```

to ask Maple to evaluate

$$\sum_{k=m}^{n} f(k).$$

To get a sum to infinity, use `infinity` as the upper limit. For example, try

```
sum(1/k^2,k=1..infinity);
```

If `f` is a function, you will need to write `f(k)` in the command (not just `f`).

You can use similar methods to evaluate products such as

$$\prod_{k=m}^{n} f(k).$$

Just use `product` instead of `sum`.

Note that there will be many cases where Maple cannot find a simple expression for the answer and it may just return your question as its answer. For example, try asking Maple to find

$$\sum_{k=2}^{\infty} \frac{1}{k^2 \ln k}.$$

Maple also has the command `add` and `mul`, which should be used in certain situations in place of `sum` and `product` — see Maple's help pages.

2.6 Equations.

In Maple an expression of the form

leftside = *rightside* ;

is an **equation**. Note that in this case the `=` symbol is used *alone* and not with the `:` which appears in assignment commands. Make sure that you do not confuse equations with assignment commands.

Maple allows you to perform various operations on equations. You can add two equations, or multiply an equation by a constant, or add a constant to it (i.e. to both sides of it). You can use this to solve simple simultaneous equations. For example,

```
> e1 := 2*x+y=5:              # e1 is the first equation
> e2 := x-y=4:                # e2 is the second equation
> e1+e2;                      # add the equations
```

$$3x = 9$$

```
> %/3;                        # divide result by 3
```

$$x = 3$$

```
> 2*e2-e1;                    # twice 2nd eqn minus 1st
```

$$-3y = 3$$

```
> %/(-3);                     # divide result by -3
```

$$y = -1$$

```
> e2+(y=y);                   # add y to both sides of e2
```

$$x = 4 + y$$

2.6.1 Solving Equations.

Maple provides two basic commands for solving equations: `solve` and `fsolve`. In general, neither of these procedures tries to find *all* solutions to an equation. However, both of them will try to find all real solutions to a *polynomial* equation.

If you give one of these solvers an *expression* instead of an equation, it will assume that you want to solve the equation *expression* `= 0` .

Using `solve`

This tries to find *exact* solutions to an equation or set of equations. For example, to solve $3x + 4 = 5x$, use

```
    solve( 3*x+4=5*x );
```

which gives the answer 2.

If the equation involves more than one variable, you will have to tell Maple which variable to solve for. For example,

```
    solve( a*x^2+b*x+c, x );
```

Notice that in this case we gave `solve` an expression rather than an equation and Maple will assume that we want to solve the equation $ax^2 + bx + c = 0$.

If `solve` finds more than one solution then you may want to select a particular one. This will be discussed later in section 2.5.

To solve more than one equation in more than one variable, use

`solve( [`*eqn1*`,`*eqn2*`,...], [`*var1*`,`*var2*`,...] );`

Note the brackets `[ ]` which tell Maple that we are giving it a *list* of equations and a *list* of variables. You can also use braces (`{ }`) to give a *set* of equations etc (which is what we do below). See section 2.5 for more detail on sets and lists.

For example, the following is a piece of Maple which solves a pair of equations (in fact, the ones we solved above) and then checks the results by substituting them back into the equations.

```
> e1 := 2*x+y = 5:
> e2 := x-y = 4:
> solve( {e1,e2}, {x,y} );
```

$$\{x = 3, y = -1\}$$

```
> assign(%):
> x; y;
```

$$3$$

$$-1$$

```
> e1; e2;
```

$$5 = 5$$

$$4 = 4$$

Notice the use of the procedure `assign`. It takes the solution values from `solve` and assigns them to the appropriate variables. The results from the last command show that when these values have been assigned to the variables, the values of the two sides of each equation are the same, so the equations are satisfied. Do not forget to unassign `x` and `y` after these steps, in case you want to use them again as unknowns in other equations.

When using `solve`, you will often get answers involving expressions like

$$\mathrm{RootOf}(\_Z^5 - 5\_Z - 1)$$

It is not possible to solve the equation $z^5 - 5z - 1 = 0$ in rational numbers or radicals, so Maple just says the answer is a *general* root of this equation. The variable $\_Z$ is just a dummy variable, introduced by Maple to state the formula for the polynomial. You should not use $\_Z$ yourself.

You can assign this general root to a variable and do calculations with this variable as though it were a number, but remember that it really stands for more than one value (in the above example, 5 values).

If you have done some calculations involving a `RootOf` expression and want to see *all* the corresponding answers, use the procedure `allvalues`. This will produce a *sequence* (see section 2.5) whose entries are the values that you want. If the roots can be expressed in a simple form, this sequence will be useful. Otherwise (as in the case of the solutions of $z^5 - 5z - 1 = 0$), you will simply be given all the answers in symbolic form. This may be overcome by using `evalf` to get the approximate answers. For example, try entering

```
alpha := 2 + 3*RootOf(_Z^5-5*_Z-1);
allvalues(alpha):
map(evalf,[%]);
```

Using `fsolve`

This tries to find *approximate floating point* solutions to an equation or set of equations. Note that in general `fsolve` only finds one solution even when many solutions exist. For example, the command

```
fsolve( tan(sin(x))=1, x );
```

will only give you the solution $.9033391108$ (though there are obviously many other solutions — as can be seen by plotting the graph of $\tan(\sin x)$).

If you give `fsolve` a *range of values* for the variable, it will try to find a solution in that range. For example, the command

```
fsolve( tan(sin(x))=1, x, 2..3 );
```

will give you another solution 2.238253543.

Sometimes `fsolve` may not find any solution even though one exists. This usually happens because `fsolve` is using a method of successive approximations and none of the starting points which it tried has produced a convergent sequence of approximations. In this case, try suggesting a range of values within which you know that there is a solution (from something like the Intermediate Value Theorem or by plotting the graph). This gives Maple a better idea of values it should try as starting points for its approximations.

Remember that for a *polynomial* equation, `fsolve` will try to find all real solutions (or all real solutions in a specified range). For example, the command

```
fsolve( x^3-3*x^2+2*x, x, -1..3/2 );
```

will give you the solutions $0.$ and 1.000000000, but not $x = 2.000000000$.

If you want `fsolve` to find all *complex* (including real) roots of a polynomial, insert the keyword `complex`. For example,

```
fsolve( x^2-2*x+2, x, complex );
```

Maple provides other solvers for special situations, for example `dsolve` for differential equations and `rsolve` for recurrence relations — see sections 2.15 and 2.17.

2.7 Complex Numbers.

In Maple, complex arithmetic is *normally* done automatically with `I` standing for $\sqrt{-1}$ (for example, if you square `I` you will get -1, not I^2). But Maple does not *always* automatically evaluate an expression involving complex numbers. For example, it may leave an expression as the product of some complex numbers or as an expression involving a root of a complex number. In these cases, apply the function `evalc` to force Maple to *evaluate as a complex number.*

For example, Maple will leave

```
(-2*I)^(1/2);
```

as $\sqrt{-2I}$ until you apply `evalc`. Applying `evalc` gives the result $1 - I$. Note that `evalc` does not give you *both* the square roots of `-2*I` — it only gives the 'principal value' of the root. If you want both the roots, use

```
solve( z^2=-2*I );
```

You will find that `evalc` often gives its results in terms of trigonometric functions and this may make the results hard to read. You may find the approximate floating point form more readable. To get this, apply `evalf` (with or without `evalc`).

When solving for the roots of a polynomial with real or complex coefficients, the result may be a sequence of complex numbers or a `RootOf` expression (which gives a sequence of complex numbers when you apply `allvalues`, as explained above). These sequences are often in a very messy form and you need to convert them to something simpler using `evalc` and/or `evalf`. Note carefully that you must enclose a sequence of results in *square brackets* before you apply `evalc` nor `evalf` to it. (The reasons for this will be explained in section 2.5).

For example, to find the cube roots of the complex number $2 + i$, we would use

