Source: Maple 9 Learning Guide
(Version 3, Sep. 30, 2014)

Begin important:

Please carefully note the following:

1) This worksheet has been modified since I went over it in class.

2) The array command, which was used in the original worksheet to demonstrate basic usage of Maple commands is now a deprecated (obsolete) feature of Maple.

3) The command which replaces it is Array (upper case), which is very similar in spirit to array, but which has additional functionality, and which operates somewhat differently.

4) I have changed all the usages of the array command in Version 1 of this worksheet to Array, and have modified the discussion/annotations appropriately, since some of the old comments no longer apply.

5) When you make your facsimile of the worksheet to answer problem 1 of Homework 2, please make a facsimile of this version, not the original, and be sure to type the section on array usage using the Array command.

6) Similarly, the discussion of the subs command has been modified to reflect that fact that substitutions can be made directly into the array elements (i.e. without using evalm) for arrays created using Array.

7) You will also be working with Arrays in other parts of the assignment. Again, be sure not to use the array command.

8) VERY IMPORTANT: You do not need to include any of the annotations that appear in italic (slanted font), just the ones in regular font (usually bolded), which are typically section headings, or which provide synopses of commands, syntax etc. The annotations that you can (and should) omit include these instrutions themselves i.e. everything between the Begin important and End Important "delimiters", inclusive.

End Important

2. Mathematics with Maple: The Basics

2.1 Introduction

NOTE: *When entering commands using "Maple Notation" vs "2-D Math Notation" (see Tools->Options->Display->Input Display), and especially when programming in Maple (writing scripts/procedures),* **you should be sure to end every statement with a semi-colon, or colon: the colon suppresses output.**

```
> 1 + 2;
```

$$3$$

```
> 1 + 3/2;
```

$$\frac{5}{2}$$

```
> 2*(3 + 1/3)/(5/3 - 4/5);
```

$$\frac{100}{13}$$

```
> 2.8754/2;
```

$$1.437700000$$

```
> 1+1/2;
```

$$\frac{3}{2}$$

2.2 Numerical Computations

Integer computations

```
> 1 + 2;
```

$$3$$

```
> 75 - 3;
```

$$72$$

```
> 5*3;
```

$$15$$

```
> 120/2;
```

$$60$$

```
> 100!;
```

$$933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000$$

Note the use of % in the following, which is short-hand for "last result". Similarly, %% and %%% can be used to retrieve the second-to-last and third-to-last results

respectively.

```
> length(%);
```

$$158$$

Commands for Working With Integers

```
> ifactor(60);
```

$$(2)^2\,(3)\,(5)$$

```
> igcd(123, 45);
```

$$3$$

```
> iquo(25, 3);
```

$$8$$

```
> isprime(18002676583);
```

$$true$$

Exact Arithmetic - Rationals, Irrationals, and Constants

```
> 1/2 + 1/3;
```

$$\frac{5}{6}$$

```
> Pi;
```

$$\pi$$

```
> evalf(Pi, 100);
```

```
3.14159265358979323846264338327950288419716939937510582097494459230\
   78164062862089986280348253421170 68
```

```
> 1/3;
```

$$\frac{1}{3}$$

```
> evalf(%);
```

$$0.3333333333$$

```
> 3/2*5;
```

$$\frac{15}{2}$$

```
> 1.5*5;
```

$$7.5$$

```
> sqrt(2);
```

$$\sqrt{2}$$

```
> sqrt(3)^2;
```

$$3$$

```
> Pi;
```

$$\pi$$

```
> sin(Pi);
```

$$0$$

```
> exp(1);
```

$$e$$

```
> ln(exp(5));
```

$$5$$

```
> 1/3 + 1/4 + 1/5.3;
```

$$0.7720125786$$

```
> sin(0.2);
```

$$0.1986693308$$

```
> Digits := 20;
```

$$Digits := 20$$

```
> sin(0.2);
```

$$0.19866933079506121546$$

Restore default number of digits for floating point expressions (10)

```
> Digits := 10;
```

