

## Lesson 02

---

### Week of Monday January xx

#### Variables, Expressions and Functions

OUTLINE : Reading Napolitano Chapter 2

1. Numbers and Operators
2. Variables
3. Expressions
4. Functions
5. Additional Exercises

##### **EXERCISE 0** (review exercise)

Use Mathematica to calculate ...

(A) the *exact* solutions of the equation  $37x^2 + 41x - 47 = 0$  ;

(B) the *approximate* solutions accurate to 6 significant figures .

WRITE THE ANSWERS ON THE ANSWER SHEET.

---

### [1] Numbers and Operators

First, recall some things that we learned last time.

- Mathematica does exact calculations with numbers.
- The familiar arithmetic operations are written as  $^$   $*$   $/$   $+$   $-$
- There are many built-in mathematical functions, e.g., **Sqrt[x]** and **Cos[x]**.
- There are other built-in functions for other purposes, e.g. , **N[expr,d]** and **Plot[...]**.
- All Mathematica built-in functions start with a capital letter.
- The arguments of a function are always enclosed in square brackets.
- We can get help on a Mathematica command by typing **?Command**
- To execute the commands in an Input Cell, hit Shift-Enter.

---

### [2] Variables

Mathematica can operate on numbers. The operations are  $+$   $-$   $*$   $/$   $^$

In other words, Mathematica can do **arithmetic**.

What makes Mathematica very powerful is that Mathematica can operate on variables.

Again, the operations are  $+$   $-$   $*$   $/$   $^$  .

In other words, Mathematica can do **algebra**.

A **variable** is a letter, or a word, or a string of alphanumeric symbols.

There are some rules about naming variables, which you will eventually learn. Perhaps the simplest way to define a variable is to use lower-case letters (a,b,c,...) or words (mass, time, temp...) or a letter with a number (c1,c2,c3,...) or a greek letter ( $\alpha, \beta, \gamma, \dots$ ).

For example, to get  $\alpha$ , type `<esc>a<esc>` where `<esc>` is the escape key.

Normally (but not always) a variable stands for a number. The number may be known or unknown or constant or variable. It's just like in algebra.

`In[ ]:=`

```
Remove["Global`*"];
```

### Example Problem

A farmyard contains some chickens and some goats.

The number of heads is 16 and the number of legs is 44.

How many chickens are there?

### Solution

Define these variables:

ch = number of chickens; go = number of goats;

he = number of heads; le = number of legs.

Then  $ch + go = he = 16$  and  $2*ch + 4*go = le = 44$ .

Now solve these equations for ch.

We can use Mathematica to do the calculation (try it yourself!) ...

`In[ ]:=`

```
soln = Solve[{c + g == 16, 2 * c + 4 * g == 44}, {c, g}] (* Shift-Enter *)
```

`Out[ ]:=` `{{c -> 10, g -> 6}}`

### Comment on the command Solve

The Mathematica command `Solve[eqs, vars]` is good for solving systems of linear equations.

`In[ ]:=` `? Solve`

`Out[ ]:=`

Symbol ?

`Solve[expr, vars]` attempts to solve the system  
*expr* of equations or inequalities for the variables *vars*.

`Solve[expr, vars, dom]` solves over the domain *dom*. Common  
 choices of *dom* are Reals, Integers, and Complexes.

▼

In the Example Problem, `eqs` is `{ c + g == 16 , 2 * c + 4 * g == 44 }`;

that is, `eqs` is a list containing two equations. The double equals sign (`==`) tells Mathematica that these are equations.

In the Example Problem, `vars` is `{ c , g }`; that is, `vars` is the list of variables.

In the Example Problem, I gave the output from the Solve command the name “soln”. What is “soln”? “soln” is a list of two “replacement rules”. The replacements  $\{c \rightarrow 10\}, \{p \rightarrow 6\}$  solve the equations. Note that “soln” is not the solution! Rather it is a list of replacements; those replacements are the solution. So the use of **Solve** is tricky. We apply **Solve** to lists of equations and variables; Mathematica returns replacement rules. We have a little more work to do to actually extract the solutions from the replacement rules.

Now let's check the solution to the Example, using Mathematica,

```
In[ ]:= (* Check the answer *)
ch = c /. soln[[1]] (* extracts the number of chickens *)
go = g /. soln[[1]] (* extracts the number of goats *)
ch+go (* calculates # c + # g; it should be 16 *)
2*ch+4*go (* calculates 2 # c + 4 # g; it should be 44 *)
```

Out[ ]= 10

Out[ ]= 6

Out[ ]= 16

Out[ ]= 44

The replacement command  $\equiv /. \equiv$  "slash-dot" is explained below.

