

Introduction to *Mathematica*

Type your name here.

This document follows many of the examples from: <http://www.wolfram.com/language/fast-introduction-for-programmers/>

You are responsible for completing all of the exercises, designated by ■ **Exercise**.

Notebook Documents

What you are looking at is called a *Mathematica notebook*, which mixes text, graphics, interfaces, etc. with code.

Notebooks are organized into **cells**, indicated by brackets on the right: →

Double-click a cell bracket to open or close a group of cells:



Click between cells to get a horizontal insertion bar to create a new cell:



Then just start typing!

■ Exercise

Create a new cell below and type $2+2$.

Mathematica distinguishes between **input** cells and **text** cells. Currently, this is a text cell. An input cell has a different font (normally) and looks

like:

```
3 * 5 / 4 + 6
```

You actually created an input cell above when you typed $2+2$. (Notice the different font.) We will focus on input cells because this is where we will be typing our executable code. To execute code in an input cell, you must press SHIFT+ENTER.

■ Exercise

Evaluate the input cell below.

```
3 + 5 / 4 + 6
```

If all went well, you should get something that looks like this:

```
In[16]:= 3 + 5 / 4 + 6
```

```
Out[16]=  $\frac{41}{4}$ 
```

```
]]
```

The `In[16]` is just a reference to the input cell. The `Out[16]` is a newly created **output** cell. Note that your numbers might be different than 16. The 16 just refers to the fact that I have executed 16 things in *Mathematica* in this session.

You can delete cells by highlighting the cell bracket and pressing DELETE.

■ Exercise

Delete the cell below.

```
you better delete me !
```

Common Operations

Let's discuss some common tasks that you will perform in *Mathematica*.

Order of Operations

Mathematica is like a big calculator. You need to tell it what operations to use: addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (^). Parentheses matter! Never forget PEMDAS.

For example, I can enter the expression $\frac{1}{5} + 2 \cdot 5^4$ as:

```
1 / 5 + 2 * 5 ^ 4
```

■ Exercise

Create a new cell below and type the expression $\frac{17+49}{102^5 - \frac{326}{7}}$ using only PEMDAS operations. Be careful with your parentheses.

Comments

Any code that is between (* and *) will be ignored by *Mathematica*. We can use this to create comments in code if we wish. For example:

```
(* this is a comment, I am adding up four numbers with the code below *)
```

```
3 + 4 + 5 + 6
```

```
18
```

■ Exercise

Create a new cell below and copy/paste the code you used in the previous exercise. Above that code add a comment of, "I love *Mathematica*!" Execute your code.

Numerical Approximation

Mathematica tries its best to do everything using exact arithmetic:

```
Sqrt[3] + Sqrt[4]
```

$$2 + \sqrt{3}$$

Sometimes you may want the decimal approximation of something. To do so, you use the `N` function. It takes one argument, and gives the numerical expression of that argument. For example:

```
N[Sqrt[3] + Sqrt[4]]
```

```
3.73205
```

You can get more decimals by using a second argument. `N[expr, n]` attempts to give the numerical value of `expr` using `n`-digit precision. For example,

```
N[Sqrt[3] + Sqrt[4], 20]
```

```
3.7320508075688772935
```

■ Exercise

Create a new cell below and compute the numerical value of $\frac{17+49}{102^5 - \frac{326}{7}}$ using 15 digit precision.

Mathematical Constants and Mathematical Functions

The mathematical constant $e = 2, 71828 \dots$ is represented as `Exp[1]` in *Mathematica*. Natural log (ln), also known as log base e , is represented as `Log`:

```
N[Exp[1]]
```

```
2.71828
```

```
Log[Exp[1]]
```

```
1
```

You can change the base of the logarithm by including an additional argument to `Log`. For instance, `Log[b, z]` gives the logarithm of `z` base `b`, which is represented mathematically as $\log_b(z)$.

```
Log[10, 1000000]
```

```
6
```

■ Exercise

Create a new cell below and compute a numerical approximation of $\log_7(9)$.

