Lecture 2: Introduction to Mathematica

Expressions and Evaluation

There are very many ways to learn how to use MATHEMATICA® . Nearly all of the best ways involve performing examples from the very beginning. That is how we are going to start—with examples. Using MATHEMATICA® 's *FrontEnd* you may execute a command by pressing Shift-Enter; simply pressing Enter tells MATHEMATICA® 's that you merely wish to have a "carriage return" on the screen.

Mathematica's syntax will feel fairly natural after a while. Use the following notebook to get started. Execute a few commands until you get a sense for what output MATHEMATICA® will produce; try editing the commands; try to make MATHEMATICA® do something strange—just try playing with it and you will soon get the hang of what is going on.

One way to use MATHEMATICA® is simply as a calculator that allows symbols to get carried along. MATHEMATICA® will usually try to resolve every symbol and return precise information about it. If something is undefined to MATHEMATICA® , it simply returns it as a symbolic expression.

A number is not returned until all of the symbols in an expression are defined as numbers. MATHEMATICA® will try to be exact—it does not calculate $\frac{1}{3} + \frac{1}{2}$ by adding $0.33333\cdots + 0.5 = 0.83333\ldots$, it has an algorithm for adding rational numbers and gives $\frac{5}{6}$.

Lecture 02 MATHEMATICA® Example 1

Getting Started

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

There are a variety of ways to get MATHEMATICA® started and these are specific to the operating system your computer uses. A license must be purchased to run MATHEMATICA® code, but free MATHEMATICA® -display tools can be obtained from Wolfram.

The FrontEnd is the graphical interface between the user and MATHEMATICA® —you arrange your MATHEMATICA® input, sometimes with text-like comments, in the FrontEnd. The user must request the FrontEnd to pass something to MATHEMATICA® 's *kernel*, by pressing Shift-Enter. The kernel is the resident symbolic algebra software engine behind MATHEMATICA® .

The appearance of the FrontEnd depends on either provided or user-designed *StyleSheets*. The StyleSheet for this course can be downloaded from the course website. The course style is particulary ugly—it is hoped that this will provide an incentive for students to create their own style.

First you must locate or obtain Mathematica and permission to use it.

The colors on your sceen may not appear to be the same as what is presented in the lectures. I use my own style sheet, you may download my style sheet from http://pruffle.mit.edu/. Information on where to put the style sheet can be found below. Let's get started with some simple *Mathematica* commands.

If you are reading this in the *Mathematica* FrontEnd, then you can go ahead and familiarize yourself with some basics by executing the following lines which are *Mathematica* Input. Typically, *Mathematica*'s FrontEnd asks *Mathematica*'s Kernal to do its job of evaluating by hitting SHIFT—ENTER while the mouse in an "Input Cell." Input Cells can be identified with the ↧-thingy at the right. One can evaluate one or more cells by selecting their ↧-thingies and hitting SHIFT—ENTER.

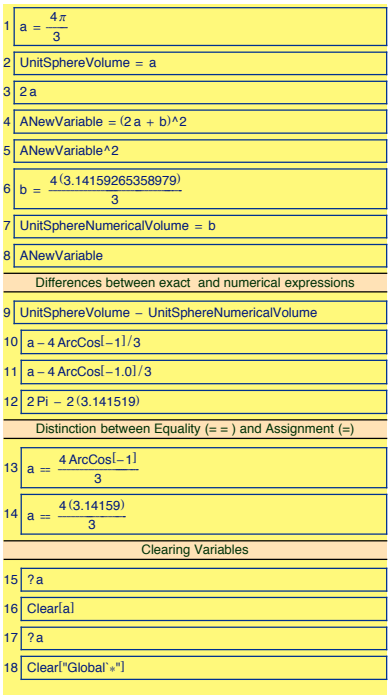
Try executing the examples given below. Try to guess what the output might be or represent—and observe carefully whether *Mathematica* is doing what you would anticipate. Notice that answers can depend on the history of commands that precede it.

Lecture 02 MATHEMATICA® Example 2

Basic Input and Assignment

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

The methods of assigning expressions (`expr`) to symbolic variables (`SomeVariable`) via `expr = SomeVariable`. Differences between exact (symbolic) objects and numerical objects. Logical equalities (`==`) and Clearing symbols. Many bugs crawl into MATHEMATICA® from "uncleared" symbols.



1: A symbol is assigned to an expression with an equals sign `=`. Some symbols, such as π, are already defined—in MATHEMATICA® it is *exactly* the ratio of a circle's circumference to its diameter.

2: If a previously assigned symbol is used in a new assignment, MATHEMATICA® will usually incorporate the previously assigned symbol's properties. Here, the variable is given a descriptive and long name: this is good practice if you want your code to self-document. Your definitions can be recalled and used to build up more complicated expressions.

6: It is possible to assign symbols to numberical objects; here `b` is a different kind of object (numerical) than the previously defined `a` (symbolic). There are built-in object-types created from combinations of other types.

