Mathematica

Mathematica is an interactive mathematical programming language. Interactive means that commands may be entered one line at a time so that new calculations can be made after the old calculations have been examined. Mathematica does symbolic, Boolean and numerical calculations as well as plotting. It also has the ability to match patterns and many of the advanced programming routines take advantage of this feature.

Mathematica comes with a library consisting of several additional packages and routines that need to be loaded by the user when he needs them. There are several additional packages from the Wolfram Company (Mathematica's creator) and third party sources called Mathematica Notebooks. These Notebooks serve as templates. They contain various additional routines and functions programmed in Mathematica code. The Notebooks tie into the Mathematica kernel to complete their calculations.

The Mathematica program has essentially two parts: (1) a Kernel - a machine independent program where the input from the Front End is evaluated and a (2) Front End or Notebook - an interface with the particular platform being used, eg., PC or Mac. The Front End takes word processing text and comments independently of the Kernel.

Mathematica is available to students at the special student price of $125 – 150 at the University Book Store. This student version is supposedly fully configured and contains all the packages. The usual limitation on a student version is the so-called stack size or the amount of volatile memory that the kernel will draw for storing information as the commands are executed. For example, student Mathematica accepts a very large matrix but only be able to execute a routine on a very small matrix.

Notation for this Document

A scheme like

Edit > Default Output Type >Traditional Form

gives a route for finding Traditional Form from the menu. Words written in `CourierFont` are Mathematica statements.

Opening Mathematica and Executing a Command

Open Mathematica by double clicking on its `Notebook` icon. A connection between the Notebook and to the so–called `Default Kernel` for the `Notebook` is made using a program called MathLink when you execute the first command needing the kernel. If you wish to activate the kernel immediately, execute a simple statement like 2 + 2. Each cell is attached to a kernel so that a `Notebook` could have links to several Kernels at the same time. You can set the evaluator of your `Notebook` to any available Kernel. You can see what your default Kernel is and what Kernels are available in the `Kernel Menu`. Incidentally, `MathLink` can be programmed to link Mathematica to many different sorts of devices and programs.

Occasionally, Mathematica fails to make an automatic connection to a Kernel. The error message is "`low level error`" which indicates an improper log off or an improper setting on the `Notebook`. To deal with this error, make sure all kernels are shut down and the Default Kernel is set to an available Kernel. The next execution in the Front End should automatically make the connection to the Kernel.

You can tell that execution of a cell is occurring by the highlighted cell bar. To get execution, press enter on the lower right hand side of your keyboard or `Shift Return` (Mac or PC) or `Shift Enter` (the Enter/Carriage Return on the main part of the keyboard for a PC). Many cells can be executed successively from top to bottom by highlighting the cell bars and executing. Shift option Return (evaluation of next input) fits a new execution on top of a running execution. The running execution will continue either be interrupted at a suitable point or the new execution will be fit into idle process times. Other keystrokes specific to the front end can be obtained from the Help file item for the particular front end. For example, setting up a new cell below the current cell with the same format as the current cell and executing a cell with the cursor going to the next input cell after execution can be done with keystrokes. Note the `Enter/Return` key is merely a carriage return to start a new line in the same cell while and is more or less inert as far as execution is concerned. `Alternate (or Option) Return/Enter` produces a new cell with the same format type at the first available place below the current cell.

All Mathematica keyboard commands available for a particular front end are listed under `Other Information` in the Help menu.

Saving a Mathematica Session

Mathematica occasionally locks up. For this reason, you should save your notebook as soon as possible. Mathematica files, especially those with graphics and formatted text, can be quite large and sometimes too large to fit on a floppy disk. It is best to have a ZIP disk or FLASH MEMORY.

If you press `Control s` the Mathematica session will be saved to a designated site. The first time you save, Mathematica will present a dialogue box asking for the destination of the saved session and for the name. There are various options for the save command available in the File Menu. Quitting and re-opening a Mathematica document gives only the typescript that was present when the Mathematica file was last saved. You will have to re–execute the text commands to arrive at the point indicated by your saved text file. Mathematica will retain cell information and might even retain the information on what kernel was used to evaluate a particular cell. So you may restore Mathematica's internal position to that of the typescript simply by selecting all cells with `Control A` and then executing with <ENTER>.

You may save various things like Tables of calculated values, symbols, equations, formulae, etc. with the `Save, DumpSave,` or the Put command followed by what you want to save. Further information on what survives file transfers will be given later. Using `DumpSave` and `DumpGet`, it is possible to save the entire internal state of Mathematica so that when you reopen you will be at exactly the same place you were when left off. Mathematica will also save automatically in the `Edit > Preferences > NotebookAutoSave`.

Finally, you may compress your Mathematica document by closing very large cells or groups of cells in the `Cell Menu > Cell Grouping > … . Closed cells` retain all information necessary to regenerate themselves when the cells are re-opened. Closing cells is useful for writing reports on top of Notebooks that contain calculations.

Documents may usually be printed after re–opening with no further work. For massive Mathematica documents with graphics a re–execution may be necessary. Closing large cells may be necessary to fit your Mathematica document onto your floppy disk. Re–execute as described in the preceding paragraphs with `Control Shift A` and ENTER. Here your program must be readable from top to bottom without a hitch in order for the re-execution to take place.

When Mathematica locks up, you can generally perform a save command before you exit. However, you can also try to release the lock up by aborting a calculation. Abort the calculation in the Kernel Menu, A more dramatic step is cutting the connection between the FrontEnd and the Kernel. This will generally clear a lockup but at the expense of knocking everything out of the memory. This is called a "long jump". To execute a long jump use `Control Shift Period` or`.` You will need to re-execute as explained in the previous paragraph.

As a last resort to relieve a lock up with restarting the computer (which is done with the restart button on the processor box), try `Alternate-Window Escape` whatever the computer operator system requires. To avoid lock ups, think about using `TimeConstrained` when executing a function for the first time. Here a limit can be set on the amount of processor time used for the `TimeConstrained` function.

Recalculating and Clearing Erroneous Input

Sometimes you have made several experiments to reach a final goal. It is good to go back and check that your calculations are clean. The invisible internal state sometimes will hinder the checking due to unwanted and hidden assignments. One way of checking is saving your Notebook, quitting the kernel from the Kernel Menu, re–executing your saved text automatically using Select All from the menu or the keyboard and ENTER.

Executing the command `Quit[]` will also close down the kernel. Executing `Clear["Global`*"]` will delete all previous global definitions. Note that Mathematica designates all user defined terms by the designation `Global``.

Occasionally, commands seem to be evaluating incorrectly. This is almost always due to incorrect input. When this happens, the best remedy is shutting down the kernel.

In order to insure a smooth recalculation, make sure that all steps are listed in order. For example, putting an assignment at a later point in the notebook from where it is used or loading a package after a function in the package is used will result in the entry of an undefined and unerasable term in the name file.

Notebooks

Notebook is the name for a Mathematica document. Notebooks are generated by a program called the Front End or Mathematica 5.2 with as a dodecahedron icon The Front End is dependent on the computer platform being used although there is much uniformity in the different front ends. Typing can be done in a Mathematica Notebook more or less like a word processor. Formulae can be entered either using word processing keystrokes or using a code that resembles the mathematical typesetting program TeX. You can examine the textual code used to generate any cell by clicking on the cell bar and using the menu `Format > Show Expression`. You can toggle out of the show expression in the same way.

Mathematica will accept text from another program using the usual copy and paste technique and will accept pictures from other programs like paint programs or equation makers using special techniques discussed below. Mathematica has its own type setting and equation making capacity and so importing files is necessary only for previously prepared work. Mathematica output will set up in mathematical notation and this copied into text lines. Mathematica also displays its own graphical output like Tables and plots. Tables, matrices, etc. can be aligned with commands like `TableForm` or `MatrixForm`. There are many devices in Mathematica (for example `Hold` or `HoldForm`, `SequenceForm`) to hold expressions in unevaluated form for display purposes.

To get your output in traditional Mathematical form, select `Cell Menu > Default Output Form > Traditional Form`. A check will appear next to the selected item.

There are many options for formatting your notebook. You can examine the options that are being used in your current notebook either in the preference file or using the usual command FullOptions[notebook name, name of option]. To find out what Mathematica is calling the currently open notebooks (called `NotebookObjects` by Mathematica) execute `Notebooks[]`. If you want to change some option that needs to be changed in each session, you might set up an init.m file. To set up an init.m file, see the section on Initialization.

Cells

Mathematica Notebooks are organized into Cells. The size of a Cell and what is in it will be indicated by the box outlines on the right margins of the notebook called Cell Brackets. There are many types of cells: Input, Text, Graphics, Unevaluable (for examples that you want to keep but do not want to evaluate when you evaluate programs that the examples tested), Initialization (for things that you want to load automatically), Locked (for routines you do not want to inadvertently change or delete), Fixed Height, Closed Group, `Evaluation Group`, or `Running or Evaluating Cells`. The cell brackets indicate the type of Cell. For example, an `Evaluating Cell` (a cell the kernel is currently processing) is denoted by a double Cell Bracket that looks something like]]. The Cell Bars will not print unless specifically requested. There are also many commands for opening or compressing cells, merging cells or dividing cells, etc. that can be found under the Cell Menu. You can change a Cell type by using the menu Style > Cell Style (or keyboard equivalents mentioned below) or by using a special pop-up menu that appears when the Ruler is activated. Cells containing unneeded calculations can be collapsed or in Mathematica terminology closed. Collapsed cells can be re–opened and the material therein is then available for copying, recalculating, or pasting. Cells coming under one of the Header forms (discussed later) on the Style menu can be collapsed as a unit.

Generally, you should reserve various types of cells for various types of input in case a re-execution becomes necessary. You can make almost any change you want in a cell: dividing or merging cells, changing from input to text, etc. See the Cell Menu for this. Note commands on the menu that can be used to initialize the cells. For example, `alt 7` gives a text cell where the wrap feature of a word processor is operational, `alt 9` gives an input cell for the kernel, and `alt 1, 2, 3` give various title formats. Look under the Style Menu for the various types of styles. You can determine the style of a cell by highlighting the cell bar and invoking `Edit > Options Inspector.` You will see the style of the cell near the end of the cell information.. If you know how the Mathematica formatting commands work, the Options Inspector allows you to get your output, input or text into a desired form. For example, an output cell will be labeled near the end with Output". Changing "Output" to "Text" will set the cell to a Text cell, which can be edited. Note that output cells cannot be edited. Trying to change an output cell will result in a copy of the cell as an input cell directly below the output.

Style Sheets

The `Format > Style Sheet` gives various template styles. The `Format > Style Sheet >Default Style` is what appears when you open a new notebook from the File Menu. You can make a template document for yourself by selecting the Default Style and editing the default styles just as you would a regular document. Saving the Default Style document will set up the currently saved version as the new Default Style.

Style Sheets (Templates) are actually kept in notebooks inside `Mathematica > System > Front End > Style Sheets.` You can change the name of any of the supplied style sheets and make it into the `Default.nb` sheet. Be sure to keep a backup copy of the `Default.nb` notebook supplied by Mathematica either by changing its name or moving it to a new location. Alternately, you can create a your own style. For example, if you want the TEXT cells to appear in Times Font, you need to change the font in the TEXT cell in the Style Sheet to Times using the regular Mathematica menu. Note that Mathematica allows you to have a several different choices for each item. For example, you can choose Red as the color of the output on the screen, Green for presentation output and black as printout since red printout on a black and white printer may be unsatisfactory. Using Copy and Paste , you can also import into your Style Sheet features that you like from any of the other Style Sheets. For example, one Style Sheet has a feature for `NumberedEquations.` This will automatically assign a consecutive number to each designated equation. To get this feature or any other such feature in your default style sheet or any other style sheet, you just need to copy this item from the `Classic Style Sheet` and place it into your Default Style Sheet. Be sure to keep a copy of anything you are editing at least until you are satisfied with the operation of the new item.

You can also change the style of your document using `Edit > Preferences`. There are a very large number of choices many relating to type setting features. Some of the categories are best left at their defaults until a specific need is identified. One change for example could be the removal of the Monkey Bong every time a cell is executed. The changes of the `Default Style` described in the previous paragraphs automatically will be added to the `Preference File`.

A list of author tools can be found in File > Palettes > Open Author Tools. These tools are useful for creating indices, table of contents, and bilateral cells. Bilateral cells are used extensively in the Mathematica manuals and have text on one side of a cell and Mathematica code or output on the other side of the cell.

Restoring Mathematica to its Installation Configuration

To restore Mathematica to its original factory configuration, hold down the option or alternate key when opening Mathematica.