The mathematical constant $\pi = 3.14592 \dots$ is represented as `Pi` in *Mathematica*.

```
N[Pi]
```

```
3.14159
```

Mathematica has lots of built-in mathematical functions. A big list can be found here. Here are some examples:

```
Sqrt[x^2]
```

$$\sqrt{x^2}$$

```
Abs[3 + x]
```

```
Abs[3 + x]
```

```
Sin[3 * Pi]
```

```
0
```

```
ArcTan[1]
```

```
 $\frac{\pi}{4}$ 
```

■ Exercise

Create a new cell below and compute $\arcsin(\sqrt{3}/2)$.

Hiding Output

Sometimes you may want to evaluate some code and hide the output. To do this, you include a semi-colon (;) at the end of your input. Compare:

```
2 + 2;
```

which has no output when you evaluate it, to

```
2 + 2
```

```
4
```

which has an output when you evaluate it.

Assigning Variables

A **variable** or scalar is a storage location and an associated symbolic name (an identifier) which contains some known or unknown quantity or information, a **value**. Values can be assigned to variables using =, For example, to assign the value 5 to the variable a, we do:

```
a = 5
```

```
5
```

I would suggest when doing variable assignments that you include a semi-colon at the end to hide the output. This keeps the notebook less cluttered:

```
a = 5;
```

Whenever we use the expression **a** in our input, *Mathematica* will replace it with 5:

```
a + 3
```

```
8
```

At any point, we can change the value **a** has by evaluating a new assignment:

```
a = 3;
```

```
a + 3
```

```
6
```

An “undefined variable” is called a symbol in *Mathematica*. Hence “undefined variables” just stand for themselves:

```
x + 5
```

```
5 + x
```

Variable names can be any number of characters long. It is conventional to start variable names with lowercase letters and use upper case letters for each additional word.

```
thisIsACoolVariableName = 4;
```

You cannot add spaces between variable names. *Mathematica* interprets a space as multiplication. For example, the following is interpreted as the variable `spaces` times the variable `are` times the variable

Bad:

```
Spaces Are Bad
```

```
Are Bad Spaces
```

■ Exercise

Create a new cell below and assign the number 10 to the variable `a` and add it to 356.

Be careful when working with variables. If we have variables `c` and `d` and we want to compute the product, we must write `c*d` rather than `cd`. This is because Maple will interpret `cd` as a *variable* but `c*d` as a *product*.

```
c = 2;
```

```
d = 4;
```

```
cd
```

```
c * d
```

```
cd
```

```
8
```

Until you get comfortable with *Mathematica* syntax, I suggest always using a `*` symbol to designate multiplication rather than using a space:

```
(* USE THIS *)
```

```
c * d
```

```
(* INSTEAD OF THIS *)
```

```
c d
```

```
8
```

```
8
```

I also suggest never assigning values to the variables `x`, `y`, or `z`.

Unassigning Variables

Not very common, but sometimes you want to clear a variable and turn it back into a symbol. You can do that by using the `clear` function:

```

a = 2;
a + 2
Clear[a]
(* notice the output is no longer 4 *)
a + 2
4
2 + a

```

Functions

A **function** is a routine that performs a specific task. It has inputs and outputs.

Built-In Functions

Mathematica has about 5000 built-in functions. All have names in which each word starts with a capital letter. Some we saw earlier: `Sqrt`, `Sin`, etc... The `Integrate` function helps us calculate integrals:

```
Integrate[x^2, x]
```

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

■ Exercise

Judging from the input and output above, what do you think the *Mathematica* code `Integrate[x^2, x]` does? What is the significance of the x^2 versus the x ? Type your answer below.

Answer:

An **argument** to a function is a specific input to the function. In the `Integrate` function above, there are two inputs. The first input is x^2 and the second input is x . Arguments to functions are always separated by commas.