```
solve( z^3=2+I );
```

This produces a sequence of 3 complex numbers expressed in a messy form. If you want these exact values to now be expressed in the standard $a + ib$ form, use

```
evalc([%]);
```

This will give an even bigger mess (though exact and in the standard form), so you will probably want to see the approximate floating point values. To get these, use

```
evalf([%]);
```

OR get them directly with

```
fsolve( z^3=2+I, z, complex );
```

(as explained in the section 2.6.1).

Maple provides a number of built-in mathematical functions which can be used with complex arguments. These include

`Re(z)`	real part
`Im(z)`	imaginary part
`conjugate(z)`	conjugate
`abs(z)`	complex modulus
`argument(z)`	argument (in range $-\pi < \arg \leq \pi$)

2.8 Plotting.

Maple has the ability to plot data points and graphs and parametrically defined curves and curves in polar coordinates. It can also do 2-dimensional representations of surfaces in $\mathbb{R}^3$.

The simplest way to get Maple to draw a graph is a using a command like

`plot(`*expression*`,`*variable*`=`*start*`..`*end*`);`

In this case Maple will decide the appropriate vertical scale. If you want to specify minimum and maximum values on the vertical scale, use

`plot(`*expression*`,`*variable*`=`*start*`..`*end*`, `*min*`..`*max*`);`

Note that there must be exactly TWO full stops between *start* and *end* and between *min* and *max*. (This notation is used whenever you refer to a **range** of values in Maple.)

This command will plot the *expression* for values of the *variable* over the range from the *start* value to the *end* value and the range on the vertical axis is restricted to the range from the *min* value to the *max* value. The *expression* should normally depend on the *variable* and the *variable* must be an unassigned variable (otherwise you will only get an error message, an empty plot or a straight line).

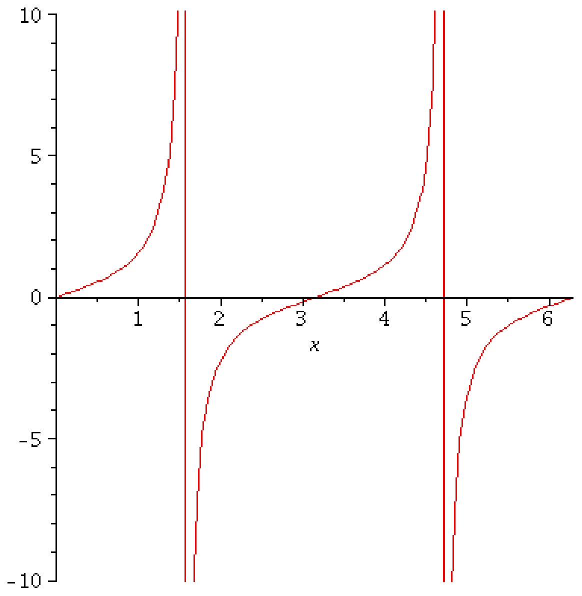


Figure 2.1: Plot of $\tan x$

For example, figure 2.1 was produced using

```
plot(tan(x), x=0..2*Pi, -10..10);
```

to plot $\tan x$ from 0 to 2π with the vertical scale restricted to $[-10, 10]$.

The plot will appear in the output region linked to this command (be patient — it may take a little time to appear). Note what Maple has done with the discontinuities at $\pi/2$ and $3\pi/2$.

The appearance of this plot can be changed. Click anywhere on the graph and you will notice two things will happen:

- a window will appear around the graph with 'markers' on the edges and vertices: these can be used to re-size this window in the usual way;
- the context bar (just above the worksheet — see section 1.2) will be replaced by a new set of buttons (use the `Tool Tips` from the `Options` submenu of Maple's `Tools` menu, see 3.11.1, to find out about these), with a box on the extreme left showing the approximate coordinates of the point where you clicked: this is useful

for getting approximate coordinates for a point of intersection of two graphs or the intercepts of a graph.

Note: As Maple scales each axis differently, it may give a misleading impression of the graph. To get a truer scale, add the option `scaling=CONSTRAINED` at the end of the `plot` command, or click on the button `1:1` in the context bar.

If you want to compare plots, you can have two or more plot windows open at the same time *or* you can plot more than one curve on the same set of axes. For example, to plot $y = f(x)$, $y = g(x)$ and $y = h(x)$ simultaneously on the same axes use

```
plot( {fx,gx,hx}, x=start..end);
```

when `fx`, `gx` and `hx` are expressions depending on `x`, or

```
plot( {f(x),g(x),h(x)}, x=start..end);
```

when `f`, `g` and `h` are functions.
Note the use of *curly brackets* (**braces**) `{ }` here.

The `plot` command offers options which control the number of points at which the function is plotted, the number of tick marks on the axes and the placing of titles on the graph. Read the help page on `plot` to find out about these options.

To get a printout of a Maple plot, *right click* while pointing at the graph. A menu will pop up: select the option Export and the option "Encapsulted PostScript". A dialogue box will open and you will have to enter the name of an appropriate file (for example `temp.eps`) and then click "Save". This will create a new file which you can find and view in the file manager. The file manager will open this file in a postscript viewer and you can print it using this viewer. See the lab notes for details on using file manager and printing.

2.8.1 Plotting Piecewise-defined Functions

If you wish to plot an expression which is defined in a piecewise way, then use the Maple command `piecewise`. For example, to plot the expression `fx` which is mathematically

$$fx = \begin{cases} 0 & \text{if } x < 0; \\ x^2 & \text{if } 0 \le x < 1; \\ 2x - 1 & \text{if } x \ge 1. \end{cases}$$

you can use the command:

```
fx := piecewise( x<0,0, x<1,x^2, 2*x-1 ):
```

For example the value of `fx` will be 0 when $x = -1$, $1/4$ when $x = 1/2$, and 5 when $x = 3$. Note that the ranges are stated before the expressions, that an earlier range ($x < 0$) takes precedence over a later one ($x < 1$) and the last range ($x \ge 1$) can be left out. An alternate way of doing this is given in section 2.21.1.

Now to plot this for x between -5 and 5, simply type

```
plot(fx, x=-5..5);
```

2.8.2 Plotting Data Points

To plot x-y data points defined by two lists

```
x := [x1,x2,...,xn]   y := [y1,y2,...,yn]
```

you should first construct a *list* of the coordinate pairs [*x1*,*y1*], [*x2*,*y2*], ... , [*xn*,*yn*] and then plot this list. (For details on lists, see section 2.5.) For example,

```
datapoints := [ [1,sin(1)], [2,sin(2)], [3,sin(3)], [4,sin(4)],
                [5,sin(5)], [6,sin(6)], [7,sin(7)], [8,sin(8)] ];
```

or (see section 2.5 for the `seq` command)

```
datapoints := [seq([i,sin(i)],i=1..8)];
```

Note the use of *square* brackets here. The square brackets in `[1,sin(1)]` etc. are to indicate the coordinates of a point in the x, y-plane (an ordered pair of numbers — i.e. a list with two members). The outermost square brackets are to make the sequence of coordinate pairs into a list.

When you have created the list of coordinate pairs, execute the plot by

```
plot(datapoints);
```

2.8.3 Parametric Plots

To plot a curve defined by

$$(x, y) = (ft, gt)$$

for t from a to b when `ft` and `gt` are *expressions depending on* `t`, use

```
plot( [ft,gt,t=a..b] );
```

Note the use of *square brackets* `[ ]` here. For example, try plotting a cusp with