Cell Hierarchy

Cells are arranged automatically in hierarchal fashion like an outline. For example, a cell with a large header, like a `alt 2` header, will have a cell bar connecting it and all subsequent cells with a lower hierarchal order. In the Mathematica Default Style, cells generated by cmd 3-6 have various types of headers, `alt 7` is a text cell, cmd 8 is a graphics command cell and its attached graphics, `alt 9` is an input cell. Output cells are editable only by a special process and are automatically attached to the input cells that generates them. Trying to edit an output cell without a special process will result in its echo to an input cell. So all cells of type `alt 3 - 9` written below a header cell of type `alt 2` will be attached to the `alt 2` cell with a cell bar. The same is true for `alt 3` headers, etc. An `alt 1` header below `alt 2` header will not be attached to the alt 2 header. Note that the grouping of the cells can be changed by `Cell > Divide or Cell > Cell Grouping > Group`. Also groupings can be set up after cells lower cmd cells have been typed by inserting the header cells at the appropriate points. The grouping is useful: cells can be hidden under a cell of higher hierarchy using Cell > Cell Grouping > Close All Subgroups. The closed cells can be reopened without any loss of information using `Cell > Cell Grouping > Open All Subgroups`. The commands `Cell > Groupings > Open/Close` also do the same job. A double click of the mouse will also open cells. In certain cases. the cell bars must be highlighted before Close or Open is invoked. The last `Open/Close` is a toggle action, i.e., selecting open gives closed and selecting closed give.

Note that you can group cells or divide cells using menu commands. Only cells of the same type can be merged. Some instructions will be given later for editing the style documents.

Cell Tags

The information about the type of cell, like Input, Output, Title, Section, is called a `Cell Tag`. `Cell Tags` different from the ones given in the Style Menu can be created in the notebook. You can use the additional cell tags to post automatically numbers on the cells or to reference the cells for quick scrolling. All Mathematica cells receive a cell number. Other Cell Tags that number equations, sections, etc., are discussed Style Sheets. Cell Tags can also be used as a device for a finding line, for example, a line with important formula. For example, create a formula and highlight the cell bar and type in the name Wavelet Generator in `Add/Remove Cell Tag` dialogue. Typing Wavelet Generator on a command line and then accessing `Find in Cell Tags` will bring you to the Cell labeled Wavelet Generator.

Automatic Numbering and Reference of Equations and Figures

In certain style sheets, the cell styles NumberedEquation, NumberedTable, and NumberedFigure, etc. are available,. These create a dynamical numbering of the objects. This allows the insertion new objects at arbitrary places with the ensuing updates of the numbering. You can make a reference to a numbered object in the body of the text. You need to give the object a cell tag (cf. Cell Tags). For example, give the object the tag eq1. When you want to reference eq1, set your cursor at the insertion point and access the `Input >Create Automatic Numbering Object` menu. In the dialogue box, write eq1. The counter reference will appear at the insertion point and automatically be updated whenever the number on the cell eq1 is updated.

Automatic numbering for any cell style can also by obtained without a tag using the Input >Create Automatic Numbering Object. For examples, section headings can be automatically numbered starting with a certain cell.

Displaying Input and Output

You can control the way Mathematica displays its input and output. There are four pre-programmed options called `InputForm`, `Output`, `StandardForm`, and `TraditionalForm`. `TraditionalForm` is set in mathematical format, standard form is a form peculiar to Mathematica to prevent mistakes in interpretation. Many failures in execution results from using Input in TraditionalForm since extra parentheses important for Mathematica are suppressed. For example, parentheses and braces have special meaning in Mathematica. In Traditional Form braces are replaced by parentheses while in standard form, the distinction between braces and parentheses is maintained. Input Form is a text form usually with a non proportional font designed to provide Mathematica with the clearest picture of what it needs to do. Output form is an inert form that cannot be modified. Attempts to modify output will result in the modified line being reproduced as an input line below the modified form.

Expressions set in `TraditionalForm` generally have a so-called `Wrapper` around the actual mathematical expression. The wrapper is a set of Mathematica commands that give the formatting structure. Invoking `ToExpression` before using the formatted expression will remove the wrapper. If a formatted expression is copied an pasted into a unformatted space, the wrapper together with the expression will often appear as plain TEXT together with several formatting commands. To avoid this string of formatting commands, you should either 1) paste the formatted expression in a `Box` obtained using `Control (` (i.e., `Control Shift 9`) or 2) highlight the string of pasted TEXT and type `Control (.` To escape from a Box or use `Control - Spacebar` or `Control . (. = Period.)`

The Kernel

The mathematical engine that stores and evaluates the input is called the Kernel. The Front End automatically sends the input to its `Default Kernel`. Each Cell in a notebook has its own kernel usually the the so–called Default Kernel. If you do not specify a kernel, the Front End will automatically route all calculations to the Default Kernel. If you are using Mathematica on your own personal computer, the Default Kernel will be the so-called Local kernel, which is the kernel on you own hard disk. Machines in a computer laboratory may have a different default kernel, usually a kernel on a powerful remote work station with very large amounts of RAM. Here the Front End serves as a communications software program for the remote Kernel. Since each cell can have its own kernel, you can choose a different kernel, for example, a more powerful kernel at a remote site to do some especially difficult computation. An evaluator for a single cell can be chosen from the Kernel menu. Using more than one kernel may be advantageous in that the different kernels will read only their own inputs. So if x = 4 in the first kernel, it could be taken to be a symbol in a second kernel. The selection `Evaluate in a Subsession` in the Kernel menu refers to the simultaneous evaluation of different cells in different kernels

If your Kernel or `FrontEnd` runs out of memory, an alarm bell sounds and the computer locks up. So you should set your memory allocation sufficiently large to prevent this. If Mathematica locks up due to insufficient memory, you will hear a fire alarm bell sound. You may need to use a forced quit of the Mathematica and this will cause a loss of your data. Sometimes, your clicking with the mouse on an area off the active window will shut the alarm down and allow you to re-entry and execute a save.

To attach a new kernel to your notebook, go to the menu `Kernel > Kernel Configuration Options`. You need to see the desired kernel in the dialogue boxes as we traverse the options. Also certain communication tools need to be present in the system folder. These are supplied with Mathematica. The program that does the communication is called `MathLink`. This program is supplied with Mathematica and its `Help file` is one line as part of the regular Mathematica help file.

Input and Output Numbers

If you enter a certain Mathematica command, Mathematica will immediately append a line number to the input, e.g.. `Input[79]`means more or less that this is the 79th Input in this session. The output for this line will be referenced as `Output[79]`. If you wish to suppress a certain output as, for example, the output from `Input[79]` which is too complicated to read anyway, type a semicolon ; after the input. Mathematica will not display Output[79] but will have it available for future use. `Output[79]` is a Mathematica assignment just like all the other assignments made in a session. If you wish to display the previous input line (for possible editing) use `Alt l` (small el). The previous output line is also available with `Alt Shift L`. If you edit the previous Input line and re–enter it, Mathematica will give the previous Input line a new Input number.

If you do not want to see the Input/Output notices you can prevent these from appearing by checking the appropriate item in the File Menu. Mathematica will keep track of the input and output numbers but not display them. Previous lines can be accessed with the Mathematica `dittos %, %% or %[n]` which means n from the last.

Notebook Management

You can clean up unwanted input or output by using the using the `Clear` in the `Edit Menu` or its keyboard equivalent `Control Shift X` or the `Delete key` on the keypad. Highlight what you want to discard or click with the mouse on the right hand cell box if you wish to discard the whole cell and then access the `Clear` or type `Control Shift X`. Other standard editing can be done in a similar fashion.

It is very important to note that this EDITING DOES NOT REMOVE THE INPUT in the Mathematica kernel. For example, the function f[x] may be entirely absent from the Notebook but still be present in the kernel. To remove f[x] you need to use something `like Clear, ClearAll, Remove, or Clear["Global`*"]`. Even removing f[x] leaves some of the properties of f[x] called Attributes behind and sometimes these have to be removed separately. If things are really behaving in an erratic fashion, go to the kernel menu and shut down the kernel. This will clear absolutely everything out of the memory but leave the notebook alone for re-execution.

Mathematica also has a Find feature and other helpful devices to allow you to access your previous input. You can designate cells or groups of cells by a name called a Keyword for reference purposes. There is a standard undo command (which restores the program to the status it had right after the last deletion or the last save) for Mathematica but this undo feature seems to relate to the word-processing and not to the mathematical calculations.

Also make sure that your output prints on one page rather than on two or three side-by-side pages. Do this by using the `Edit>Preferences > Action Menu` to set the screen width to 78. Also make sure that the `Word Wrap` has been activated so that Mathematica will not print on two or 3 side by side pages. To activate the `Word Wrap` for the entire Mathematica notebook, open the `Edit Style` work sheet in the manner described in the preceding section on "Cells". Then highlight the cell bars where you wish to invoke the `Word Wrap`. Then choose `Style > Alignment > Word Wrap`. Upon closing the `Edit Style` work sheet, you will find that the `Word Wrap` is engaged in all the cell types that you have chosen.

The Notebook that you use to solve a problem can also be presented as a final report. Your professors may be interested in your Mathematica code but your clients certainly will not be. You can create a professional looking report in the same Notebook you used for your calculations. To do this add a narrative description of your problem together with a narrative of your solution. Include only as much of your calculations as are needed to support the narrative. Also include any equations that you might need. These can be imported into Mathematica as described in the section "Interfaces … ". All cells containing technical Mathematica code should be closed before printing as described in the section "Cells". Now the printed copy of your Notebook will look like a professional report while your personal electronic copy of your Notebook will have all the codes ready for inclusion in some other Notebook or revisions of the current Notebook.

Your Style Sheet allows you to print your Notebook in a style different from the one in which you created the notebook. For example, you may wish to leave some of your input and output lines but you may wish these to be printed in `Traditional Form` rather than `the Input Form` and `Output Form`. On the Default Style Sheet, find the Input Printout and go to the menu `Cell > Convert To > Traditional Form`. You can also do this from the keyboard using `StyleData["Input", "Printout", FontFamily->"Times", FontSize->12]`

Mathematica automatically post page breaks. Sometimes these come at inconvenient places. The placement of page breaks can be controlled in the `Format > Show Page Break` menu. The location of page breaks can be viewed using the File > Printing Settings menu. Page breaks can be controlled manually by adding `PageBreakAbove -> True, PageBreakBelow->True` at the end of Cell.

Note that you can have several notebooks open at the same time. Notebooks that feed into the same kernel will be linked in that any assignment made in one of the notebooks will be carried over to the linked notebook. This can be both beneficial in that you can do side calculation but this can also be detrimental if the two notebooks being executed interfere with each other. However, it you execute each notebook in a separate kernel, there will be no interference.

Fonts

Mathematica 5.2 has a powerful word processing and type setting capabilities and virtually any font can be used. Mathematica will choose a non-proportional font like Courier or Monaco for input and output in Standard Form. `Courier` is the standard IBM typewriter font and the `Monaco` is another non-proportional font. The font will look strange but it will preserve the spacing of the output and make the input and output easier to read. Mathematica 5.0 will preserve the spacing of input and output so the use of a non-proportional font is not necessary for legible input and output. One can set the set the `Cell > Default Input Type`, etc. to anything that is aesthetically pleasing. However, it might be best to have the input and the output in the so-called `Standard Form` since Mathematica has less trouble interpreting input in `Standard Form` and you will have less trouble reading output in Standard Form. For Presentation (cf. this category in the `Edit Style` page), you can request input and output in `Traditional Form`, which is Mathematical type setting form.

<u>Only characters typed from standard keyboard or the standard shifted keyboard should be used.</u> Special characters typed using other combinations of keys or even different fonts often are not reproduced correctly by the printer or by file transfers between platforms. Instead <u>you should use Mathematica's special character sets</u>. These characters are carried in a plain text format that can be read by email programs and all the different Mathematica Front Ends. For example, the Greek character alpha can be typed using the abbreviated form ⋮a⋮ or the long form `\[Alpha]`. Here ⋮ denotes typing the `Escape key`. Virtually every mathematical symbol is found in Mathematica's special character set either in the formal form `\[NameOfSymbol]` and often a simpler abbreviated form. The formal form is always available. For example, the symbol ≈ can be typed as either `\[TildeTilde]` or ⋮ ~~ ⋮. Special symbols typed in the Mathematica character set are usually mathematically active when transferred to input lines and are usually correctly set in mathematical form on output lines. The symbols are always in the correct form on text lines. To find the Mathematica name of a displayed symbol, you can use `FullForm[symbol]`.Every symbol also has a unique number associated with it called its character code. There are some 10,000 symbols including several thousand for oriental languages. To find the character code of a symbol, you can use `ToCharacterCode["the symbol"]`. Also `FromCharacterCode` sets up the symbol.

