MATLAB Workshop 8 - More on Functions

Objectives: Learn more about functions: how values are passed to and returned from functions and how functions interact with the MATLAB workspace.

- **More on functions**

An understanding of some of the properties of the functions that you are learning to create interact with the MATLAB workspace is critical for control of information flow (data or values) in your scripts. It is also important for ***debugging*** functions that you design. Frequently, functions will have errors when first created. The process of removing the errors so that the functions work properly and provide correct answers is called ***debugging***.

This workshop requires that MATLAB scripts named **`wkshp8_ac1.m`**, **`wkshp8_ac3.m`**, and **`wkshp8_ac4.m`** be in the current directory. These scripts are available on the program disk. Please copy them into the current MATLAB directory before proceeding.

(1) Passing values to functions.

Whenever a function is *called*, that is, whenever a function is used in the workspace, in a script, or in another function, MATLAB seeks to pass *values* (not variables) that will be needed by the function to do its job. Consider the function **`wkshp8_ac1.m`**, defined here.

```
function  [out1, out2] = wkshp8_ac1(in1, in2)
% Workshop 8, Activities 1 and 2
% John F. Patzer II
% MATLAB Primer
% 26 June 2003

% variable dictionary
%   in1     incoming value
%   in2     incoming value
%   out1    outgoing value
%   out2    outgoing value

% calculations
    out1 = in1+in2;
    out2 = in1-in2;
```

Start MATLAB, or if MATLAB is already running, enter

```
» clear all
```

to start with a clear workspace. Next enter

```
» [a,b] = wkshp8_ac1(1,1)
a =
     2
b =
     0
```

This statement *called* **`wkshp8_ac1.m`** and sent it the two values listed in parentheses after the function name. The values were provided as numbers. The first value in the list was assigned to the first input variable in the function definition (**`in1`**) and the second value to the second input variable (**`in2`**) in the function definition.

MATLAB then performed the calculations listed in the function definition and returned two values as listed in the function definition. The first value in the output list (associated with **out1**) was assigned to the first variable on the output side of the function call (**a**). The second value in the output list (associated with **out2**) was assigned to the second variable on the output side of the function call (**b**).

- MATLAB transfers values, one-for-one, for both the input list and output list of functions.

Now enter

```
» [c,d] = wkshp8_ac1(a,b)
c =
     2
d =
     2
```

This time, MATLAB obtained the *values* associated with **a** (= 2) and **b** (= 0) and passed them to the function. Again, the first value in the list was assigned to the first input variable in the function definition (**in1**) and the second value to the second input variable (**in2**) in the function definition. The values resulting from the calculations were returned with the first (**out1**) being assigned to **c** and the second (**out2**) being assigned to **d**.

Now try this

```
» [e,f] = wkshp8_ac1(a*c,b+d)
e =
     6
f =
     2
```

Again, MATLAB performed the indicated arithmetic operations to produce the *values* associated with **a*c** (= 4) and **b+d** (= 2) before passing them on to the function. The first value in the list was assigned to the first input variable in the function definition (**in1**) and the second value to the second input variable (**in2**) in the function definition. The values resulting from the calculations were returned with the first (**out1**) being assigned to **e** and the second (**out2**) being assigned to **f**.

- Values passed to a function can be specified by
  - Listing an actual value, e.g., **1626**. MATLAB will transfer the value directly.
  - Listing a variable name, e.g., **angle**. MATLAB will get the associated value for transfer.
  - Listing an arithmetic expression, e.g., **2*pi**. MATLAB will do the calculation and transfer the value.

- A corollary to the above is that variable names in the calling statement and function definition do not need to be the same!!! (Note that none of the variable names used in the calling statements were the same as those used in the function definition (**in1**, **in2**, **out1**, and **out2**).
- Only the position in the list matters since MATLAB transfers the values one-for-one into the function and one-for-one out of the function.

(2) Functions do not add their variables to the workspace.

Recall that whenever a script is run, all variables in the script on the right hand side of the assignment operator are added to the workspace. Functions do their calculations on a piece of scrap paper. Values are transferred to the scrap paper, calculations performed on the scrap paper, and then values are transferred from the scrap paper back to the

workspace. As a result, variables defined inside functions are never associated with the workspace. To clearly see this, enter

```
» who
Your variables are:
a  b  c  d  e  f
```

Only the variables defined by placing them on the right hand side of the function calling statement in the three calls to **wkshp8_ac1** are in the workspace. **in1**, **in2**, **out1**, and **out2**, which belong to the function are not present. They were only present on a piece of scrap paper which has now been discarded.