■ Exercise

How many arguments does the function `Range` have in the example: `Range[1.2, 2.2, 0.15]`? Type your answer below.

Answer:

Making Your Own Functions

A mathematical function is a rule that assigns outputs for each input. For example $f(x) = x^2$ is a function named f that takes x as input and outputs x^2 . In *Mathematica*, a function is designated using the `SetDelayed` operator (`:=`) like so: `functionName[input_] := output`. For example,

```
f[x_] := x^2;
```

The above command assigns to f the function that takes x as input and gives x^2 as output. Now we can use f like we would if we were doing math by hand. For example, if we wanted to “plug-in 3 in to f ”

we would do

```
f[3]
```

```
9
```

Keep in mind you need to tell Maple that we are using a function. If I had done,

```
f = x^2;
```

instead, *Mathematica* thinks of f as an *expression* rather than a *function*. If I try,

```
f[3]
```

```
x2[3]
```

we see *Mathematica* returns a weird output which is not what we expected.

■ Exercise

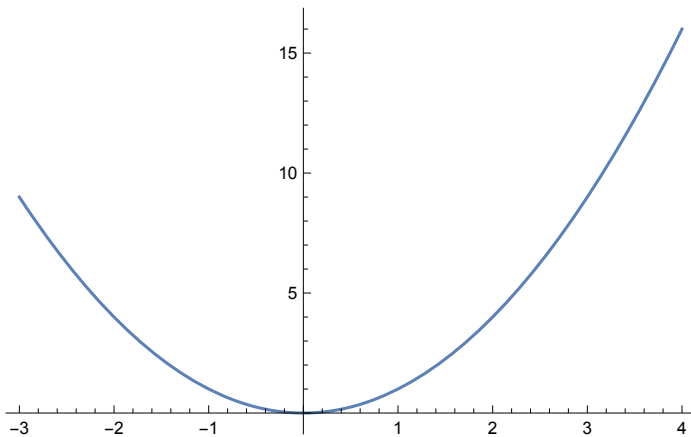
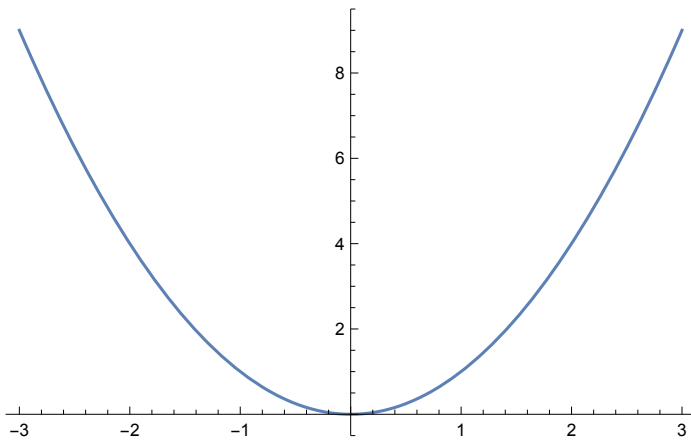
Create a new cell below. Create a function g that takes y as input and outputs \sqrt{y} . Test is by computing $g(3)$.

■ Exercise

Compare the output of $h[3]$ and $h[4]$ below. Describe the output of the function h in terms of the input $b \geq 0$.

Answer:

```
h[b_] := Plot[x^2, {x, -3, b}];
h[3]
h[4]
```



Making Substitutions

In *Mathematica*, you will work with lots of things called expressions. Below, we assign the expression x^2 to f :

```
f = x^2;
```

As we saw above, to plug-in $x = 3$ we cannot simply do $f[3]$ because to *Mathematica*, f is not a function, it is an expression. To substitute in to expressions, we use the `Replace` command which has the form `ReplaceAll[expression, rules]`. A rule takes the form `lefthandside -> righthandside` and represents a rule that transforms `lefthandside` to `righthandside`. For example, if we want to plug $x = 3$ into the expression f , we do

```
ReplaceAll[f, x -> 3]
(* or *)
ReplaceAll[x^2, x -> 3]
9
9
```