$$Digits := 10$$

Arithmetic with Special Numbers

Note: I = sqrt(-1)

```
> (2 + 5*I) + (1 - I);
```

$$3 + 4\,I$$

```
> (1 + I)/(3 - 2*I);
```

$$\frac{1}{13} + \frac{5}{13}\,I$$

```
> convert(247, binary);
```

$$11110111$$

```
> convert(1023, hex);
```

$$3FF$$

```
> 27 mod 4;
```

$$3$$

Mathematical Functions

```
> sin(Pi/4);
```

$$\frac{1}{2}\sqrt{2}$$

```
> ln(1);
```

$$0$$

```
> ln(Pi);
```

$$\ln(\pi)$$

2.3 Basic Symbolic Computations

```
> (1 + x)^2;
```

$$(1+x)^2$$

```
> (1 + x) + (3 - 2*x);
```

$$4 - x$$

```
> expand((1 + x)^2);
```

$$x^2 + 2\,x + 1$$

```
> factor(%);
```

$$(1+x)^2$$

Symbolic differentiation and integration

Maple has separate 'inert' (upper-case) and 'active' (lower-case) commands for differentiation/integration; the 'inert' forms return the usual mathematical representation of the operations, unevaluated

```
> Diff(sin(x), x);
```

$$\frac{\mathrm{d}}{\mathrm{d}x}\sin(x)$$

```
> value(%);
```

$$\cos(x)$$

```
> diff(sin(x), x);
```

$$\cos(x)$$

```
> Int(cos(x)^2, x);
```

$$\int \cos(x)^2 \,\mathrm{d}x$$

```
> value(%);
```

$$\frac{1}{2}\cos(x)\sin(x)+\frac{1}{2}x$$

```
> int(cos(x)^2, x);
```

$$\frac{1}{2}\cos(x)\sin(x)+\frac{1}{2}x$$

```
> Sum(n^2, n);
```

$$\sum_n n^2$$

```
> value(%);
```

$$\frac{1}{3}n^3-\frac{1}{2}n^2+\frac{1}{6}n$$

```
> sum(n^2, n);
```

$$\frac{1}{3}n^3-\frac{1}{2}n^2+\frac{1}{6}n$$

```
> rem(x^3 + x + 1, x^2 + x + 1, x);
```

$$2 + x$$

```
> series(sin(x), x = 0, 10);
```

$$x - \frac{1}{6} x^3 + \frac{1}{120} x^5 - \frac{1}{5040} x^7 + \frac{1}{362880} x^9 + O(x^{11})$$

2.4 Assigning Expressions to Names

Syntax for Naming an Object (a.ka. Variable)

name := expression;

```
> var := x;
```

$$var := x$$

```
> term := x*y;
```

$$term := x\,y$$

```
> eqns := x = y + 2;
```

$$eqns := x = y + 2$$

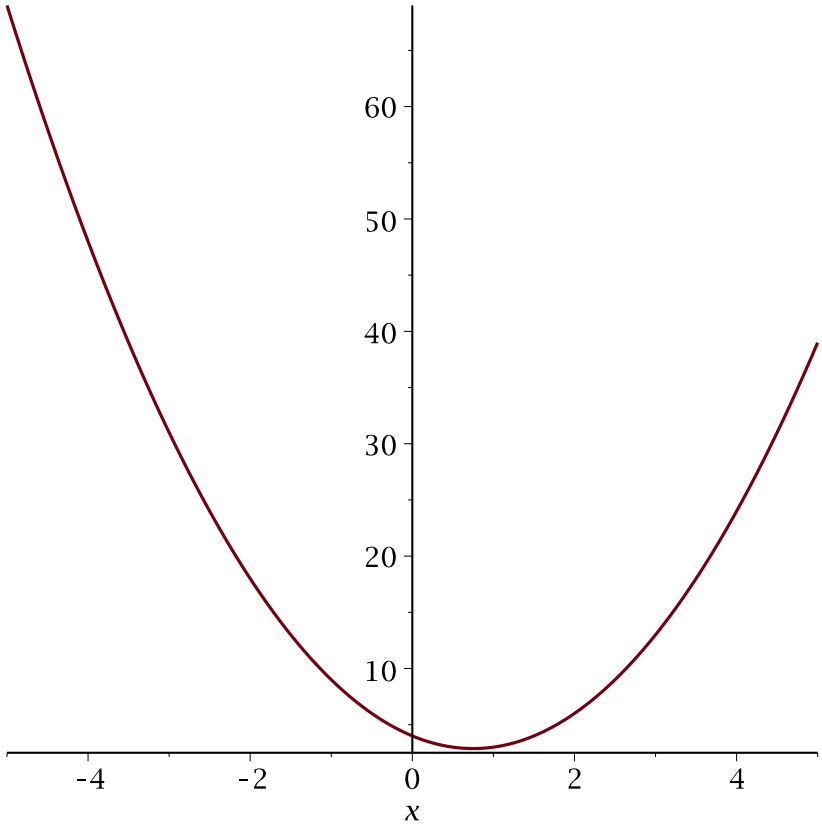
BEWARE OF USING = WHEN YOU MEAN := !!!