Advanced users may wish to experiment with Latex (called `TeXForm` in Mathematica). Mathematica will supply the code to set up its output in TeX, which is a mathematical typesetting program. Also certain TeX style sheets are supplied with Mathematica so that the Mathematica TeX can be correctly interpreted by the TeX program. Output from TeXForm can be imported into a TEX compiler in order to create type set quality work.

If Mathematica is displaying gibberish, the Mathematica fonts are not loading correctly. Bring this to the attention of the laboratory supervisor immediately.

Packages, which contain additional functions that you wish to have readily available or make available to others, need to be saved in Text format. You can set up your package in Mathematica notebook format with special symbols and invoke `File > Save as Special > Text`. Mathematica should automatically set up your saved package with all the symbol names write in `FullForm`. Occasionally, even the `FullForm` will not suffice. In this case, one needs to use `ToCharacterForm` and `FromCharacterForm`. This assigns a number to each character. All characters used in printing around the world has a number from 0 to about 10,000 associated with it. Mathematica will recognize the Character code for each character whose font it can find on your computer. Using the character code number together with the `ToCharacterCode` and `FromCharacterCode` allows the characters to survive the translation to and from Text.

Word Processing

Word-processing commands may be found under `Edit > Expression Input`. For example, to actually see a superscript a_2 in an input or text line, use a `Control shift 6`. This execution will leave you in the superscript position. To exit any level of formula making use `control spacebar`. A full set of keyboard strokes can be found in the Help file under `FrontEnd`.

You need to set in line word equations off by themselves. Otherwise, features of word processing, like the word wrap, will not work. Start your in line formulae with `control(` and end them with `control )`.

Initialization

You can create an Initialization Cell for your notebook. This cell can be used to load automatically at startup any set of commands or packages that you need on a regular basis for your work. The initialization cells are named init.m. Several packages have init.m associated with them. Some of the packages have a complete file. When you load this complete file, the init.m file for the package is loaded. The init.m file has a directory in an initialization cell. The directory has all the names of all the functions in subpackages. So the net effect of loading the master file is creating a link to all the commands in all the subpackages. The commands in the subpackages are now available for use.

Mathematica also has a master file called init.m that controls all the preferences. One should keep a copy of this file in a safe place since the original file sometimes becomes corrupted. Mathematica will not work with a corrupted "master"init.m file. If the master init.m file gets corrupted, you can remove the file from its folder. Mathematica will generate a new master init.m minus any of your personalized preferences.

Finally, you can create your own init.m file. This file should be put in the Autoload folder and will supplement the master init.m. For example, Mathematica displays the text of its graphics in Courier font. There does not seem to be a mechanism in the Preference of StyleSheet for changing the Courier font. If you set

$TextStyle={FontFamily->"Times"}

in an initialization cell in your "private" init.m and put the init.m into the Autoload, all the Graphics text will be rendered in Times as a default. You can change the rendering in the notebook to anything else if you wish. When you save a notebook with a initialization cell, mathematica will assist you with the set up.

Conventions

Mathematica has the following conventions:

1) all Mathematica functions and calls begin with a capital letter;
2) Mathematica distinguishes between lower and upper case letters but often will give a spelling error message if two words differ only in capitalization;
3) Spaces and carriage returns are not read except where a space denotes multiplication;
4) Parentheses (…) are used only for grouping;
5) Braces { … } are used for lists, n-tuples, vectors, matrices;
6) Brackets […] are used for function arguments;
7) Double brackets [[…]] are used to access pieces of expressions;
8) Starred parentheses (* … *) are used for comments on execution lines;
9) Many commands have both a long form and an abbreviated form and these can be either postfixed, infixed or prefixed;

10) Expressions that Mathematica cannot evaluate are echoed unchanged (e.g., when a pattern cannot be matched).
11) Formulae and mathematical typesetting are enclosed in invisible boxes starting with Control (and ending with Control).
12) Many special characters can be called out by using the escape key using the "alias" of the character. Virtually every mathematical and technical symbol is available in the special character sets including special alphabets like the Greek and German (Faktur) alphabets and script and double struck Roman characters. For example, the symbol ∂ would be written <escape> pd <escape> and a would be written <escape> a <escape> where <escape> means pressing the escape key. In Mathematica manuals the escape key is denoted by ⋮. A long form for special characters is always available using `\[NameOfCharacter]` but commonly used characters also have abbreviated forms called aliases. For example, the symbol ∂ has the long form `\[PartialD]` and the ⋮pd⋮. Characters that are available on a standard keyboard are called "raw" characters. In general, the use of characters typed with the option key should be avoided.

Syntax Errors

Mathematica will beep when you have made a syntax error and will indicate briefly where your error is below the input line. Sometimes the error will be trivial, like forgetting to capitalize or not balancing parentheses. Mathematica will help with the balancing of parentheses by highlighting the corresponding forward parenthesis [, {, (or " when the backward parenthesis], },), " is typed and give a warning sound for unbalanced back parentheses. Mathematica will also help do the balancing by showing what parentheses (braces, brackets, quotes, etc.) correspond with each other. Look in the Edit menu for the keyboard command that will do the balancing (`Control Shift B`). Set your cursor and execute the balancing command and Mathematica highlights the balancing.

Mathematica will echo your input if you are not getting execution. Also the output will be strange if you are not getting the desired execution. Mathematica will often do calculations in unexpected waves especially if you are a novice. So all results should be thoroughly checked.

In longer more involved statements, it might be advisable to execute steps one at a time to see where an error occurs. Mathematica will attempt to execute step by step if Trace is used. Also `FullForm` or `OutputForm` will show all the steps Mathematica has used label the position of an expression. These are useful when Mathematica gives a surprising execution to your commands.

Mathematica provides tools to examine how a routine is working. Trace applied around an execution command will cause all intermediate steps to be displayed. Even though output for `Trace` can be very long, it is often useful in pinpointing why Mathematica is not producing the desired results. Other commands like `Sow, Reap, Throw, Catch` allow the display of intermediate steps that ordinarily would be hidden. Sometimes a long calculation can be `TimeConstrained` to prevent locking up the program and a forced quit of the Kernel.

A common error arises during plotting. Mathematica reports that a plot is not a `machine sized value`. Mathematica does not graph but plots by computing coordinates of points to be plotted. The error means that Mathematica cannot compute numerical values. Numerical evaluation of functions that can be evaluated numerically can be forced by using Evaluate. Without this, Mathematica reaches back only one level into an assignment path. If Evaluate does not force evaluation, you should check (by running a few values) that numerical values are actually being generated.

Comments

Comments help you remember what you are doing or assist you with reporting your results. The comments can be entered anywhere if they are entered as enclosed in starred parentheses `(*Comment*)` where `Comment` can be anything

Output

Output is often very complicated. Things that are very easy to do with pencil and paper are not so easy to do with Mathematica. So Mathematica has several editing and control features that allow you to manipulate the output and the data without actually touching either the output or the data. The primary data structure for Mathematica seems to be a List. So there are very many control routines to manipulate lists. In order to handle efficiently problems some of these control routines should be learned. These are listed below under the heading Control and in other places.

Output to the monitor is suppressed by ending the command line with a semicolon ;. In fact, several statements can be written on the same line if they are separated by a semicolon. The semicolon serves this same suppression and separation purpose in the programming calls like Module.

Interfaces with Word Processors and Equation Editors

This should be necessary only if you have non-Mathematica work you wish to avoid redoing.

You can paste pictures from other programs into Mathematica and from Mathematica into a graphics program or a word processor. For example, you can paste graphs or equations made by an equation maker like `Expressionist` into Mathematica. To paste a picture either to or from Mathematica, you can use the following procedure:

1) Copy the picture from your graphic or equation-maker program. The picture will now be on the so-called Clipboard (buffer). It is necessary to know how the picture is being carried on the clipboard. Some experimentation should be sufficient. For example, most of the graphics made by a Macintosh are carried as PIC files. The modern graphics programs generally have a way of setting a default for pictures being copied, e.g., PIC, TIF, Postscript. Set the default to suit your needs.
2) Enter Mathematica.
3) Convert the Cell where the picture is to be Pasted into a Graphic Cell in Cell > Convert To.
4) Highlight the Cell bar with your mouse and Edit >Paste. The graphics should appear at the left had margin. You can grab the graphics with your mouse and move it to any desired position on the line. You can also convert the graphics at this time (for placing on your home page or fancy printing) or you can resize.
5) To copy a pasted object out of Mathematica, highlight the cell bar and use `Edit > Copy As`. For example, a formula from an equation maker pasted into Mathematica can be brought back to the equation maker for editing using this procedure..

Mathematica also has a special link to Microsoft Word.

Importing into Word Processors

You can import formatted text or graphics into your word processors using Copy Special and choosing one of the graphics formats that works well with your word processor. For example, PICT works well with Mac word processors.

Interfaces with Scanners

Scanned graphics may be inserted into Mathematica. You will be able to read the coordinates of scanned graphic objects just as with other plots created by Mathematica itself. This could be useful in converting pictures to mathematically active data.

Start with a computer file of a scanned graphic. It is possible that Mathematica will not be able to open this computer file. If Mathematica cannot open the computer file, use a program like Graphics Converter. Programs like Graphics Converter are able to open automatically almost all of the many types of graphics files. Use the graphics program to make changes in the computer file to get it in a form, which Mathematica can read. If you are making the scan yourself, then the scanner software should allow you at the outset to prepare your files for Mathematica without using another graphics program.

To find out what graphics files your Mathematica Front End will read access the File Menu > Open as Special in Mathematica. For example, the Front End of the Macintosh will read PIC files. Using `Save As` with the correct choice of format for your front end in the dialogue box, save your open scanned graphics file to your hard disk or floppy. Other necessary editing can also be done in the graphics program. For example, I recently had a scanned black and white plot that showed up with 256 shades of gray. I was unable to transfer this to Mathematica. The graphics program allowed me to change this back to black and white reducing the file size from say 2.5 Meg to 17k. Mathematica read the new 17k PIC file.

At this stage you may be able to copy and paste from the graphics program directly into Mathematica. Here a Clipboard Conversion and Graphic Cell will be necessary (cf. Interfaces with Word Processors and Equation Editors). If copy and paste does not work, close the graphics file in the necessary format and ask Mathematica to open (File Menu > Open as Special) the graphics file. Mathematica will open the file as a separate document. A copy and paste from this separate document into your Notebook should be possible.

Scanned graphics can be sent as attachments to e-mail over the Internet without loss of data. Attachments are retrieved in a folder called Attachments, which sits in the folder containing the e-mail program.

Corrupted Mathematica Documents

Sometimes Mathematica will report that a cell, like Cell 2256, is corrupted. Even though 2256 is not visible, you can locate the cell with a series of commands `OpenRead, SetStreamPosition,Skip, Read`. Once the corrupted cell is located, an ordinary `Find` will usually locate the contents. The topic of reading documents with OpenRead will be discussed in more detail elsewhere.

Importing Data

Mathematica has the ability to open and read data files made by other programs, e.g., a word processor or spread sheet, and even open and read data files made on other platforms, e.g., open and read PC file on the Macintosh. This ability is important because it prevents errors inherent in recopying data. Mathematica easily reads files formatted as TEXT but has enough control features to read other formats. A TEXT file is a file having only the very basic word processing control features common to almost every word processor. Many programs permit saving a document as TEXT. To save as TEXT you should choose the Save As option in the file menu and then choose the option TEXT in the dialogue. Also use TEXT when downloading a file from the Internet; otherwise, unwanted formatting options might be present in the downloaded data. To open the TEXT document, choose the Open in the Mathematica File Menu and locate the document you wish to open. The TEXT document will open in a new window. At this stage the data in the open document can be moved to the Mathematica work sheet using copy and paste.