- Functions do not add variables to the workspace.

(3) Functions will not overwrite existing values for variables of the same name in the workspace.

Unlike scripts, which operate in the workspace, functions operate on a piece of scrap paper. Only values are transferred between the workspace and the function. Hence, functions can have variables of the same name as those in the workspace without interfering with the values associated with variables in the workspace. (Another reason to like functions - they do not mess with your workspace!).

Consider the function **wkshp8_ac3.m**, defined here.

```
function  [out1, out2] = wkshp8_ac3(in1, in2)
% Workshop 8, Activity 3
% John F. Patzer II
% MATLAB Primer
% 26 June 2003

% variable dictionary
%   in1      incoming value
%   in2      incoming value
%   a        an internal variable
%   b        an internal variable
%   c        an internal variable
%   d        an internal variable
%   out1     outgoing value
%   out2     outgoing value

% calculations
    a = 1234;
    b = 6 + 3*i;
    c = a*b;
    d = c/b;
    out1 = in1+in2;
    out2 = in1-in2;
```

This function makes assignments to the variables **a**, **b**, **c**, and **d**, as well as **out1** and **out2**. Remember, **a** (=2), **b** (=0), **c** (=2), and **d** (=2) are variables currently in your workspace with the values indicated. Run the function by entering

```
» [g,h] = wkshp8_ac3(1,3)
g =
     4
```

```
h =
    -2
```

Now check on the values of the variables **a**, **b**, **c**, and **d** by entering the variable name at the command prompt. They should be the same as shown above - not the strange values assigned to them in the function.

- Running a MATLAB function does not affect any variable values in your workspace.
  - Variable names in functions are associated with the function only.
  - You can use the same variable name in functions without changing its value in the workspace.
  - Names are local: Jane Doe in Washington, D.C. is not the same as Jane Doe in Vancouver, B.C.

(4) Functions will only use values passed in the calling statement.

This issue is important in designing and debugging functions. A common method for developing functions that can lead to problems is to initiate the function design as a script. Because scripts have access to the workspace and functions do not, this can lead to some frustrating consequences unless you realize that functions actually operate on a piece of scrap paper and do not have access to any variables and their associated values in the workspace.

Consider the function **wkshp8_ac4.m**, defined here.

```
function  [out1, out2] = wkshp8_ac4(in1, in2)
% Workshop 8, Activity 4
% John F. Patzer II
% MATLAB Primer
% 26 June 2003

% variable dictionary
%   in1      incoming value
%   in2      incoming value
%   a        an internal variable
%   b        an internal variable
%   c        an internal variable
%   d        an internal variable
%   out1     outgoing value
%   out2     outgoing value

% calculations
    c = a*b;
    d = c/b;
    out1 = in1+in2;
    out2 = in1-in2;
```

This function was developed originally in script form and only later translated to a function. Execute the function by entering

```
» [p,q] = wkshp8_ac4(1,3)
??? Undefined function or variable 'a'.
Error in ==> C:\temp\wkshp8_ac4.m
On line 18  ==>  c = a*b;
```

MATLAB objects because the indicated calculation requires a value for **a** that was never assigned in the function and is, therefore, not available. **a** may have been present in the workspace when the function was being developed as a script (which would have access to the workspace), but **a** is definitely not defined in the function (which does not have access to the workspace) prior to trying to use it. Hence the error message.

- Avoid developing functions from scripts. Follow the design procedure in Workshop 7 instead.

(5) Semicolons are used to suppress display from functions.

We seek to explicitly control display from our scripts and functions to be user friendly and show only what we want to show in the format we want to show it. Hence, we use semicolons at the end of assignment statements in functions to suppress unwanted display from the function (the results of intermediate calculations are rarely of interest to the user).

Edit the function **`wkshp8_ac1.m`** to remove the semicolons following the assignment statements for **`out1`** and **`out2`**. Run the function. What displayed? Even though the results of executing the assignment statements in the function were displayed, the variables **`out1`** and **`out2`** have not been added to the workspace. (Functions cannot add variables to the workspace!) This is easily demonstrated by entering **`out1`** at the command prompt. What happens?

Recap: You should have learned

- Functions run on scrap paper.
- Only values, not variables, are passed between the workspace and a function.
- Functions cannot add variables to the workspace.
- Functions cannot change variable values in the workspace.
- Functions cannot access variables or variable values in the workspace.
- A MATLAB command in a function has nothing to do with a MATLAB command in the command window.
- Semicolons are used to suppress unwanted display from functions.