Maple Arrow Notation for Defining functions

```
> f := x -> 2*x^2 - 3*x + 4;
```

$$f := x \rightarrow 2\,x^2 - 3\,x + 4$$

```
> plot(f(x), x = -5 .. 5);
```



More on the Assignment Operator

```
> f := x -> x^2;
```

$$f := x \rightarrow x^2$$

```
> f(5);
```

$$25$$

```
> f(y+1);
```

$$(y+1)^2$$

Predefined and Reserved Names

```
> Pi := 3.14;
```

Error, attempting to assign to `Pi` which is protected. Try declaring `local Pi`; see ?protect for details.

```
> set := {1, 2, 3};
```
Error, attempting to assign to `set` which is protected. Try declaring `local set`; see ?protect for details.

2.5 Basic Types of Maple Objects

Expression sequences

```
> 1, 2, 3, 4;
```
$$1, 2, 3, 4$$

```
> x, y, z, w;
```
$$x, y, z, w$$

```
> 10, x, sin(y), (w + z)^3, 3.0;
```
$$10, x, \sin(y), (w+z)^3, 3.0$$

Simultaneous assignment of sequence of names (variables)

```
> f, g, h := 3, 6, 1;
```
$$f, g, h := 3, 6, 1$$

```
> f ;
```
$$3$$

```
> h;
```
$$1$$

Lists

```
> data_list := [1, 2, 3, 4, 5];
```
$$data_list := [1, 2, 3, 4, 5]$$

```
> polynomials := [x^2 + 3, x^2 + 3*x - 1, 2*x];
```
$$polynomials := [x^2+3, x^2+3\,x-1, 2\,x]$$

```
> participants := [Kathy, Frank, Rene, Niklaus, Liz];
```
$$participants := [Kathy, Frank, Rene, Niklaus, Liz]$$

Order and repeats of elements in lists is preserved

```
> [a, b, c], [b, c ,a], [a, a, b, c, a];
```
$$[a, b, c], [b, c, a], [a, a, b, c, a]$$

```
> letters := [a, b, c];
```

$$letters := [a, b, c]$$

Extraction of list element using indexing (subscript) operator [ ]

```
> letters[2];
```

$$b$$

```
> letters[0];
Error, invalid subscript selector
```

```
> letters[4];
Error, invalid subscript selector
```

Length of list

```
> nops(letters);
```

$$3$$

Sets

```
> data_set := {1, -1, 0, 10, 2};
```

$$data_set := \{-1, 0, 1, 2, 10\}$$

```
> unknowns := {x, y, z};
```

$$unknowns := \{x, y, z\}$$

Order of elements in sets is arbitrary, no repeated elements

```
> {a, b, c}, {c, b, a}, {a, a, b, c, a};
```

$$\{a, b, c\}, \{a, b, c\}, \{a, b, c\}$$

```
> {1, 2, 2.0};
```

$$\{1, 2, 2.0\}$$

```
> {a, b, c} union {c, d, e};
```

$$\{a, b, c, d, e\}$$

```
> {1, 2, 3, a, b, c} intersect {0, 1, y, a};
```

$$\{1, a\}$$

```
> nops(%);
```

$$2$$

More Operations on Sets and Lists

```
> participants := [Kate, Tom, Steve];
```

$$participants := [Kate, Tom, Steve]$$

```
> member(Tom, participants);
```

$$true$$

```
> data_set := {5, 6, 3, 7};
```

$$data_set := \{3, 5, 6, 7\}$$

```
> member(2, data_set);
```

$$false$$

```
> participants[2];
```

$$Tom$$

```
> empty_set := {};
```

$$empty_set := \{\}$$

```
> empty_list := [];
```

$$empty_list := [\,]$$

```
> nops(empty_list);
```

$$0$$

```
> old_set := {2, 3, 4} union {};
```

$$old_set := \{2, 3, 4\}$$

```
> new_set := old_set union {2, 5};
```

$$new_set := \{2, 3, 4, 5\}$$

```
> third_set := old_set minus {2, 5};
```

$$third_set := \{3, 4\}$$

Arrays (regularly structured collections of values/expressions)

IMPORTANT: As mentioned above, the Array command replaces the deprecated array command in newer versions of Maple. The latter should no longer be used.