In order to read data into Mathematica in the correct way you need to have an idea of how the data us presented. For example, the data

1972 195
1974 226
1976 283
1978 302
1980 343
1982 369
1984 384
1986 412

was contained in a text file called Test2. The file Test2 was opened to see its contents using

```
!!MacintoshHD:Test2.
```

The symbol !! is the read command. The name that Mathematica needed to access the file Test2 was found by setting the cursor after `!!` and the using menu `Action>Prepare Input > Paste File Pathname`. Note first that the result of `Paste File Pathname` will come in quotation marks, viz., "`MacintoshHD:Test2`" which must be removed manually before execution can take place. Note second that the file that being read must be on the same disk as the Mathematica Kernel being used to evaluate the `!!`. It may be necessary to designate a different Mathematica Kernel like the Local Kernel for the particular cell being used to execute `!!` and the `ReadList` explained below rather than a remote kernel which is often the default for programs running on a file server or it might be necessary to transfer the file being read to the file server (being aware of the 4 hour grace period). To change the evaluator this access the menu `Style > Evaluator` and choose the kernel that is located on the same disk as the data you wish to read. The kernel for subsequent evaluations will revert to the default kernel of the notebook. Once the data that you wish to read is displayed, you can load it into to Mathematica using `ReadList`. Here you need to designate the way you want the data loaded. At the very least you have to designate whether the data consists of Word, Number, etc. The file is then read into Mathematica with commas placed in the proper place. Otherwise, spaces separating the data will be read as multiplication by Mathematica. To read Test2 into the Mathematica as a Table with the commas and the brackets placed in the correct position to reflect the displayed data, you can use

```
ReadList["Macintosh HD:Mathematica Hdbk Folder:Test2", {Number, Number}]

{{1970, 164}, {1972, 195}, {1974, 226}, {1976, 283}, {1978, 302},
{1980, 343}, {1982, 369}, {1984, 384}, {1986, 412}}
```

Some other useful type specifications besides number are String, Word, Real. Here note the quotation marks given in Action>Prepare Input > Paste File Pathname are needed.

The data was read quickly because the Test2 was written as a TEXT file. If the file had been written by a word processor, then there would likely be some control features that would cause the file to load incorrectly or cause the ReadList to terminate prematurely. So some option in `ReadList` would be needed. For example, if the preceding file where made by Microsoft Word, you could use

```
ReadList["Macintosh HD:MathematicaHdbkFolder:Test2", Number,  RecordLists->True, RecordSeparators->{"\r"}]
```

where the `RecordLists->True` instructs Mathematica to make a separate sub list for each separate line and `RecordSeparators` instructs Mathematica to look for the hidden character "`\r`" that marks the end of the Microsoft Word line. Here the `\r` was found by instructing Mathematica to read the data file as Character. The \r was not visible even when the data file was read as Character. However, Mathematica found the control symbol \r when it was ordered using cmd cap L to drop the output line to an input line. There are several symbols of this kind like `\r`, `\t` that can also be used in other places in Mathematica to control the style of the output.

It might be impossible to load the data as just a list of numbers because of inclusion of descriptive column headings or the like. In these cases, a certain amount of additional work may be needed to read the data. For example, the date is opened first with !!. Then `an OpenRead` is used to open the file. The open file is said to be a `Stream`. The cursor by default is set at the beginning of the file. Using `SetStreamPosition` or Skip the cursor can be set to a better position which is a position where the data actually begins. Then a `ReadList` can be performed on the data with the cursor in a better position.

Exporting Files and Data

Mathematica output or data can be exported to various types of files. For example, data generated by Mathematica can exported to a Text file. First set up a Text file, with name for example mydata, using File>Save as Special >Text . Then **`Export[data[1], Path to mydata]`** will send whatever is labeled data[1] in the notebook to the file mydata. Here the path to mydata is found using **`Input > Get File Path`** menu. The data[1] will be read into the text file. Opening the file and reading it into a program like Excel is now possible.

Other types of files can be exported in the same way

Graphics

Plots are entered automatically in the Notebook as they are computed unless the graphical output is suppressed as explained below. You can copy the plots to a word processor by highlighting the plot with the cell bar and copying and pasting. You will need a Convert Clipboard for this. You can also grab plots with the mouse to reposition them or change their size within the Mathematica notebook. Also you can change the size of the plot by using the Size menu located at the bottom of the screen. Note that the plots are mathematically active. For example, clicking with the mouse on a point in a plot while holding down the `Control key` will cause the coordinates under the cursor to appear in the lower left hand corner of your screen. This selection feature is quite useful, for example, to get a starting point for certain algorithms or to get coordinates for placing labels of various features in the plot. After selecting one or more points or even lines, use Copy to copy their coordinates into the Clipboard to use with routines like `Fit` or `ListPlot`.

Several other options are available in the Input Menu in addition to Plot[Options] in Mathematica or several of the special packages supplied with Mathematica.

Graphics are organized in the following way. On the primary level are the so-called graphics primitives like `Line, Circle, Disk, Rectangle, Polygon`, `Raster, Raster Array`, etc. Mathematica can draw the graphics primitives with minimal amount of input, e.g., a list of coordinates for a polygonal line in `Line`. Options can be added to the graphics primitives when they are being computed. Also certain rendering instructions like `RGBColor, Thickness, PointSize`, etc., can be bundled with the graphics primitives by enclosing rendering instructions and the graphics primitives with the command `Graphics` or `Graphics3D`. In any case `Graphics` or Graphics3D needs to surround a graphics primitive to activate a graphics primitive for rendering with `Show`.

At a a higher level are the graphics commands like `ListPlot`, `Plot, Plot3D, ContourPlot, ContourPlot3D, ImplicitPlot, ImplicitPlot3D, ParametricPlot, ParametricPlot3D`, etc. These commands are really programs for generating enough graphics primitives to represent the object and for displaying the graphics primitives. For example, `Plot3D` generates enough planar polygons made up of 3 or 4 vertices to give a picture of a surface $z = f(x, y)$. Graphics like `Plot3D` can be manipulated by manipulating the graphics primitives that comprise the `Plot3D`.

Graphics can be formatted in several ways. Mathematica will make a default choice for the particular front end being used. The choice can be changed using the various options. Color choices made in one or more 3 dimensional plots are ignored by Show. To prevent this from happening use the option `Lighting -> False` in the show. There is also a Plot Viewer to change the orientation of graphics. The viewer can be accessed through the menu or by typing cmd V.

Text Editing in Notebooks

Once something has been entered into the Mathematica it cannot be altered as one would alter a word processing document. For example, if you defined p as a certain polynomial and entered it and then found that you made a mistake in one entry, erasing the mistaken entry and replacing it with the correct one will produce no change. You must re-execute the new *p* in order for Mathematica to pick up this change. Re-entering will delete the former *p* in the internal state of the Mathematica as well as change the p on the screen. However, even this does not work completely and even the more powerful statement `Clear[p]` does not work completely in that there are certain attributes associate with each assignment that persist. Sometimes `Remove[p]` will get rid of the Attributes and sometimes a Kernel shut down must be used. The fact that the monitor screen does not accurately reflect the internal state of Mathematica is a major cause of errors. For example, if you assign a number to x, then x will no longer be a variable. Mathematica will remember that x is a number until you restore x to an unassigned symbol using Remove. You can check on the status of any name or symbol by using `Context`. Note that Mathematica remembers the order in which you enter the commands and not the order the commands appear on your Notebook.

If Mathematica is giving what appears to be erroneous or erratic answers, you might have to remove the assignments. The easiest way to do this is by executing `Clear["Global`*"]`. You can also disconnect the kernel from the notebook and re-execute everything. The easiest way to do this is to save your notebook, close down and reopen the just closed notebook. Also most manuals on Mathematica recommend that every assignment be preceded by a Clear operation on the new variables.

File Transfers, Mathematica Reader, PDF Files, HTML Files

Mathematica can be run on many different types of platforms, e.g. PC's, Sun Workstations, and Macintoshes. Mathematica notebooks are written and saved in plain TEXT using only ASCII characters and therefore should be transferable by email, disk, zip disk or on the internet without any further intervention. You should NOT encode your notebook before sending it as an email attachment or as an FTP document. The plain text has all the necessary formatting commands for Mathematica to reconstitute itself as a Mathematica document. Note that Macintosh will read a PC formatted disk but a PC will not read a Macintosh disk. So you should format your disk as a PC disk.

Occasionally, documents sent over email or the Internet lose their creator code necessary for Mathematica to read the disk. In this case, it is necessary to reconstitute the Mathematica document as a TEXT document using Word or some other application like MacLink. Also it might be necessary to open a TEXT document from within Mathematica. This is usually the case when the Mathematica document does not have its distinctive icon. The first save of the opened document will restore the distinctive icon. In all case, the Mathematica document should have the suffix `.nb`.

To read a Mathematica document, you need a copy of Mathematica or the `Mathematica Reader`. The Mathematica Reader allows you to read and print Mathematica documents but not to execute them or to edit them. The Mathematica Reader can be obtained free from

`www.Wolfram.com`

On rare occasions, Mathematica will encounter an error when trying to reconstitute a document. The line where the error is encountered will be part of the error message. You can open the Mathematica document with `File > Open Special > Text` as a text file and use the `Find` utility to locate the place where the error occurs on the chance that you will be able to correct the error or delete the surrounding material. For more subtle errors where the line the error is given rather than the actual error, you can use the methods outlined in the section Correcting Corrupted Mathematica Files.

A pdf file is a document readable using Adobe Acrobat Reader. Anyone who uses the internet will have a a Reader and be able to read a Mathematica notebook in pdf format. To create a pdf file from a Mathematica notebook using a Macintosh, select `File >Print` and select `PDF` on the print dialogue. For the PC, again select `File >Print` and then select `Save As File`. The saved file will be in postscript format and needs to be converted to pdf format using `Adobe Acrobat`.

Mathematica has translators to make documents in HTML or TeX format. HTML takes formatted text into an encoding of the format using only the lower case and upper case keys with no alternate keys. HTML documents are read by web browsers. TeX is a standard mathematical type setting format again in standard typewriter characters. Publishers use TeX formatted documents to create galley proofs directly from an electronic copy of the document. The HTML format generated by Mathematica is a long file of JPG or picture files.

Hyperlinks and Buttons

You can set up hyperlinks within your Notebooks. The links can be either within the document, to other Mathematica documents or to Internet locations. Use the `Input > Create Hyperlink` menu.

To set up hyperlinks to the Internet, you need to set up a `Button` in your notebook. Buttons are executable routines embedded in the document. For example,

```
Button[www.wolfram.com, ButtonData:> {URL["http://www.wolfram.com"], None},
                        ButtonStyle -> "Hyperlink"]
```

sets up an executable link between your notebook and Wolfram's homepage.

Help

Help is built into the Mathematica program. You can access the Help through the Help on the Menu Bar or use the Help Key on the computer keyboard or cmd Shift F or Shift F1 key. If your cursor is set behind a Mathematica command or even a command in a previously loaded package, Mathematica will read the command and go to the required help file automatically without retyping (This seems to be available only on the Macintosh.) The help file opened by Mathematica will have examples and hyperlinks to the complete Mathematica and Package manuals and to similar subjects. Pages of the manual can be printed for easy reference. The manual pages are actual Mathematica notebooks and can be executed and changed as desired. The manual should reconstitute itself in its original form when closed so that you do not need to be concerned with making a irrevocable change. There is a special style for the manual listed in the style sheets. This should not be edited since the manual without its style sheet is more or less unreadable.

If you remember part of the name of a command, type the part that you remember and then go to `Input>CompleteSelection` or its keyboard equivalent `Control k`. Mathematica will open a popup menu where you can complete your selection. If you forget the syntax for a command, type the command and then go to `Input>MakeTemplate` or its keyboard equivalent `Control Shift K`. Mathematica will supply the syntax in the form of recognizable symbols. You can replace Mathematica's symbols with your own input.

A short list of commands and reserved words Mathematica is appended to this Introduction. Note first that commands and reserved words generally resemble the English words. All reserved Mathematica words are capitalized and user defined terms should generally start with lower case letters to avoid conflicts and error messages. To learn if a certain command will do what you have in mind or to learn the format (syntax) of a command, you can use the 'help' utility in Mathematica in the following way. For example, typing

```
?PolynomialDivision
```

and

```
?Simplify
```

brings up the help file on `PolynomialDivision` and `Simplify` respectively. Most Help Files have examples, which can be overwritten even in the Help File. The overwritten examples will not remain a permanent part of the Help File. To bring up more information use ??. For example ??Map will bring up a more complete list of information about the command Map. Many times the commands permit extra variables called Options. To find the Options available with say Plot type

`Options[Plot].`

If you find an option that you want to see, e.g.. AxesOrigin in Options[Plot], type

`?AxesOrigin`

to find the syntax or

`??AxesOrigin`

to get the details.

Other forms of help are available using

`Information[Map].`

To find a list of Mathematica commands that contain a certain name or match a certain pattern, like `Plot,` use

`Names["Plot"].`

This does not give much but Plot is often part of a compound word. These can be accessed by a utility program that matches a string:

```
?*letters          gives all commands beginning with letters
?*letters*     gives all commands with letters
?letters*          gives all commands ending with letters.
```

For example, `??*@@*` will give an almost complete list (running to 10 pages) of Mathematica commands since @@ is a Mathematica programming shortcut for `Apply` the subject to the predicate. You can also see what packages are available to add extra capabilities using pre-programmed functions in the Function Browser.

