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.