```
plot( [t^2,t^3,t=-5..5] );
```

or try plotting a circle with

```
plot( [cos(t),sin(t),t=0..2*Pi] );
```

where $\cos(t)$ and $\sin(t)$ are expressions made up using the *functions* $\cos$ and $\sin$.

2.8.4 Polar and Implicit Plots

The package `plots` provides special plotting procedures. Once you have loaded the package by doing

```
with(plots):
```

you can use `polarplot` for polar curves and `implicitplot` for implictly defined curves.

Figure 2.2 is a plot of the the cardioid $r = 1 - \cos\theta$ for θ from 0 to 2π done using the command

```
polarplot(1-cos(t),t=0..2*Pi);
```

For convenience, we have used `t` as the name for the variable representing θ — we do not have to name it `theta`.

You can plot the same curve in a parametric format as

```
polarplot( [1-cos(t),t,t=0..2*Pi] );
```

and you can plot several polar curves on the same diagram by using curly brackets in the same way as for the command `plot`.

Try plotting the implicitly defined ellipse $x^2 + y^2/4 = 1$ to true scale by

```
implicitplot(x^2+y^2/4=1,x=-3..3,y=-3..3,scaling=CONSTRAINED);
```

where the `scaling=CONSTRAINED` forces Maple to use the same scale on the two axes (otherwise the graph looks like a circle).

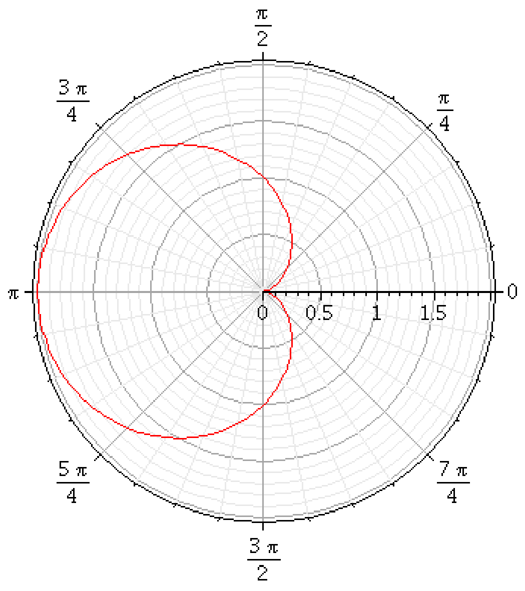


Figure 2.2: Plot of cardioid with `polarplot`

2.8.5 3-D plots

Maple provides lots of nice features for 3-D plots. For example, you can change the plot style, the style and scale of the axes and the point from which the surface is viewed. If you click on the graph, you will get a window and new buttons in the context bar (in a similar way to 2-dimensional plots). If you continue to hold down the left mouse button (while pointing the mouse at the plot), and move the mouse around, the orientation will change so that you can get a different perspective on the plot.

3-D plotting is not required for first year Mathematics subjects in any serious way, and we do not give the details here. The command for 3-D plots is `plot3d` and you should see Maple's Help if you wish to use it.

2.9 The `student` Calculus Package.

One of the aims of the `student` calculus package is to help you to practice the sort of methods which are needed to find integrals when you do not have access to software like Maple. When you use commands from this package, *you* can specify the method of integration and Maple just does the calculations.

The package also contains other procedures of interest to students of calculus (e.g Riemann sums, Simpson's Rule, maxima and minima). Some of these are described in this section and you can use Maple's Help to find out about the others.

As mentioned in section 2.3.1, the procedures described in this section will not work unless you have loaded the `student` package by typing

```
with(student):
```

2.9.1 Inert procedures.

If you are trying to practice integration by substitution then you do not want Maple to go ahead and carry out the integration as soon as you enter the integral — you want Maple to transform it into another integral. Maple provides the command `Int` (note the *upper case* `I`) to allow you to enter an integral without having Maple go ahead and evaluate it. This is an example of what is called an *inert procedure*. Maple also provides the inert procedures `Diff`, `Limit`, `Sum` and `Product` for entering derivatives, limits, sums and products.

After you have entered an expression with one of these commands and applied various procedures to the expression, you can finally evaluate it with the command `value`, as in

```
    value(%);
```

Note that `value` only works with `Int`, `Diff`, `Limit`, `Sum` and `Product`.

2.9.2 Change of Variable

To change the variable in an integral (i.e. to apply the method of integration by substitution) use the command `changevar`. The general form of this command is

```
    changevar(s,F,u);
```

where

`u` is the new variable which is being introduced

`s` is an equation of the mathematical form $h(x) = g(u)$ which implicitly or explicitly defines the old variable x in terms of the new variable u

`F` is an expression like `Int(f,x)` or `Int(f,x=a..b)` when `f` is an expression in `x`; or `Int(f(x),x)` or `Int(f(x),x=a..b)` when `f` is a function.

The result of the command is an expression for `F` with the variable changed. For example,

```
  changevar( 1+x^2=u, Int(x/(1+x^2),x=a..b), u );
```

$$\int_{1+a^2}^{1+b^2} \frac{1}{2u}\, du.$$

If you use `value` to evaluate an indefinite integral in terms of `u` then you cannot always use `changevar` in the reverse direction to get the answer in terms of `x`. You may have to use `subs` to substitute for `u` in terms of `x` (see section 2.3.2).

2.9.3 Integration by Parts

This is integration by means of the formula

$$\int u\, dv = uv - \int v\, du.$$

The general form of the command is

```
    intparts(F,u);
```

where `u` is the expression which is to play the role of u in the above formula and `F` is of the form `Int(u*`*expression*`,x)`.

For example, to integrate $x^3 \sin x$ with $u = x^3$ try

```
  intparts( Int(x^3*sin(x),x), x^3 );
```

2.9.4 Riemann Sums and Simpson's Rule

The `student` package offers you the opportunity to calculate some Riemann sums and get pictures showing the graph of the integrand together with the rectangles whose areas make up the Riemann sum. It can also show sums corresponding to approximations given by Simpson's Rule.

The command

```
  rightsum(expresssion,x=a..b,n);
```

will give you a summation expression for the Riemann sum obtained by dividing the interval from `a` to `b` into `n` equal subintervals and erecting on each subinterval a rectangle whose height is the value of *expression* at the righthand end of the subinterval. If you use `rightbox` instead of `rightsum`, you get the diagram for the same Riemann sum.

There are corresponding commands `leftsum` and `leftbox` which use values at the lefthand ends of the subintervals and `middlesum` and `middlebox` which use values in the middle of the subintervals.

The command

```
    simpson(expression,x=a..b,n);
```

gives the summation expression which would arise if you used Simpson's Rule to get an approximate value for the integral of *expression* from `a` to `b`.

The expressions given by the comands `rightsum`, `leftsum`, `middlesum` and `simpson` are not numerical values — they are in the form of a sum such as

$$\frac{1}{10}\left(\sum_{i=1}^{10}\left(\frac{1}{100}i^2\right)\right).$$

To see the actual value of the summation, apply `value`. This will, in general, give an algebraic expression and you may need to use `evalf` in order to get a meaningful answer. If the value of `n` is large then `value` will produce an enormous amount of messy output, so it is best to combine all these procedures into one command by entering

```
    evalf( value(rightsum(f(x),x=a..b,n)) );
```

(or alternatively, use a colon at the end of the `value` command line).

2.10 Vectors and Matrices

2.10.1 Vectors.

Maple uses a capital V for vectors when you use the LinearAlgebra package. (Be careful not to use vector with a small 'v'. You will not get an error as this is used by an older package called `linalg`, but things may not work as you expect.)

You create Vectors by using the `Vector` procedure, or more easily by angle brackets (the $<$ and $>$ signs):

