Chapter 03 - Variables

Table of Contents

Like most programming languages, Matlab let you use variables to store values.

Declaring Variables as Values

In Matlab you do not need to declare variables or state their types, just jump in and use them. For example, try the following:

```
x = 10


x =

    10

```

This will create a variable called `x` (or re-use the existing one if you've previously assigned a value to `x` during this session), storing `10` as its value. You can now use `x` in any calculation and it will be replaced with the value `10`. It also parrots back to you what the variable is, just to notify you what you did. We'll see how to keep it quiet soon.

You should know that in Matlab all of your variables are actually storing matrices, not just an individual number. In the example above, the variable `x` is really being initialized with a one-by-one matrix that contains the value `10`. This observation will become important later but you can totally ignore it for now.

Accessing Stored Values

Once you've stored one or more values in variables, you can recall them at any time by using the variable names. For example just typing the variable name will tell you what it contains:

```
x


x =

    10

```

Below is a very simple example of a session where we store some values in variables and then use the variables later in a calculation:

```
width = 5


width =

     5


length = 10


length =

    10


area = length * width


area =

    50
```

Of course the three variables (`width`, `length` and `area`) created above can all be used in any subsequent calculations that we might want to do.

Altering Variables Depending on Themselves

We can also do something that might confuse the math majors at first. Let's first do:

```
x=7


x =

     7
```

and then

```
x=x+1


x =

     8
```

Do you see what this does? In mathematics you may think of `x=x+1` as an equation, and one with no solution. However in this context in Matlab it's treated as an assignment. The way it works is that the right side `x+1` is evaluated to get `8` and then this is assigned to the left side `x`. The result of this line is that `x` is now `8`. We can even do more:

```
x=x+2
```

```
x =

    10
```

Now `x` is `10`.

```
x=x/2
```

```
x =

     5
```

Now `x` is `5`. And so on.

A note about = vs ==

If you're not a programmer it will help for you to understand that in most programming languages `=` is used to assign values and `==` is used to compare them. We'll work with `==` later.

A Warning About Stored Variables

It is easy to forget that a variable contains a value. Later on when we deal with functions you may need to figure out `f(x)` and if you've forgotten that `x=10` you may get some unexpected results. So pay attention to the variables you've assigned.

Using ans

By now you must have noticed that when you ask Matlab to do a calculation for you (and you are not storing the result explicitly in a variable), Matlab responds with something like:

```
2*5
```

```
ans =

    10
```

Conveniently, `ans` is itself a variable that you can use in subsequent calculations. It simply stores whatever the result of the most recent calculation was. Here is a simple example of a session in which we make use of `ans`

```
15 * 3 + 11
```

```
ans =

    56
```

```
ans / 7
```

```
ans =

     8
```

Notice that now `ans` is `8`.

```
ans + 9
```

```
ans =

    17
```

Clearing the Variable List

If you'd like to start from scratch and eliminate all of the variables that you've created, just type the clear command:

```
clear
```

And now your variables are gone, so our `x` from way above has vanished:

```
        x

        Undefined function or variable 'x'.
```

Be sure that you really don't need the values of the variables before you do this!

Semicolons at the end to keep quiet!

Last but absolutely not least, if you put a semicolon at the end of a Matlab command the command is still executed but nothing is given as `ans`. For example note the difference between:

```
a = 3*4
```

```
a =

    12
```

and

```
a = 3*4;
```

In both cases `a` is assigned the value `12` but in the second case Matlab doesn't say anything about it. Very often in Matlab you want something done and you want Matlab to be quiet about it.

Published with MATLAB® R2016b