No help for routines in packages is available until the package is loaded (cf. Packages section). To find out what kinds of routines are available in a loaded package, you can use `Names`. For example, typing

`Names["Graphics`Graphics*"]`

produces a long list

`{BarChart, BarEdges, BarEdgeStyle, ... , SkewGraphics, StackedBarChart, TextListPlot, TransformGraphics, UnitScale}`

of routines available in the package.

Some third party packages come with their own help file. The package instructions will give information on the placement of the help file. These help files often appear in the help menu and work like Mathematica help files. You can make our own help files to go with your own Mathematica packages.

Third party on–line help files are available. For example, a hypercard program (an on–line file card system) by Robert Campbell is available. See the Documentation below.

Help for your front end including a list of `Key Board Shortcuts` is available under

`Help > Help Browser > Other Information >Your Platform.`

Packages

Certain of the Mathematica routines are contained in special packages. On the Macintosh you can open a package from the `Help > Help Browser` menu. Find the package you want from the Function Browser. To see the packages you must highlight the middle radio button `for Add Ons`. Most of the packages are `Standard Packages`. Highlight the Standard Packages in the left hand scroll box, then highlight the package you want in the middle scroll box, and double click on the sub package in the right hand scroll box. The page in the Mathematica menu corresponding to the package will appear. The page in the manual is actually an active Mathematica notebook. Find the command to load the package, usually near the top of the page, and copy it and execute it in your notebook. Alternately, execute the package in the help file. You can also type the loading command in your notebook and execute it without accessing the help file. Almost no one can get this method of loading a package to work and so using the help file is best. Also executing in a package without first loading the package has some bad effects and thus it is necessary to be careful. For example, `if FilledPlot` is used before its package is loaded, it must be removed using `Remove[FilledPlot]` before the package is loaded. Your vacuous `FilledPlot` will overshadow Mathematica's own `FilledPlot`.

Sometimes commands from a packages will interfere with commands in the main program or with commands in a different part of the same package. For example commands in `VectorCalculus` interfere with the main program and the command Mean in various parts of the statistics package interfere with each other. In the former case, you need to use Remove to get rid of the `VectorCalculus` package after it has been used. In the latter case, you might wish to consider what you will need for your work session and then open the smallest package up to the master package, eg., `Master.m`, where all the routines from the package are available together with some method of resolving conflicts. You can also resolve conflicts by designating the desired function by a very specific name which includes a complete pathway back to the package in which the desired function lies.

There are many additional packages available from the Wolfram Development Corporation for registered users. Some of these packages as well as current information on Mathematica can be obtained via e-mail from the Wolfram Development Corporation. You can try logging on as an anonymous user at info@wolfram.com. Also try sending the e–mail message Help or Intro to mathsource@wri.com. Other access routes are

FTP: mathsource.wri.com
Gopher: mathsource.wri.com (port 70)
WWW: http://www.wolfram.com/
Dialup: 217-398-1898 (8N1)

Other packages come with various textbooks. Sometimes these packages require the installation of a program called MathLink which is supplied by Wolfram Corporation to users of Mathematica. MathLink can also be used to create your own programs that will access the computational engine of the Mathematica kernel.

Your own Packages

You can create your own packages. These will contain functions that you want to have available or make available to others. You set up these packages using exactly the same syntax as Mathematica's packages. The packages are saved in Text files ending with the designation .m. You can write a large package as a collection of smaller packages and link all the smaller packages together using a directory file which has the name `init.m`. To set up the package, you create a folder and give it the name of the package, for example `MyLinearAlgebra`. Inside the folder MyLinearAlgebra, you create a second folder called Kernel. The Kernel contains a special file named `init.m`. Inside `MyLinearAlgebra` but outside Kernel, you set you individual package files, for example, a file called `mySpecialMatrices.m`, which sets up special types of matrices you use in your work, for example, torsionmatrix and stressmatrix. You now insert in your init.m file DeclarePackage lines, which tell Mathematica to look for the `torsionmatrix` and the `stressmatrix` functions in `the SpecialMatrices.m` file once the package MyLinearAlgebra has been opened.

Your placement of your own package files is important. When you instruct Mathematica to open your package or any of the packages supplied by Mathematica, Mathematica will automatically search certain folders to find the package to be opened. You should place your packages in one of the folders that is automatically searched such as the `AddOns > Applications Folder` in the Mathematica Folder. You should place your folders in a location that you remember for the following reason: when you upgrade Mathematica, you might find that all the Mathematica folders have been overwritten. The `AddOns > Applications` is supplied as an empty folder on the Search Path and an update might overwrite this folder with another empty folder. So you should at minimum physically remove your packages at upgrade or perhaps even default regeneration and at best always have a current backup copy saved in a safe location.

It might happen that you cannot access the folders in the default search path. This will occur if you use a remote kernel on a protected file server or use a laboratory machine. In this case, you may still have access to a portion of the file server or hard disk on the laboratory machine. To use your own package in this case, you can put your package on the available portion of the file serve or hard disk. You then need to locate your package.

You can do this by using the Input > Find File Path menu to find the location of your package. This will be part of the information you need. Some hidden information may be necessary. This is obtained by executing `Directory[]`. Together these two procedures should give you enough information to find your package. Either you can use this information to open your package directly using the form

```
DirectoryName1`DirectoryName2`...`MyLinearaAlgebra
```

where DirectoryName1 is a file on the search path. Alternately, you can append a new search path

```
$Path = Append[$Path, YourNewSearchPath]
```

to in case you have many packages.

Note that packages can be set in the AutoLoad folders or in initialization cells which cause items to be loaded at start up of any notebook or the particular notebook respectively. To create the an initialization cell, type a command line and highlight the cell bar and use Cell > Cell Properties > Initilialization..

Patterns

Much of the strength of Mathematica comes from its ability to match patterns. The patterns are matched in a deterministic way; however, care must be exercised in order to insure that the patterns are matched as intended.

Documentation

Copies of the Mathematica manuals are available in the Engineering College Library. the Geology/Physics Library, and the Mathematics Library. There are many specialized books. A sample selection is listed below. Note that many textbooks are now appearing with Mathematica appendices or work books.

1. Martha Abell and James Braselton, Mathematica by Example, Revised Edition, Academic Press, Boston, 1994, $39.95 - various examples from beginning college mathematics displaying the scope and power of Mathematica
2. _____, The Mathematica Handbook, Academic Press, 1992, $34.95 a detailed look at all commands
3. _____, Differential Equations with Mathematica, Academic Press, Boston, 1993, $44.95 -First course in ordinary differential equations with some separable partial differential equations. New edition in 1997 uses Mathematica 3.0.
4. Nancy Blochman, Colin Williams, Mathematica, A Practical Approach, 2/e, Prentice-Hall, 1998. Information of programming, packages, importing, exporting
5. _____, Mathematica, Quick Reference, Addison-Wesley, 1992, $18.25 - annotated list of all Mathematica commands with cross references.
6. Robert Campbell, Mathematica Help, Variable Symbolics Inc.,1994. On-line help file in a hypercard (file card) format with examples, attributes, bugs for all Mathematica commands organized both by category and by name in either Macintosh or Windows format -$129.
7. Kevin R. Coombes, Brian R. Hunt, Ronald L. Lipsman, John E. Osborn, Garrett J. Stuck, Differential Equations with Mathematica, John Wiley and Sons, New York, 1995. Manual to accompany Boyce and De Prima but can be used by itself.
8. Samuel Dick, Alfred Riddle, Douglas Stein, Mathematica in the Laboratory, Cambridge University Press, Cambridge, 1997, $29.95. How to get Mathematica hooked up to your laboratory equipment.
9. James Finch and Millianne Lehmann, Exploring Calculus with Mathematica, Addison-Wesley, 1992, $10.75
10. Richard Gass, Mathematica for Scientists and Engineers, Prentice-Hall, Upper Saddle River (NJ), 1998
11. Alfred Gray, Modern Differential Geometry of Curves and Surfaces with Mathematica, CRC Press, Boca Raton, 1998, - Curvature, first and second fundamental forms, etc.

12. John W. Gray, Mastering Mathematica, Academic Press, Boston, 1994, with Mathematica problems on disk -$44.95. Explores many of the fine points while also treating basics.
13. Eugene Johnson, Linear Algebra with Mathematica, Brooks/Cole, 1995
14. Stephan Kaufman, Mathematica as a Tool, Birkhauser, 1994, $34.85. Much information on system features
16. Roman Maeder, Programming in Mathematica, 2cd Ed., Addison-Wesley, 1991. Standard reference for those who wish to learn Mathematica programming.
17. _____, The Mathematica Programmer, Academic Press, Boston, 1994. $44.95. Programming in Mathematica - comes with a disk.
18. Peter V. O'Neil, Advanced Engineering Mathematics Mathematica Lab Manuel, PWS, Boston, 1995. Lab manual to accompany text with ODE, PDE, Fourier analysis, matrices, vector calculus and analytic functions.
19. Mark Pinsky, Partial Differential Equations and Boundary–Value Problems with Applications Appendix on Mathematica by Alfred Gray, McGraw-Hill, 1991 - Various routines from partial differential equations with Fourier Series using Mathematica.
18. John S. Robertson, Engineering Mathematics with Mathematica, McGraw-Hill, 1995, New York - mainly ordinary and partial differential equations using Mathematica.
20. William T. Shaw, Jason Tigg, Applied Mathematica: getting started, getting it done, Addison-Wesley, Reading, Mass., 1994.
21. Steven Skiena, Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Addison–Wesley, 1990. Main reference for discrete mathematics package of Mathematica. $53.95
22. Arthur Sparks, John Davenport, James Braselton, Calculus Labs using Mathematica, Harper-Collins, 1993, $10.95.
23. K.D. Stroyan, Calculus using Mathematica, Academic Press, 1992
24. Tom Wickham-Jones, Mathematica Graphics, Springer-Verlag, 1994, $44.95. Explores graphics capabilities. Comes with disk with routines (e.g. labels for contour plots) for all personal computer platforms without the need for translators.
25. S. Wolfram, Mathematica: A System for Doing Mathematics by Computer 5/e, Cambridge University Press, 1999, New York, $49.95. This is the manual that accompanies the main part of Mathematica 5.0. It is sold separately from the program and it is on-line for Mathematica 5.0. (not available as of Sept.1, 2003)
26. Wolfram Research, Mathematica 4 Standard Add-on Packages, Wolfram Media Inc., 1999, Urbana $49.95 paper. This is the manual for the add-on packages.

Reporting Requirements

All reports should contain the problem statement and a narrative of the method of solution with as many intermediate steps as necessary to clarify the narrative. In this sense your report should be like a laboratory report. At first you may leave the Mathematica commands in the reports. At a later more advanced stage the Mathematica commands should definitely not be included in the report and only as much of the actual mathematical output included to clarify the solution. The material in the section Making your Notebooks Readable must be implemented.

Getting Started with Mathematica

Execution of Commands

Enter Key (Lower Right Corner Enter Key)		Executes commands at the current cursor position
Shift Return Key		Executes commands at the current cursor position
Option Enter		First hit sends cursor to the next input line and second hit causes execution of this input line
Shift Option Return		Interrupts current execution and causes current selection to be executed as soon as possible
Cmd Return		Creates a new cell. Creating a new cell can also done with the cursor key. When a horizontal line forms across the monitor, a new cell is ready to be formed.
Control e		Enters an expression
Control period		Exits an expression and highlights what has been exited
Control Shift Enter		Replaces input with output .in highlighted section

Help (cf. Section on Help)

cmd ?		Creates a question mark pointer for front end
Help Key		Opens an extensive help file called The Function Browser
cmd cap F (shift F1 on PC)		Opens the Function Browser. Brings the reference manual on line. Note the reference manual for the packages requires highlighting the packages radio button.
?expr		Information at various levels of detail Basic Level
??expr		Detailed Level
??*expr*		Displays a list of anything in the help file that contains the string expr
Names		To see what is available in a package
Make Template		control Shift K
Make Template		Highlight the command an go to Input > Create Template. Mathematica will use generic terms to show the syntax
Complete Selection (control k)		Highlight the first few letters of a command and go to Input > Complete Selection. Mathematica will create a pull down menu of all commands starting with your letters
Options		Options[ListPlot] will show all the options for ListPlot
InputForm, FullForm, OutputForm		Shows everything Mathematica is using to generate expressions or compute
FullForm		Can be used to find Mathematica's name of a symbol or the way Mathematica is reading an object.
Trace		Surrounding a command will print the complete execution for seeing where something is going wrong
Hold down option key while opening Mathematica		Causes the preference file to be set to Mathematica's default. Save a copy of preference files and init's in case they get corrupted.

.