■ Exercise

Use the code in the cell below to plug $y = 4$ in to the expression stored in the variable `someExpression`.

```
someExpression = Sqrt[y] * Pi / 4;
```

Lists

A **list** is a finite ordered collection of values, where the same value may occur more than once. In *Mathematica*, lists are indicated by curly braces, and each value is separated by a comma. They can contain any kinds of expressions, and even images:

$\{3, 4, 5, 7/8, x, y, x^2 + 3x^3, \{a, b, c\},$  $\}$

■ Exercise

Create a new cell below that includes a list of the first 10 numbers of the Fibonacci sequence.

■ Exercise

Execute the input cell below. Then, answer the following. Type your answers below.

1) Describe the output.

Answer:

2) What do you think the `Fibonacci[n]` command does (where n is a natural number)?

Answer:

3) Make a guess as to what the `Table` command does.

Answer:

```
Table[Fibonacci[n], {n, 10}]
```

■ Exercise

How many arguments does the function `Integrate` have in the example:

`Integrate[x^2, {x, 0, 4}]`? Type your answer below.

Answer:

Parts of lists are indexed starting at 1, and can be extracted using `[[...]]`. For example, in our Fibonacci list, we can extract that 6th entry of the list like so:

```
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55}[[6]]
8
```

Negative indices count from the end:

```
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55}][[-2]]
34
```

Using double brackets in this way is a shorthand version of the command **Part**.

■ Exercise

Highlight and copy the following list:

```
{3, 4, 5, 7/8, x, y, x^2 + 3 x^3, {a, b, c},
```



Create a new cell below. Assign the list to the variable `myList`. Retrieve the fourth element from `myList`.

A fast way to create lists is using the **Table** command. `Table[expression, {n, max}]` generates a list of the values of `expression` when `n` runs from 1 to `max`. We used the **Table** command above to get the first 10 numbers in the Fibonacci sequence:

```
Table[Fibonacci[n], {n, 10}]
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55}
```

The command above creates the following list

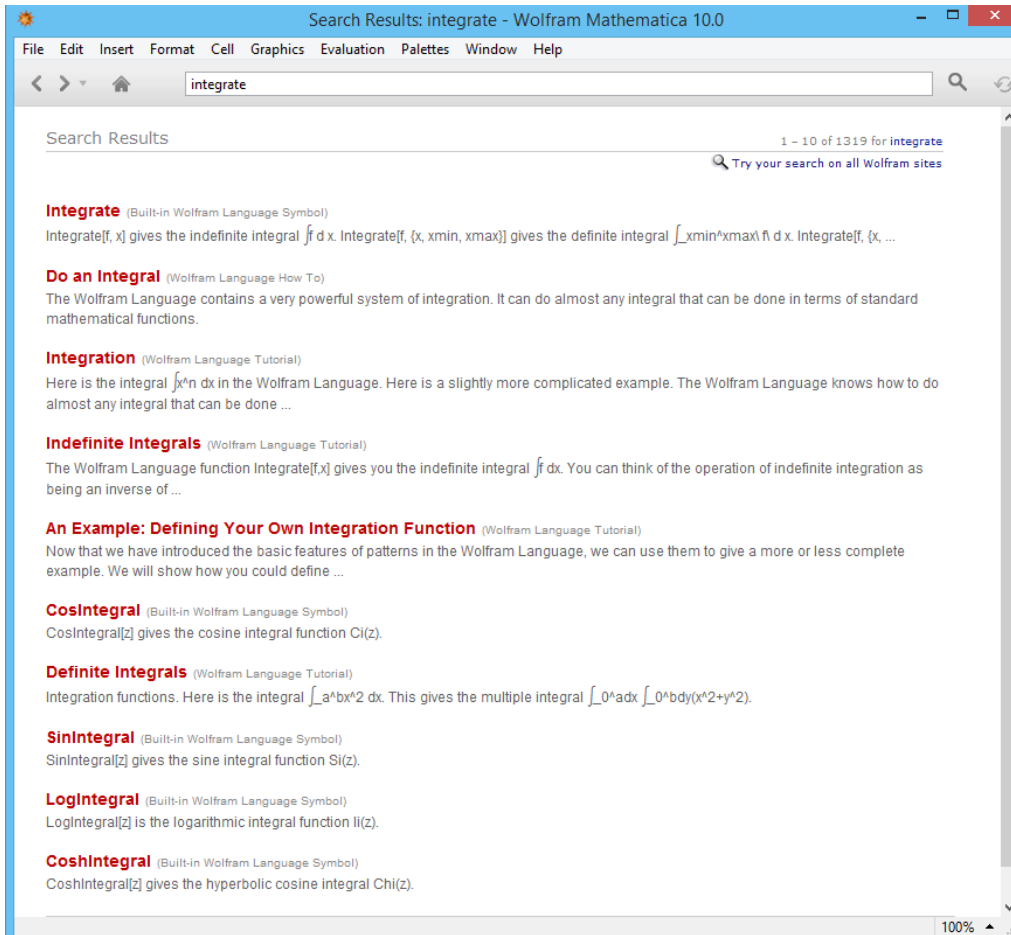
```
{Fibonacci[1], Fibonacci[2], Fibonacci[3], Fibonacci[4], Fibonacci[5],
 Fibonacci[6], Fibonacci[7], Fibonacci[8], Fibonacci[9], Fibonacci[10]}
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55}
```