```
   <sequence>;
```

where *sequence* is a sequence of expressions (which *can* be numbers). For example, the command

```
    v := < 1,-5,4 >;
```

assigns to the variable `v` a value which is a 3-dimensional Vector with components 1, -5 and 4 (in that order). Alternatively, one could use

```
    v := Vector( [ 1,-5,4 ] );
```

You can also use fuctions to define Vectors. For example, to create a Vector in $\mathbb{R}^3$ whose entries are 1,4 and 9 respectively, use

```
    Vector(3, i -> i^2);
```

2.10.2 Matrices

Maple uses a capital M for a matrix in the `LinearAlgebra` package. (Be careful not to use matrix with a small 'm'. You will not get an error as this is used by an older package called `linalg`, but things may not work as you expect.)

You create a Matrix using pairs of angle brackets (or the `Matrix` procedure). You enclose the columns of the Matrix as a collection of `Vectors`, separated by vertical lines `|` and linked together with an outer set of angle brackets. For example, you can assign the matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & -1 & 5 \\ 3 & -2 & 4 \end{pmatrix}$$

to the variable `A` with the command

```
A := < <1,3,3> | <2,-1,-2> | < 3,5,4> >;
```

Note the outer set of angle brackets `<  >` which make the object into a `Matrix`. These outer brackets must be there even if the matrix only has one column, as in

```
A := < <1,2,3> >;
```

Note that Maple does NOT regard a Vector as the same thing as a Matrix with one column, but the differences will probably not be relevant to you in first year.

An alternative way to enter matrices is by rows rather than columns: all you need to do is "swap the commas and bars around". So the prevous matrix can be defined by

```
A := < <1|2|3> , < 3|-1|5> , <3|-2|-4> >;
```

However, it is better to think of matrices in columns rather than rows so the 'collection of columns' method is preferable. It is also easier to type commas rather than the vertical bars, of course, and there are fewer bars in the collection of columns method.

You can also create a Matrix using a function that tells Maple how to calculate each entry. For a Matrix you need a function of two variables (and to give two dimensions), for example,

```
Matrix(3,2, (i,j) -> i^2+j^3);
```

It is conventional in mathematics to use a single capital letter to denote a matrix, but `Matrix` names in Maple do not have to follow this convention — they could be a lower case letter or any valid Maple name. But remember that `A` and `a` are not the same name. Also be warned that some letters (such as `D`) are reserved.

Note that the default behaviour of Maple, for both Vectors and Matrices, is to display a placeholder when any dimension of a Vector or Matrix is larger than 10. For example, try

```
Vector(11, i -> i^3);
```

To see how to alter this behaviour look up the Maple help page for `interface`.

2.10.3 Selecting Components of Vectors and Matrices

If the value of `A` is a Matrix, you can use the notation `A[i,j]` to refer to the entry in the `i`th row and `j`th column of `A`. For example, if the value of `A` is

$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & -1 & 5 \\ 3 & -2 & 4 \end{pmatrix}$$

then `A[3,2]` has the value `-2`.

Similarly, if the value of a variable `v` is a Vector, you can use `v[n]` to refer to its `n`th entry. For example, if (the list) `v` has been assigned a value by

```
v := [1,2,-4,9];
```
then `v[3]` will have the value -4 because the third entry in the list is -4. Note that the same notation also works for lists.

You can use these notations in commands to *change* the value of entries. For example, the commands

```
v[2] := 12:    A[2,1] := 13:
```
will assign the value 12 to the second entry of the list `v` and the value 13 to the entry in the second row and first column of the Matrix `A`.

You can do the same thing with a set, but this is dangerous because the order of entries in a set is not fixed, so you cannot be sure which entry Maple will regard as being the `n`th entry.

2.10.4 Manipulating Vectors and Matrices

The last section shows how to create and display Vectors and Matrices. However, we cannot do any useful linear algebra without first loading the `LinearAlgebra` package, which provides many procedures for dealing with Vectors, Matrices and linear equations. In these Notes we will only give an outline of some of the basic procedures. To find out more about these and other procedures available in `LinearAlgebra`, use Maple's Help.

To use the procedures in this package, it is easiest to first load it with the command

```
with(LinearAlgebra):
```
Note the use of the *colon* here, which supresses the (very long) list of new funcitons in `LinearAlgebra`. You can use a command from this package directly by giving its **full name**: you do this by adding `LinearAlgebra:-` to the start of the command. This is what Maple will do if you operate with the context sensitive menus (see section 1.3.4).

The package `LinearAlgebra` also allows you to enter a *diagonal* matrix (i.e. a matrix A with $a_{ij} = 0$ for $i \neq j$) by using `DiagonalMatrix`. You can use either a list or a Vector as the argument of `DiagonalMatrix`. For example,

```
A := DiagonalMatrix([1,2]);
```

$$A := \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

There is the useful command `IdentityMatrix` for creating (of course) identity matrices. For example, the 6×6 identity matrix is

```
id6 := IdentityMatrix(6);
```

Also, new Matrices can be constructed by combining Vectors and/or Matrices. If we have Vectors, say `v1`, `v2` and `v3`, then we can form a Matrix which has these vectors as *columns* by using the angle brackets and vertical bars

```
< v1 | v2 | v3 >;
```
Similarly, if we have matrices `A`, `B` and `C` of appropriate sizes we can combine them *side by side* (called **augmenting**) into a large matrix by using

```
< A | B | C > ;
```
or *one on top of the other* (called **stacking**) by using

```
< A , B , C >;
```

Conversely, you can extract vectors from matrices by using `Column(A,i)` to create a *Vector* whose components are the entries in the `i`th column of the Matrix `A`. For example,

```
    v := Column(A,2);
```

We also have `Row` for rows — but this creates a **row Vector**, which is not something we'll be dealing with.

If you use a range (for example `2..3`) you can extract more than one column (as a sequence).

Note that the things created by `Column` (and `Row`) are *Vectors* (or sequences of Vectors) and not Matrices

Another procedure which selects parts of matrices is `SubMatrix`. The command

```
    SubMatrix( A, a..b, c..d );
```

(where `a..b` and `c..d` are *ranges* of positive integers) produces a smaller Matrix whose entries are the values of `A[i,j]` for `a` ≤ `i` ≤ `b` and `c` ≤ `j` ≤ `d`, arranged in the same relative positions as they had in `A`.

You can also use `SubMatrix` in the form

```
    SubMatrix( A, [1,4..6], [2..4] );
```

to get the submatrix whose entries are the values of `A[i,j]` for `i` in the list `[1,4,5,6]` and `j` in the list `[2,3,4]`.

Maple also let's you define Vectors and Matrices using an *indexing function.* If f is a function that takes one argument, `Vector(5, f)` gives a vector with the 5 entries, $f(1)$, $f(2)$, $f(3)$, $f(4)$, $f(5)$. Similarly, if g is a function that takes two arguments, `Matrix(2, 3, g)` produces a Matrix with 2 rows and 3 columns in which the element in the i^{th} row and j^{th} column is given by $g(i,j)$. See section 2.21 for examples.

2.11 Gaussian Elimination.

The `LinearAlgebra` package provides the procedure `RowOperation` which allows you to solve systems of simultaneous linear equations by going step by step through the steps of Gaussian elimination. The one procedure will do different things, depending on what arguments you give it. So:

`< A | b>`	augment Matrix `A` by Vector `b` to give `(A|b)`
`RowOperation(A,[i,j])`	swap rows `i` and `j` in `A`
`RowOperation(A,i,m)`	multiply row `i` of `A` by the value of `m`
`RowOperation(A,[i,j],m)`	replace row `i` with row `i` + (row `j`)*`m` in `A` i.e. add `m` times row `j` to row `i` (NOTE THE ORDER CAREFULLY)

Note that the Matrices produced by these procedures are NOT assigned as new values for the original variable `A` (unless you specifically instruct Maple to do that by adding the option `inplace=true`, see the help page). Usually you will assign the resulting matrix to a new variable (as in the following example) OR just allow Maple to display the result and use `%` to refer to it in your next command.

Example

If you want to apply Gaussian elimination to the augmented matrix

$$(\mathrm{A}|\mathbf{b}) = \left(\begin{array}{cccc|c} 0 & 1 & 2 & 1 & 1 \\ 2 & -1 & -2 & 1 & -1 \\ 1 & 2 & 4 & 0 & 5 \end{array}\right)$$

then you could proceed as follows. (Note that some of the command lines end in a colon. This is to save space by suppressing display of the result.)