WordProcessing

⁝alias⁝		The symbol ⁝□used in the Mathematica manual for the escape key in upper left corner of keyboard. Called AliasIndicator. The alias will be set up in type. E.g., ⁝int⁝ gives an integral sign.
Control ^		Sets Superscript - this is also good for input

Control _		Sets Subscript
Control =		Sets Overscript
Control 7		Sets Underscript
Control 5		Moves to next position, eg., after setting upper limit of an integral with int control^, type control 5 to reach the lower limit position
Control spacebar		To move out of a formatted item
Control 2		Radical
Control /		Fraction
\[name of symbol]		Symbol
⋮a⋮		Greek letter a corresponding to the Roman a. Type Escape key a Escape key.
FullForm[α]		Gives \[Alpha]
ToCharacterCode		For the international character number of a particular symbol. Useful in getting package functions to carry out formatting commands. Packages must be saved as plain TEXT and often lose formatted characters unless the characters are carried as their character code.
FromCharacterCode		Take a character number into a symbol
SequenceForm[HoldForm[Integrate[x, x]], " = ", Integrate[x, x]]		To get equation set up. SequenceForm allows for mixed Input, Text and Output and can be used to format equations: the actual input is both held in symbolic form with HoldForm and then evaluated. ReleaseHold can also be used to cause evaluation of HoldForm.
Control CapC		Brings up matrix dialogue box
HoldForm		Keeps Mathematica from evaluating but will set expression into Mathematical form. HoldForm distributes sums, viz., a + b + c//HoldForm gives HoldForm[a] + HoldForm[b] + HoldForm[c]
Boxes to hold input (in matrices, etc.)		Called a PlaceholderBox
\		To connect characters that should be written together but are likely to appear on separate lines due to word wrap. Mathematica sometimes places these automatically. For example, 1000\000 is 1 million.
\t, \r		Characters of this kind control tabs, new lines. These can be used on Input lines to control the format of the output especially output set up in SequenceForm.
Control (Control) Control Space bar		This sets a format box for typing formatted text. This will automatically appear with control -^, etc. Control) or Control spacebar escapes from expressions or subexpressions one level at a a time.

Boolean

expr[1] === expr[2]		Returns True if expr[1] and expr[2] are identical and False otherwise
&&		And to set up simultaneous equations. Can be applied to a list of words True or False, to give True if all true and false if at least one is False.
||		Or Can be applied to a list of words True, False.
True		To set a default for a conditional statement

Automatic		Mainly used as the setting for an option which activates Mathematica's built in default. Ticks ->{{- 2, 0, 2}, Automatic} can be used as an option in Plot
All		Used to include everything of a given object. PlotRange ->All includes

Arithmetic

1 + 2*3^4		^ for exponents nd * for multiplication. Note that a space works for multiplication, e.g., x y or x*y is x times y. The formal names are of the functions are Plus, Times, Power. Plus[x, y] is x + y. To get Plus and Times to work on a list, see Apply or @@
4x		No space between the 4 and x is neccessary
1/4		Fraction. Note that 1/ 4 2 is 1/2 while 1/(4 2) is 1/8
N[1/4]		N stands for decimal evaluation
N[Pi, 20]		Returns first 20 digits of Pi.
Im[a]==0		Gives the condition that a is a real number

Complex Numbers

I		The complex number
ComplexExpand[…]		Expand for complex numbers
Conjugate		Complex Conjugate
Arg		Argument
a /: Im[a] = 0		Instructs Mathematica to treat a as a real number
Im I a ^=0		Also gives the condition that a is a real number

Assignments

=		An assignment read as Set. Table[Set[f[k], expr[k]], {k,1,10}] defines a function f with domain {1, 2, … , 10}
==		Equal. A == B returns true if A and B are identical
LHS === RHS		SameQ. Answers the question: is LHS identical to RHS? No simplifications will be performed resulting in many erroneous False's.
Exp[g[x_]] ^= x^2		Creates a function associated with the name g. Mathematica will not accept Exp[g[x_]] = x^2 since it uses the reservered term Exp. This is called an UpSet
g/:Exp[g[x_]] = x^2		An alternate form of the preceding
:=		A delayed assignment where everything is held unevaluated
x =.		Removes the value associated with x
Clear[a]		Removes a, for example, when a is a function
a = .		Another form of Clear
Remove[Name]		Removes Name from the list of things that Mathematica recognizes. Useful if a name is invoked before the package where it resides is opened.

Algebra

f[x_, y_] = x^2*Cos[y]		Defines the given function. This need not define only an algebraic property but can be used to call up a subroutine (cf. Programming Section)
Factor(x^2 - 4*x + 3)		Factors the polynomial over the integers if possible. The option GausianIntegers Æ True forces factorization over the complexes. Other forms include FactorList
Factor[(2*x^2 + 25*x+ 72)/(-14*x^2 -47*x + 72)]		Reduces the fraction to lowest terms
Solve[x^2 - 4*x + 3= = 0]		Mathematica will try to solve for all variables. Since only x is present, this gives no problem. Note = = means "equal" while = means assignment.
Solve[{x^2 + 2*b*x + c = = 0}, {x}]		We need to give the variable here in order to carry c as a constant.
NSolve[x^2+x+1= =0,{x}, 5]]		Solves numerically for x with precision 5 digits
PolynomialDivision[x^3 + x + 1, x + 1]		Gives the quotient together with the remainder as the second term
PolynomialRemainder[x^3 + x + 1, x + 1, x]		Gives the remainder as the first term and the quotient as the second term
Eliminate[eqns, vars]		eliminates variables in a set of simultaneous equations
Expand[expr]		Expands and simplifies the expression expr
ExpandNumerator[expr]		For fractions
ExpandDenominator[expr]		
TrigExpand, TrigFactor, TrigReduce		For handling trigonometric expressions
Cancel[…]		Cancels common factors in numerator and denominator.
Together[…]		Puts a sum of fractions over a common denominator
Trig->True		To engage trigonometric routines in Simplify
Apply[Plus, {x^2, y^2, 3}]		Converts the list {x^2, y^2, 3} into variables so Plus will act on it. Useful in all conversions of lists to variables for multivariable functions. Here gives x^2 + y^2 + 3. An abbreviated way of implementing Apply is Plus@@{x^2, y^2, 3}
Times@@{x^2, y^2 3}		3*x^2*y^2
Coefficient[x^2*y^3 + 3*x*y^3 + y^2,y,3]		Returns the coefficient x2 + 3x of y^3
Coefficient[x^2*y^3 + 3*x*y^3 + y^2,y^3]		Does the same as the preceding

Control

Quit Kernel		Removes everything from memory and a restart in place
Cmd Comma		Pause in an execution
Cmd Period		Abort an execution

Cmd Shift Period		A so-called Long Jump which disconnects the Notebook from a remote Kernel in desperate situations. A save is generally possible even in a lock up.
cmd l (el)		Copy input from above
cmd L		Copy output from above
TimeUsed[ ]		The amount of time already used in the session. Can be used to compute how long a routine takes. Also Timing for this.
TimeConstrained[expr, n]		Evaluates the expression for n seconds and then aborts
nSix[x_]:=N[x,6]		A possible 6 decimal place output function
$Post = nSix		A short program to set precision at 6 places. Mathematica does not seem to have this built in. Default is 16 place. To force decimal instead of rational evaluation, try entering at least one number as a decimal, eg., enter 2.0 instead of 2
expr//Function		So-called Postfix form is the same as Function[expr]
expr//Expand		Expands the previous expression
expr//N		Evaluates numerically the previous expression
{2#, 3#}& ~ Outer ~ {a, b}		So-called Infix form is the same as Outer[{2#, 3#}&, {a, b}] giving {{2a, 3a}, {2b, 3b}}
%		Last output
%%		Second to last output
%%%		Third to last output. You can use as many % as you like
Short[expr]		An one line view of expr
Shallow[expr]		Shows the top level of expr in abbreviated form. More detailed information can be requested with Shallow[expr, {Depth. Length}]
a = x^3		a is now assigned the value x^3
<<Algebra`SymbolicSum		Loads the SymbolicSum routines in the Algebra package. Packages can also be loaded via FunctionBrowser.
TrigReduce		expands trigonometric identities. In the Algebra`Trigonometry package
PowerExpand		To deal with nested radicals
Needs[…]		To load packages for the PC. Better to load packages via the Function Browser
<<Algebra`Master		Loads all routines in the Algebra package
Off[General::Spell1]		Prevents Mathematica from giving too many error messages for permissible assignments, supposed conflicts in name assignments, etc.
Clear[f]		Removes any assignments from f. To be used before defining functions, etc. due to our imprecise knowledge of Mathematica's internal state (cf. inquiry routines for internal state). Clear[f, g] will clear f and g simultaneously.
Clear["Global'*"]		Clears all global definitions and acts as a restart
expr /.{x->x^2 + 2*y, y->w^2 + 3}//Simplify		First applies the rules {x->x^2 + 2*y, y->w^2 + 3} to expr and then Simplifies, i.e., substitutes the x^2 + 2*y and w^2 + 3 for x and y in expr and simplifies. Here /. is an abbreviated form of ReplaceAll
x^5 + 3 x^4 + 2 x^2 + 2/.%		Returns x a2 + a2 + 2a + 2 contrasted with the next substitution
x^5 + 3 x^4 + 2 x^2 + 2/.{x^2->a}		Returns x5 + x4 + 2a + 2
Table[x, {5}]/.x ->Random[ ]		Returns {0.7244, 0.7244, 0.7244, 0.7244, 0.7244} since Random[ ] is evaluated before being substituted for x
Table[x, {5}]/.x:>Random[ ]		Returns {0.321771, 0.987824, 0.459796, 0.414655, 0.483086} since Random[] is substituted for x before Table is evaluated

log[a b c d]/. log[x_y_] -> log[x] + log[y]		Gives log[a] + log[b c d]
log[a b c d]//. log[x_y_] -> log[x] + log[y]		Gives log[a] + log[b] + log[c] + log[d] since //. is an abbreviation for ReplaceRepeated
ReplaceAll		Alternate way of doing the substitution /.
ReplacePart		Replaces a specified part of an expression
Button		Sets up a template for an executable Mathematica command. Used for hyperlinks or palettes.
Sequence		Splices objects into input for a functions, e.g., f[a, Sequence[b, c]] = f[a, b, c]
Simplify[expr, opts, {assumptions}]		Applies simplification rules to expr. Options like Trig->True are available. The optional assumptions part have the form of a list like {n ∈ Integers, x >= 0} which will, for example, set Sin[n π] to 0, and simplify fractional powers in x,
FullSimplify		Applies the full range of simplification rules known to Mathematica. The same optional assumptions as Simplify can be used. This also has a time-out feature.
Refine		Used to define conditions for a function
Assuming		Assuming[assumptions, expr] simplifies expr according to assumptions. Actually, FullSimplify with assumptions added to list of Mathematica simplification routines.
FunctionExpand		Tries to expand out special functions using built in rules. The same optional assumptions as Simplify can be used.
Sow[expr]		Specifies expr should be collected by the nearest enclosing Reap. Useful in seeing the operation of a program. Does not interfere with the evaluation.
Reap		Used to collect the Sow items.
Dispatch		Optimizes a list of rules for fast execution.
Unevaluated		Unevaluated[expr] represents the unevaluated form when it appears in a function. Insert[{a, b, c, d}, Unevaluated[Sequence] @@ {1, 2}] -> {a, 1, b, c, d} and is Insert[{a,b,c,d}, 1, 2]