where `Fibonacci[n]` gives the `n`th Fibonacci number. Notice that `n` increased from 1 to 10 (max).

Help Documentation

The help documentation in *Mathematica* is great. To access it, look for a Help menu in the toolbar and access Wolfram Documentation in the submenu. You can also highlight any text and press F1 to do a search for that text.

For example, when we search for integrate:



we see the first result is the `Integrate` command. Clicking it leads us to this help file:

Integrate (\int)

UPDATED
show changes

`Integrate[f, x]` (1)
gives the indefinite integral $\int f dx$. (2)

`Integrate[f, {x, xmin, xmax}]`
gives the definite integral $\int_{x_{min}}^{x_{max}} f dx$.

`Integrate[f, {x, xmin, xmax}, {y, ymin, ymax}, ...]`
gives the multiple integral $\int_{x_{min}}^{x_{max}} dx \int_{y_{min}}^{y_{max}} dy \dots f$.

`Integrate[f, {x, y, ...} ∈ reg]`
integrates over the geometric region *reg*.

Details and Options

Examples (85)

Basic Examples (7)

Indefinite integral:

In[1]:= `Integrate[1/(x^3+1), x]`

Out[1]= $\frac{\text{ArcTan}\left[\frac{-1+2x}{\sqrt{3}}\right]}{\sqrt{3}} + \frac{1}{3} \text{Log}[1+x] - \frac{1}{6} \text{Log}[1-x+x^2]$

Definite integral:

In[1]:= `Integrate[1/(x^3+1), {x, 0, 1}]`

Out[1]= $\frac{1}{18} (2\sqrt{3} \pi + \text{Log}[64])$

(1): The function specification.

(2): Description of the function specification.

Notice that there are plenty of examples to help illustrate the use of the function.

Common Problems

Mathematica will do its best to alert you of an error. In the example below, we see that we forgot a closing bracket:

(* after clicking *)

`Integrate[x^2, x`



Syntax::bktmcp : Expression "Integrate[x^2, x" has no closing "]"

If you misspell a command, often *Mathematica* will just spit your input right back at you:

```
(* forgot to capitalize the first letter *)  
sqrt[3]  
(* misspelled Sqrt *)  
Sqtr[3]  
sqrt[3]  
Sqtr[3]
```