```
> with(LinearAlgebra):
> A := < <0,2,1> | <1,-1,2> |< 2,-2,4> | <1,1,0> >:
> b := < 1,-1,5 >:
> m1 := < A | b >;
```

$$m1 := \begin{bmatrix} 0 & 1 & 2 & 1 & 1 \\ 2 & -1 & -2 & 1 & -1 \\ 1 & 2 & 4 & 0 & 5 \end{bmatrix}$$

```
> m2 := RowOperation( m1,[1,2] );
```

$$m2 := \begin{bmatrix} 2 & -1 & -2 & 1 & -1 \\ 0 & 1 & 2 & 1 & 1 \\ 1 & 2 & 4 & 0 & 5 \end{bmatrix}$$

```
> m3 := RowOperation( m2,[3,1],-1/2 );
```

$$m3 := \begin{bmatrix} 2 & -1 & -2 & 1 & -1 \\ 0 & 1 & 2 & 1 & 1 \\ 0 & 5/2 & 5 & -1/2 & 11/2 \end{bmatrix}$$

```
> m4 := RowOperation( m3,[3,2],-5/2 );
```

$$m4 := \begin{bmatrix} 2 & -1 & -2 & 1 & -1 \\ 0 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & -3 & 3 \end{bmatrix}$$

This matrix is now in echelon form. We could use the procedure `BackwardSubstitute` at this point but instead we will carry out the backsubstitution on the matrix itself.

```
> m5 := RowOperation( m4,3,-1/3 );
```

$$m5 := \begin{bmatrix} 2 & -1 & -2 & 1 & -1 \\ 0 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

```
> m6 := RowOperation( m5,[2,3],-1 ):
> m7 := RowOperation( m6,[1,3],-1 );
```

$$m7 := \begin{bmatrix} 2 & -1 & -2 & 0 & 0 \\ 0 & 1 & 2 & 0 & 2 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

```
> m8 := RowOperation( m7,[1,2],1 );
```

$$m8 := \begin{bmatrix} 2 & 0 & 0 & 0 & 2 \\ 0 & 1 & 2 & 0 & 2 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

```
> reduced := RowOperation( m8,1,1/2 );
```

$$\text{reduced} := \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 & 2 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

We now have the fully reduced form and we can read off the general solution as

$$[1, 2-2\lambda, \lambda, -1]$$

where λ is an arbitrary parameter representing an arbitrary choice of the third variable.

When Maple is solving systems of linear equations, it uses names like $\_t_1$, $\_t_2$ for arbitrary parameters which may occur in the general solution. So the solution of the above set of equations would be displayed as

$$[1, 2-2\_t_1, \_t_1, -1]$$

if you were using Maple, or as `[1, 2-2*_t[1], _t[1], -1]` in text files or when you are using plain Maple without a Maple window.

In the above example, intermediate matrices were assigned to named variables so that we could go back to an earlier stage if we found that we had made a mistake. Alternatively you could simply use `%` to apply each command to the previous result. The above example would then start with something like

```
<A | b>;
RowOperation( %,1,2 );
RowOperation( %,[3,1],-1/2 );
RowOperation( %,[3,2],-5/2 );
```

The `LinearAlgebra` package also provides procedures which do all or parts of the Gaussian elimination process in one step. In particular, you can use:

`GaussianElimination(A)`	gives the row-echelon form of `A`
`ReducedRowEchelonForm(A)`	gives the fully-reduced form of `A`
`BackwardSubstitute(W)`	does backsubstitution on a matrix `W` in echelon form, assuming that `W` has been been derived from an augmented matrix of the form `(A|b)`
`LinearSolve(A,b)`	solves `A.x=b`
`Basis( {v1,v2,...,vn} )`	finds a basis for the vector space spanned by the set $\{v1, v2, \ldots, vn\}$ of Vectors.

NOTES:

1. Maple does all its reductions using rational numbers, unless the original matrix had some floating point numbers in it. If you want to see the results in floating point, apply `evalf` in the way which was described in section 2.5.7. If the matrix involved is large and has floating point entries, the procedures `GaussianElimination`, `ReducedRowEchelonForm` and `LinearSolve` *will not always give exactly correct answers* because of problems with rounding errors when doing floating point arithmetic.

2. Maple can solve linear sytems with unknown parameters in them. In particular, it can solve systems in which each of the entries in the righthand side is an *unassigned* variable, for example

```
b := < b1 , b2 , b3 >;
```

3. If you are doing MATH1231 or MATH1241, you should look up the Maple help files for the procedures `Rank`, `RowSpace`, `ColumnSpace`, `Transpose`, `Determinant`, `Eigenvalues`, `Eigenvectors`, `CharacteristicPolynomial`. To anyone who has studied the linear algebra section of MATH1231/1241, the effects of these procedures should be obvious from their names.

2.12 Vector and Matrix Arithmetic

The last section shows how to create and display Vectors and Matrices. However, we cannot do any useful linear algebra without first loading the `LinearAlgebra` package, which provides many procedures for dealing with Vectors, Matrices and linear equations. (Technically, `LinearAlgebra` is called a **module**, but you can ignore this distinction.) In these Notes we will only give an outline of some of the basic procedures. To find out more about these and other procedures available in `LinearAlgebra`, use Maple's Help.

To use the procedures in this package, it is easiest to first load it with the command

```
with(LinearAlgebra):
```

Note the use of the *colon* here, which supresses the (very long) list of new funcitons in `LinearAlgebra`.

However, you can use a command directly by giving its **full name**: you do this by adding `LinearAlgebra:-` to the start of the command. This is what Maple will do if you operate with the context sensitive menus (see section 1.3.4).

Vectors and Matrices can be added, subtracted and multipled by scalars using the usual operators `+`, `-` and `*`. Two Matrices can be multipled using the `.` operator. The `.` operator is also used to give the dot product of two Vectors. A square Matrix can be raised to an integer power using `^`.

For example, to find the linear combination $2\mathbf{v} - 3\mathbf{w}$ of the vectors $\mathbf{v}$ and $\mathbf{w}$ we use

```
2*v - 3*w;
```

and we can enter the formula for a general point on the line through the points with position vectors $\mathbf{v}$ and $\mathbf{w}$ (i.e. the formula $\mathbf{x} = (1-\lambda)\mathbf{v} + \lambda\mathbf{w}$, where λ is an arbitrary parameter). We do this by

```
x := (1-lambda)*v + lambda*w;
```

Note that `A^(-1)` gives the inverse matrix.

When using these operations, you must of course ensure that the Matrices and Vectors are of the appropriate dimensions and, in the case of the negative power of a matrix, that the matrix is invertible.

If you enter an expression involving these operations, Maple will automatically carry out the operations. For example, to enter the mathematical expression $A\mathbf{x} + \mathbf{b}$, we use