Functions

f[x_,y_]= x^2+y^3		The function $f(x, y) = x^2 + y^2$. Here f[z^2, 2] will produce $z^4 + 8$
g[x_]={x,x^2}		Definition of a function of a single variable into two space
f[x_]:= x		Delayed assignment =: where there is a possibility that Mathematica cannot make sense of the function. If Mathematica returns an error message to the usual assignment =, try =
x=4		This will not affect the f[x_]:=x in the previous line since := is used instead of =.
f[x_]:= x^2;f[5]		Returns 25 since the delayed assignment tells f to match the pattern and wait for the value of x
Map[f, {1,2,3}]		Returns {1,4,9}.
Scan[f, {1,2,3}]		Same as Map except that Scan discards the results of its execution. Useful for carrying out operations which have a side effect like an assignment.
{1,2,3}//f		Also returns {1, 4, 9}
f[{1,2,3}]		Also returns {1, 4, 9}

f/@{1,2,3}		Also returns {1, 4, 9} since /@ is Map
f@@{1,2,3}		This is not evaluated since @@ means Apply
Composition[f,g][x]		Returns f(g(x)) where f and g are functions. Also written as f@g@x or f@g[x] or x//g//f.
ComposeList[{f1,f2, … , fn},x]		Returns {x, f1(x), f2f1(x), … , }
NestList[f,x, 3]		Returns {x, f(x), f(f(x)), f(f(f(x)))}. This can be used to carry out a fixed number of steps of an algorithm like the Newton–Raphson Algorithm.
Nest[f,x, 3]		Returns the last term f(f(f(x))) of NestList
FixedPoint, FixedPointList		Runs a Nest until there until two successive terms are the same. FixedPointList permits the option SameTest ->(ComparisonFunction)
Thread		Applies a function to all elements of a list. Thread[{x, y} -> {1,2}] gives {x –> 1, y –> 2}
MapThread MapIndex MapAll MapAt		MapThread[f,{a, b}, {c, d }] gives {f[a, c], f[b, d]}, MapIndexed[f, {a,b}] gives {f[a, {1}], f[b,{2}} MapAll[f, {{a,b}, {c,d}}] gives f[{f[{f[a],f[b]}], f[{f[c],f[d]}] An option is the level at which f acts Applies a function at a designated place in an expression.
Through Distribute Operate Outer Inner		Through[p[f, g][x]] -> p[f[x], g[x]] Distribute[f[a + b, c + d]]->f[a, c] + f[a, d] + f[b, c] + f[b, d] Operate[p f[x, y]] ->p[f][x, y] Outer[f, {a, b}, {c, d}]-> {f[a, c], f[a, d], f[b, c], f[b, d]} Inner[f, list1, list2, g] where f = Times and g = Plus in the usual Dot product
Function[{x,y},x^2 + y^3}][a, b]		Gives the function $a^2 + b^3$ without the need to use up a name for a function
((#1)^2 + (#2)^3) & [w^2, 2]		Another way of writing the preceding function just to get the evaluation $w^4 + 8$. Used for intermediate steps of control functions for Cases, Select, etc.
Map[Function[x,{x[[2]], x[[1]]}], {{1, 2}, {3, 4}}]		Gives {{2, 1}, {4, 3}}
g[{x_, y_}] = x^2 + y^2		
{1,2}//g		Returns 5
f/@{a, b, c}		Returns {f[a], f[b], f[c]} being another form of Map
Apply[Plus, {1,2,3}] Plus@@{1,2,3}		Returns 6. Used to get functions of two or more variables to work on lists especially points.
Sin[{2,4,6}]		{Sin[2], Sin[4], Sin[6]} Functions that behave this way are called Listable
Evaluate[expr]		Forces full evaluation, e.g., a function defined with := in Plot because symbolic data in a plot causes an error. Should be used in Package functions and Modules to prevent failure to execute.
Return		Forces an immediate exit from a control function in a routine
SetAttributes		Every symbol has a certain set of attributes that aids ievaluation. These can be modified or augmented. Eg. Sin[{1 ,2, 3}] = {Sin[1], Sin[2], Sin[3]} since Sin has the attribute Listable.
Orderless, Flat, OneIdentity, ...		Various attributes that can be assigned to a symbol

Unprotect		Necessary to change or expand attributes of built in functions. Changing definition of a built in function does not survive a kernel shutdown. To prevent changes use Protect
Fold[f, a, {b, c, d}]		Returns f[f[f[a, b], c], d] where f is a function of two variables. Useful for updating lists, matrices, etc. where f could be Append, ReplacePart, etc.
Throw		Stops a routine upon execution
Catch		Encloses a routine with Throw. It gives the nearest value when Throw is executed

Lists

{2, 3, 1}		An order triple
{2, 3, 1}[[2]]		Returns element 2 of the list, viz., 3. Formal name is Part (cf. Below)
{{1,2,3},{4,5,6},{7,8,9}}[[2,3]]		Returns the 2,3 entry of the matrix, viz. 6
{{1,2,3},{4,5,6},{7,8,9}}[[{1, 2},3]]		Returns the 1,3 and {2,3} entry of the matrix written as a list viz. {3, 6}
{{1,2,3},{4,5,6},{7,8,9}}[[{1, 2},{1, 3}]]		Returns the submatrix with listed rows and columns, viz. {{1, 3},{4, 6}}
{{1,2,3},{4,5,6},{7,8,9}}[[All,{1, 3}]]		Returns the submatrix with columns 1 and 3, viz., {{1, 3},{4, 6},{7, 8}}. A similar statement can be used for getting rows
Shallow		Returns a shortened skeleton form of a list for inspection. Options allow for in depth inspection
Short		Returns a one line skeleton of a list
Take[{1,2,3,4,5},3]		Takes the first 3 elements from the list {1,2,3,4,5} to form a new list {1,2,3}
Take[{1,2,3,4,5},{2,4}]		Takes the 2, 3, 4 elements of {1,2,3,4,5} to from the new list {2, 3, 4}
Take[{1,2,3,4,5}, -2]		Takes the last 2 elements from {1,2,3,4,5} to form the list {4, 5}
First, Last, Rest, Most		For extracting the first, last, all but the first, all but the last elements respectively from a list
Length[{1,2,3,4,5}]		The number of terms in the list {1,2,3,4,5}
Position[{1,2,3,4,5},3]		Returns the position of 3 in the list {1,2,3,4,5}
Part[{1,2,3,4,5},{2,4}]		Returns the second and fourth elements of {1,2,3,4,5} to give the new list {2, 4}
Part[x^2 + 3 x + 4,{1,2}]		Returns 4 + 3x
{a, b, c, d, e}[[{1, 2, 4, 3, 5}]]		Returns {a, b, d, c, e}. Useful for making permutations.
Take[{2, 3, 1, 0}, -2]		Gives a list {1, 0} of the last two elements of the given list
Drop[{1,2,3,4,5}, {3,5}]		Drops the 3 through 5 elements of the list to give the new list {1,2}
Extract		Equivalent to Part but interacts better with the output of Position. Extract[A,{{1,2},{3,4}}] is a list of the 1,2 and 3,4 elements of the matrix A.
Range[2.1,14, 3.2]		{2.1, 5. 3, 8,5, 11.7}
Partition[{1,2,3,4,5,6}, 3]		{{1, 2, 3}, {4, 5, 6}} a matrix with 3 columns
Partition[{1,2,3,4,5,6}, 3, 1]		{{1, 2, 3}, {2, 3, 4}, {3, 4, 5}, {4, 5, 6}}. The offset is very useful in constructing points to be connected in a plot. The 1 in the partition gives the speed at which the list is traversed. Other options allow for cycling and padding and overhangs at the beginning and the end.
Flatten[{{1, 2}, {3, 4}, {5, 6}}]		{1,2,3,4,5,6} Removes all braces except outside ones

Flatten[{{{1, 2}},{{3, 4}}}, 1]		{{1, 2},{3, 4}} Removes 1 set of braces starting at the inside
FlattenAt		A method of removing some of the parenthesis from a complicated nested list
Transpose{{1, 2, 3},.{4, 5, 6}}		Gives {{1, 3}, {2, 5}, {3, 6}} which is a useful manipulation for plotting, etc.
Join[{1,2,3},{6,2,4}]		Returns the list {1,2,3,6,2,4}
Union[{1,2,3},{6,2,4}]		Returns the list {1,2,3,4,6}. An option SameTest permits you to specify what redundancies should be eliminated.
Append[list, element]		Adds element to the end of list
Prepend[list, element]		Adds element to the front of list
RotateLeft[list, n]		Cycles elements of list n place to the left
RotateRight		Cycles elements of a list to the right
All		For selecting the everything available. Mat[[All,{1,2,3}]] is the first 3 columns of the matrix Mat
Sort		To sort a list in lexicographic order. Sort admits a function as an option. The function allows sorting in user specified order.
Position[{x^2, 3, y^2},_^_]		Returns the position of objects in the list that match the pattern
Split[{a, a, 1,1,1, b, c, c, 1}]		Splits into successive recurrences {{a, a},{1, 1, 1},{b},{c, c},{1}}

Strings

Print["The input is incorrect!"]		The input is incorrect!
StringForm["The value of `` is ``.,x^3,9]		Prints The value of x^3 is 9.
Fold[SequenceForm[#1," = ", #2]&, a, {b,c,d}]		Returns a = b = c = d as a typeset expression
Subscripted[x[k]] Subscripted[x[s, t, u], {1,2}]		x_k and $x_{s, t}$[u] holdovers from earlier Mathematica editions. Keyboard can be used to insert subscripts and superscripts
Subscripted[x[SequenceForm[a,b]]]		x_{ab}
x Superscript[k]		x^k. There is no Superscripted since a superscript should be interpreted in most cases as a power. Can be typed in using ^ key or Power[x, "a b c"]
SubsuperscriptBox[x,k, j]// DisplayForm		Makes k into a subscript and j into a superscript on x
ToString		Makes a string out of an expression
StringTake		Takes elements out of a string
ToCharacterCode FromCharacterCode		Gives the character numbers of a string and sets the character up from its number. ToCharacterCode["x ≤ 4"] gives {120, 32, 8804, 32, 52}. Note spaces have code 32. FromCharacterCode acts on a list and reverses ToCharacterCode. This is useful in setting up formatting commands in packages.

Data Types

{2, 3, 17}		An ordered set called a List - here a 3-tuples. The List is the primary (perhaps only) data type
Table[expr, iterator1, iterator2, …]		Returns a list of iterates of the expression. Expression can be anything involving one or more parameters The iterator is of the form {i, imin, imax, stepsize} with not all options needed for execution with some additional options. For example {i, 4} is the same as {i, 1, 4}.

Table[s[i, j]. {i, 1,4}, {j, 1, 5}]		Creates a 4 by 5 list
Table[i, {5}]		{i, i, i, i, i}
Table[3i, {i,5,11}]		{15, 18, 21, 24, 27, 30, 33}
TableForm		Causes a table to be listed in a column form, cf. options for this command especially TableDepth which specifies the maximum number of levels
Options[Table]		Many options exist setting up labeled tables
Array[f, {3, 4, 5}]		Creates a nested list of elements f[i1, i2, i3] with 1 ≤ ij ≤ nj Arrays can be any dimension.
Array[List, {3,4,5 }]		Creates a nested list {i, j, k} with 1 ≤ i ≤ 3, 1 ≤ j ≤ 4, 1 ≤ k ≤ 5
Set		Does not seem to be available in Mathematica. To remove repeated elements from list write Union[list]

Patterns

x_		A pattern that matches a single expression. Used for defining functions.
x_Integer		A pattern called Blank that matches a specific type which is an integer. There are many other types like Real, Complex, Rational, etc.
x__ (2 consecutive underbars)		A pattern (called a BlankSequence) that matches one or more expressions, e.g., f[x_,y__]=Plus[x,y] yields f[1,2,3] -> 6 but f[1] -> f[1]
x___ (3 consecutive underbars)		A pattern (called a BlankNullSequence) that matches 0 or more expressions, e.g. f[x_,y___]=Plus[x,y] yields f[1,2,3] -> 6 and f[1] -> 1.
{aa, bb, cc}/.opts/.{aa->True, bb->False, cc->True}		Form for adding optional input for a function f[x_, opts___] with built in defaults. For example, if opts are set as aa -> False, then the control list {aa, bb, cc} will appear as {False, False, True}.
x_?Real		A pattern that matches a single expression when the answer to the inquiry Real? is true
x_/;x > 1		A pattern that matches only if x is greater than 1
x_:1		A pattern that matches an expression and has default 1, e.g., y_^(x_:1) puts 1 in for the exponent on y when y appears without an exponent
x_.		A pattern in which a default is assigned by Mathematica if x is missing. The default is assigned so as not to change the value of the expression in which x appears
f/:f[a_] + f[b_] = f[a + b]		The symbol /: defines a TagSet. Here f[1] + f[2] becomes f[3]
f[a..]		Allows pattern to be repeated any number of times. Cases[{f[a,a,a],f[a,b,a], f[a,a], f[a]},f[a..]] gives {f[a,a,a], f[a,a], f[a]}
FreeQ[Sin[x^2] + 1, _^_]		Matches the pattern _^_. Here returns False
FreeQ[Sin[x^2]+1, _^_ , {1}]		Matches the pattern _^_ at level 1 Here returns True

Graphics

Plot[{Sin[x], Cos[x]},{x,-2,2}]		Graphs sin x and cos x in the given range on the same axes
Clear[h]; h[x_,y_]= (x^2+y^2)^(1/3);		Graphs the surface given by h(x, y). To graph the surface according to a color directive, eg. red, graph {h[x,y], RGBColor[1,0,0]}. If no

