Variables in Matlab

Making variables in Matlab are a lot easier than doing so in other languages because Matlab is inferred. In Matlab, when declaring a variable, you *only* need to clarify the variable name and the value(s) assigned to it. There does not need to be a "variable type" in front of the name like in many other languages.

However, the naming rules of Matlab are similar to those of other languages:

- No punctuations are allowed except underscore (_)
- Numbers cannot be the first letter
- Only English Alphabets, numbers, and underscores are allowed in naming
- The name cannot be one of the reserved words, such as "if", "else", "for", "while" … etc.
    - You can always use the function "iskeyword" to test if the variable name is one of the reserved word. the command would be like this:
      ans = iskeyword('your variable name')
      If it is an illegal name, the function will return 1 and vise versa. Using iskeyword function without any input will return a list of reserved words.

Even though you are not required to define variable types, Matlab still has a couple variable types, and it decides which one is correct by the value you assign to the variable. Also, Matlab is a weakly typed programming language because its variables are implicitly converted, meaning that if you want to change a variable's type, you only have to assign it a new value and Matlab will automatically change it to the default type of the input value. Here are the most common types:

A. Double:

Double (Double-Precision Floating Point), the default variable types. It can store a single number that is positive, negative or zero. It can contain decimals. Everything would be rounded up to the fifteenth significant number. The declaration syntax is (variables) = (assigned value). For examples:

input:

```
>> f = 0.1234567890123456
```

output:

```
f =
   0.123456789012346
```

(if you got something different, it is because your output format isn't the same as mine's! how to change it : http://kb.mit.edu/confluence/pages/viewpage.action?pageId=3907432)

input:

```
>> f = 1234567890123456
```

output:

```
f =
  1.23456789012346e+18
```

B. Vectors (Arrays):

Vectors in Matlab are easy to make and have a lot of uses like plotting data points. There are no limitations on the amount of information you can store into vectors.

Of course, you can use any variables you created earlier in the declaration of vectors (see examples below). A vector can only have one row, if it has more than one row, then it would become a matrix. The declaration syntax is (variable name) = [(values inside vector)]. Use a comma (,) or space to separate values. When creating a set of numbers that have a pattern, there is a shortcut - colons. The syntax of colons is (starting number):(ending number), which will give you a set of numbers that range from the starting number to the ending number with the increment of 1, and (starting number):(increment):(ending number), which will give you a set of numbers that range from the starting number to the ending number with the increment of increment.

For example:

input:

```
>> x = [1,2,3]
```

output:

```
x =
    1    2    3
```

input:

```
>> d = 1
>> x = [d, d+1 ,d+2]
```

output:

```
x =
    1    2    3
```

input:

```
>> x = 1:5
```

output:

```
x =
    1    2    3    4    5
```

input:

```
>> x = 1:2:9
```

output:

```
x =
    1    3    5    7    9
```

input:

```
>> x = [1:2:9,1]
```

output:

```
x =
    1    3    5    7    9    1
```

C. Matrices (2D arrays and so on):

One of the main reasons why many people use computers and calculators are for Matrices, because they are simply too hard to be done by hands. In Matlab, it keeps the dimensions of a matrix consistent, meaning that if a matrix is 2 by 5, there must be five numbers in each row and 2 in each column, otherwise Matlab will throw an error message such as below:

```
Error using vertcat
```

Dimensions of matrices being concatenated are not consistent.

Just like vectors, you can use any variables you created earlier in the declaration of a matrix as long as their variable types match. You can also use any previously created vectors. The declaration syntax for a matrix with two rows is (variable name) = [(values inside matrix); (values inside matrix)]. Use a comma (,) or space to separate values in each row and a semicolon (;) to separate rows in the matrix.

For example:

input:

```
>> x = [1,2;3,4]
```

output:

```
x =
    1    2
    3    4
```

input:

```
>> a = [1,2]
>> b = [3,4]
>> x = [a;b]
```

output:

```
a =
    1    2
b =
    3    4
x =
    1    2
    3    4
```

input:

```
>> a = [1,2, 3, 4, 5, 6]
>> b = [11,12]
>> x = [a; 7:10, b]
```

output:

```
a =
    1    2    3    4
b =
   11    12
x =
    1    2    3    4    5    6
    7    8    9    10    11    12
```

Beyond 2D matrices, there are 3D matrices, 4D matrices and so on. The declaration syntax is: (variable) = zeros((dimension), (dimension), (dimension)...). Note that this is not the actual way to create a matrix, it actually means to create a multi-dimensional matrix that has 0 for all the numbers.

D. String:

A String is basically a text and does not have to be more than one letter. Strings are commonly used as arguments and titles of anything in matlab, but of course they have other functions. A String can contain any character, as long as the characters are surrounded by a pair of single quotes. The only two characters that will cause some trouble are single quotes (‘) and new line characters (char(10) or \n). If you want to include a single quote in your string, you have to put two consecutive single quotes (“), so Matlab does not take it as the end of string. If you want to include a new line character, you will have to declare a string by either using the method sprintf or declare it in the form of a string vector. To avoid getting off-topic, I will not go over the rules of sprintf. The declaration syntax is (variable) = ‘(any character(s))’. For example,

input:

```
>> d = {‘Hi!’ ,char(10), ‘ H0w’, ‘ @r3’, ‘ Y&ou’, ‘ t*#oday??’}
```

output:

```
d =
    Hi!
    H0w @r3 Y&ou t*#oday??
```

input:

```
>> d = ‘I’’m fine!’
```

output:

```
d =
     I'm fine!
```

input:

```
>> d = sprintf('That is great!\nBye!')
```

output:

```
d =
     That is great!
     Bye!
```

(char(10) and \n are the new line symbols)

E. Boolean:

Booleans, in numerical form, only have two values, 0 and 1, which translate to false and true. Booleans can be used in logical statements such as if-statements. Double variables can also be used as booleans, except that in double, 0 is false and everything else is true. The declaration syntax is (variable) = (true or false). If a number is put instead of true or false, Matlab will make it a double instead of a boolean. For example,

input:

```
>> x = true
```

output:

```
x =
    1
```

input:

```
>> x = false
```

output:

```
x =
    0
```

There are more data types such as Integer, Single, Structures, Listeners, etc. Those are more advanced topics that will not be covered in this tutorial. If you are interested in those, please check the link below:

http://www.mathworks.com/help/matlab/data-types_data-types.html