```
A.x + b ;
```

If values have already been assigned to `A`, `x` and `b` then `A.x+b` will be evaluated, but if some of these variables are unassigned then you will get an answer involving the unassigned variables.

A useful convention in Maple is that, in expressions involving matrices, a scalar constant can be treated as that scalar multiple of an appropriate identity matrix I. For example, if `A` is a square matrix then the command

```
   1 + A + 2*A^2 ;
```

evaluates the polynomial expression $I + A + 2A^2$, where I is the appropriate identity matrix. This convention enables you to substitute matrices into polynomials. For example,

```
  subs( x=A, 1+x+2*x^2 );
```

gives the expression $I + A + 2A^2$ (because the 1 is interpreted as an identity matrix).

If you use the dot with two Vectors of the same size, Maple will calculate their dot (inner) product. So

```
  v := <2,1,2> : w := <-4,2,1>:
  v.w;
```

will return -4, the dot product of the two vectors.

The transpose of a Matrix or Vector is given by the `Transpose` function from the `LinearAlgebra` package. For example, if `A` is a 2 by 3 Matrix, then `Transpose(A)` is the 3 by 2 Matrix which is the transpose of `A`. If `v` is a Column vector, then `Transpose(v)` is the corresponding row Vector.

2.13 Vector Geometry

Maple provides facilities for doing geometry in $\mathbb{R}^n$ and, in particular, in $\mathbb{R}^3$.

2.13.1 Dot and Cross Products, Length

The `LinearAlgebra` package provides the procedures

`DotProduct(v,w)` (or `v.w`)	dot product of $\mathbf{v}$ and $\mathbf{w}$
`CrossProduct(v,w)`	cross product of $\mathbf{v}$ and $\mathbf{w}$

To calculate the length of a vector, you need to use the procedure `Norm`, in the `LinearAlgebra` package, but it must be used with care. If `v` is a Vector, then the command `Norm(v,2)` will give the length of the Vector. (The reason for the `2` is that `Norm` can be used for rather more general purposes which you will not need to know about in first year mathematics.)

There is also a `distance` procedure in the `student` package, which gives the distance between two points (represented by lists or Vectors). Use Maple's Help for more information.

2.13.2 Three-dimensional Geometry.

The package `geom3d` contains many useful procedures for solving problems in $\mathbb{R}^3$. In fact, many of the geometric problems involving planes from the MATH1131 Algebra Notes can by solved with procedures in `geom3d`. The following is an outline of some of the things which you can do with `geom3d`.

The method of assigning a name to a point, line or plane is not quite what you might expect. To say that A is the point $(1, 2, 3)$, you do NOT enter `A := [1,2,3];` — this is assigning a point to a name. You enter `point(A,[1,2,3]);` — you are assigning a name to a point. The command `line` is used to assign a name to a line. The line may be specified by giving two points on it or a point on it and a direction parallel to it. The command `plane` is used to assign a name to a plane. The plane may be specified by giving a cartesian equation for it or three (non-collinear) points on it or a point on it and a normal direction or a point on it and two lines parallel to it. The command `sphere` is used to assign a name to a sphere. A sphere may be specified by giving its cartesian equation or four points on it or the end-points of a diameter or its center and its radius. To display the specifications of one of these things you need to use `detail` (see the example below). For a plane, the detail includes a cartesian equation for the plane. If you want to find a normal to a plane `p` use

```
    NormalVector(p);
```

Be warned that if you specify a plane or sphere by means of an equation then Maple will want you to specify the names of the variables which are associated with the three axes. You can do this by listing them as a third argument to the `plane` or `sphere` command, as in

```
     plane(P,x+y+z=1,[x,y,z]);
```

If you leave out the `[x,y,z]` then Maple will, rather strangely, prompt you with the request `enter the name of the x-axis`, to which you reply `x;`, and similarly for the other two axes. The semi-colons are *compulsory* here. Depending on the way Maple is set up, this might be done using a pop-up dialogue box.

When you have set up objects of these types you can, for example, use the command `distance` to find the distance between two of them or the command `intersection` to find the intersection of two of them or the command `FindAngle` to find the angle between two of them (where appropriate).

You can also use `Equation` to find a cartesian equation for a line, plane or sphere and `center` for the center of a sphere and `coordinates` to find the coordinates of a point.

Use the Maple `Help` to find out more about any of these commands and to find out about the many other commands available in `geom3d`.

In the following example, we first label the points $A(0,1,2)$ and $B(2,3,1)$ and the line AB through A and B. Applying `detail` to AB shows that the direction of the line is $(2,2,-1)$ and the line can be expressed in parametric vector form as

$$\mathbf{x} = \begin{pmatrix}0\\1\\2\end{pmatrix} + t\begin{pmatrix}2\\2\\-1\end{pmatrix}, \quad t \in \mathbb{R}.$$

Then we assign the label P to the plane through $C(4,5,6)$ with normal $(1,1,1)$ and use `detail` to find that P can be described by the cartesian equation

$$x + y + z = 15.$$

Then we assign the label X to the point of intersection of the line AB and the plane P and find that the coordinates of X are $(8,9,-2)$. Next we find that the plane ABC through the three points A, B, C can be described by the cartesian equation

$$12x - 12y + 12 = 0.$$

Finally, we find the sphere S passing through A, C, X and $Y(0,7,4)$ this time giving the coordinates axes as x, y and z and calling its centre Q, and see that it has volume 288π, its equation is

$$5 + x^2 + y^2 + z^2 - 8x - 10y = 0,$$

and so on.

Notice that we use a colon : to suppress the output of most of the commands because the output would just be an echo of assigned names.

```
 with(geom3d):
Warning, the name polar has been redefined
 point(A,[0,1,2]),point(B,[2,3,1]):
 line(AB,[A,B]):
 detail(AB);
   Warning, assume that the parameter in the parametric equations is _t
Warning, assuming that the names of the axes are _x, _y, and _z
 name of the object: AB
     form of the object: line3d
     equation of the line: [_x = 2*_t, _y = 1+2*_t, _z = 2-_t]
 point(C,[4,5,6]):
 plane(P,[C,[1,1,1]]):
 detail(P);
Warning, assuming that the names of the axes are _x, _y, and _z
 name of the object: P
     form of the object: plane3d
     equation of the plane: -15+_x+_y+_z = 0
 intersection(X,AB,P):
 detail(X);
 name of the object: X
     form of the object: point3d
     coordinates of the point: [8, 9, -2]
 plane(ABC,[A,B,C]):
 detail(ABC);
Warning, assuming that the names of the axes are _x, _y, and _z
 name of the object: ABC
     form of the object: plane3d
     equation of the plane: 12+12*_x-12*_y = 0
 point(Y,[0,7,4]):
 sphere(S,[A,C,X,Y],[x,y,z],centername=Q):
 detail(S);
 name of the object: S
     form of the object: sphere3d
     name of center: Q
     coordinates of center: [4,5,0]
     radius of sphere: 6
     surface area of sphere: 144*Pi
     volume of sphere: 288*Pi
     equation of the sphere: 5+x^2+y^2+z^2-8*x-10*y = 0
```

2.14 Partial Derivatives

If `f` is an *expression* which contains several unknowns (one of which is `x`) then the result of the command

```
diff(f,x);
```

is the *partial* derivative $\partial f/\partial x$. Similarly, the command

```
diff(f,x,x);
```

gives you the second order partial derivative with respect to `x`. If `f` is an expression in unknowns `x` and `y` (and maybe other unknowns as well), the mixed second order partial derivative $\partial^2 f/\partial y \partial x$ is given by