1-dimensional arrays

```
> squares := Array(1..3);
```

$$squares := \left[\begin{array}{ccc} 0 & 0 & 0 \end{array}\right]$$

Note that Maple initializes array elements to 0 by default.

Individual elements selected/specified using indexing operator [ ] as with lists, but lists and 1-dimensional arrays are distinct Maple objects

```
> squares[1] := 1; squares[2] := 2^2; squares[3] := 3^2;
```

$$squares_1 := 1$$

$$squares_2 := 4$$

$$squares_3 := 9$$

```
> cubes := Array(1..3, [1, 8, 27]);
```

$$cubes := \left[\begin{array}{ccc} 1 & 8 & 27 \end{array}\right]$$

Selection of single array element using [ ]

```
> squares[2];
```

$$4$$

For named arrays that have a relatively small number of values (25 or fewer), Maple will display all values when the name is entered at the command line ...

```
> squares;
```

$$\left[\begin{array}{ccc} 1 & 4 & 9 \end{array}\right]$$

However, for larger arrays, such as ...

```
> larger_array := Array(1..30);
```

$$larger_array := \left[\begin{array}{c} 1 \text{ .. } 30 \text{ Array} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran_order} \end{array}\right]$$

... Maple outputs a summary format of the array. In such cases, Maple does not provide a straightforward mechanism for viewing all elements "in-line" (even using the evalm or print command described in the Maple 9 Learning Guide) , rather one can double-click on the summary format, which will open a data browsing window that will allow you to view and change the elements of the array. (You might want to try this as you make your copy of the worksheet).

The maximum size of the arrays that Maple will display inline can be changed using the interface command - specifically, interface(rtablessize=n), where n is an integer, will set the size to n.

2-dimensional arrays

```
> pwrs := Array(1..3, 1..3);
```

$$pwrs := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Individual element definition / display uses "double index"

```
> pwrs[1,1] := 1; pwrs[1,2] := 1; pwrs[1,3] := 1;
```

$$pwrs_{1,1} := 1$$
$$pwrs_{1,2} := 1$$
$$pwrs_{1,3} := 1$$

```
> pwrs[2,1] := 2: pwrs[2,2] := 4: pwrs[2,3] :=8:
```

```
> pwrs[3,1] := 3: pwrs[3,2] := 9: pwrs[3,3] := 27:
```

Since pwrs is a small array ("small" means 1 or 2 dimensional, with no more than 25 elements per dimension), Maple will display all values in-line when we enter the name to which the array has been assigned:

```
> pwrs;
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 8 \\ 3 & 9 & 27 \end{bmatrix}$$

```
> pwrs[2,3];
```

$$8$$

```
> pwrs2 := Array(1..3, 1..3, [[1, 1, 1], [2, 4, 8], [3, 9, 27]] );
```

$$pwrs2 := \begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 8 \\ 3 & 9 & 27 \end{bmatrix}$$

3-dimensional arrays

```
> array3 := Array(1..2, 1..2, 1..2,
> [ [ [1, 2], [3, 4] ], [ [5,6], [7, 8] ] ] );
```

$$array3 := \begin{bmatrix} 1..2\ x\ 1..2\ x\ 1..2\ Array \\ Data\ Type: anything \\ Storage: rectangular \\ Order: Fortran_order \end{bmatrix}$$

The subs Command (substitutes sub-expressions with other expressions)

```
> expr := z^2 + 3;
```

$$expr := z^2 + 3$$

```
> subs( {z = x + y}, expr );
```

$$(x+y)^2 + 3$$

Note: With arrays created using Array, we do not have to use evalm to force substitution in array elements; use of the subs command works as expected.

