Programming: Step 0

You write instructions to the computer in a *computer language*, like c, c++, fortran, java, assembler, perl, Matlab, etc. We will use the Matlab language in this course. Your instructions, written in the computer language, are called *source code*. The source code is read and interpreted by the computer and saved in a language suitable for that particular type of computer (*executable code*). When you ask that your program run, the computer attempts to follow your (translated) instructions.

Usually your program will manipulate data (like plot a picture or solve an equation, etc.). We will assume that the data resides in computer memory. In your source code you will give names to data, called *variable names*. You can picture a variable name as an alias for the place in memory where the datum is stored. As in mathematical notation, we can use symbols to represent objects that may or may not have fixed values. For example, $ax^2 + bx + c$ represents "parabola" whether or not we have fixed values for a, b or c, and if a real value for x is fixed, then $ax^2 + bx + c$ is a number.

An *assignment statement* is a way to associate a variable name like a or x or `MaxIterations` to a piece of data. The variable name `fbase` can be assigned to the number 2 with the assignment statement `fbase = 2` This does *not* represent an equation. It looks like an equation, but it isn't; other notations are (or have been) used: `fbase` $\leftarrow$ `2` or `fbase` := `2` or even `let fbase = 2` Now if we wanted to store 34 in `fbase`, we could write `fbase = 17*fbase` To swap the values in `bop` and `cool`, one might use the assignment statements

```
t = bop        bop = cool        cool = t
```

A *branching statement* is a way to instruct the computer to do or not do something based on some condition. The fundamental branching statement (or conditional) is the `if` statement. Here is an "*if-then-else*" construct:

`if x,` *dosomething* `else,` *dosomethingelse* `end`

Here, x is an expression that can be interpreted as either true or false. If x is true, then the computer will execute the instructions *dosomething*, and if x is false it will execute *dosomethingelse*. Now *dosomething* or *dosomethingelse* might consist of any number of instructions (including other if statements). Different languages have different syntaxes, but the "if-then-else" construct is common to all.

A *looping statement* is a way to instruct the computer to execute something over and over, until some condition occurs. The fundamental looping statement is the `while` statement:

`while x,` *dosomething* `end`

Here, if x is true, then the computer will execute the instructions *dosomething*, and then it will loop back to the `while` and see if x is still true. It will continue to do this until x is false, in which case it continues executing statements after the `end`. Again, *dosomething* might be any number of instructions.