```
diff(f,x,y);
```

If `f` is a *function*, then you need to use either of the two methods in section 2.4.2. For example, if `f` is a function of two variables, then the first method becomes

```
diff(f(x,y),y);
```

In the second method, use the `D` operator followed by numbers in square brackets to indicate the variable with respect to which you are differentiating. For example, if `f` has been defined as a function of two variabels then

```
D[2](f);
```

gives you the partial derivative with respect to the second variable and

```
D[1,2](f);
```

gives you function of two variables given by the mixed second order partial derivative $\partial^2 f(x,y)/\partial y \partial x$ and

```
D[1,2](f)(3,-2);
```

gives the value of that mixed derivative evaluated at the point $(3,-2)$.

Note: All derivatives in Maple are partial derivatives and this is the reason why Maple insists on a second argument for the `diff` command.

2.15 Ordinary Differential Equations.

The procedure `dsolve` is used to solve ordinary differential equations (ODEs). You enter the ODE as an equation involving the function, say `y(x)`, differentiated appropriately with respect to `x`. For example to solve (i.e. find the general solution of) $y' + y = \sin x + x$ you would use

```
dsolve( diff(y(x),x)+y(x)=sin(x)+x, y(x) );
```

and for the more complicated equation

$$xy' + y(x^2 + \ln y) = 0$$

you would use

```
dsolve( x*diff(y(x),x)+y(x)*(x^2+ln(y(x)))=0, y(x) );
```

and to solve

$$y'' - 2y' - 3y = 2x$$

you would use

```
dsolve( diff(y(x),x,x)-2*diff(y(x),x)-3*y(x)=2*x, y(x) );
```

In the resulting general solutions, arbitrary constants are shown as `_C1`, `_C2` etc.

NOTES:

1. The dependent variable `y(x)` must always be written completely as `y(x)` and never just as `y` — Maple will give you a warning message if you mix the two. The derivatives must be written out explicitly and you must say what variable you are solving for. In some cases Maple will not solve for this variable explicitly — it will only give you the implicit definition of the solution (just as you would if you were doing it yourself). Also, you may of course use variables other than `x` and `y`.

2. It is important to be aware that the output of dsolve is an *equation*. It does *not* define $y(x)$ as a function of x. If you want to use the expression on the right hand side of the output equation, you must extract it using the command `rhs(%)`.

Often it is useful to have the differential equation in pieces so that you can manipulate it outside the `dsolve` or use it more than once. For example, the second order ODE given above could have been solved via the following:

```
leftside := diff(y(x),x,x) - 2*diff(y(x),x) - 3*y(x);
ODE := leftside = 2*x;
dsolve( ODE,y(x) );
```

which is perhaps more readable and more useful if you want to go on to find a particular solution subject to some initial conditions. *Resist the temptation to use the names* `left` *and* `lhs` *in this context: they are Maple reserved words.*

Solving ODEs subject to initial conditions is treated as solving a *set* of equations, some of which are simply the initial conditions. For example, the condition $y(0) = 3$ is simply written as `y(0)=3`. When there are derivatives in the initial conditions, you must use the `D` operator to differentiate so that you can evaluate at the same time. For example, the condition $y'(0) = 1$ is entered as `D(y)(0)=1`. Notice that this is the derivative of `y` evaluated at `0`. (It is not `D(y(0))`, which is just 0 because `y(0)` is a constant.)

To solve the equation `ODE` subject to the initial conditions $y(0) = a$, $y'(0) = b$, use the command

```
dsolve( {ODE,y(0)=a,D(y)(0)=b}, y(x) );
```

Note the curly brackets `{ }` which indicate that we are giving Maple a *set* of equations.

There are options to solve ODEs using series and other methods. Use Maple's Help to get more details.

2.16 Taylor Series.

Taylor series can be found using Maple via the `taylor` procedure. For example,

```
taylor( exp(x), x=0, 5 );
```

will give you first five terms (i.e. the constant through to the term of degree 4) in the well known Taylor series for e^x. The display will end with `O(x^5)` to show you that the remainder is of order x^5.

For series about other points, simply put a different value after the `x=`. For example, the command

```
s := taylor( exp(-x)*sin(x), x=Pi/2, 8 );
```

will find the first eight terms (i.e. up to and including the term of degree 7) in the Taylor series about the point $x = \frac{\pi}{2}$ (i.e. in powers of $(x - \frac{\pi}{2})$) for the the function $e^{-x} \sin x$ and it will assign the result to the variable `s`.

Note that Taylor series are *not* polynomials and cannot be treated as such, hence there is a procedure to convert a Taylor series into the corresponding Taylor polynomial. For example, to convert the Taylor series that is in the variable `s` above into a degree 7 polynomial in powers of $(x - \frac{\pi}{2})$ use

```
    convert(s,polynom);
```

2.17 Discrete Mathematics.

Maple provides a number of procedures which will be useful to students in the first year Discrete Mathematics courses. In this section we only give hints as to what is available. See also an example of a recursively defined function in section 2.21.1.

2.17.1 Greatest Common Divisors

The procedures `igcd` and `ilcm` are used for finding the greatest common divisors and least common multiples of integers (the `i` in these words stands for integer). For example

```
    igcd(234,57);
```

gives an answer of 3 .

You can also find numbers x and y satsifying $ax + by = \gcd(a, b)$. Use the command

```
    igcdex(a,b,x,y);
```

where `x` and `y` must be unassigned variables. This procedure will calculate the gcd and assign the appropriate values of x and y to the variables `x` and `y`. You can then see these values by entering `x;` and `y;`. The `ex` at the end of the `igcdex` command name is there since Maple is using the *extended* Euclidean algorithm.

The variables `x` and `y` used in this command must be *unassigned*. If necessary, put them in forward-quotes to remove any previous assignment, as in

```
    igcdex(a,b,'x','y');
```

2.17.2 Modular Arithmetic

There are a number of procedures for working with modular arithmetic.

`a mod m`	positive remainder when a is divided by m
`Power(a,b) mod m`	a^b with all arithmetic done mod m
`a^(-1) mod m`	inverse of a mod m
etc	

Note the use of the inert procedure `Power` rather than the usual `a^b`. The `Power` form does not evaluate until the `mod` part is reached. If you used `a^b mod m` then Maple would first compute a^b and afterwards reduce the answer mod m , whereas `Power(a,b) mod m` will do *all* the calculations mod m and this makes the calculation much faster.

2.17.3 Set Algebra

There are procedures `union`, `intersection` and `minus` for doing set algebra. Use Maple's Help to check the details. Note that there are also functions `member` and `subset`, which return *true* or *false* as appropriate.

2.17.4 Solving Recurrence Relations

Recurrence relations (or difference equations) can be solved by using the command `rsolve`.

When solving for a sequence $\{a_n\}$, write `a(n)` for a_n, `a(n-1)` for a_{n-1}, etc. For example, to find the general solution of the Fibonacci recurrence $a_n = a_{n-1} + a_{n-2}$, use

```
    rsolve( a(n)=a(n-1)+a(n-2), a(n) );
```

The `a(n)` after the comma tells Maple which variable to solve for. Note that `a(n)` must be written out in full, do not just put `a`.

Solving a recurrence relation subject to initial conditions is treated as solving a *set* of equations, some of which are the initial conditions. For example, to solve the Fibonacci recurrence with the initial conditions $a_0 = 0$ and $a_1 = 1$ (hence giving the Fibonacci numbers) you can use