```
> subs( {2 = 9}, pwrs );
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 9 & 4 & 8 \\ 3 & 9 & 27 \end{bmatrix}$$

See also the algsubs command (illustrated at end of worksheet), which is considerably more powerful and flexible. Also note that in the above expressions the braces { } around the substitutions to be performed are not necessary (but are if there is more than one substitution to be made)

Strings (delimited with double quotes ", NOT single quotes ')

```
> "This is a string.";
```

"This is a string."

```
> "my age" := 32;
Error, invalid left hand side of assignment
```

```
> mystr := "I ate the whole thing.";
```

$$mystr := \text{"I ate the whole thing."}$$

Substring extraction

```
> mystr[3..5];
```

"ate"

Negative range values index the string from the right end, rather than the left

```
> mystr[11..-2];
```

"whole thing"

```
> newstr := cat("I can't believe ", mystr);
```

$$newstr := \text{"I can't believe I ate the whole thing."}$$

```
> length(newstr);
```

38

2.6 Expression Manipulation

The simplify Command

```
> expr := cos(x)^5 + sin(x)^4 + 2*cos(x)^2 -
>   2*sin(x)^2 - cos(2*x);
```

$$expr := \cos(x)^5 + \sin(x)^4 + 2\cos(x)^2 - 2\sin(x)^2 - \cos(2x)$$

```
> simplify(expr);
```

$$\cos(x)^4 (\cos(x) + 1)$$

```
> simplify(sin(x)^2 + ln(2*y) + cos(x)^2);
```

$$1 + \ln(2) + \ln(y)$$

```
> simplify(sin(x)^2 + ln(2*y) + cos(x)^2, 'trig');
```

$$1 + \ln(2y)$$

```
> simplify(sin(x)^2 + ln(2*y) + cos(x)^2, 'ln');
```

$$\sin(x)^2 + \ln(2) + \ln(y) + \cos(x)^2$$

The factor Command

```
> big_poly := x^5 - x^4 - 7*x^3 + x^2 + 6*x;
```

$$big_poly := x^5 - x^4 - 7x^3 + x^2 + 6x$$

```
> factor(big_poly);
```

$$x(x-1)(x-3)(2+x)(1+x)$$

```
> rat_expr := (x^3 - y^3)/(x^4 - y^4);
```

$$rat_expr := \frac{x^3 - y^3}{x^4 - y^4}$$

```
> factor(rat_expr);
```

$$\frac{x^2 + x\,y + y^2}{(x + y)\,(x^2 + y^2)}$$

The expand Command

```
> expand((x + 1)*(x + 2));
```

$$x^2 + 3\,x + 2$$

```
> expand(sin(x + y));
```

$$\sin(x)\cos(y) + \cos(x)\sin(y)$$

```
> expand(exp(a + ln(b)));
```

$$e^a\,b$$

The convert Command

```
> convert(cos(x), exp);
```

$$\frac{1}{2}\,e^{Ix} + \frac{1}{2}\,e^{-Ix}$$

```
> convert(1/2*exp(x) + 1/2*exp(-x), trig);
```

$$\cosh(x)$$

```
> A := Array(1..2, 1..2, [ [a, b], [c, d] ]);
```

$$A := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
> convert(A, 'listlist');
```

$$[[a, b], [c, d]]$$

```
> convert(A, 'set');
```

$$\{a, b, c, d\}$$

```
> convert(%, 'list');
```

$$[a, b, c, d]$$

The combine Command

Roughly speaking, combine can be viewed as an inverse of expand

```
> combine(exp(x)^2*exp(y), exp);
```

$$e^{2x+y}$$

```
> combine((x^a)^2, power);
```

$$x^{2a}$$

The lhs and rhs Commands

```
> eqn1 := x + y = z + 3;
```

$$eqn1 := x + y = z + 3$$

```
> lhs(eqn1);
```

$$x + y$$

```
> rhs(eqn1);
```

$$z + 3$$

The numer and denom Commands

```
> numer(3/4);
```

$$3$$

```
> denom(1/(1 + x));
```

$$1 + x$$

Substitutions using subs and algsubs

```
> expr := a^3*b^2;
```

$$expr := a^3 b^2$$

*The following will make the substitution only when the expression literally contains the sub-expression a*b*

```
> subs(a*b=5, expr);
```

$$a^3 b^2$$

Get the desired effect using algsubs

```
> algsubs(a*b=5, expr);
```

$$25\,a$$