Introductions:

At its heart, MATLAB® is a big calculator. To calculate something simply type it in at the "command prompt" and press Enter. Thus, to calculate 1 + 1 we type it in and press Enter. The screen should show:

**.**

*Run MATLAB, find the command window and the blinking cursor. Find the answer to the following arithmetic problems:*

* 1234+4321=?
* 104−765=?
* 47∗33=?
* 34=? *(The operator for "power" is the circumflex ^, usually found by pressing* Shift ⇑ 6
* *Find an approximation to*

 31 *(while you can of course use the fact that*x=x0.5*, you can also "look for" a dedicated function square root by learning how to use the* lookfor *command....)*

You may have noticed in the exercises that the answer is only given with 5 digits of accuracy (at most). For example, we can ask MATLAB for the value of π and get:

>> pi

ans =

3.1416

*Internally*, MATLAB keeps a 16 (more-or-less) digit version of the number it shows us, but to keep things orderly, it only displays the answer rounded to show 5 digits (by default). We can change this by issuing a command:

>> format long

>> pi

ans =

3.141592653589793

We can see this, by subtracting part of π from ans, which always holds the full, unrounded answer to the previous, unassigned expression:

>> format short

>> pi

MATLAB® is particularly convenient at calculating with lists of numbers. In fact, it was built for manipulating two-dimensional lists called matrices. An n-by-m matrix has n rows and m columns of numbers, and many MATLAB commands know how to work correctly and efficiently with them.

For example, if we have 10 grocery items whose price we would like to add up, we can write

>> sum([2.35 3.45 10.55 12.32 1.99 5.43 2.66 3.78 10.21])

ans =

52.7400

Here we used a function sum and its argument was a (row) vector we created "manually". Other vectors have shorthand notation (try them out with various numbers):

* Many zeros: zeros(n,m) (n and m must be positive integers)
* Many ones: ones(n,m) (same)
* An increasing list (step =1): n:m (m must be greater than n)
* An increasing list with step-size s: n:s:m (m might not be the last element of the list)
* A column vector (manual): [3 ; 2; 6 ; 7] (notice the semicolons)
* A column of increasing numbers (using transpose) (n:m)'

















