```
   reln := a(n) = a(n-1) + a(n-2);
   fibn := rsolve( {reln,a(0)=0,a(1)=1}, a(n) );
```

Note the use of curly brackets here to denote a *set*.

The solution to a recurrence will often often involve messy expressions (as in the Fibonacci example). You may need to use the simplification commands described in section 2.3.3. For example, after carrying out the above Fibonacci example try

```
   subs( n=7, fibn );
   radsimp( %, 'ratdenom' );
```

You will get a complicated expression after the first command which simplifies to give 13 as the value of the seventh Fibonacci number. In other cases you may need to use `evalf` to get an approximate floating point answer.

You can also use loops to calculate the values of a recurrance relation: see section 2.20.1 for an example.

2.18 Assuming properties.

The current version of Maple has a feature that enables you to assume that variables have certain properties. (Early versions did not have this feature and consequently sometimes made mistakes.)

For example, Maple cannot give you a value for $\lim_{x\to\infty} e^{-ax}$ unless it knows whether a is positive, negative or zero. You can tell Maple that a is positive by giving the command

```
    assume(a,positive);
```

before giving the command to evaluate the limit.

Maple sometimes needs an assumption to be made when you might not think it is necessary. This is because Maple usually assumes that variables take complex number values and this may cause problems. If Maple does not evaluate something when you think it should, it might be useful for you to `assume` that the variables are `real`, `integer` or `positive`.

The syntax for assuming properties for variables is

`assume(`*variable*`,`*property*`);`

where `property` can be one of many things including `complex`, `real`, `positive` and `integer`. To see a full list of the possible properties, look at the help entry for `assume`.

If a range of values is being assumed, you can use a single inequality of the type

`assume(`*variable*`>`*value*`);`

Only one assumption at a time can be made using `assume`. If you use `assume` twice on the same variable then the first assumption is lost. However, you can make additional assumptions with the command `additionally`. It is used in the same way as `assume`.

To find out what assumptions have been made about a variable, use

```
about(variable);
```

and Maple will display all assumptions about that variable.

For example, to assume that the variable `a` is a positive integer and check the assumptions on `a` and then evaluate the limit of e^{-ax} as $x \to \infty$ you could say

```
assume(a,integer);
additionally(a>0);
about(a);
limit( exp(-a*x), x=infinity );
```

To show that a variable has had an assumption made about it, Maple puts a `~` after the variable name. For example, the variable `a` with an assumption having been made about it will always be displayed and printed as `a~`. This sometimes looks like a `-`, so take care when reading the Maple output. There is no need for you to type the `~` when using the variable.

An assumption about a variable can be removed by unassigning the variable in any of the ways described in section 2.2.3.

There are other details about making assumptions, such as the `assuming` command, and you can check the relevant help entries if you want to know more.

Exercise What happens when you enter

```
sqrt(x^2) assuming( x,negative );
about(x);
```

2.19 Conditions and the `if` – `then` construction.

Suppose that you want a sequence of commands to be carried out if and only if a certain condition is satisfied. This often happens when you are writing the definition of a procedure. You can achieve the desired effect with an `if` ... `then` construction, but first you need to know how you can express conditions in Maple.

2.19.1 Conditions

Conditions are equations or inequalitities or other expressions that are either `true` or `false`. In stating conditions, you can use `=`, `>`, `>=`, `<`, `<=` and the symbol `<>` which means 'not equal'. Conditions can be combined by using the logical `or`, `and` and `not`. For example

```
if x>3 or y<=2 then ...
if a>b and not (c>d) then ...
if not isprime(x) then ...
```

In the last example, `isprime` is a Maple procedure which gives the value *true* if `x` is a prime number and *false* otherwise. There are other procedures of this sort — use Maple's Help Browser to find them.

Conditions are also called **Boolean expressions**, and if you use them in an `if` statement, for example, and an error occurs in the condition, Maple will refer to the condition as a boolean, which should give you a clue as to where the error is.

2.19.2 The `if` – `then` construction

As an example, the following commands are a procedure to find the absolute value of a real number.

```
absval := proc(x)
  if evalf(x)>=0 then
    x;
  else
    -x;
  end if;
end proc;
```

The general shape of the `if` command is something like

```
if condition1 then
     commseq1
elif condition2 then
     commseq2
elif condition3 then
     commseq3
else commseq4
end if:
```

where *commseq1* etc are sequences of commands (each finishing as usual with `:` or `;`) and *condition1* etc are conditions that are tested for their truth. The action of such an `if` command is to test if *condition1* is true and if it is true then execute (carry out) the commands in *commseq1*, while if it is not true then Maple tests *condition2*. If this is true then Maple executes *commseq2*, and if it is not true then Maple tests *condition3*. If this is true then Maple executes *commseq3*, and if it is not true then Maple executes *commseq4*.

An `if` command needs the first `if` and `then` and the final `end if`, but the other parts are optional, depending on what is required. We have seen this above and there are further examples in section 2.21.1.

You do not have to put `:` or `;` after the `then` (though it will not matter if you do), but you MUST put one after the `end if`.

2.20 Looping with `for` and `while`.

Suppose that you want to execute a set of Maple commands several times, changing the value of one variable n at each repetition. This is called creating a **loop**, and is very common in scientific programming.

The way you create the loop depends on whether you know in advance exactly how may times you want to repeat the commands or not. If you know that you want to repeat the commands 100 times then you can use a construction of the type '`for n from 1 to 100 do ...`'. If you do not know how many repetitions you want to make then you will have to tell Maple to keep repeating until some condition is no longer satisfied, using a construction of the type '`while ... do ...`'.

As an example of the first type of situation, consider the following piece of Maple.

```
for n from 0 by 2 to 4 do
    print(`(a+b) to the power`,n,`equals`,expand((a+b)^n));
end do:
```

The resulting display will be

$$(a+b) \text{ to the power, } 0, \text{ equals, } 1$$
$$(a+b) \text{ to the power, } 2, \text{ equals, } a^2+2ab+b^2$$
$$(a+b) \text{ to the power, } 4, \text{ equals, } a^4+4a^3b+6a^2b^2+4ab^3+b^4$$

This display shows the expansion of $(a+b)^n$ for each of the values `n = 0, 2` and `4`. We terminate this 'for–loop' with `end do:`, using a `:` rather than a `;`, so that any intermediate results will not be printed on the screen. We use the `print` procedure to force display of the things that we want to see. In the argument to the `print` procedure, text contained between back-quotes will be echoed in the printout and variables which occur outside back-quotes will have their values printed.

It is possible to print things nicely using a specified **format**. This is done using the `printf` procedure which works the same as it does in the C programming language. We will not go into any details here — use Maple's Help if you want more details.

As an example of a situation where you need to use `while`, here is a loop that finds the first value of n for which the sum

$$\sum_{k=1}^{n} \frac{1}{k}$$

is greater than 5.

```
total := 0; # Note that  sum  is a reserved word
for n while total<5 do
      total := total+1/n;
end do:
n;
```

The `for n` part tells Maple that n is the variable which should be increased at each repetition. Since no starting value or increment are specified, Maple assumes that you want to start at $n = 1$ and increase n by 1 at each repetition. Again note the full colon `:` at the end of the loop rather than a semicolon `;` — this is to stop the display which would otherwise be produced at the end of each loop.

The most general possible form of Maple's looping construction is quite complicated, but you can leave out the parts you do not need. The general form is

`for` *variable* `from` *start* `by` *step* `to` *finish* `while` *condition* `do`
command sequence
`end do:`

The *command sequence* is called the **body** of the loop.

You do NOT need a `;` or `:` after the `do` (though it does not matter if you put one there), but you MUST have a `:` or `;` after the `end do`.

The body of the loop is repeated once for each value of the 'index' variable, say n. Initially n is given the value of *start*. If this value is less than or equal to that of *finish* and