10: MATHEMATICA® will to be exact—the `ArcCos` of -1 is exactly π.

11: However, the `ArcCos` of -1.000 is a numerical approximation to π—the result will be numerical (not exact).

13: A `Rational` expression is another type of MATHEMATICA® object. There are built-in rules to treat rational expressions exactly. The result of applying `N` to the rational $\frac{5}{6}$ is a numerical approximation to 5/6.

14: Assignment `=` is *different* from *logical equality* `==`. For a logical equality, the result is either `True`, `False`, or MATHEMATICA® cannot determine and returns an expression.

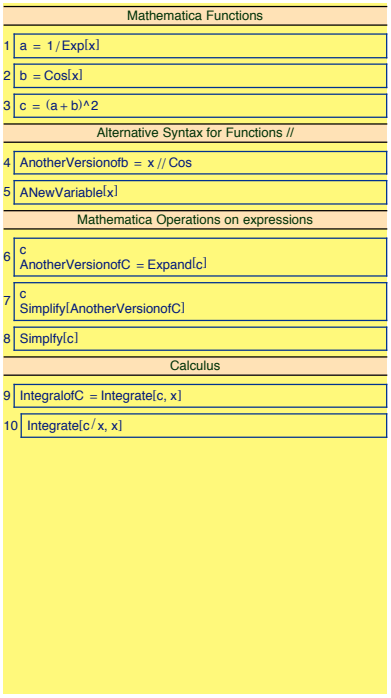
17: It is possible to clear *all* user-made definitions. A last resort would be to instruct the FrontEnd to kill MATHEMATICA® 's kernel and restart it. clearing symbols.

Lecture 02 MATHEMATICA® Example 3

Built-in Functions and Operations on Expressions

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

MATHEMATICA® has many, *very many*, built-in mathematical functions; it's probably impossible to memorize all of them. You can usually find what you need by using the Help Browser, or by querying `?` with a wildcard `*` such as in `?*Geom*`.



1: MATHEMATICA® has a fairly consistent function naming strategy The first letter of a word is always captialized; compound words are concatenated together while maintaining the first letter capitalization; thus `InverseBetaRegularized`. A function is just another symbol—if a symbol is followed by square brackets `[]` the stuff inside the brackets become the argument(s) for the function.

4: There are usually more than one way to do things. The operator `//` is a short-hand way of applying the function that follows `//` to the expression that proceeds it (e.g., `(Pi/2)//Sin`). You can also use `@` to prefix a function (e.g. `Sin@(Pi/2)`).

6: There are methods designed to improve or alter the appearance of complicated expressions.

8: Spelling errors can be very difficult to spot and debug—as a matter of practical advice, look for MATHEMATICA® 's spelling warnings. They can be turned off with `Off[General::spell]`, but it is not generally a good idea.

9: MATHEMATICA® has a sophisticated symbolic integration algorithm...

10: even if sometimes it expresses it in special functions

Lecture 02 MATHEMATICA® Example 4

Calculus and Plotting

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

The derivative and integration methods are introduced. Simple plotting methods are demonstrated with an example of annotating a plot.

2: If MATHEMATICA® can't differentiate or integrate a function, it will be left in a symbolic form.

5: MATHEMATICA® applies the fundamental theorems of calculus...

10: The calculus operations will often create long and complicated expressions. That two expressions are equivalent can *sometimes* be shown with built-in functions such as `Simplify`, `FullSimplify`, `Factor`, `Expand`, `Collect`, etc., but sometimes it is an art to turn an expression into an aesthetic form.

12: This is the simplest form of `Plot`. The second argument is a list giving the variable and its bounds. The first argument should have a numerical value at most of the points within the variable's bounds.

15: To find all the possible options for a function with their default values, the `Options` function provides a way to decipher what aspects of a plot can be changed easily. Demonstration of a function that MATHEMATICA® does not know how to integrate or differentiate.

16: Here is an example with a plot title, axes labels, different colors and thickness for the curves.

```
1  ?ExpIntegralEi
2  D[ANewVariable[x], x]
3  Integrate[ANewVariable[x], x]
4  D[ANewVariable[x], z]
5  tempvar = Integrate[ANewVariable[x], {x, 0, y}]
6  D[tempvar, x]
7  D[tempvar, y]
8  Factor[IntegralofC]
9  IntegralofC
   AnotherVersionofIntegralofC =
     Integrate[AnotherVersionofC, x]
10 c
   D[IntegralofC, x]
11 Factor[c]
   Simplify[D[IntegralofC, x]]
```

Plotting Functions

```
12 Plot[IntegralofC, {x, 0, 10}]
13 Plot[{IntegralofC, c}, {x, 0, 10}]
14 Plot[c, {x, 0, 10}, PlotRange → {0, 0.0001}]
15 Options[Plot]
16 Plot[{IntegralofC, c}, {x, 0, 10},
     PlotStyle → {{RGBColor[1, 0, 0], Thickness[0.005]},
         {RGBColor[1, 0, 1], Thickness[0.0075]}},
     TextStyle → {FontFamily –> "Helvetica", FontSize → 24},
     PlotLabel –>
       " A Function (Purple)\nand Its Integral (Red)\n",
     AxesLabel → {"Value", "Argument"},
     ImageSize → 800
   ]
```

Lecture 02 MATHEMATICA® Example 5

Lists, Lists of Lists, and Operations on Lists

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

Lists are useful ways to keep related information together, and MATHEMATICA® uses them extensively. Lists could be created in MATHEMATICA® by using the `List` function, but they are usually entered in with curly-brackets {}.