Plot3D[h[x,y],{x,-2,2},{y,-2,2}]		color directive is used, Mathematica will disregard other color directives and will choose its own colors for 3D plots shown together.
Plot[{Sin[x],Cos[x]},{x,-2,2}, PlotStyle->{RGBColor[1,0,0], RGBColor[0,1,0]}		Colors sin x red and cos x green
Plot[f[x], {x, 0, 4*Pi}, PlotRange -> {-12, 12},		Plots a previously defined function. The option
DisplayFunction -> $DisplayFunction		DisplayFunction -> Identity aborts
Show[plot1, plot2]		To display previously defined Plot or Graphics structures. Show displays on the basis of overlays with the first plot1 on top. Sometimes the order of the objects in the Show must be rearranged to avoid one from blocking another.
ListPlot[{{1,2},{2, 4},{4,8}}]		Plots points. Other Methods of doing this, eg. Graphics[Point[{1,2}], Point[{2, 4}], Point[{4, 8}]] with various options
Graphics[Text['y = x', {1, 2}, {0,0}]]		Centers text y = x at the point {1 ,2}
Show[GraphicsArray[{{plot1, plot2},{plot3,plot4}}]]		Displays previously defined plots plot1, plot2, plot3, plot4 in a 2 ¥ 2 array.
Rectangle, Circle, Point, Line, Disk		Various so-called Graphics primitives to create various geometric figures. These are to be used Graphics. Line will connect many points in order. See section on Graphics for information on generating coordinates
Cone, Sphere, Cylinder		Various so-called 3D Graphics primitive in the Graphics`Shapes` Package. Must be surrounded by Grahics3D. Draws shapes with given dimensions in standard position. Using a negative number for height.
RotateShape, TranslateShape		To move 3D Shapes which are initially constructed in standard position. In Graphics`Shapes` Package.
Show[Graphics[{GrayLevel[.6], Rectangle[{1,0},{2,1}]}]]		A shaded rectangle with opposite corners. GrayLevel[1] is white while GrayLevel[0] is black.
Options[Plots], Options[Graphics]		To see all the different formats that are available for plotting.
<<Graphics`Legends`		One of many special packages for platting. Other draw vector fields, do implicit plotting, etc. See the Function Browser to see the various possibilities
GraphicsArray		For making a rectangular array of graphic objects
ParametricPlot		For producing parametric plots
3D		Added to plot and graphics commands, e.g. Plot3D
PlotStyle-> {various options}		For adding features to Plot, ParametricPlot and ListPlot
AspectRatio -> a		Plot option calling for ratio of height/width = a. Default is 1/GoldenRatio. In general, Mathematica chooses a suitable scale. AspectRatio -> Automatic calls for an absolute x and y scale
Plot[t^3,{t,0,1}, Epilog->{Pointsize[.03], Point[{0,0}],		Epilog is plotted after the main plot.
Point[{1,1}]] }]		There is also Prolog
Hue, RGBColor, GrayLevel, CKMYKColor		Various functions to control the color of graphics. Can be used with automatic call, eg. ColorFunction -> Hue.

ParametricPlot3D		Graphs 3 coordinates as a function of 2 or 3 variables. A fourth coordinate can give the color directive. To make the color apperar, the option Lighting->False must be used
ParametricPlot3D [{3+t^2,2+t+t^2,175-100/(1+t^2), Hue[1]}, {t,0,17}]		Plots a parametric curve in red
{ DisplayFunction->$DisplayFunction]		For forcing a graphics to be displayed. Used after DisplayFunction->Identity
ImplicitPlot, ContourPlot		For graphing functions given implicitly. ImplicitPlot requires a special package.
TextStyle		An option in plot commands to control the text, eg., TextStyle -> { FontFamily -> "Times", FontSize->10, FontWeight -> "Bold"}
Lighting		Lighting ->False will permit user defined colors to appear in Show of several 3D graphs. Lighting -> True is the default and allows Mathematica to choose contrasting colors without regard to user choice
AmbientLight		Specifies the direction of the light sources which is reflected by Lambert's Law
SurfaceColor		A function that allows user defined 3D plots to be colored. Set up as Graphics3D[SurfaceColor[RGBColor[.45,.45,0], RGBColor[.2,.2,0], 10], your3Dgraph] where the second and third variables in SurfaceColor relate to the specular color and are optional.
FullOptions		Useful for discovering how Mathematica is setting its automatic options. For example, FullOptions[plt[1], PlotRange] which show the plot range of plt[1].
PlotRange -> All		Forces Mathematica to put all objects into graph. Sometimes Mathematica will choose a PlotRange that leaves out points.
FullOptions[yourgraphname, PlotRange]		Shows the PlotRange for the plot yourgraphname. This will work to reveal any option setting used in any output.
MeshRange->{{0,1},{2, 3}}		Changes the x and y coordinates to run between 0 and 1 and 2 and 3 on the x and y axes respectively. Useful in graphing various matrix inputs which are graphed by default on intervals [0, m] by [0, n] for matrices of dimensions {m, n}.
Mesh->False		Removes the boundary of the polygons used to draw 3D surfaces. Also cf. EdgeForm.
EdgeForm		To draw edges of polygons used in 3D graphics
SurfaceColor		Directive for 3D graphics which permits coloring and reflection off the surface. Mathematica will give a surface color using its default unless Lighting -> False is invoked.
<<RealTime3D`		Loading this experimental package permits changing the ViewPoint in 3D graphics by manipulating the graphics with the mouse
<<Default3D`		Restores the default for manipulating the ViewPoint after RealTime3D` has been loaded

Calculus

f[x_]= x*Sin[2*x]*Exp[x^2]		Note x_ in the function argument.

Exp[x]		The exponential function ex
Log[x]		ln x. Logarithms to other bases are available.
f ’ [x]		Derivative of a function with an apostrophe Higher order derivatives can be obtained by ‘’, etc.
Derivative[2, 3] [f] [x,y]		The derivative of a function f[x_,y_] twice with respect to the first variable x, 3 times with respect to the second variable y
D[f[x,y],{x,2},{y,3}]		The partial derivative of f[x_,y_] twice with respect to x and three times with respect to y
D[x*Sin[2*y]*Exp[x^2],{x, 2},{y,3}]		Useful for an expression
Dt[[x*Sin[2*x]*Exp[x^2],x]		Assumes y is a function of x.
D[x^2 + y[x]^2 ==1,x]		Returns 2x + 2y[x] y’[x] = 0
D[x^2 + y[x]^2 ==1, x, Nonconstants–>{y}]		The same as the preceding
Integrate[x^2, x]		Indefinite integration
⁝int⁝ control^ 2 control spacedd x		$\int x^2$ dx set in symbols and mathematically active
⁝int⁝ (control ^) 3 (control %) (control space) x (control ^) 2 ddx		The definite integral or type Integrate[x^2, {x, 1, 3}] // TraditionalForm
Integrate[y*Sin[x] -x*Sin[y], {x,0,1},{y,1,2}]		For definite integrals. Iterated indefinite integration does not seem to be implemented and must be done by hand or some other type of recurrence
Integrate[Integrate[x^2*y,x],y]		For iterated indefinite integrals
NIntegrate		A numerical integrator with many options for controlling accuracy, dividing interval into several pieces, etc.
FindRoot[f[x], ==a, {x,x0}]		Starts the Newton–Raphson algorithm on f[x] at x0. This will also solve systems of equations. Syntax needs to be carefully prepared
Expand Collect Exponent PolynomialQuotient PolynomialRemainder CoefficientList		Tools for dealing with polynomials. Exponent is the highest degree in a specified polynomial variable.

Series

Sum[k^3,{k,1,n}]		Mathematica will try to compute a closed form. First load <<Algebra`Symbolic Sum
Series[f[x],{x, a, n}]		Computes first n terms of the Taylor series of f[x] in the variable x about the point a. The last term O[x - a]^(n + 1) represents the omitted terms
Normal[%]		Removes the Big Oh from the preceding power series
Series[Cos[x],{x,0,6}]//Normal		The first 6 terms of the Taylor polynomial of cos x about x = 0

Limits

Limit[Sin[x]/x, x ->0]		Form of Limit
Limit[(x + 2)/(2*x + 1), x-> Infinity]		Will attempt this evaluation.
Limit[Sqrt[x^2]/x, x->0, Direction->-1]		Direction -> -1 takes the limit from above and Direction ->1 takes the limit from below. Default uses Direction -> -1

Differential Equations

DSolve[eqn, y[x],x]		The option for this is the initial conditions

Linear Algebra

matrixa ={{1,2,3},{3, 4, 5}}		A matrix is identical to a rectangular list of lists and any operation on a list of lists is a matrix operation
Table		Can be used to create a matrix
Partition		To enter a matrix as a list and then partition cf. Control
a//MatrixForm		Displays a in rectangular form. Use a non proportional font like Courier to keep columns straight.
matrixa = Array[a, {2,3}]		Defines a 2 ¥ 3 matrix with i, j entry denoted by a[i, j]
Array[f, {i1,i2},{j1, j2}];		Gives a matrix with i, j entry f[i, j] where f is a function and i and j run between i1 ≤ i ≤ i2 and j1 ≤ j≤ j2 where 0 < i1 ≤ i2, 0 < j1 ≤ j2
Table[Subscripted[b[i],{i,1,3}]]		Returns {b1, b2, b3}
2*a + a.b		Matrix operations
Det[a]		Determinant
Inverse[a]		Finding the inverse of a square matrix
Transpose[a]		Transpose of the matrix a
{1,2,3,4}		A vector or a 1 ¥ 4 matrix
Reaching inside a matrix		These are the same as those given in the Control section for lists of lists
v.w		The dot product of the vectors v~ and w
<< LinearAlgebra`MatrixManipulation		Loads package to do the succeeding manipulations
RowReduce[a]		Returns complete row reduce form for a matrix a
AppendRows[a, b]		Appends the row b to the matrix a.
AppendRow[a, {{1},{2},{3}}]		Appends the column given by {{1},{2},{3}} to the 3 ¥ n matrix a
CharacteristicPolynomial[a, x]		Returns the characteristic polynomial of of the square matrix a in the variable x
Eigenvalues[a]		Returns eigenvalues of the square matrix a
Eigenvectors[a]		Returns the eigenvetors of the square matrix a
Eigensystem[a]		Returns the eigenvalues and eigenvectors of of the square matrix a

LinearAlgebra`Orthogonalization`		This package contains the GramSchmidt process

Linear Calculus

Grad		The gradient in the <<Calculus`VectorAnalysis package. Works only for 3 coordinates
Div		Divergence in same package as Grad. For more than three coordinates you must write your own program

Programming

Do[Body, {i, imin, imax, step}]		Performs the iteration Body which depends on the parameter i from start imin to end imax in steps step. Not all the parameters need to be present.
For[start, test, incr, body]		Executes start then repeatedly executes body until test fails to return True
If[condition, t, f, u]		Returns t if condition is true and f if the condition is false, and u otherwise. If u is not specified nothing is done.
Module[{x, y, …}, body]		This is the main programming tool. The list {x, y, … } are so–called local variables or assignments used only inside the module and body is some sequence of Mathematica commands (all except the last punctuated with a semi-colon) that are to be executed. Also assignments like x = 1, etc in the local variable declaration are permitted
Block[{x, y, …}, body]		Like Module with differences in way the local variables are loaded. In Module, local variables are assigned a unique name, the Module is rewritten in terms of the unique names, and the new Module is evaluated with the input. In Block, local variables do not get a unique name. Module can be used to evaluate an expression with special assigned values that will not persist outside the Block
With[{x = a, y= b}, expr]		Evaluates expr with substitutions in {}. Items like x, y remain variables outside of With.
f[x_]:=Module[{i},Sum[x^i,{i,1,5}]]		Yields f[i] → i + i^2 + i^3 + i^4 + i^5
g[x_]:=Block[{i},Sum[x^i,{i,1,5}]]		Yields g[i] → 3413
With[{x,y, … }, body]		Functions like Module or Block but allows for importing or exporting assignments and local variables
While[test, body]		While test is true, body is evaluated
clear[f]		
f[x_]:= 0 /; x <=0		A Conditional evaluation for a piecewise linear
f[x_]:= x /; (0 < x && x < = 1)		function. Note && which is And. Mathematica
f[x_]:= 1 /; 1 < x		does not accept 0< x <= 1. Or is given by ||.
Which[test1, value1, test2, value2, …]		Evaluates test1 and evaluates value1 if test1 is true; if test1 is false, evaluates test2 and if test2 is true, evaluates value2. Continues until first match is true.
Switch[expr, form1, value1, form2, value2, …]		Evaluates expr and compares it to form1 and evaluates value1 if there is a match. Switch continues until the first match is found.
opt___?OptionQ		Used to get Mathematica to recognize the options that go with a user

		defined function. This should be the last variable in the function and consists of one or more rules.
FilterOptions		Used to make a sublist of options from a user-defined function for insertion in a Mathematica function. In Utilities` package.

Revision 19 on 3/06