\*\*\*

#### **EXERCISE 1**

Solve this system of equations for x, y, z:

$\{ 2x+y+z == 5, x+2y+z == 7, x+y+2z == 9 \}$

WRITE THE ANSWERS ON THE ANSWER SHEET.

Another way to solve a system of linear equations is to use matrices and vectors.

We will study linear algebra later in the semester.

\*\*\*

### **Summary of “variables”**

We have seen several examples of variables.

Remember, a variable is a letter or a word or a list of alphanumeric characters.

Some variable names are not allowed. For example, these are examples of disallowed variable names:

C, D, E, I, N, Sqrt, Cos, Solve, 1a, 2a, 3b, 4c

They are not allowed because they are already used for some purpose by Mathematica.

Type ?C to see why C is disallowed.

Type ?D to see why D is disallowed.

Also, a variable name must start with a letter.

#### **EXERCISE 2**

Type ?C to see why C is disallowed.

Type `?D` to see why `D` is disallowed.  
Write the reasons on the answer sheet.

### [3] Expressions

Now that we have variables, we can make expressions.

A “simple expression” is a combination of numbers, variables, numerical operations, and parentheses (`()`).

The elements (i.e., numbers, variables, operations, parentheses) must obey the usual algebra formatting rules. You should already know the algebra formatting rules.

Numerical operations are `+` `-` `*` `/` `^`

Parentheses are `()` which enclose sub-expressions.

**You cannot use brackets, i.e., `[]` or `{}`, in an expression.**

Mathematica will usually allow you to type an invalid expression. However, if you try to execute an invalid expression, Mathematica will probably give you an error message. Then you must go back to the Input Cell and correct the error.

#### **EXERCISE 3**

Examples of expressions. Suppose  $x = 3$  and  $c = 5$ . Then calculate

(a)  $2x+c$ ;    (b)  $x^2+c$ ;    (c)  $x^2+c$ ;    (d)  $x^2*c$     (e)  $x^2c$

WRITE THE ANSWERS ON THE ANSWER SHEET.

A “complex expression” is a combination of simple expressions separated by numerical operations. The simple expressions may be enclosed in parentheses, but that is not always necessary. Sometimes we use parentheses to make an expression more readable, even if it is not necessary. However, it is better not to clutter up an expression with a bunch of unnecessary parentheses. That will make it less readable.

#### **EXERCISE 4**    *THE REPLACEMENT COMMAND; SLASH-DOT*

Evaluate  $(x^2 + 3x + 5) / (11x^2 + 13x + 15)$  at  $x = 4.66$ .

WRITE THE ANSWER ON THE ANSWER SHEET.

`In[ ]:=`

```
Remove[x]
expr1 = x^2 + 3 * x + 5
expr2 = 11 * x^2 + 13 * x + 15
answer = expr1 / expr2 /. {x -> 4.66}
```

`Out[ ] =`  $5 + 3x + x^2$

`Out[ ] =`  $15 + 13x + 11x^2$

`Out[ ] =`  $0.129418$

### The “slash-dot command”


The replacement command is `/.` (“slash dot”)

We used this command in **EXERCISE 4** and in the example about chickens and goats.

Now we need to understand it, because we will often use it.


Type this into Mathematica:

`?/.` (then hit Shift-Enter)

From the help information that comes up, do you understand the meaning of `/.`? This command is very important, and we’ll use it a lot, so be sure that you understand it. If you don’t fully understand the help information, then click on  and read the additional information, which includes some examples.

In[ ]:= `?/.`

Out[ ]:=

Symbol 

`expr/.rules` applies a rule or list of rules in an attempt to transform each subpart of an expression `expr`.  
`ReplaceAll[rules]` represents an operator form of `ReplaceAll` that can be applied to an expression.

▼

## /4/ Functions

### Mathematical definition

A function is a map from one set of numbers (called the domain of the function) to another set of numbers (called the range of the function). For each number in the domain there is an associated number in the range.

We might write the function as  $f(x)$  where  $x$  stands for a number in the domain and  $f(x)$  is the associated number in the range; for example, **Sin[x]** is the sine function of trigonometry.

In another problem we might write the function as  $x(t)$ . Then  $t$  is called the “independent variable”—a real number; and  $x(t)$  is called the “dependent variable”—the real number associated with  $t$  by the function  $x$ ; for example,  $t$  might denote time and  $x$  might denote position. For example, the height of a particle in free fall near the Earth’s surface is

$$y(t) = y_0 + v_0 t - \frac{1}{2} g t^2 \quad \text{where } g = -9.81 \text{ m/s}^2.$$

### Functions and Expressions

For our purposes there will be an equation that defines the function we are interested in, such as

$$f(x) = \text{an expression that depends on } x \text{ and other things} \quad (1)$$

The right-hand side of (1) is an expression, which may depend on variables, operations, functions, parentheses, etc, etc, etc, etc, etc, etc. The expression should depend in some way on  $x$ .