3: Some functions, such as `Cos` here, are *threadable functions*; when called on a list-argument, they will produce a list of that function applied to each list element.

4: A list's parts (or, elements) can be picked out in a variety of ways. The `Part` function has a shorthand double-bracket form.

7: There are plenty of functions designed to operate on lists.

8: Logical operations, such as `NumberQ`, can be used to select elements by their characteristics.

13: A list's elements can be lists themselves. For example, a matrix is represented by list of a list. And their are higher-dimensional structures such as tensors. `Dimensions` is a useful way of learning about such structures.

14: Here, the *post-fix operator* for a function is used to change the way a matrix is displayed. *Note, the result is* ***not*** *a matrix, but a* `DisplayForm` *of a matrix.*

22: This is a fairly advanced example of extracting the odd-numbered columns of a matrix. The list `IntList` is simply the integers for each column; its odd-numbered members are selected and become the second (i.e., column) argument of the `Part` selection. The first argument is `All`, so the entire row is captured for each selected column.

Lists {} and Matrices {{}} (Lists of Lists)

```
1  AList = {a, b, 2, 7, 9, 1.3, π/2, 0}
2  Length[AList]
3  Cos[AList]
4  AList[[2]]
5  AList[[{3, 6}]]
6  AList[[-2]]
7  Sort[AList]
8  Select[AList, NumberQ]
9  Reverse[Sort[Select[AList, NumberQ]]]
10 Select[AList, EvenQ]
11 Select[AList, PrimeQ]
12 Perms = Permutations[Select[AList, ExactNumberQ]]
13 Dimensions[Perms]
14 Transpose[Perms] // MatrixForm
15 TranPerms = Transpose[Perms];
16 TranPerms[[3]]
17 TranPerms[[4, 1]]
18 TranPerms[[1, 4]]
19 TranPerms[[5, 1]]
20 TranPerms[[{1, 2}]]
21 IntList = Table[i, {i, 1, Length[TranPerms[[1]]]}]
22 TranPerms[[All, Select[IntList, OddQ]]] // MatrixForm
```

Lecture 02 MATHEMATICA® Example 6

Rules (→) and Replacement (/.)

Download notebooks, pdfs, or html from http://pruffle.mit.edu/3.016-2006.

A rule `leftvar` → `rightvar` is *similar* to assignment in that it associates a new symbol (`leftvar`) with something else, but it the value is not assigned—it does not effect future values of the left-hand-side symbol. Rules are often used in conjunction with replacements. Many of MATHEMATICA® functions, (e.g., `Solve`) return rules as a result.

1: The *rule* `a` → $\pi/3$ is *assigned* to the symbol `ARule`

3: A rule can be applied with the function Replace, but the syntax (.) is typically used instead.

4: Rules can be collected into lists.

5: Assigment of `a` is reflected in the form of `ARule`.

8: Rules are necessary for manipulations in MATHEMATICA® , but can be used to generate "mistakes." Think of Rule and Replace acting on an expression as "What would the expression be if a certain rule were applied to it?" If the rule is wrong, the resulting expression will be as well.

Rules → and Replacement /.

```
1  ARule = a → π/3
2  a
3  AList
   AList /. ARule
4  SomeRules = {ARule, b -> π/12}
5  AList /. SomeRules
6  a = SomeOtherSymbol;
7  AList
8  StrangeRule = {Rational[x_, y_] :> y/x}
9  (AList /. SomeRules) /. StrangeRule
```

Getting Help on Mathematica

MATHEMATICA® 's built-in help functions are very useful. This was true even before the whole MATHEMATICA® manual was incorporated into the Help Browser. In the old days, one would memorize large portions of the MATHEMATICA® book—which has grown continuously heavier since its first publication in the early 1990's—and rely on the useful "?" and "??" operators. The use of "?" with the wildcard "" enabled a beginning user to track down almost any MATHEMATICA® function. The Options function is also a very efficient way to discover alternative ways of getting results.

I would have recommended 'scanning' the *entire* MATHEMATICA® manual in a single three hour sitting (about 600 pages per hour) as an effective way to acquire a working familiarity with the software, but I don't because the built-in browser is so easy to use.

I encourage you to idly explore the MATHEMATICA® Help Browser. You will not only learn about MATHEMATICA® , but also about mathematics.