PSYCH 20A (PROF. FRANE) – LECTURE 1
INTRODUCTION TO MATLAB

Basic arithmetic

`3 + 7`	addition
`3 - 7`	subtraction
`3 * 7`	multiplication
`5 / 2`	division
`3^2`	exponentiation
`(3 + 7) * 2`	Matlab follows order-of-operations (compare to 3 + 7 * 2)

Defining (assigning values to) variables

`x = 4`	the = sign means set the variable on the left of the = sign equal to what's on the right
`x`	now when you type `x`, Matlab outputs what `x` is (4)

Now you can use x just like any other number:

`x + 1`	outputs 5 (note that this doesn't change what `x` is; it just tells you what `x + 1` is)
`x + 2`	outputs 6
`x * 2`	outputs 8
`x^2`	outputs 16
`y = 2 + 3`	sets y equal to 5
`y`	outputs what y is (5)
`x + y`	outputs 9 (because x is 4 and y is 5)
`x + z`	here we get an error because we haven't defined z as anything
`x = x + 1`	changes x to the current value plus 1 (so after this command runs x will be 5)
`myAge = 19`	variable names should typically be more descriptive than just a single letter
`myAge`	outputs what `myAge` is (19)
`Myage`	here we get an error, because variable names are case-sensitive and we haven't defined `Myage` as anything

Vectors

A vector is a series of values in some particular order. Vectors are useful when we have a set of measurements of a single variable (such as 4 students' scores on a math test).

Defining vectors using square brackets:

`mathScore = [84  75  72  82.5]`	define a vector called `mathScore` containing those 4 values
`mathScore = [84, 75, 72, 82.5]`	same as above (commas between values are optional)
`mathScore`	outputs the values in `mathScore` to the command window

Artithmetic operations using a vector and a constant:

`mathScore + 1`	outputs `mathScore` but with 1 added to each value
`mathScore * 10`	outputs `mathScore` but with each value multiplied by 10

Defining vectors of equally spaced values using the colon shortcut:
To make a vector of values that increase in increments of positive 1, enter the lower bound, followed by a colon, followed by the upper bound. For example:

```
0:10                          outputs the vector [0 1 2 3 4 5 6 7 8 9 10]
oneToFour = 1:4               defines oneToFour as the vector [1 2 3 4]
```

To make a vector of equally spaced values using an increment other than positive 1, enter the lower bound, followed by a colon, followed by the increment, followed by a colon, followed by the upper bound. For example:

```
zeroToTenByTwos = 0:2:10      defines zeroToTenByTwos as the vector [0 2 4 6 8 10]
tenToOne        = 10:-1:1     defines tenToOne as the vector [10 9 8 7 6 5 4 3 2 1]
```

Example exercise (defining a vector of transit times in minutes, then converting those times to hours)**:**

```
transitMin  = [15 17 5 7 8 22 40 2 18 6 23 11 15 16 35]
transitHour = transitMin / 60
```

Indexing

The *index* is the position of a value in the vector.

```
transitTime(3)                outputs the 3rd value in the vector
transitTime(end)              outputs the last value in the vector
transitTime(3:5)              outputs the 3rd through 5th values in the vector
transitTime([2 5 6])          outputs the 2nd, 5th , and 6th values in the vector
transitTime(7:end)            outputs the 7th through last values in the vector
transitTime(end-1)            outputs the next-to-last value in the vector

transitTime(2) = 12           changes the 2nd value in the vector to 12
```

Note that unlike in some programming languages, in Matlab the first index is 1 (not 0).

Vector orientation and transposition

Generally speaking, a vector doesn't necessarily have a particular orientation (e.g., horizontal or vertical); it can just be a series of values in a particular order. However, in Matlab, a vector always has an orientation.

So far, the vectors we've defined have been *row vectors*. That is, they are oriented horizontally from left to right. We can convert a row vector to a *column vector* (or vice versa) by *transposing* it. We do this using the straight single-quote (apostrophe) symbol. For instance, if we define `mathScore` as the vector [84 75 72 82.5] then `mathScore'` outputs the following column vector:

```
84
75
72
82.5
```