### Built-in functions and User-defined functions

We have already seen some built-in functions in Mathematica, such as  $\text{Sqrt}[x]$  ( $\equiv$ the square root of  $x$ ) and  $\text{Cos}[\theta]$  ( $\equiv$ the cosine of  $\theta$ ).

Mathematica has hundreds of built-in functions. These are pure mathematical functions.

We will also need USER-DEFINED functions. (You are the User!) For example, the solution to a physics problem might be a user-defined function.

We need to tell Mathematica when we (the User) define a function.

The most commonly used Mathematica command to define a function looks like this,

$$f[x_] = \text{expr}$$

where

- $f$  is the function name,
- $x$  is the independent variable name,
- $\text{expr}$  is an expression that depends on  $x$  and other things.

Important!  
Note the proper use of  $[\ ]$  and  $x_$

### A subtle point

Sometimes we do not write  $f[x_] = \text{expr}$  to define a function. Instead we write  $f[x_] := \text{expr}$ . You could try to understand the difference between  $:=$  and  $=$  by using the help files. Normally, use  $=$  to define a function. But if something is wrong, try  $:=$  instead of  $=$ .

$\text{In}[ ] :=$

? :=  
? =

$lhs := rhs$  assigns  $rhs$  to be the delayed value of  $lhs$ .  $rhs$  is maintained in an unevaluated form. When  $lhs$  appears, it is replaced by  $rhs$ , evaluated afresh each time.  $\gg$

$lhs = rhs$  evaluates  $rhs$  and assigns the result to be the value of  $lhs$ . From then on,  $lhs$  is replaced by  $rhs$  whenever it appears.

$\{l_1, l_2, \dots\} = \{r_1, r_2, \dots\}$  evaluates the  $r_i$ , and assigns the results to be the values of the corresponding  $l_i$ .  $\gg$

**EXERCISE 5** An example of a user-defined function ...

Define the function  $z(t) = A \cos(\omega t) \exp(-\beta t)$  where  $A = 10$ ,  $\omega = 1.57$  and  $\beta$

= 0.13. Plot the function for  $t \geq 0$ . [Use an appropriate domain for the values of  $t$ , i.e., large enough to see what is happening as  $t$  increases.] DESCRIBE  $z(t)$  IN WORDS ON THE ANSWER SHEET.

```
In[ ]:= (* First define the function. *)
Remove["Global`*"]
z[t_] = A * Cos[ $\omega$  * t] * Exp[- $\beta$  * t]
{A,  $\omega$ ,  $\beta$ } = {10, 1.57, 0.13}
(* Now make the plot. *)
Plot[z[t], {t, 0, 20},
  PlotRange -> {{0, 20}, {-12, 12}}]
(* Hit Shift-Enter to execute the plot *)
```

### Graphs of functions

To understand a function, we often plot a graph of the function, over some domain and range.

#### Command Plot

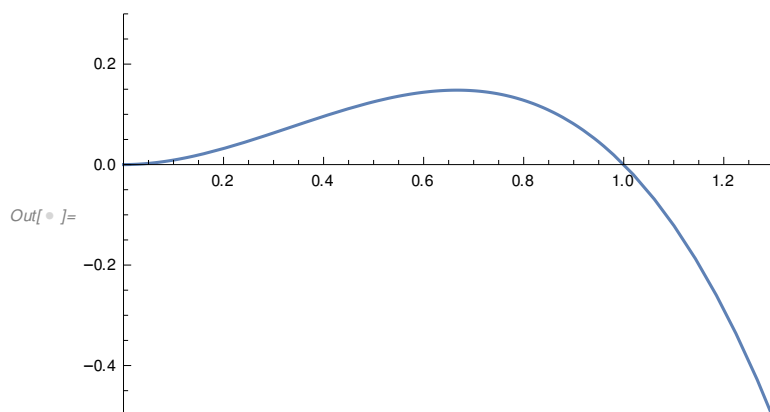
If we know  $f(x)$  then the simplest way to make a plot is the Copmmand Plot.

The general syntax is

```
Plot[f[x], {x, x1, x2},
  PlotRange -> {{x1, x2}, {f1, f2}}]
```

For example, plot  $x^2 - x^3$  for  $x$  from 0 to 1.3, using an appropriate plot range.

```
In[ ]:= Plot[x^2 - x^3, {x, 0, 2}, PlotRange -> {{0, 1.3}, {-0.5, 0.3}}]
```



#### Command ParametricPlot

Sometimes we have two functions  $f(x)$  and  $g(x)$  that both depend on one independent variable  $x$ .

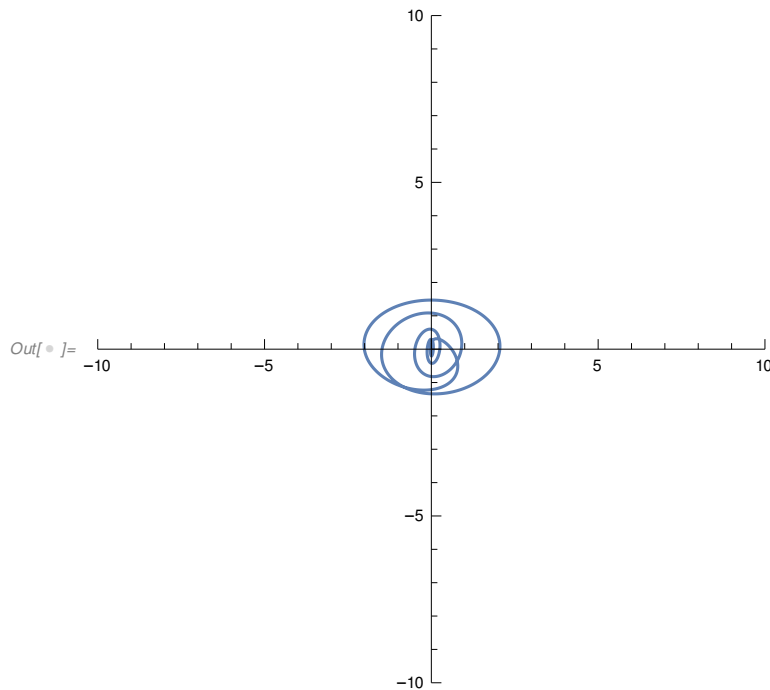
Then we can use **ParametricPlot** to make a graph of  $g$  versus  $f$ .

For example, suppose  $f(\tau) = \tau^2 e^{-\tau/2} \sin(1.57 \tau)$  and  $g(\tau) = \tau e^{-\tau/4} \cos(1.57 \tau)$ .

Plot  $g$  versus  $f$  for  $\tau$  from 0 to 30. Use an appropriate plot range.

In[ ]:=

```
ParametricPlot[{τ2 * Exp[-τ / 2] * Sin[1.57 * τ], τ * Exp[-τ / 4] * Cos[1.57 * τ]},
{τ, 0, 30}, PlotRange → {{-10, 10}, {-10, 10}}
```



## [5] More Exercises

### **Exercise 6** [BUILT-IN FUNCTION ; uses the command **Plot**]

Plot the function  $G(x) = \ln(x)$  for  $x$  from 0 to 10.

Sketch the result on the Answer Sheet. Neatness counts!

### **Exercise 7** [SUPERIMPOSE PLOTS ; uses **Plot** ]

Make a graph that shows both  $\ln(x)$  and  $x-1$  on the same graph.

You can see that  $x-1$  is a good approximation of  $\ln(x)$  for  $x$  near 1.

Explain that from Taylor's theorem.

Write the explanation on the Answer Sheet. Neatness counts!

### **Exercise 8** [MYSTERY FUNCTION ; uses **Plot** ]

Consider this function of complex numbers:  $F(\omega) = 3 / (\omega - 5 - 0.1i)$ .

Here  $i = \text{Sqrt}[-1]$ ; in Mathematica,  $\text{Sqrt}[-1] = I$ .

Plot the real and imaginary parts of  $F(\omega)$  on the same graph.

Use an appropriately large plot range.

What might this function represent in Physics?

Write the answer on the Answer Sheet. Neatness counts!



An Example from Astronomy

**Exercise 8 / A KEPLERIAN ORBIT ; uses ParametricPlot /**

The Cartesian coordinates of an asteroid in orbit around the sun are

$x(\varphi) = r(\varphi) \cos(\varphi)$  and  $y(\varphi) = r(\varphi) \sin(\varphi)$ , where  $r(\varphi) = a(1 - e^2) / [1 + e \cos(\varphi)]$ .

Notation:  $x, y$  = Cartesian coordinates;  $r, \varphi$  = plane polar coordinates.

Assume  $a = 4$  AU and  $e = 0.75$ .

Plot a graph of the orbit, i.e.,  $y$  versus  $x$ , for  $\varphi$  from  $0$  to  $2\pi$ .

Sketch the graph accurately on the Answer Sheet. Neatness counts! Sloppiness counts negative.